Oracle9*i*

Recovery Manager User’s Guide

Release 2 (9.2)

March 2002

Part No. A96566-01

ORACLE®

Oracle9*i* Recovery Manager User’s Guide, Release 2 (9.2)

Part No. A96566-01

Primary Author: Lance Ashdown

Contributors: Beldalker Anand, Tammy Bednar, Don Beusee, Senad Dizdar, Muthu Olagappan, Francisco Sanchez, Steve Wertheimer

Graphic Designer: Valarie Moore

Contents

17 Querying the RMAN Repository

Send Us Your Comments

Oracle9*i* Recovery Manager User's Guide, Release 2 (9.2)

Part No. A96566-01

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

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

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

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227   Attn: Server Technologies Documentation Manager
- Postal service:
  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

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

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

Preface

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

Oracle9*i* Recovery Manager User’s Guide is intended for database administrators who perform the following tasks:

- Back up, restore, and recover Oracle databases
- Perform maintenance on backups and copies of database files

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

- Relational database concepts and basic database administration as described in *Oracle9i Database Concepts* and the *Oracle9i Database Administrator’s Guide*
- Basic backup and recovery concepts and strategies as described in the *Oracle9i Backup and Recovery Concepts*
- The operating system environment under which you are running Oracle

Organization

This document contains:

Part I, "Getting Started with Recovery Manager"

This section gives you basic information necessary to use Recovery Manager (RMAN).

Chapter 1, "Introduction to Recovery Manager"

This chapter gives a short explanation of the purpose and functionality of the RMAN utility.

Chapter 2, "Connecting to Databases with RMAN"

This chapter describes how to start RMAN and connect to target, catalog, and auxiliary databases.

Chapter 3, "Quick Start to Recovery Manager"

This chapter gives a simple introduction to using basic RMAN features.

Part II, "Recovery Manager Architecture and Concepts"

This section gives a detailed conceptual account of RMAN functionality.

Chapter 4, "Recovery Manager Architecture"

This chapter describes the basic components of the RMAN environment and the nature of the RMAN command interface.

Chapter 5, "RMAN Concepts I: Channels, Backups, and Copies"

This chapter describes RMAN channels, both manual and automatic, as well as the uses of the `BACKUP` and `COPY` commands.

Chapter 6, "RMAN Concepts II: Restore, Recovery, and Duplication"

This chapter describes the uses of the `RESTORE`, `RECOVER`, and `DUPLICATE` commands.

Chapter 7, "RMAN Concepts III: Maintenance"

This chapter describes the various commands used to generate reports, perform maintenance on backups, and store files and scripts in the recovery catalog.

Chapter 8, "Configuring the Recovery Manager Environment"

This chapter describes how to set up and configure the RMAN environment.

Part III, "Performing Backup and Recovery with Recovery Manager"

This section describes how to perform backup and recovery operations.

Chapter 9, "Making Backups and Copies with Recovery Manager"

This chapter describes how to use the `BACKUP` and `COPY` commands.

Chapter 10, "Restoring and Recovering with Recovery Manager"

This chapter describes how to use the `RESTORE` and `RECOVER` commands.

Chapter 11, "Performing RMAN Tablespace Point-in-Time Recovery"

This chapter describes how to recover one or more tablespaces to a noncurrent time without affecting the rest of the database.

Chapter 12, "Duplicating a Database with Recovery Manager"

This chapter describes how to use `DUPLICATE` to create a copy of the target database.

Chapter 13, "Creating a Standby Database with Recovery Manager"

This chapter describes how to use `DUPLICATE` to create a standby database.

Chapter 14, "Tuning Recovery Manager"

This chapter gives tips for improving RMAN backup and restore performance.

Chapter 15, "Recovery Manager Troubleshooting"

This chapter gives tips for diagnosing and responding to RMAN problems.

Part IV, "Maintaining the Recovery Manager Repository"

This section describes how to create the RMAN repository, query the RMAN repository, and maintain the backups recorded in the repository.

Chapter 16, "Managing the Recovery Manager Repository"

This chapter describes how to create and manage a recovery catalog, as well as how to run RMAN without using a recovery catalog.

Chapter 17, "Querying the RMAN Repository"

This chapter describes how to access the information stored in the repository.

Chapter 18, "Performing Maintenance with Recovery Manager"

This chapter describes how to use maintenance commands such as `CHANGE`, `CROSSCHECK`, and `DELETE` to operate on backups and copies.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Recovery Manager Reference*
- *Oracle9i Backup and Recovery Concepts*
- *Oracle9i User-Managed Backup and Recovery Guide*
- *Oracle9i Database Utilities*
- `http://www.oracle.com/database/recovery`

You can access information about the Backup Solutions Program at

`http://otn.oracle.com/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. Refer to *Oracle9i Sample Schemas* for information on how these schemas were created and how you can use them yourself.

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

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

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

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

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

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

`http://otn.oracle.com/admin/account/membership.html`

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

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

To access the database documentation search engine directly, please visit

`http://tahiti.oracle.com`

Conventions

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

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true.` Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
`[ ]`	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`

`|`	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}`
		`[COMPRESS | NOCOMPRESS]`

`...`	Horizontal ellipsis points indicate either:
	■ That we have omitted parts of the code that are not directly related to the example
	■ That you can repeat a portion of the code
		`CREATE TABLE ... AS `*`subquery`*`;`
		
		`SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`

.
.
.
	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	`SQL> SELECT NAME FROM V$DATAFILE;`
		`NAME`
		`------------------------------------`
		`/fsl/dbs/tbs_01.dbf`
		`/fs1/dbs/tbs_02.dbf`
		`.`
		`.`
		`.`
		`/fsl/dbs/tbs_09.dbf`
		`9 rows selected.`

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);`
		`acct    CONSTANT NUMBER(4) := 3;`

Convention	Meaning	Example

`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME = `*`database_name`*

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

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

Conventions for Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32`

Convention	Meaning	Example
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`

Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (’) do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job=’SALESMAN’ and sal<1600\"`
		`C:\>imp SYSTEM/`*`password`*` FROMUSER=scott TABLES=(emp, dept)`

`HOME_NAME`	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_ BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default used one of the following names: ■ `C:\orant` for Windows NT ■ `C:\orawin98` for Windows 98 This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\ora`*nn*, where *nn* is the latest release number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle9i Database Getting Started for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*\ORACLE_ HOME`\rdbms\admin` directory.

Documentation Accessibility

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

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

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

What's New in Recovery Manager?

This section describes new features of Recovery Manager in Oracle9*i* Release 2 (9.2) and provides pointers to additional information. New features information from previous releases is also retained to help those users migrating to the current release from releases prior to Oracle8*i*.

The following sections describe the new features in Recovery Manager:

- Oracle9i New Features in Recovery Manager
- Oracle8i New Features in Recovery Manager

Oracle9*i* New Features in Recovery Manager

This section contains these topics:

- Oracle9i Release 2 (9.2) New Features in Recovery Manager
- Oracle9i Release 1 (9.0.1) New Features in Recovery Manager

Oracle9*i* Release 2 (9.2) New Features in Recovery Manager

The Oracle9*i* Release 2 (9.2) features and enhancements increase manageability and expand functionality.

- **Backup of Server Parameter Files**

  The `BACKUP` command can back up the current server parameter file. RMAN automatically backs up the current server parameter file whenever it includes the current control file in a backup set. If the server parameter file is lost, then you can start an instance without a parameter file, and then run `RESTORE SPFILE` to restore it.

  > **See Also:** *Oracle9i Recovery Manager Reference* for `BACKUP` syntax, and "Configuring the Control File and Server Parameter File Autobackup" on page 8-17 to learn how to enable the control file autobackup feature

- **Control File Autobackups After Structural Changes to the Database**

  If `CONFIGURE CONTROLFILE AUTOBACKUP` is set to `ON` (it is `OFF` by default), then RMAN automatically backs up the control file and server parameter file after structural changes to the database. The target database records in the autobackup in the alert log.

  > **See Also:** "Configuring the Control File and Server Parameter File Autobackup" on page 8-17 to learn how to enable the control file autobackup feature

- **Backing Up Archived Logs That Need Backups**

  You can use `NOT BACKED UP` *`integer`* `TIMES` clause of the `BACKUP ARCHIVELOG` command to back up only those logs that have not been backed up at least *`integer`* times. When calculating the number of backups for a file, RMAN only considers backups created on the same device type as the current backup. This option is a convenient way to back up archived logs on specified media (for example, you want to keep at least three copies of each log on tape).

See Also: *Oracle9i Recovery Manager Reference* for BACKUP syntax, and "Creating a Weekly Secondary Backup of Archived Logs: Example" on page 9-48 for an example

- **Managing Space When Restoring Archived Logs**

  The MAXSIZE option of the RECOVER ... DELETE ARCHIVELOG command limits how much disk space RMAN uses when restoring logs during media recovery.

  See Also: *Oracle9i Recovery Manager Reference* for RECOVER syntax, and "Restoring and Recovering the Whole Database in the Default Location" on page 10-8 for an example

- **Automatic Tags for Backup and Copies**

  If you do not specify a tag for a backup or copy, then RMAN automatically assigns one. The default format includes the date and time for the start of the backup.

  See Also: *Oracle9i Recovery Manager Reference* for a description of the default format for the TAG parameter of the BACKUP command

- **DBNEWID Utility Changes DBID or Database Name**

  You can use the DBNEWID utility to change the database name or DBID of a database. If you copy a database without using the RMAN DUPLICATE command, then the original and copied database share the same DBID. Because RMAN distinguishes databases by DBID, you cannot register multiple databases with the same DBID in the same recovery catalog. By using DBNEWID to change the DBID values for copied databases, you can register them in the same recovery catalog.

  See Also: *Oracle9i Database Utilities* to learn about the DBNEWID utility

- **New V$ Views**

  The V$DATABASE_BLOCK_CORRUPTION view records the corrupt blocks in a file after the most recent backup, backup validation, or copy of the file. The CORRUPTION_TYPE column shows the type of corruption. Run the BLOCKRECOVER command with the CORRUPTION LIST clause to recover all corrupt blocks recorded in the view. After a corrupt block is repaired, the row describing the corruption remains in the view until the next RMAN backup of the affected file. The `V$DATABASE_BLOCK_CORRUPTION` view has a corresponding recovery catalog view called `RC_DATABASE_BLOCK_CORRUPTION`.

The `V$DATABASE_INCARNATION` view lists all incarnations of the database. A new incarnation is created whenever the database is opened with the `RESETLOGS` option. The `V$DATABASE_INCARNATION` view has a corresponding recovery catalog view called `RC_DATABASE_INCARNATION`.

> **See Also:** "Block Media Recovery with RMAN" on page 6-10 for concepts, and *Oracle9i Recovery Manager Reference* for `BLOCKRECOVER` syntax

- **Default Autolocation for Real Application Clusters**

  RMAN automatically discovers which nodes of an Oracle Real Application Clusters configuration can access the files that you want to back up or restore. RMAN autolocates the following files:

  – Backup pieces during backup or restore

  – Archived redo logs during backup

  – Datafile or control file copies during backup or restore

  RMAN enables the autolocation feature whenever the allocated channels have different `PARMS` or `CONNECT` strings.

  Prior to Oracle9*i* Release 2, you had to manually enable this option with `SET AUTOLOCATE`, and the option only applied to backup pieces.

  > **See Also:** "Automatic Location of Backups When Restoring in Real Application Clusters" on page 6-5

- **Diagnostics for Media Manager Function Calls**

  You can now query dynamic performance event views to obtain diagnostic data about RMAN calls to the media manager. An event name corresponds to every media management function. These event names can be used to diagnose problems during RMAN backup, restore, and maintenance jobs.

  > **See Also:** "Monitoring RMAN Interaction with the Media Manager" on page 15-22

- **FORCE Option for DELETE Command**

  Sometimes the status of an object in the RMAN repository does not reflect the status of the object on the media. For example, someone deletes a backup piece with an operating system utility before `CROSSCHECK` is run. If the object is listed as `AVAILABLE` or `EXPIRED` but the reality on the media is otherwise, and if you run the `DELETE` command on the object, then RMAN does not delete it. You can override this behavior with the `FORCE` option of the `DELETE` command.

  > **See Also:** "Behavior of DELETE Command When the Repository and Media Do Not Correspond" on page 7-14

- **Deletion of Files Already Backed Up to a Device**

  You can delete files that have already been backed up a specified number of times to a device. For example, you can delete all archived redo logs that have been backed up at least twice to tape.

  > **See Also:** "Deletion of Archived Redo Logs That Are Already Backed Up" on page 7-13

- **DUPLICATE Enhancements**

  The `SKIP TABLESPACE` option of the `DUPLICATE` command enables you to exclude a list of tablespaces from the duplicate database. Also, you can specify the `UNTIL` clause on the `DUPLICATE` command to recover the duplicate database to a noncurrent time.

  > **See Also:** "Database Duplication Options" on page 12-4

- **RMAN Error Output Improved**

  The RMAN error output is more compact and informative. If an RMAN command fails, then the error stack is always followed by `RMAN-03002` or `RMAN-03009` stating which command failed. If the errors are generated from the target database, then RMAN does not explicitly indicate that they are from the target database; however, if the errors are from the catalog or auxiliary database, then RMAN indicates this fact in a separate message.

Oracle9*i* Release 1 (9.0.1) New Features in Recovery Manager

The Oracle9*i* Release 1 (9.0.1) features and enhancements increase manageability and greatly expand functionality.

- **Persistent RMAN Configurations**

  The `CONFIGURE` command creates persistent RMAN settings that apply to any session. You can configure a variety of features including automatic channels, channel parallelism, retention policies, backup options, auxiliary filenames, and so forth.

  > **See Also:** Chapter 8, "Configuring the Recovery Manager Environment" for configuration procedures, and *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax

- **Automatic Channel Configuration**

  The `CONFIGURE` command stores a persistent set of channel settings. RMAN automatically allocates these channels when you execute commands such as `BACKUP` or `COPY` at the RMAN prompt, abolishing the necessity for manual channel allocation.

  Use the `CONFIGURE` command to define backup attributes such as parallelism, duplexing, and channel control options. These settings are used by all commands that require channels unless you manually allocate channels in a `RUN` command.

  > **See Also:** "Automatic and Manual Channel Allocation" on page 5-4 for concepts, and *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax.

- **More Commands Available at the RMAN Prompt**

  Formerly, most backup and recovery commands had to be executed within a `RUN` command. Now, many commands such as `BACKUP, COPY,` and `RESTORE` can be executed at the RMAN prompt.

- **Retention Policies**

  A retention policy is a configuration that specifies how long backups of control files and datafiles should be retained, as well how many of these backups should be retained. The retention policy also determines which archived logs are obsolete because they are not needed to recover these backups.

  The `REPORT OBSOLETE` and `DELETE OBSOLETE` commands use the criteria specified in the retention policy to determine what is obsolete. You can specify a retention policy in these ways:

- Using a recovery window, which defines a window of time extending backward from the present. The database must be recoverable to any point of time in that window.
- Using redundancy, which specifies that any number of backups or copies beyond a specified number need not be retained.

See Also: "Backup Retention Policies" on page 5-49 for concepts, and *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax.

- **Long-Term Backups Exempt from Retention Policy**

  The `KEEP` option of the `BACKUP` command marks a backup as exempt from the retention policy. The `REPORT OBSOLETE` and `DELETE OBSOLETE` commands only consider the backup as obsolete when the `KEEP` time expires. You can alter the status of the backup with the `CHANGE ... KEEP` command.

  See Also: "Backups Exempt from the Retention Policy" on page 5-55 for concepts, and *Oracle9i Recovery Manager Reference* for `BACKUP` syntax and *Oracle9i Recovery Manager Reference* for `CHANGE` syntax

- **Automatic Control File Backup and Restore**

  If you set `CONFIGURE CONTROLFILE BACKUP` to `ON`, then RMAN makes a mandatory and automatic control file autobackup after you run the `BACKUP` or `COPY` command. RMAN gives this backup a default name (which you can change using `CONFIGURE CONTROLFILE BACKUP FORMAT`), thereby allowing RMAN to restore this control file even in a disaster scenario in which the recovery catalog and current control file are lost.

  See Also: "Control File and Server Parameter File Autobackups" on page 5-47 for concepts

- **Block Media Recovery**

  Block media recovery can perform media recovery on individual blocks in a datafile while the datafile remains online. The `V$DATABASE_BLOCK_CORRUPTION` view displays corrupt blocks in the most recent backup or image copy of each datafile.

See Also: "Block Media Recovery with RMAN" on page 6-10 for concepts, and *Oracle9i Recovery Manager Reference* for `BLOCKRECOVER` syntax

- **Archived Redo Log Improvements**

  RMAN enhances its treatment of archived redo logs in the following ways:

  - The `BACKUP ... PLUS ARCHIVELOG` command switches out of and archives the current online log, backs up archived logs, then performs another log switch and backs up remaining logs. This operation guarantees that the backed up datafiles can be recovered to a consistent state.
  - At the beginning of a backup of archived logs, RMAN automatically switches out of and archives the current online redo log if needed.
  - RMAN does not issue an error if a `BACKUP ARCHIVELOG` command does not discover any logs to back up.
  - RMAN performs archived log failover automatically. If RMAN discovers a corrupt or missing log during a backup, then it considers all logs listed in the repository as alternative candidates for the backup.
  - The `DELETE ALL INPUT` option of the `BACKUP ARCHIVELOG` command deletes all logs that match the specified criteria. In this way, you can delete logs located in multiple archiving destinations.
  - The `BACKUP_COUNT` column in `V$ARCHIVED_LOG` shows the number of times that a specified archived log was backed up.

  See Also: "Backups of Archived Logs" on page 5-15 for concepts, and *Oracle9i Recovery Manager Reference* for `BACKUP` syntax

- **LIST Enhancements**

  The `LIST` command syntax is now more similar in structure to the `CHANGE`, `CROSSCHECK`, and `DELETE` commands. You also have more control over how the `LIST` output is displayed.

  See Also: "LIST Command Output" on page 7-2 for concepts, and *Oracle9i Recovery Manager Reference* for `LIST` syntax

- **CROSSCHECK Enhancements**

  The functionality of the `CROSSCHECK` command is expanded, and the `CHANGE ... CROSSCHECK` is deprecated. The `CROSSCHECK` command is now similar in syntax to the `DELETE` and `CHANGE` commands.

  > **See Also:** "Crosschecks of RMAN Backups and Copies" on page 7-8 for concepts, and *Oracle9i Recovery Manager Reference* for `CROSSCHECK` syntax

- **CHANGE Enhancements**

  The `CHANGE` command can make files available or unavailable, catalog or uncatalog them, and change the `KEEP` setting, regardless of whether the backup repository is the control file or recovery catalog (except for `CHANGE ... KEEP FOREVER`, which requires a catalog). The `CHANGE` command operates on more types of files, and is now similar in syntax to the `DELETE` and `CROSSCHECK` commands.

  > **See Also:** "Changes to Availability of RMAN Backups and Copies" on page 7-16 for concepts, and *Oracle9i Recovery Manager Reference* for `CHANGE` syntax

- **DELETE Enhancements**

  The `DELETE` command combines the functionality of the previous `CHANGE ... DELETE` and `DELETE EXPIRED` commands. Additionally, you can run `DELETE OBSOLETE` to remove all files that are no longer needed. The `DELETE` command is now similar in syntax to the `CHANGE` and `CROSSCHECK` commands.

  Note that by default, `DELETE EXPIRED` and `DELETE OBSOLETE` prompt for confirmation before deleting files. In releases prior to Oracle9*i*, RMAN did not prompt when you ran `DELETE EXPIRED`.

  > **See Also:** "Deletion of RMAN Backups and Copies" on page 7-10 for concepts, and *Oracle9i Recovery Manager Reference* for `DELETE` syntax

- **Backup of Backup Sets**

  You can now back up backup sets from disk to tape or from disk to disk. If RMAN discovers a corrupt block or missing backup piece during the backup, then RMAN automatically performs failover to an existing intact copy.

See Also: "Backups of Backup Sets" on page 5-24 for concepts, and *Oracle9i Recovery Manager Reference* for BACKUP syntax

- **Enhanced Duplexing Functionality**

  You can specify up to four different FORMAT strings when generating a backup set, so each duplexed copy can have a different filename.

  You can enable duplexing by specifying the COPIES parameter on the following commands: CONFIGURE, SET, and BACKUP.

  See Also:

  - "Duplexed Backup Sets" on page 5-55 for concepts
  - *Oracle9i Recovery Manager Reference* for BACKUP syntax
  - *Oracle9i Recovery Manager Reference* for CONFIGURE syntax
  - *Oracle9i Recovery Manager Reference* for SET syntax

- **Backup Optimization**

  You can configure RMAN to skip backups of files that have already been backed up. In this way, you can avoid making multiple backups of unchanging files such as archived logs. You can override this functionality with the FORCE option of the BACKUP command.

  See Also: "Backup Optimization" on page 5-56 for concepts, and *Oracle9i Recovery Manager Reference* for CONFIGURE syntax

- **Restartable Backups**

  You can specify the NOT BACKED UP SINCE clause on the BACKUP command to back up only those files that were not backed up after a specified time. Hence, if a backup fails partway through, you can restart it and back up only those files that were not previously backed up.

  See Also: "Restartable Backups" on page 5-61 for concepts, and *Oracle9i Recovery Manager Reference* for BACKUP syntax

- **Restore Optimization**

  When RMAN is restoring files, it checks the files on disk to determine whether a restore is necessary. If not, it does not restore the file. Hence, if a restore fails partway through, you can restart it and restore only those files that were not previously restored. You can override this behavior with the FORCE option.

> **See Also:** "Restore Optimization" on page 6-4 for concepts, and *Oracle9i Recovery Manager Reference* for RESTORE syntax

- **Multiple Block Size Support**

  Oracle allows different tablespaces in a database to use different block sizes. RMAN backs up tablespaces with different block sizes in the same BACKUP command, but it never mixes datafiles with different block sizes in a single backup set. Each backup set contains only files with a single block size.

- **Exclusion of Tablespaces From Backups**

  You can configure RMAN so that a specified tablespace is excluded from whole database backups.

  > **See Also:** "Configuring Tablespaces for Exclusion from Whole Database Backups" on page 5-61 for concepts, and *Oracle9i Recovery Manager Reference* for CONFIGURE syntax

- **NOCATALOG Mode Is Default**

  If you do not specify either CATALOG or NOCATALOG on the command line, and if you do not run CONNECT CATALOG at the prompt, then RMAN defaults to NOCATALOG mode when you run a command requiring a repository.

  > **See Also:** "Starting RMAN Without Connecting to a Database" on page 2-4 for concepts

- **RMAN Message Output Improvements**

  The RMAN prefix is removed from non-error messages.

- **SBT Library Architecture**

  On platforms that support third-party media managers through the Oracle Media Management (SBT) API, the new SBT_LIBRARY parameter controls which media management library that RMAN uses on channels of DEVICE TYPE sbt. Use SBT_LIBRARY in the PARMS setting of the ALLOCATE CHANNEL or CONFIGURE CHANNEL command to specify the filename of the shared library to be loaded. You can also specify SBT_LIBRARY=oracle.disksbt, which causes the server to load Oracle's disk sbt library (formerly called the "dummy API").

If no `SBT_LIBRARY` parameter is specified in `PARMS`, then the Oracle server attempts to load the dynamic libraries `orasbt.dll` on Windows NT and `libobk.so` on UNIX. If Oracle cannot locate the library, then the server returns an error.

> **See Also:** "Configuring RMAN to Make Backups to a Media Manager" on page 8-2 to learn about linking to a media manager, and *Oracle9i Recovery Manager Reference* for `PARMS` syntax

Oracle8*i* New Features in Recovery Manager

The Oracle8*i* features and enhancements described in this section profoundly expanded the 8.0 RMAN functionality. Some of these features were not in all Oracle8*i* releases.

- **RMAN Support for Standby Database**

  In release 8.1.7, you can use RMAN to create and back up a standby database.

  > **See Also:** Chapter 13, "Creating a Standby Database with Recovery Manager"

- **Making Test Backups**

  In Oracle8*i* release 3 (8.1.7), the `BACKUP VALIDATE` command tests a backup without actually producing output files.

  > **See Also:** "Test Backups Using RMAN" on page 5-68 for concepts, and *Oracle9i Recovery Manager Reference* for syntax

- **Setting the Default Location of the RMAN Snapshot Control File**

  In Oracle8*i* release 3 (8.1.7), you can use the `SET SNAPSHOT CONTROLFILE LOCATION TO DEFAULT` command to set the default filename that is used for the snapshot control file. This default value is platform-specific and depends on the location of the Oracle home.

- **Crosschecking and Deleting on Multiple Channels**

  In Oracle8*i* release 3 (8.1.7), you can allocate multiple maintenance channels before issuing RMAN maintenance commands. RMAN checks on all channels that have the same device type as the channel used to create the backup.

See Also: "Allocation of Multiple Channels for RMAN Maintenance Commands" on page 7-15

- **Fewer Maintenance Commands Require a Catalog**

  In Oracle8*i* release 3 (8.1.7), fewer maintenance commands require the use of a recovery catalog. The only options of the `CHANGE` command that require a recovery catalog are the following:

  - `CHANGE ... AVAILABLE` (note that backup sets, backup pieces, and proxy copies do not require the use of a recovery catalog)
  - `CHANGE ... UNAVAILABLE`

  The `CROSSCHECK` and `DELETE EXPIRED` commands no longer require a recovery catalog in release 8.1.7.

- **Setting Recovery Catalog Compatibility Level No Longer Required**

  In Oracle8*i* release 2 (8.1.6), the `CONFIGURE COMPATIBLE` command set the compatibility level in the recovery catalog. This command helped to solve problems resulting from the way in which RMAN updated and deleted catalog records. These problems are solved in Oracle8*i* release 3 (8.1.7), so the `CONFIGURE COMPATIBLE` command is still accepted for compatibility, but has no effect.

- **Automatic Location of Backups on Oracle Real Application Cluster Nodes**

  In Oracle8*i* release 2 (8.1.6), the `AUTOLOCATE` option of the `SET` command automatically discovers which nodes of an Oracle Real Application Clusters configuration contain the backups that you want to restore.

  See Also: *Oracle9i Recovery Manager Reference* for `SET` syntax

- **Recovery Catalog Compatibility Setting**

  In Oracle8*i* release 2 (8.1.6), the `CONFIGURE COMPATIBLE` command controls the compatibility of the recovery catalog packages with the RMAN executable.

- **ALTER DATABASE RESETLOGS Command**

  In Oracle8*i* release 2 (8.1.6), the `RESETLOGS` option of the `ALTER DATABASE` command opens the database and reset the online redo logs.

  See Also: *Oracle9i Recovery Manager Reference* for `ALTER DATABASE` syntax

- **Deletion Commands Remove Repository Records**

  In Oracle8*i* release 2 (8.1.6), the `CHANGE ... DELETE`, `DELETE EXPIRED`, and `BACKUP ... DELETE INPUT` commands can now remove catalog records rather than update them to status `DELETED`.

- **Version 2.0 of the Media Management API**

  Oracle releases the Media Management API, Version 2.0. Support for the version 1.1 Media Management API is maintained. The following are features of the new API:

  – Through an enhancement called **proxy copy**, the media management vendor software is able to take over data movement involved in a backup or restore.

  – Some media management software allows backup media to be arranged into storage pools based on media type, retention period, or other criteria. The `POOL` parameter of the `BACKUP` command provides integration between such products and RMAN.

  – When a channel is allocated, RMAN displays in its log a message identifying the media management product used to take backups on that channel. When the media manager encounters an error, it returns an error message explaining the error, which is displayed in the RMAN log.

  – The `SEND` command allows commands to be sent directly from an RMAN session to the media management software.

  See Also: *Oracle9i Recovery Manager Reference* for `BACKUP` syntax and see *Oracle9i Recovery Manager Reference* for `SEND` syntax

- **Crosscheck and Delete Functionality**

  The commands `CROSSCHECK BACKUP`, `CHANGE ... CROSSCHECK`, and `DELETE EXPIRED BACKUP` allow for the synchronization of the recovery catalog with the media manager's catalog. RMAN can determine whether backups and copies are on disk or tape and update their repository record if they are not.

  See Also:

  - *Oracle9i Recovery Manager Reference* for `CHANGE` syntax
  - *Oracle9i Recovery Manager Reference* for `CROSSCHECK` syntax
  - *Oracle9i Recovery Manager Reference* for `DELETE` syntax

- **Improved LIST Output**

  The output of the LIST BACKUP command now prints the list of backups belonging to a backup set in a separate section of the report from the list of data files or archived logs included in the backup set.

  > **See Also:** *Oracle9i Recovery Manager Reference* for LIST syntax

- **Reports on Needed Backups**

  A new command, REPORT NEED BACKUP REDUNDANCY, alerts the user that a new backup is required when fewer than a user-specified number of backups of a datafile exist.

  > **See Also:** *Oracle9i Recovery Manager Reference* for REPORT syntax

- **CREATE CATALOG Command**

  The CREATE CATALOG command creates the recovery catalog. It replaces `catrman.sql` and associated scripts in the `?/dbs/admin` directory.

  > **See Also:** *Oracle9i Recovery Manager Reference* for CREATE CATALOG syntax

- **UPGRADE CATALOG Command**

  Previously it was necessary to run a SQL script to perform a recovery catalog upgrade. Now you can upgrade the catalog with the UPGRADE CATALOG command.

  > **See Also:** *Oracle9i Recovery Manager Reference* for UPGRADE CATALOG syntax

- **DROP CATALOG Command**

  The DROP CATALOG command removes the recovery catalog schema.

  > **See Also:** *Oracle9i Recovery Manager Reference* for DROP CATALOG syntax

- **STARTUP, SHUTDOWN, and ALTER DATABASE Commands**

  The RMAN commands `STARTUP`, `SHUTDOWN`, and `ALTER DATABASE` (`MOUNT` and `OPEN` options only) have the same syntax as their equivalent SQL*Plus commands.

  See Also:

  - *Oracle9i Recovery Manager Reference* for `STARTUP` syntax
  - *Oracle9i Recovery Manager Reference* for `SHUTDOWN` syntax
  - *Oracle9i Recovery Manager Reference* for `ALTER DATABASE` syntax

- **Database Duplication**

  The `DUPLICATE` command allows creation of a new database using the backups of another database.

  See Also: Chapter 12, "Duplicating a Database with Recovery Manager"

- **Disk Affinity in Oracle Real Application Clusters**

  When backing up on multiple nodes of an Oracle Real Application Clusters configuration, it is possible that some disks have affinity to certain nodes in the cluster such that access to those disks is faster from those nodes than from other nodes in the cluster. RMAN recognizes node affinity, if it exists, and attempts to schedule datafile backups on channels allocated at nodes that have affinity to those files.

- **Duplexed Backups**

  RMAN can create up to four concurrent copies of each backup piece.

  See Also: "Duplexed Backup Sets" on page 5-21

- **RMAN-Generated Backup Piece Names**

  RMAN no longer requires that backup piece names be explicitly specified using the `FORMAT` parameter. By default, RMAN chooses a unique name for each backup piece.

- **Backup Pieces Not Overwritten**

  A backup piece is no longer overwritten if an attempt is made to create a backup piece with the same name as an existing one. Instead, RMAN issues an error message.

- **TSPITR Does Not Require a Catalog**

  You can perform TSPITR (Tablespace Point -in-Time Recovery) without a recovery catalog.

  > **See Also:** "Performing TSPITR Without a Recovery Catalog" on page 11-5

- **New V$ Views for Monitoring Backup Performance**

  Two new views are available to monitor the progress and performance of Recovery Manager backups: `V$BACKUP_SYNC_IO` and `V$BACKUP_ASYNC_IO`.

  > **See Also:** "Step 8: Query V$ Views to Identify Bottlenecks" on page 14-16

Part I
Getting Started with Recovery Manager

Part I describes gives a general introduction to Recovery Manager and explains how to use RMAN to connect to target databases. This part contains these chapters:

1

Introduction to Recovery Manager

This chapter describes the definition and advantages of the Recovery Manager (RMAN) utility.

This chapter contains these sections:

- About Recovery Manager (RMAN)
- Why Use RMAN?
- Overview of the RMAN Environment

About Recovery Manager (RMAN)

Recovery Manager (RMAN) is an Oracle utility that can back up, restore, and recover database files. The product is a feature of the Oracle database server and does not require separate installation.

Recovery Manager is a client/server application that uses database server sessions to perform backup and recovery. It stores metadata about its operations in the control file of the target database and, optionally, in a recovery catalog schema in an Oracle database.

You can invoke RMAN as a command-line executable from the operating system prompt or use some RMAN features through the Enterprise Manager GUI.

Why Use RMAN?

Most production database systems impose stringent requirements on backup and recovery. As a DBA in charge of backup and recovery, you must:

- Manage the complexity of backup and recovery operations
- Minimize the possibility of human error
- Make backups scalable and reliable
- Utilize all available media hardware
- Make backups proportional to the size of transactional changes, not to the size of database
- Make recovery time proportional to the amount of data recovered

You have two basic methods for performing these backup and recovery tasks on an Oracle release 8.0 or higher database:

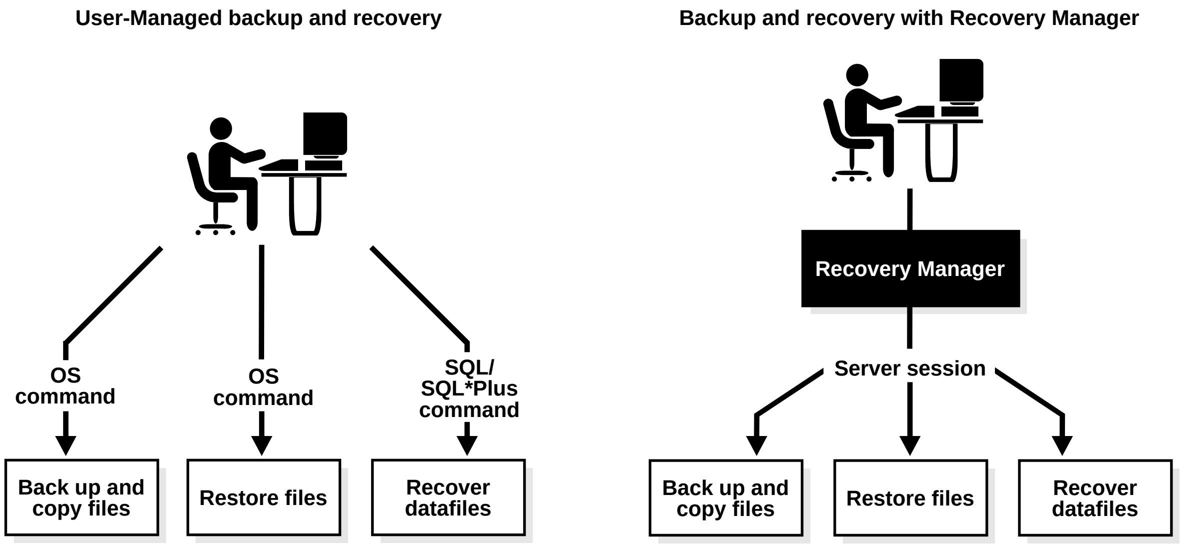
- Using operating system commands to perform backup and restore operations, and SQL or SQL*Plus statements to perform recovery
- Using Recovery Manager for backup, restore, and recovery

Note: RMAN was introduced in Oracle release 8.0 and is not compatible with Oracle databases prior to release 8.0.

Why use one method rather than the other? As illustrated in Figure 1–1, RMAN uses server sessions to perform backup and recovery operations and stores metadata in a repository. RMAN automates backup and recovery, whereas the user-managed method requires *you* to keep track of all database files and backups. For example, instead of requiring you to locate backups for each datafile, copy them to the correct place using operating system commands, and choose which logs to apply, RMAN manages these tasks automatically.

The advantage of using RMAN is especially true if you use Oracle Managed Files. When you let Oracle name and manage your datafiles, control files, and online redo logs, the system becomes easier to use. On the other hand, it may be harder for you to keep track of the filenames of the various database files because you have not named them yourself. RMAN users do not suffer from this problem because RMAN handles all record keeping.

Figure 1–1 Comparison of RMAN Automated and User-Managed Procedures



Besides the obvious advantage of automation, RMAN provides a host of other useful features. Table 1–1 compares some of the differences between the RMAN methodology and the traditional user-managed methodology.

Table 1–1 Comparison Between RMAN and User-Managed Methods **(Page 1 of 2)**

Recovery Manager	User-Managed Method
Uses a media management API so that RMAN works seamlessly with third-party media management software. More than 20 vendors support the API.	Does not have support of a published API.
When backing up online files, RMAN rereads fractured data blocks to get a consistent read. You do not need to place online tablespaces in backup mode when performing backups.	Requires placing online tablespaces in backup mode before backing them up, and then taking the tablespaces out of this mode after the backup is complete. Serious database performance and manageability problems can occur if you neglect to take tablespaces out of backup mode after an online backup is complete.
Performs incremental backups, which back up only those data blocks that changed after a previous backup. You can recover the database using incremental backups, which means that you can recover a `NOARCHIVELOG` database. However, you can only take incremental backups of a `NOARCHIVELOG` database after a consistent shutdown.	Backs up all blocks, not just the changed blocks. Does not allow you to recover a `NOARCHIVELOG` database.
Computes checksums for each block during a backup, and checks for corrupt blocks when backing up or restoring. Many of the integrity checks that are normally performed when executing SQL are also performed when backing up or restoring.	Does not provide error checking.
Omits never-used blocks from datafile backups so that only data blocks that have been written to are included in a backup.	Includes all data blocks, regardless of whether they contain data.
Uses the repository to report on crucial information, including: ■ Database schema at a specified time ■ Which files need a backup ■ Which files have not had a backup in a specified number of days ■ Which backups can be deleted because they are redundant or cannot be used for recovery ■ Current RMAN persistent settings	Does not include any reporting functionality.
Stores RMAN scripts in the recovery catalog.	Requires storage and maintenance of operating system-based scripts.

Table 1–1 Comparison Between RMAN and User-Managed Methods (Page 2 of 2)

Recovery Manager	User-Managed Method
Allows you to easily create a duplicate of the production database for testing purposes, or easily create or back up a standby database.	Requires you to follow a complicated procedure when creating a test or standby database.
Performs checks to determine whether backups on disk or in the media catalog are still available.	Requires you to locate and test backups manually.
Performs automatic parallelization of backup and restore operations.	Requires you to parallelize manually by determining which files you need to back up and then issuing operating system commands in parallel.
Tests whether files can be backed up or restored without actually performing the backup or restore.	Requires you to actually restore backup files before you can perform a trial recovery of the backups.
Performs archived log failover automatically. If RMAN discovers a corrupt or missing log during a backup, then it considers all logs and log copies listed in the repository as alternative candidates for the backup.	Cannot failover to an alternative archived log if the backup encounters a problem.

Overview of the RMAN Environment

The RMAN environment consists of the utilities and databases that play a role in a backup and recovery strategy. A typical RMAN setup utilizes the following:

- RMAN executable
- Target database
- Recovery catalog database
- Media management software

Of these components, only the RMAN executable and target database are required. RMAN automatically stores its metadata in the target database control file, so the recovery catalog database is optional. Nevertheless, maintaining a recovery catalog is strongly encouraged. If you create a catalog on a separate machine, and if the production machine fails completely, then you have all the restore and recovery information you need in the catalog.

This section contains these topics:

- About the RMAN Executable
- About the Target Database

- About the RMAN Repository
- About the RMAN Media Management Interface

About the RMAN Executable

The RMAN executable is automatically included with the Oracle software installation. Its location is platform-specific and is typically located in the same place as the other Oracle executables. On Unix systems, for example, the RMAN executable is located in `$ORACLE_HOME/bin`.

To start the executable, simply enter the filename on the command line. For example, on a UNIX system, enter:

```
% rman
```

About the Target Database

The target database is the database that RMAN is backing up, restoring, or recovering. You can use a single recovery catalog in conjunction with multiple target databases. For example, assume that your data center contains 10 databases of varying sizes. You can use a single recovery catalog located in a different data center to manage the metadata from all of these databases.

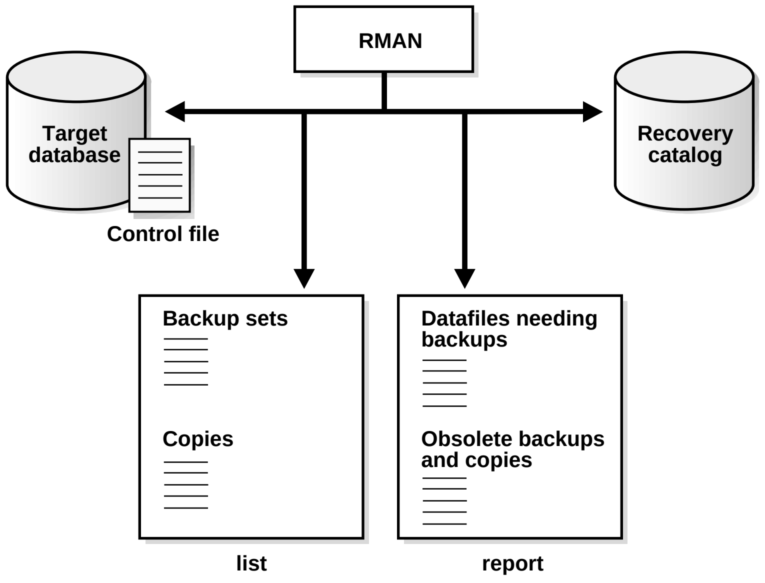
About the RMAN Repository

The RMAN repository is a set of metadata that RMAN uses to store information about the target database and its backup and recovery operations. Among other things, RMAN stores information about:

- Backup sets and pieces
- Image copies (including archived redo logs)
- Proxy copies
- The target database schema
- Persistent configuration settings

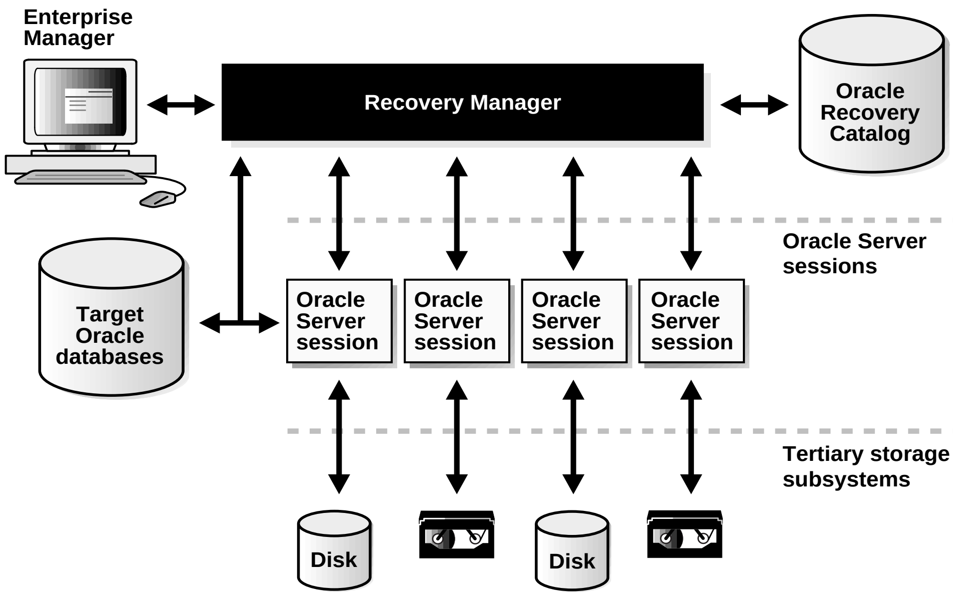
You can access this metadata by issuing `LIST`, `REPORT`, and `SHOW` commands in the RMAN interface, or by using `SELECT` statements on the catalog views (only if you use a recovery catalog). Figure 1–2 illustrates how RMAN issues lists and reports.

Figure 1–2 RMAN Lists and Reports



You can either create a **recovery catalog** in which to store the repository, or let RMAN store the repository exclusively in the target database control file. Figure 1–3 depicts RMAN using a recovery catalog.

Figure 1–3 RMAN with Optional Recovery Catalog



Although RMAN can conduct all major backup and recovery operations using just the control file, note these advantages of using the catalog:

- Some RMAN commands and operations function only with a catalog.
- The recovery catalog retains historical backup information that can get overwritten in the control file.
- The recovery catalog stores information about backups from different incarnations of the database.

The recovery catalog is maintained solely by RMAN; the target database never accesses it directly. RMAN automatically propagates information about the database structure, archived redo logs, backup sets, and datafile copies into the recovery catalog from the target database's control file. You can also propagate this information to the catalog manually using the `RESYNC CATALOG` command.

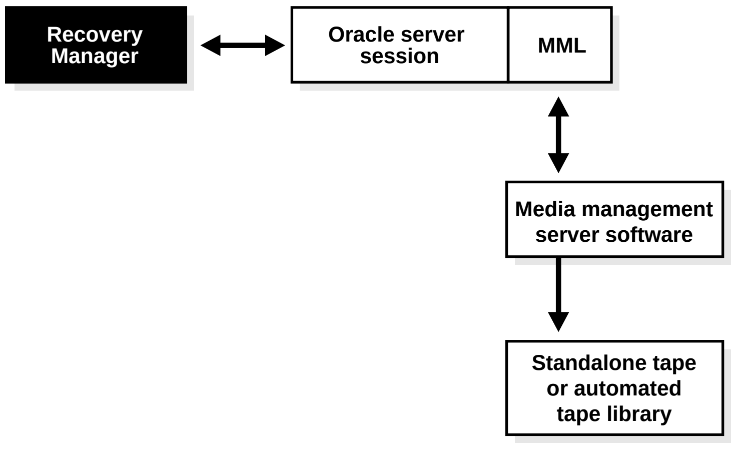
See Also:

- Chapter 16, "Managing the Recovery Manager Repository"
- "Deciding Whether to Use RMAN with a Recovery Catalog" on page 3-13
- "Understanding Catalog-Only Command Restrictions" on page 16-31

About the RMAN Media Management Interface

To store backups on tape, RMAN requires a **media manager**. A media manager is a software program that loads, labels, and unloads sequential media such as tape drives used to back up and recover data. Figure 1–4 shows the architecture for a media manager integrated with Oracle.

Figure 1–4 Architecture for MML Integrated with Oracle


Recovery Manager
Oracle server session
MML
Media management server software
Standalone tape or automated tape library

The Oracle server session is the same type of server session used when a client such as SQL*Plus connects to the database. The media management library (MML) in Figure 1–4 represents vendor-supplied media management software library that can interface with Oracle. Oracle calls MML software routines to back up and restore datafiles to and from media controlled by the media manager.

See Also: "How Oracle Interacts with the Media Manager" on page 8-3

2 Connecting to Databases with RMAN

This chapter describes how to start and stop the Recovery Manager (RMAN) command-line interface and make database connections. This chapter contains these topics:

Starting RMAN: Overview

You have the following basic options for starting RMAN:

- Start the RMAN executable at the operating system command line while connecting to one or more databases, as in these examples:

```
% rman TARGET / CATALOG rman/cat@catdb
% rman TARGET SYS/oracle@trgt NOCATALOG
% rman TARGET / CATALOG rman/cat@catdb AUXILIARY SYS/oracle@auxdb
```

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

```
% rman
```

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

If you start RMAN without connecting to the target database, then you must issue `CONNECT TARGET` command at the RMAN prompt before you can begin performing backup and recovery operations.

Types of Database Connections

You can connect to the following types of databases.

Database	**Connection**
Target database	RMAN connects you to the target database with the `SYSDBA` privilege. If you do not have this privilege, then the connection fails.
Recovery catalog database	This database is optional: you can also use RMAN with the default `NOCATALOG` option.
Auxiliary database	You can connect to a standby database, duplicate database, or auxiliary instance (standby instance or tablespace point-in-time recovery instance).

Authentication for Database Connections

When connecting to a target or auxiliary database, you must have the `SYSDBA` privilege. You can connect as `SYSDBA` using a password file or using operating system authentication.

Note: You do not need to specify the SYSDBA option because RMAN uses this option implicitly and automatically. You must have the SYSDBA privilege to connect to the target database.

If the target database uses password files, then you can connect using a password. Use a password file for either local or remote access. You must use a password file if you are connecting remotely as SYSDBA using a net service name.

If you connect to the database using operating system authentication, remember to set the environment variable specifying the Oracle SID. For example, to set the SID to `trgt` at the UNIX command line enter:

```
% ORACLE_SID=trgt; export ORACLE_SID
```

Note that a SYSDBA privilege is not required when connecting to the recovery catalog. The only requirement is that the RECOVERY_CATALOG_OWNER role be granted to the schema owner.

See Also: The first chapter of the *Oracle9i Database Administrator's Guide* to learn how to authenticate users on a database, and to create a password file

Command-Line Options When Starting RMAN

RMAN provides a number of command-line options that you can specify when starting RMAN. For example, you can start RMAN:

- In batch mode by specifying a command file that contains a series of RMAN commands
- In interactive mode
- With a command pipe
- With a log file that redirects RMAN output

See Also:

- *Oracle9i Recovery Manager Reference* for RMAN command line options
- *Oracle9i Recovery Manager Reference* for CONNECT syntax
- *Oracle9i Database Administrator's Guide* to learn about password files

Starting RMAN Without Connecting to a Database

You can start Recovery Manager at the operating system command line without connecting to a database by issuing the RMAN command without any arguments. For example, enter:

```
% rman
```

If you did not specify the LOG option at the command line, then RMAN displays the RMAN prompt:

```
RMAN>
```

After the RMAN prompt is displayed, you can issue further commands to connect to the target database, recovery catalog database, or auxiliary database.

If you start RMAN without specifying either CATALOG or NOCATALOG on the command line, then RMAN makes no repository connection. The first time a command is issued that requires the repository, if no CONNECT CATALOG command has been issued yet, then RMAN automatically connects in the default NOCATALOG mode. After that point, the CONNECT CATALOG command is not valid in the session.

Connecting to the Target Database Without a Recovery Catalog

In these examples, assume that these variables have the following meanings.

Variable	Meaning
SYS	User with SYSDBA privileges
oracle	The password for connecting as SYSDBA specified in the target database's orapwd file
trgt	The net service name for the target database

Connecting to the Target Database Without a Catalog from the Command Line

To connect from the operating system command line, enter the connection as in the following examples:

```
# example of operating system authentication
% rman TARGET / NOCATALOG

# example of Oracle Net authentication
% rman TARGET SYS/oracle@trgt NOCATALOG
```

Note that you can also start RMAN without specifying either NOCATALOG or CATALOG as follows:

```
# example of operating system authentication
% rman TARGET /

# example of Oracle Net authentication
% rman TARGET SYS/oracle@trgt
```

If you do not specify the NOCATALOG keyword on the command line, and if you also do not specify CONNECT CATALOG after RMAN has started, then RMAN connects in NOCATALOG mode by default the first time that you run a command that requires a repository. For example:

```
% rman TARGET /
RMAN> BACKUP DATABASE;    # RMAN defaults to NOCATALOG mode
```

Connecting to the Target Database Without a Catalog from the RMAN Prompt

Alternatively, start RMAN and connect to the target database from the RMAN prompt, as in this example:

```
% rman NOCATALOG
RMAN> CONNECT TARGET
```

This example connects to the target database by using a password file:

```
% rman NOCATALOG
RMAN> CONNECT TARGET SYS/oracle@trgt
```

Connecting to the Target Database and Recovery Catalog

In these examples, assume that you maintain a recovery catalog and that these variables have the following meanings.

Variable	Meaning
SYS	User with SYSDBA privileges
oracle	The password for connecting as SYSDBA specified in the target database's orapwd file
trgt	The net service name for the target database
rman	Owner of the recovery catalog having RECOVERY_CATALOG_OWNER privilege

Variable	Meaning
`cat`	The password for user RMAN specified in the recovery catalog's `orapwd` file
`catdb`	The net service name for the recovery catalog database

Connecting to the Target Database and Recovery Catalog from the Command Line

To connect to the target and recovery catalog databases from the operating system command line, enter the connection as in the following examples:

```
# operating system authentication
% rman TARGET / CATALOG rman/cat@catdb

# Oracle Net authentication
% rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
```

Connecting to the Target Database and Recovery Catalog from the RMAN Prompt

Alternatively, start RMAN and connect to the target database from the RMAN prompt. This example uses operating system authentication:

```
% rman
RMAN> CONNECT TARGET
RMAN> CONNECT CATALOG rman/cat@catdb
```

This example uses Oracle Net authentication:

```
% rman
RMAN> CONNECT TARGET SYS/oracle@trgt
RMAN> CONNECT CATALOG rman/cat@catdb
```

Note: If you run `CONNECT TARGET` and then do not run `CONNECT CATALOG`, RMAN connects in `NOCATALOG` mode by default the first time that you run a command that requires a repository (for example, `BACKUP`). Thereafter, you cannot run `CONNECT CATALOG` in the RMAN session.

Connecting to a Target Database in an Oracle Real Application Cluster

RMAN can only connect to one instance in an Oracle Real Application Clusters database at a time. Assume that `trgt1`, `trgt2`, and `trgt3` are net service names for three instances in an Oracle Real Application Clusters configuration. In this case, you can connect to the target database using only one of these net service names. For example, you can connect as follows:

```
% rman TARGET SYS/oracle@trgt2 CATALOG rman/cat@catdb
```

Each net service name must specify one and only one instance. You cannot specify a net service name that uses Oracle Net features to distribute connections to more than one instance.

Note that the fact that RMAN connects to only one instance for its initial target connection does not preclude running a backup using all three instances. For example, you can configure automatic channels to connect to each cluster instance as follows:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT = 'SYS/oracle@trgt1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT = 'SYS/oracle@trgt2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT = 'SYS/oracle@trgt3';
```

Then, make a whole database backup by running the following command:

```
BACKUP DATABASE;
```

Connecting to an Auxiliary Database

To use the `DUPLICATE` command or to perform RMAN TSPITR, you need to connect to an auxiliary instance. In these examples, assume that these variables have the following meanings.

Variable	**Meaning**
`SYS`	User with `SYSDBA` privileges
`oracle`	The password for connecting as `SYSDBA` specified in the target database's `orapwd` file
`trgt`	The net service name for the target database
`rman`	Owner of the recovery catalog having `RECOVERY_CATALOG_OWNER` privilege
`cat`	The password for user RMAN specified in the recovery catalog's `orapwd` file
`catdb`	The net service name for the recovery catalog database

Variable	Meaning
`aux`	The password for connecting as `SYSDBA` specified in the auxiliary database's `orapwd` file
`auxdb`	The net service name for the auxiliary database

If the auxiliary database uses password files for authentication, then you can connect using a password for either local or remote access. If you are connecting remotely through a net service name, then authentication through a password file is mandatory.

Connecting to an Auxiliary Database from the Command Line

To connect to an auxiliary instance from the operating system command line, enter the following:

```
% rman AUXILIARY SYS/aux@auxdb
```

To connect to the target, auxiliary, and recovery catalog databases, issue the following (all on one line):

```
% rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb AUXILIARY SYS/aux@auxdb
```

Connecting to an Auxiliary Database from the RMAN Prompt

Alternatively, you can start RMAN and connect to the auxiliary database from the RMAN prompt:

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

To connect to the target, auxiliary, and recovery catalog databases, issue:

```
% rman
RMAN> CONNECT TARGET SYS/oracle@trgt
RMAN> CONNECT CATALOG rman/cat@catdb
RMAN> CONNECT AUXILIARY SYS/aux@auxdb
```

See Also:

- Chapter 11, "Performing RMAN Tablespace Point-in-Time Recovery"
- Chapter 13, "Creating a Standby Database with Recovery Manager"

Hiding Passwords When Connecting to Databases

To connect to RMAN from the operating system command line and hide authentication information, you must first start RMAN and then perform either of the following actions:

- Run the `CONNECT` commands at the RMAN prompt. If the password is not provided in the connect string, then RMAN prompts for the password.
- Run a command file at the RMAN prompt that contains the connection information. You can set the read privileges on the command file to prevent unauthorized access.

For example, if you are running RMAN in an UNIX environment, then you can use the following procedure:

1. Start RMAN without connecting to any databases:

   ```
   % rman
   ```

2. Place the connection information in a text file. For example, place the following lines in a file called `connect.rman`:

   ```
   CONNECT TARGET SYS/oracle@trgt
   CONNECT CATALOG rman/cat@catdb
   ```

3. Change the permissions on the connect script so that everyone can execute the script but only the desired users have read access. For example, enter:

   ```
   % chmod 711 connect.rman
   ```

4. Run the script from the RMAN prompt to connect to the target and catalog databases. For example:

   ```
   RMAN> @connect.rman
   ```

Executing RMAN Commands Through a Pipe

The RMAN pipe interface is an alternative method for issuing commands to RMAN and receiving the output. By using a pipe, RMAN can interface with the `DBMS_PIPE` PL/SQL package and avoid the operating system command shell altogether.

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, be careful to 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 use unique pipe names for each session of RMAN. The `DBMS_PIPE.UNIQUE_SESSION_NAME` function is one method that can be used to generate unique pipe names.

Note that the order of steps in the procedure does not have to be exactly as shown in scenario 1. In scenario 2, you put commands into the pipe and then connect to the database.

To use RMAN in pipe mode (scenario 1):

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

   ```
   % rman PIPE abc TARGET SYS/oracle@trgt
   ```

   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 SYS/oracle@trgt 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. Send the `EXIT` command to force RMAN to terminate. Alternatively, use the `TIMEOUT` option, in which case RMAN terminates automatically after not receiving any input for the specified length of time).

To use RMAN in pipe mode (scenario 2):

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 SYS/oracle@trgt 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: *Oracle9i Supplied PL/SQL Packages and Types Reference* for documentation on the DBMS_PIPE package and "RMAN Pipe Interface" on page 4-10 for a brief overview of RMAN pipes

Exiting RMAN

To quit RMAN and terminate the program, type EXIT or QUIT at the RMAN prompt. For example:

```
RMAN> EXIT
```

See Also: *Oracle9i Recovery Manager Reference* for EXIT syntax or *Oracle9i Recovery Manager Reference* for QUIT syntax

3 Quick Start to Recovery Manager

This chapter describes how to get started using RMAN. This chapter contains these topics:

Using Sample RMAN Scripts and Scenarios

The `?/rdbms/demo` subdirectory (the location differs depending on your operating system) contains a number of sample RMAN scripts. The scripts are suffixed with the `.rcv` filename extension.

These files are case studies of RMAN commands that are fully documented so that you can understand the features used. You can customize them for your use in your own backup and recovery plan, or simply read them to learn how RMAN is used.

See Also:

- *Oracle9i Recovery Manager Reference* to learn how to run command files from the RMAN prompt
- *Oracle9i Recovery Manager Reference* to learn how to run command files from the command line
- "Managing RMAN Scripts Stored in the Recovery Catalog" on page 16-15 to learn how to create and execute stored scripts

Choosing an RMAN Authentication Method

As explained in "Authentication for Database Connections" on page 2-2, to run RMAN commands you must first connect RMAN to the target database with the SYSDBA role to a dedicated server process. You can either connect using operating system authentication or using Oracle Net. An Oracle Net connection as SYSDBA requires that you create a password file.

Table 3–1 Deciding on an RMAN Authentication Method

If you connect using . . .	Then . . .
Operating system authentication	You must have SYSDBA privileges on your operating system, and the initialization parameter `REMOTE_LOGIN_PASSWORDFILE` must be set to `NONE`. **See Also:** *Oracle9i Database Administrator's Guide* to learn about authentication, and your operating system specific documentation to learn about adding SYSDBA privileges

Table 3–1 Deciding on an RMAN Authentication Method

If you connect using . . .	Then . . .
Oracle Net	You must create a password file using the ORAPWD utility, set `REMOTE_LOGIN_PASSWORDFILE` to `EXCLUSIVE`, and add the user who connects to the target database to the password file. Also, you must add the net service name of the target database to the `tnsnames.ora` file and configure the `listener.ora` file so that the target database can receive connections. **See Also:** *Oracle9i Database Administrator's Guide*, and *Oracle9i Net Services Administrator's Guide*

See Also:

- Chapter 2, "Connecting to Databases with RMAN" for a complete explanation of how to start RMAN and connect to target, recovery catalog, and auxiliary databases
- "Setting Up RMAN for Use with a Shared Server" on page 8-29 to learn how to connect RMAN to a dedicated server process on a database that is running a shared server configuration

Using Basic RMAN Commands

After you have learned how to connect to a target database, you can immediately begin performing backup and recovery operations. Use the examples in this section to go through a basic backup and restore scenario using a test database. These examples assume the following:

- The test database is in `ARCHIVELOG` mode.
- You are running in the default `NOCATALOG` mode.
- The RMAN executable is running on the same host as the test database.
- You are connecting from the command line using operating system authentication (see "Authentication for Database Connections" on page 2-2).
- You are not running an Oracle Real Application Clusters configuration.

This section contains these topics:

- Connecting to the Target Database
- Starting Up and Shutting Down the Database

Connecting to the Target Database

The first task is to connect to the target database. If you have created a recovery catalog, then you can connect to it as well—although these examples assume you are connecting in the default `NOCATALOG` mode.

At the operating system command line, enter the following to connect to the target database in the default `NOCATALOG` mode:

```
% rman TARGET /
```

If the database is already mounted or open, then RMAN displays output similar to the following:

```
Recovery Manager: Release 9.2.0.0.0

connected to target database: RMAN (DBID=1237603294)
```

The `DBID` value displayed is the database identifier for the target database.

If the target database is not started, then RMAN shows the following message:

```
connected to target database (not started)

RMAN>   # the RMAN prompt is displayed
```

See Also: *Oracle9i Recovery Manager Reference* for connection options

Starting Up and Shutting Down the Database

If the database is not started, then run the STARTUP command at the RMAN prompt, specifying an initialization parameter file only if you do not use a server parameter file. This example starts the instance with the server parameter file:

```
RMAN> STARTUP MOUNT

Oracle instance started
database mounted
```

If the database is open, then you can run the following RMAN commands to close it cleanly and then mount it:

```
RMAN> SHUTDOWN IMMEDIATE

database closed
database dismounted
Oracle instance shut down

RMAN> STARTUP MOUNT
```

> **See Also:** *Oracle9i Recovery Manager Reference* for STARTUP syntax and *Oracle9i Recovery Manager Reference* for SHUTDOWN syntax

Reporting the Current Schema of the Target Database

In this example, you generate a report describing the target datafiles. Run the REPORT SCHEMA command as follows:

```
RMAN> REPORT SCHEMA;
```

RMAN displays the datafiles currently in the target database. Depending on the contents of the database, you will see output similar to the following:

```
Report of database schema
File K-bytes    Tablespace           RB segs Datafile Name
---- ---------- -------------------- ------- -------------------
1        204800 SYSTEM               ***     /oracle/oradata/trgt/system01.dbf
2         20480 UNDOTBS              ***     /oracle/oradata/trgt/undotbs01.dbf
3         10240 CWMLITE              ***     /oracle/oradata/trgt/cwmlite01.dbf
4         10240 DRSYS                ***     /oracle/oradata/trgt/drsys01.dbf
5         10240 EXAMPLE              ***     /oracle/oradata/trgt/example01.dbf
6         10240 INDX                 ***     /oracle/oradata/trgt/indx01.dbf
7         10240 TOOLS                ***     /oracle/oradata/trgt/tools01.dbf
8         10240 USERS                ***     /oracle/oradata/trgt/users01.dbf
```

See Also: Chapter 17, "Querying the RMAN Repository" to learn how to make reports, and *Oracle9i Recovery Manager Reference* for REPORT syntax

Backing Up the Database

In this task, you back up the database to the default disk location. The default location is port-specific. For example, on most UNIX systems the location is ?/dbs. Because you do not specify the FORMAT parameter in this example, RMAN assigns the backup a unique filename. If you do not manually allocate a channel, then RMAN uses a preconfigured disk channel by default.

You can make two basic types of backups: full and incremental. In a full backup, RMAN backs up all blocks of the target database files. In an incremental backup, RMAN backs up only the blocks that have changed since a previous backup.

Making a Full Backup

Run the BACKUP command at the RMAN prompt as follows to make a full backup of the datafiles, control file, and current server parameter file (if the instance is started with a server parameter file) to the default device type:

```
RMAN> BACKUP DATABASE;
```

Unless you explicitly configure the default device to tape with the CONFIGURE command, the default device is disk.

When you run the BACKUP command, RMAN creates a **backup set**, which is a logical object that contains one or more backup pieces. The BACKUP command output contains the essential information about the backup, as shown in the following example:

```
Starting backup at OCT 12 2001 19:09:48
using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=10 devtype=DISK
channel ORA_DISK_1: starting full datafile backupset
channel ORA_DISK_1: specifying datafile(s) in backupset
including current SPFILE in backupset
including current controlfile in backupset
input datafile fno=00001 name=/oracle/oradata/trgt/system01.dbf
input datafile fno=00002 name=/oracle/oradata/trgt/undotbs01.dbf
input datafile fno=00003 name=/oracle/oradata/trgt/cwmlite01.dbf
input datafile fno=00004 name=/oracle/oradata/trgt/drsys01.dbf
input datafile fno=00005 name=/oracle/oradata/trgt/example01.dbf
input datafile fno=00006 name=/oracle/oradata/trgt/indx01.dbf
input datafile fno=00007 name=/oracle/oradata/trgt/tools01.dbf
input datafile fno=00008 name=/oracle/oradata/trgt/users01.dbf
channel ORA_DISK_1: starting piece 1 at OCT 12 2001 19:09:56
channel ORA_DISK_1: finished piece 1 at OCT 12 2001 19:10:31
piece handle=/oracle/dbs/lvd6dtk1_1_1 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:39
Finished backup at OCT 12 2001 19:10:33
```

Making an Incremental Backup

Incremental backups are a convenient way to conserve storage space because they back up only database blocks that have changed. RMAN compares the current datafiles to a **base backup**, also called a **level 0 backup**, to determine which blocks to back up.

For example, you can make a full backup as a base backup and then make some updates to the test database and commit them. Then, when you run the following command, RMAN backs up only those blocks that have changed since the previous full backup:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;
```

Note that you will see lines such as the following in the output:

```
no parent backup or copy of datafile 1 found
```

This line does not indicate a problem, but simply means that no base LEVEL 0 incremental backup exists. RMAN automatically creates a LEVEL 0 backup for its base incremental backup.

See Also:

- "Backup Types" on page 5-36 to learn about the differences between full and incremental backups
- "Backing Up Database Files and Archived Logs with RMAN" on page 9-3 to learn how to make backups
- *Oracle9i Recovery Manager Reference* for BACKUP syntax

Backing Up a Tablespace

Besides backing up the whole database, you can back up a tablespace or datafile. In this example, back up the SYSTEM tablespace to disk using the preconfigured disk channel. Of course, you can choose to back up a different object.

Run the BACKUP command at the RMAN prompt as follows:

```
RMAN> BACKUP TABLESPACE SYSTEM;
```

See Also: "Backing Up Tablespaces with RMAN" on page 9-7 to learn how to back up tablespaces

Backing Up Archived Logs

Typically, database administrators back up archived logs on disk to a third-party storage medium such as tape. You can also back up archived logs to disk. In either case, you can delete the input logs automatically after the backup completes.

To back up all archived logs and delete the input logs (from the primary archiving destination only), run the `BACKUP` command at the RMAN prompt as follows:

```
RMAN> BACKUP ARCHIVELOG ALL DELETE INPUT;
```

See Also: "Backing Up Archived Redo Logs with RMAN" on page 9-11 to learn how to back up archived redo logs

Copying a Datafile

In this example, make an image copy of datafile 1 to a new location. An image copy differs from a backup set in that it is not in an RMAN-specific format. It is the equivalent of a copy made using an operating system command such as the UNIX `cp` command.

This example uses an automatically allocated disk channel to create a datafile copy named `df1.bak`. Run the `COPY` command as follows from the RMAN prompt, specifying the path name for the backup:

```
RMAN> COPY DATAFILE 1 TO '/tmp/df1.cpy';   # specify any filename that you choose
```

RMAN displays the full filename of the created file in the output, as in this example:

```
Starting copy at OCT 12 2001 19:11:28
using channel ORA_DISK_1
channel ORA_DISK_1: copied datafile 1
output filename=/tmp/df1.cpy recid=141 stamp=442955509
Finished copy at OCT 12 2001 19:11:59
Finished copy at 18-APR-01
```

See Also:

- "Image Copies" on page 5-62 for conceptual information
- "Copying Files with RMAN" on page 9-29 to learn how to make image copies
- *Oracle9i Recovery Manager Reference* for `COPY` syntax

Listing Backups and Copies

To list the backup sets and image copies that you have created, run the `LIST` command as follows:

```
RMAN> LIST BACKUP;
```

RMAN displays which backup sets and pieces it created as well as which datafiles it included in those sets, as in the following example:

```
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ -------------------
699     Full    122M       DISK        00:00:30     OCT 12 2001 19:10:23
        BP Key: 1293   Status: AVAILABLE   Tag: TAG20011012T191001
        Piece Name: /oracle/dbs/lvd6dtk1_1_1
  Controlfile Included: Ckp SCN: 322170       Ckp time: OCT 12 2001 19:09:53
  SPFILE Included: Modification time: OCT 12 2001 19:09:53
  List of Datafiles in backup set 699
  File LV Type Ckp SCN    Ckp Time            Name
  ---- -- ---- ---------- ------------------- ----
  1       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/system01.dbf
  2       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/undotbs01.dbf
  3       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/cwmlite01.dbf
  4       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/drsys01.dbf
  5       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/example01.dbf
  6       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/indx01.dbf
  7       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/tools01.dbf
  8       Full 322172     OCT 12 2001 19:09:56 /oracle/oradata/trgt/users01.dbf
```

To list image copies, run the following command:

```
RMAN> LIST COPY;
```

RMAN displays both datafile and control file copies as well as archived redo logs (an archived redo log is considered a type of copy):

```
List of Datafile Copies
Key     File S Completion Time     Ckp SCN    Ckp Time            Name
------- ---- - ------------------- ---------- ------------------- ----
141     1    A OCT 12 2001 19:11:49 322204     OCT 12 2001 19:11:29 /tmp/df1.copy

List of Archived Log Copies
Key     Thrd Seq     S Low Time            Name
------- ---- ------- - ------------------- ----
1105    1    70      A OCT 10 2001 17:55:49 /oracle/oradata/trgt/arch/archive1_70.dbf
1106    1    71      A OCT 11 2001 05:12:06 /oracle/oradata/trgt/arch/archive1_71.dbf
1107    1    72      A OCT 11 2001 14:54:36 /oracle/oradata/trgt/arch/archive1_72.dbf
1108    1    73      A OCT 11 2001 14:55:48 /oracle/oradata/trgt/arch/archive1_73.dbf
1109    1    74      A OCT 11 2001 15:13:27 /oracle/oradata/trgt/arch/archive1_74.dbf
1110    1    75      A OCT 12 2001 06:56:22 /oracle/oradata/trgt/arch/archive1_75.dbf
```

See Also:

- "LIST Command Output" on page 7-2 for conceptual information about lists
- "Listing RMAN Backups, Copies, and Database Incarnations" on page 17-3 to learn how to make lists and reports
- *Oracle9i Recovery Manager Reference* for syntax and an explanation of the column headings in the `LIST` output

Validating the Restore of a Backup

Check that you are able to restore the backups that you created without actually restoring them. Run the `RESTORE ... VALIDATE` command as follows:

```
RMAN> RESTORE DATABASE VALIDATE;
```

You should see output similar to the following:

```
Starting restore at 07-DEC-01

allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=9 devtype=DISK
channel ORA_DISK_1: starting validation of datafile backupset
channel ORA_DISK_1: restored backup piece 1
piece handle=/oracle/dbs/07db39tr_1_1 tag=TAG20011012T191001 params=NULL
channel ORA_DISK_1: validation complete
Finished restore at 07-DEC-01
```

If there are no error messages, then RMAN confirms that the database backup can be restored. When there is an error, RMAN always displays an error banner and provides messages indicating the nature of the error. For example, if you attempt to restore the database when some backups do not exist, then RMAN displays an error stack such as the following:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of restore command at 12/07/2001 17:28:24
RMAN-06026: some targets not found - aborting restore
RMAN-06023: no backup or copy of datafile 8 found to restore
RMAN-06023: no backup or copy of datafile 7 found to restore
```

See Also: "Performing a Backup Validation with RMAN" on page 9-28 to learn how to restore backups and copies, and *Oracle9i Recovery Manager Reference* for `VALIDATE` syntax

Restoring and Recovering the Database

The primary aspect of developing a backup and recovery strategy is learning what to do in case of a media failure. In this scenario, you simulate a media failure. First, shut down the database and then exit RMAN with the following commands:

```
RMAN> SHUTDOWN IMMEDIATE;
RMAN> EXIT;
```

After the database is shut down, use the output from the `REPORT SCHEMA` command in "Reporting the Current Schema of the Target Database" on page 3-5 to identify the filenames of your datafiles. Temporarily rename some or all of your database files with operating system commands (but make sure not to rename your control files). This action simulates a media failure because Oracle is able to find the datafiles during startup.

This UNIX example temporarily renames the datafile in the `tools` tablespace:

```
% mv $ORACLE_HOME/oradata/trgt/tools01.dbf $ORACLE_HOME/oradata/trgt/tools01.bak
```

Now, start RMAN and attempt to open the database as in the following example:

```
% rman TARGET /
RMAN> STARTUP

RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of startup command at 08/29/2001 13:46:04
ORA-01157: cannot identify/lock data file 7 - see DBWR trace file
ORA-01110: data file 7: '/oracle/oradata/trgt/tools01.dbf'
```

The database mounts but does not open because some datafiles require recovery. To restore and recover the database using the default disk channel, simply run the following commands from the RMAN prompt:

```
RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE;
```

RMAN uses the backups and copies that you made earlier and restores the files to their default locations. Then, it uses archived redo logs (if needed) to recover the database. After recovery is complete, open the database:

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

Restoring and Recovering a Tablespace

It is not uncommon for a media failure to affect some but not all files in a database. In this scenario, you simulate a media failure to a datafile in a single tablespace. Use the output from the REPORT SCHEMA command in the previous example to identify the filenames of datafiles in a tablespace other than the SYSTEM tablespace. This example renames datafiles in the tools tablespace.

While the database is open, rename one or more datafiles in the tablespace that you selected. This UNIX example temporarily renames a datafile in tools tablespace:

```
% mv $ORACLE_HOME/oradata/trgt/tools01.dbf /tmp/tools01.dbf
```

If you attempt to update a table located in one of the renamed datafiles, Oracle generates an error message because the datafile is unavailable. You need to restore and recover the missing datafiles. First, start RMAN and then take the tablespace offline using the SQL command, making sure to specify the name of the tablespace:

```
RMAN> SQL 'ALTER TALBESPACE tools OFFLINE IMMEDIATE';
```

To restore and recover the tablespace with the default disk channel, simply run the following commands from the RMAN prompt:

```
RMAN> RESTORE TABLESPACE tools;
RMAN> RECOVER TABLESPACE tools;
```

After recovery is complete, bring the tablespace online with the following command:

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

Showing the RMAN Configuration

RMAN contains some default configuration settings. These settings apply to all RMAN sessions until you explicitly change or disable them with another CONFIGURE command. You can always return to a default configuration setting by running the CONFIGURE ... CLEAR command.

Of the possible configurations, perhaps the most important are the automatic channels and the retention policy. RMAN is preconfigured with an automatic disk channel so that you can make backups and copies to disk without manually allocating channels.

To see all the current RMAN configuration settings, run this command:

```
RMAN> SHOW ALL;
```

See Also: Chapter 8, "Configuring the Recovery Manager Environment" to learn how to create RMAN configurations

Deciding Whether to Use RMAN with a Recovery Catalog

By default, RMAN connects to the target database in `NOCATALOG` mode, meaning that it uses the control file in the target database as the sole repository of RMAN metadata. Perhaps the most important decision you make when using RMAN is whether to create a recovery catalog as the RMAN repository for normal production operations. A recovery catalog is a schema created in a separate database that contains metadata obtained from the target control file.

In general, Oracle Corporation advises using a catalog when you manage multiple databases. If you have more than one database to back up, then you can create one systemwide recovery catalog and store metadata for all the databases in this catalog. Hence, you avoid the extra space requirements and memory overhead of maintaining multiple databases, each with a single catalog. You need to take extra precautions when backing up the catalog, however, because if you lose the catalog then you lose the metadata for multiple target databases.

This section outlines some of the costs and benefits associated with using and not using a recovery catalog. If you decide to create a catalog, refer to "Creating the Recovery Catalog" on page 16-2 for instructions.

See Also: "RMAN Repository" on page 4-11 for an overview of the function of the RMAN repository in the RMAN environment

Benefits of Using the Recovery Catalog as the RMAN Repository

When you use a recovery catalog, RMAN can perform a wider variety of automated backup and recovery functions than when you use the control file in the target database as the sole repository of metadata. The following features are available only with a catalog:

- You can store metadata about multiple target databases in a single catalog.
- You can store metadata about multiple incarnations of a single target database in the catalog. Hence, you can restore backups from any incarnation.
- Resynchronizing the recovery catalog at intervals less than the `CONTROL_FILE_RECORD_KEEP_TIME` setting, you can keep historical metadata.
- You can report the target database schema at a noncurrent time.
- You can store RMAN scripts in the recovery catalog.

- When restoring and recovering to a time when the database files that exist in the database are different from the files recorded in the mounted control file, the recovery catalog specifies which files that are needed. Without a catalog, you must first restore a control file backup that lists the correct set of database files.
- If the control file is lost and must be restored from backup, and if persistent configurations have been made to automate the tape channel allocation, these configurations are still available when the database is not mounted.

Costs of Using the Recovery Catalog as the RMAN Repository

The main cost of using a catalog is the maintenance overhead required for this additional database. For example, you have to:

- Find a database other than the target database to store the recovery catalog (otherwise, the benefits of maintaining the catalog are lost), or create a new database
- Create enough space on the database for the RMAN metadata (as described in "Configuring the Recovery Catalog Database" on page 16-2)
- Back up the recovery catalog metadata
- Upgrade the recovery catalog when necessary

Hence, unless you manage a network of databases, you may choose to avoid the overhead and use the control file as the exclusive repository of metadata. When you use a control file as the RMAN repository, RMAN still functions effectively. If you do not use a catalog, read the section "Managing the RMAN Repository Without a Recovery Catalog" on page 16-31. Specifically, make sure you:

- Consider the costs of not using a recovery catalog, as described in "Understanding Catalog-Only Command Restrictions" on page 16-31
- Develop a strategy for backing up the repository, as described in "Backing Up and Restoring the Control File" on page 16-34

See Also: Chapter 16, "Managing the Recovery Manager Repository" to learn how to manage the recovery catalog

Part II

Recovery Manager Architecture and Concepts

Part II describes the architecture of the RMAN environment and introduces basic concepts. This part contains these chapters:

4

Recovery Manager Architecture

This chapter describes the Recovery Manager (RMAN) interface and the basic elements of the RMAN environment.

This chapter contains these topics:

Note: RMAN is only compatible with Oracle databases of release 8.0 or higher.

See Also: *Oracle9i Database Migration* for information about RMAN compatibility and upgrade issues

Overview of RMAN Architecture

This section contains these topics:

- About the RMAN Environment
- RMAN Session Architecture
- Storage of RMAN Metadata

About the RMAN Environment

Recovery Manager (RMAN) is a client application that performs backup and recovery operations. The **Recovery Manager environment** consists of the various applications and databases that play a role in a backup and recovery strategy.

The RMAN environment can be as simple as an RMAN executable connecting to a target database, or as complex as an RMAN executable connecting to multiple media managers and multiple target, recovery catalog, and auxiliary databases, all accessed through Oracle Enterprise Manager. Table 4–1 lists possible components of the RMAN environment.

Table 4–1 Components of the RMAN Environment

Component	Description	Required?
Target database	The control files, datafiles, and optional archived redo logs that RMAN is in charge of backing up or restoring. RMAN uses the target database control file to gather information about the database and to store information about its own operations. The actual work of the backup and recovery jobs is performed by server sessions on the target database.	Yes
RMAN executable	The client application that manages backup and recovery operations for a target database. The RMAN client uses 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.	Yes
Recovery catalog database	A database containing the recovery catalog schema, which contains the metadata that RMAN uses to perform its backup and recovery operations.	No
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.	No

Table 4–1 Components of the RMAN Environment

Component	Description	Required?
Standby database	A copy of the primary database that is updated using archived logs created by the primary database. RMAN can create or back up a standby database.	No
Media management application	A vendor-specific application that allows RMAN to back up to a storage system such as tape. When doing backups or restores, the RMAN client connects to the target instance and directs the instance to talk to its media manager. No direct communication occurs between the RMAN client and the media manager: all communication occurs on the target instance.	No
Media management catalog	A vendor-specific repository of information about a media management application.	No
Oracle Enterprise Manager	A GUI-based application that you can use as an interface to RMAN.	No

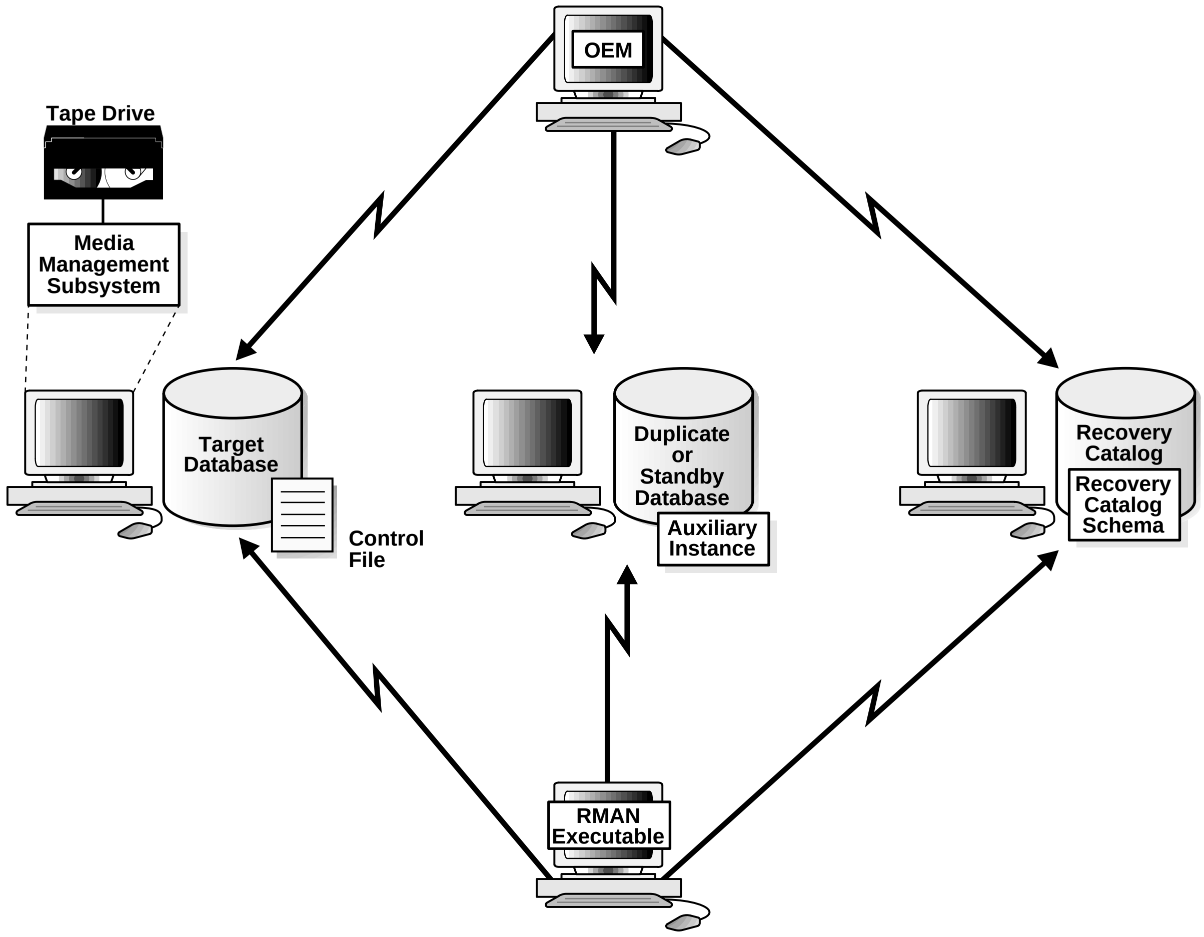
As the table illustrates, the only required components in an RMAN environment are the target database and the RMAN executable. Nevertheless, most real-world configurations are more complicated.

Figure 4–1 depicts an example of a realistic RMAN environment. In this environment, five nodes are networked together, with each machine serving a different purpose. The five nodes share duties as follows:

- One client node runs the RMAN executable
- One server node hosts the target database and media management subsystem
- One server node hosts the duplicate or standby database
- One server node hosts the recovery catalog database
- One client node runs the Oracle Enterprise Manager application, which provides a GUI interface to the databases in the system

In this scenario, you can run the RMAN executable from a client machine, and then connect to the target, catalog, and auxiliary databases. You can then run backup and recovery jobs. You can also connect to the client hosting Oracle Enterprise Manager and use the GUI interface to access RMAN.

Figure 4–1 Example RMAN Environment



RMAN Session Architecture

The RMAN client application directs database server sessions to perform all backup and recovery tasks. The meaning of "session" in this sense depends on the operating system. For example, on a UNIX system a server session corresponds to a server process. On a Windows NT system, an server session corresponds to a thread within the Oracle service.

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

Storage of RMAN Metadata

Because RMAN manages backup and recovery operations, it requires a place to store necessary information about the database. RMAN always stores this information in the target database control file. You can also store RMAN metadata in a recovery catalog schema contained in a separate database. The recovery catalog schema must be stored in a database *other than* the target database. RMAN periodically migrates information from the control file to the recovery catalog.

RMAN Command Interface

Use the RMAN interface to enter commands that you can use to manage all aspects of backup and recovery operations.

Note: All RMAN commands for Oracle release 8.0 and higher also work in Oracle9*i*.

This section contains these topics:

- RMAN PL/SQL Packages
- How RMAN Compiles and Executes Commands
- Types of RMAN Commands
- User Execution of RMAN Commands
- RMAN Pipe Interface
- RMAN Job Scripts in Oracle Enterprise Manager

See Also: *Oracle9i Recovery Manager Reference* for a complete description of RMAN commands and their syntax

RMAN PL/SQL Packages

The RMAN executable uses PL/SQL packages to communicate with the target database and recovery catalog. The packages are internal and undocumented.

How RMAN Compiles and Executes Commands

RMAN processes most commands in the two phases discussed in this section:

- Compilation Phase
- Execution Phase

Compilation Phase

During the compilation phase, RMAN determines which objects the command will access by, for example, translating a `TABLESPACE` keyword into the names of its component datafiles. Then, RMAN constructs a sequence of remote procedure calls (RPCs) that instruct the target database to perform the desired operation, such as backing up, restoring, or recovering datafiles.

Execution Phase

During the execution phase, RMAN sends the RPC calls to the target database, monitors their progress, and collects the results. If more than one channel is allocated, then RMAN can execute certain commands in parallel so that all of the channels' target database sessions are concurrently executing an RPC call. RMAN coordinates this parallel execution and monitors the progress on all channels that are doing work.

Types of RMAN Commands

RMAN commands can be divided in this way:

- Standalone commands, which are commands that can only be executed at the RMAN prompt
- Job commands, which are commands that can only be executed within the brackets of a `RUN` command
- Command exceptions, which can be executed either at the RMAN prompt or within the brackets of a `RUN` command

Besides these commands, RMAN also supports a number of command-line arguments that you can specify when you start RMAN.

> **See Also:** *Oracle9i Recovery Manager Reference* for a complete description of RMAN commands and command-line options

Standalone Commands

Unlike job commands, standalone commands cannot appear as subcommands within `RUN`. Following are some of the commands that you must use on the RMAN prompt:

- `CONNECT`
- `CONFIGURE`
- `CREATE CATALOG`, `DROP CATALOG`, `UPGRADE CATALOG`
- `CREATE SCRIPT`, `DELETE SCRIPT`, `REPLACE SCRIPT`
- `LIST`
- `REPORT`

> **See Also:** Syntax entries in *Oracle9i Recovery Manager Reference* to determine which must be executed at the RMAN prompt

Job Commands

Unlike standalone commands, job commands must appear within the brackets of a `RUN` command. Following are examples of job commands:

- `ALLOCATE CHANNEL`
- `SWITCH`

RMAN executes the job commands inside of a `RUN` command block sequentially. If any command within the block fails, then RMAN ceases processing—no further commands within the block are executed. In effect, the `RUN` command defines a unit of command execution. When the last command within a `RUN` block completes, Oracle releases any server-side resources such as I/O buffers or I/O slave processes allocated within the block.

> **See Also:** *Oracle9i Recovery Manager Reference* to learn about `RUN` command syntax

Command Exceptions

Although some commands are either standalone commands or job commands exclusively, other commands can be issued either at the prompt or within a `RUN` command. Whether issued at the prompt or within `RUN`, the commands can make use of automatic channels. Note that you can manually allocate channels only within a `RUN` command, and in this case the manually allocated channels override any configured automatic channels.

The following are examples of commands that can function as both standalone and job commands:

- `BACKUP`
- `BLOCKRECOVER`
- `COPY`
- `RESTORE`
- `RECOVER`
- `VALIDATE`

User Execution of RMAN Commands

RMAN uses a command language interpreter (CLI) that can execute commands in interactive or batch mode. You can also specify the `LOG` option at the command line to write RMAN output into a log file.

> **See Also:** "Terminating an RMAN Command" on page 15-11 to learn how to end an RMAN session

Interactive Mode

To run RMAN commands interactively, start RMAN and then type commands into the command-line interface. For example, you can start RMAN from the UNIX command shell and then execute interactive commands as follows:

```
% rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
```

After the RMAN prompt is displayed, you can enter commands such as the following:

```
RMAN> BACKUP DATABASE;
```

Batch Mode

You can type RMAN commands into a file, and then run the **command file** by specifying its name on the command line. The contents of the command file must be identical to commands entered at the command line.

When running in batch mode, RMAN reads input from a command file and writes output messages to a log file (if specified). Batch mode is most suitable for performing regularly scheduled backups through an operating system job control facility.

In this example, a sample RMAN script is placed into a command file called `b_l0.rcv`. You can run this file from the operating system command line and write the output into the log file `log.f` as follows:

```
% rman TARGET / CATALOG rman/cat@catdb CMDFILE b_l0.rcv LOG log.f
```

> **See Also:** *Oracle9i Recovery Manager Reference* for more information about RMAN command line options

Stored Scripts

A **stored script** is a block of RMAN job commands that is stored in the recovery catalog. Stored scripts allow you to plan, develop, and test commands for backing up, restoring, or recovering the database. Also, scripts minimize the potential for user errors. Note that each stored script relates to one and only one target database.

To create a stored script, either enter the script interactively into the RMAN command-line interface, or enter the RMAN commands into a command file and run the file. You must be connected to a target database and recovery catalog.

Following is an example of a stored script:

```
REPLACE SCRIPT b_whole_l0
{
  # back up whole database and archived logs
  BACKUP
    INCREMENTAL LEVEL 0
    TAG b_whole_l0
    FILESPERSET 6
    DATABASE PLUS ARCHIVELOG;
}
```

You can execute this stored script from the RMAN prompt as follows:

```
RUN { EXECUTE SCRIPT b_whole_10 };
```

View stored scripts by querying the recovery catalog view `RC_STORED_SCRIPT`:

```
SQL> SELECT * FROM RC_STORED_SCRIPT;

    DB_KEY DB_NAME   SCRIPT_NAME
---------- -------   -----------------------------------------------------
         1 RMAN      full_backup
         1 RMAN      incr_backup_0
         1 RMAN      incr_backup_1
         1 RMAN      incr_backup_2
         1 RMAN      log_backup
```

See Also: "Managing RMAN Scripts Stored in the Recovery Catalog" on page 16-15 for more on stored scripts. Also see to the sample scripts in the `?/rdbms/demo` directory.

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. RMAN does not read or write any data using the operating system shell. By 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. 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 SYS/oracle@trgt
```

RMAN opens the following pipes in the target database:

- `ORA$RMAN_ABC_IN`, which RMAN uses to receive user commands
- `ORA$RMAN_ABC_OUT`, which RMAN uses to send all output

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.

See Also: "Executing RMAN Commands Through a Pipe" on page 2-9 to learn how to execute RMAN commands through a pipe

RMAN Job Scripts in Oracle Enterprise Manager

Use the Run RMAN Script feature to issue any command or script within Oracle Enterprise Manager that can also be called from the RMAN command line. The rman script will be run as a job through the Oracle Enterprise Manager Job System when you submit or schedule it.

Oracle Enterprise Manager's Job System enables the automation of standard and administrative tasks. With the Job System, you can create and manage jobs, schedule their execution, and view and share information about defined jobs with other administrators.

RMAN Repository

The RMAN repository is the collection of metadata about the target databases that RMAN uses to conduct its backup, recovery, and maintenance operations. You can either create a **recovery catalog** in which to store this information, or let RMAN store it exclusively in the target database control file. Although RMAN can conduct all major backup and recovery operations using just the control file, some RMAN commands function only when you use a recovery catalog.

The recovery catalog is maintained solely by RMAN; the target database never accesses it directly. RMAN propagates information about the database structure, archived redo logs, backup sets, and datafile copies into the recovery catalog from the target database's control file.

> **See Also:** Chapter 16, "Managing the Recovery Manager Repository" to learn how to manage the RMAN repository, and "Understanding Catalog-Only Command Restrictions" on page 16-31 for a list of catalog-only commands

Storage of the RMAN Repository in the Recovery Catalog

The recovery catalog is an optional repository of information about the target databases that RMAN uses and maintains. It is recommended that you store the catalog in a dedicated database. RMAN uses the information in the recovery catalog, which is obtained from the control file, to determine how to execute requested backup and recovery operations.

This section contains these topics:

- Registration of Databases in the Recovery Catalog
- Contents of the Recovery Catalog
- Resynchronization of the Recovery Catalog
- Backups of the Recovery Catalog
- Compatibility of the Recovery Catalog

Registration of Databases in the Recovery Catalog

The enrolling of a database in a recovery catalog is called **registration**. You can register more than one target database in the same recovery catalog. For example, you can register databases `prod1`, `prod2`, and `prod3` in a single catalog owned by `catowner` in the database `catdb`. You can register a database only once in a given catalog schema: for example, you cannot register `prod1` in the `catowner` schema and then register `prod1` again in the `catowner` schema.

Because RMAN distinguishes databases by unique database identifier (DBID), each database registered in a given catalog must have a DBID. You cannot register two databases with the same DBID in the same catalog. For example, you cannot register database `prod1` with DBID 862893450 and database `prod2` with DBID 862893450 in the same catalog. However, each database does not have to have a unique database name. For example, you can register database `prod1` with DBID 862893450 and a different database called `prod1` with DBID 951781249 in the same recovery catalog.

If you use operating system commands to copy a database, then the copied database has the same DBID as the original database. Thus, you cannot copy a database manually and then register it in the same catalog with its parent unless you change the DBID of the copied database. For this reason, it is easiest to use the `DUPLICATE` command to create a copied database because RMAN automatically gives the copied database a different DBID. You can also use the DBNEWID utility to change the DBID (or the database name) of any Oracle database. DBNEWID is useful when you have already created multiple databases with the same DBID and you now want to register them all in the same recovery catalog.

See Also: *Oracle9i Database Utilities* to learn how to use the DBNEWID utility to change the DBID of a database

Contents of the Recovery Catalog

The recovery catalog contains information about RMAN operations, including:

- Datafile and archived redo log backup sets and backup pieces
- Datafile copies
- Archived redo logs and their copies
- Tablespaces and datafiles on the target database
- Stored scripts, which are named user-created sequences of RMAN commands
- Persistent RMAN configuration settings

Resynchronization of the Recovery Catalog

The recovery catalog obtains crucial RMAN metadata from the target database control file. 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 to update changed data, but does not resynchronize metadata about the database physical schema: datafiles, tablespaces, redo threads, rollback segments, and online redo logs. In a full resynchronization, RMAN updates all changed records, including schema records.

RMAN automatically detects when it needs to perform a full or partial resynchronization and executes the operation as needed. You can also force a full resynchronization by issuing a `RESYNC CATALOG` command, which you should do in the cases described in "When Should You Resynchronize?" on page 16-13.

To ensure that the catalog stays current, run the `RESYNC CATALOG` command periodically. A good rule of thumb is to run it at least once every *n* days, where *n* is the setting for the initialization parameter `CONTROL_FILE_RECORD_KEEP_TIME`. Because the control file employs a circular reuse system, backup and copy records eventually get overwritten. Resynchronizing the catalog ensures that these records are stored in the catalog and so are not lost.

See Also: "Types of Records in the Control File" on page 4-15 for more information about control file records

Snapshot Control File RMAN generates a **snapshot control file**, which is a temporary backup control file, each time it performs a full resynchronization. This snapshot control file ensures that RMAN has a consistent view of the control file. Because the snapshot control file is intended for RMAN's short-term use, it is not registered in the recovery catalog. RMAN records the snapshot control file checkpoint in the recovery catalog to indicate the currency of the recovery catalog.

The Oracle9*i* server ensures that only one RMAN session accesses a snapshot control file at any point in time. This safeguard is necessary to ensure that two RMAN sessions do not interfere with each other's use of the snapshot control file.

Note: You can specify the name and location of the snapshot control file. For instructions, refer to "Configuring the Snapshot Control File Location" on page 8-28.

See Also: "Managing the Control File When You Use a Recovery Catalog" on page 16-18 to learn how to resynchronize the recovery catalog, and *Oracle9i Recovery Manager Reference* for syntax

Backups of the Recovery Catalog

A single recovery catalog is able to store information for multiple target databases. Consequently, loss of the recovery catalog can be disastrous. You should back up the recovery catalog frequently.

If the recovery catalog is destroyed and no backups of it are available, then you can partially reconstruct the catalog from the current control file or control file backups. Nevertheless, you should always aim to have a valid, recent backup of the catalog.

> **See Also:** "Backing Up the Recovery Catalog" on page 16-20 to learn how to back up the recovery catalog

Compatibility of the Recovery Catalog

When you use RMAN with a recovery catalog, the RMAN environment contains the following components:

- RMAN executable
- Recovery catalog database
- Recovery catalog schema in the recovery catalog database
- Target database

Each of these components has a release number associated with it. For example, you can use a release 8.0.6.1 RMAN executable with a release 8.1.6 target database, and store the repository in a release 8.1.5 recovery catalog database whose catalog schema was created in release 8.1.6.

> **See Also:** *Oracle9i Recovery Manager Reference* for a chart describing the compatibility of the components in the RMAN environment

Storage of the RMAN Repository Exclusively in the Control File

Because most information in the recovery catalog is also available in the target database's control file, RMAN supports an operational mode in which it uses the target database control file instead of a recovery catalog. This mode is especially appropriate for small databases where installation and administration of a separate recovery catalog database is burdensome. The only RMAN feature that is not supported in `NOCATALOG` mode is stored scripts.

Types of Records in the Control File

When you do not use a recovery catalog, the control file is the exclusive source of information about backups and copies as well as other relevant information. The control file contains two types of records: **circular reuse records** and **noncircular reuse records**.

Circular Reuse Records Circular reuse records contain noncritical information that is eligible to be overwritten if the need arises. These records contain information that is continually generated by the database. Some examples of information circular reuse records include:

- Log history
- Archived redo logs
- Backups
- Offline ranges for datafiles

Circular reuse 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 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.

> **See Also:** "Monitoring the Overwriting of Control File Records" on page 16-32 to learn how to manage Oracle's treatment of circular reuse records

Noncircular Reuse Records 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 logs, and redo threads.

Recovery Without a Catalog

To restore and recover the database without using a recovery catalog, Oracle recommends that you:

- Enable the control **file autobackup feature**, which causes RMAN to automatically back up the control file and also enables RMAN to restore the control file autobackup without access to a repository (either a recovery catalog or the target database control file)
- Use a minimum of two multiplexed or mirrored control files on separate disks

- Keep all Recovery Manager backup logs

If you lose the current control files, then you can restore a control file autobackup even if you do not use a recovery catalog.

See Also:

- "Understanding Catalog-Only Command Restrictions" for a complete list of commands that are disabled unless you use a recovery catalog
- "Control File and Server Parameter File Autobackups" on page 5-47 to learn about disaster recovery using control file autobackups

Media Management

To use tape storage for database backups, RMAN requires a **media manager**. A media manager is a utility that loads, labels, and unloads sequential media such as tape drives for backing up and recovering data.

Oracle publishes a media management API that third-party vendors can use to build software that works with RMAN. To use RMAN to make backups to sequential media such as tape, integrate media management software with your Oracle software. Note that Oracle does not need to connect to the media management library (MML) software when it backs up to disk.

Some media management products can manage the entire data movement between Oracle datafiles and the backup devices. Such products may use technologies such as high-speed connections between storage and media subsystems, which can remove much of the backup load from the primary database server.

Backup and Restore Operations with a Media Manager

The following RMAN command performs a datafile backup to tape:

```
BACKUP DEVICE TYPE sbt DATAFILE 1;
```

When Recovery Manager executes this command, it sends the backup request to the Oracle server session performing the backup. The Oracle server session identifies the output channel as a media management device and makes a request to the media manager to write the output.

The media manager labels and keeps track of the tape and names of files on each tape. If your site owns an automated tape library, then the media manager automatically loads and unloads the tapes required by Oracle; if not, the media manager requests an operator to load a specified tape into the drive.

The media manager handles restore as well as backup operations. When you restore a file, the following steps occur:

1. Oracle requests the restore of a particular file.
2. The media manager identifies the tape containing the file and reads the tape.
3. The media manager passes the information back to the Oracle server session.
4. The Oracle session writes the file to disk.

Proxy Copy

Oracle has integrated **proxy copy** functionality into its media management API. Vendors can use this API to develop media management software that takes control of backup and restore operations. 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.

Media Manager Testing

A client program, `sbttest`, is a standalone test of the media management software that is linked with Oracle to perform backups to tape. Use it when Oracle is unable to create or restore backups using a vendor's media management product. Only use the `sbttest` program at the direction of Oracle support.

Backup Solutions Program

The Oracle Backup Solutions Program (BSP), part of the Oracle Partner Program, is a group of leading media management software vendors whose products are compliant with Oracle's MML specification. Software that is compliant with the MML interface enables an Oracle server session to back up to a media manager and request the media manager to restore backups.

Note that 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 that stream. When RMAN needs to restore the backup, it asks the media manager to retrieve the identical byte stream. All details of how and where that stream is stored are handled entirely by the media manager.

Several products may be available for your platform from media management vendors. For a current list of available products, you can access the Backup Solutions Program Web site at:

`http://otn.oracle.com/deploy/availability`

You can also contact your Oracle representative for a complete list.

To use a specific media management product, contact the media management vendor directly to determine whether it is a member of Oracle BSP. Note that Oracle Corporation does not certify media vendors for compatibility with RMAN, so any questions about availability, version compatibility, and functionality should be directed to the media vendor, not Oracle Corporation.

5

RMAN Concepts I: Channels, Backups, and Copies

This chapter describes the basic concepts involved in using the Recovery Manager (RMAN) utility.

This chapter contains these topics:

RMAN Automatic and Manual Channel Allocation

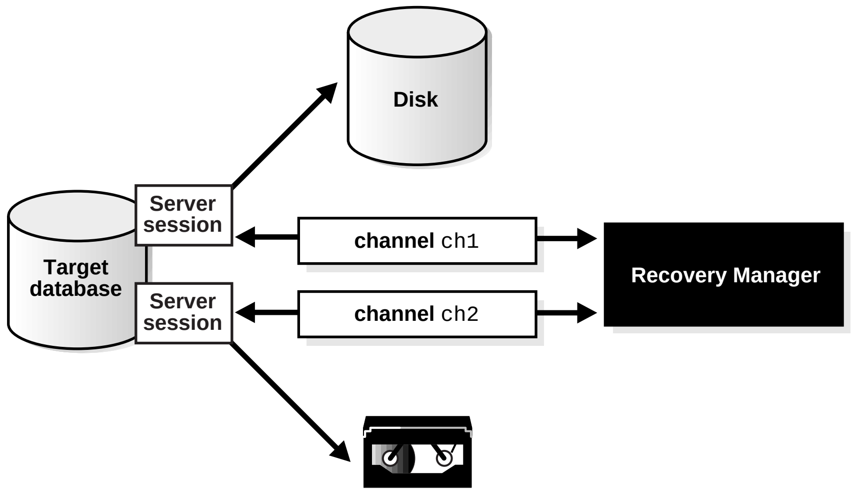
This section contains these topics:

See Also: *Oracle9i Real Application Clusters Administration* for information about channel allocation and backups in an Oracle Real Application Clusters environment

About RMAN Channels

An RMAN **channel** represents one stream of data to a device type and corresponds to one server session. Allocation of one or more RMAN channels is necessary to execute most backup and recovery commands. As illustrated in Figure 5–1, each channel establishes a connection from the RMAN executable to a target or auxiliary database instance by starting a server session on the instance. The server session performs the backup, restore, and recovery operations. Only one RMAN session communicates with the allocated server sessions.

Figure 5–1   Channel Allocation



You can either allocate channels manually within a RUN block, or preconfigure channels for use in all RMAN sessions using **automatic channel allocation**. RMAN comes preconfigured with a DISK channel that you can use for backups and copies to disk. You can also run the CONFIGURE CHANNEL command RMAN to specify automatic channels to disk or tape. In this way, you do not have to allocate channels every time you perform a backup, restore, or recovery operation.

When you run a command that requires a channel, and you do not allocate a channel manually, then RMAN automatically allocates the channels using the options specified in the CONFIGURE command. For the BACKUP command, RMAN allocates only a single type of channel, such as DISK or sbt. For the RESTORE command and the various maintenance commands (for example, DELETE), RMAN determines which device types are required, and allocates all necessary channels.

If you specify channels manually, then the ALLOCATE CHANNEL command (executed only within a RUN command) and ALLOCATE CHANNEL FOR MAINTENANCE command (executed only at the RMAN prompt) specify the type of I/O device that the server session will use to perform the backup, restore, or maintenance operation.

Whether the ALLOCATE CHANNEL command or CONFIGURE CHANNEL causes 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: *Oracle9i Recovery Manager Reference* for `ALLOCATE CHANNEL` syntax and *Oracle9i Recovery Manager Reference* on `ALLOCATE CHANNEL FOR MAINTENANCE`

Automatic and Manual Channel Allocation

You can use the automatic channel allocation feature to configure a set of persistent, automatic channels for use in all RMAN sessions. You can use the manual channel allocation feature you to specify channels for commands used within a `RUN` block.

Unless you manually run an `ALLOCATE CHANNEL` command, RMAN allocates automatic channels according to the settings in these commands:

- `CONFIGURE DEVICE TYPE ... PARALLELISM`
- `CONFIGURE DEFAULT DEVICE TYPE`
- `CONFIGURE CHANNEL DEVICE TYPE`
- `CONFIGURE CHANNEL` *n* `DEVICE TYPE`

For example, you can issue the following commands at the RMAN prompt:

```
BACKUP DATAFILE 3;
RUN { RESTORE TABLESPACE users; }
```

RMAN automatically allocates channels according to values set with the `CONFIGURE` command in the following cases:

- You use commands such as `BACKUP`, `RESTORE`, or `DELETE` outside of a `RUN` block.
- You use commands within a `RUN` block but do not allocate any channels within the `RUN` block.

You can override automatic channel allocation settings by manually allocating channels within a `RUN` block. You cannot mix automatic and manual channels, so manual channels always override automatic channels. For example, you override automatic channel allocation when you issue a command as follows:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

RMAN optimizes automatic channel allocation by leaving automatic channels allocated so long as each new command requires exactly the same channel configuration as the previous command. For example, RMAN can use the same preallocated channels for the following series of commands:

```
BACKUP DATAFILE 1;
BACKUP CURRENT CONTROLFILE;
BACKUP ARCHIVELOG ALL;
```

If you issue a command such as `ALLOCATE` or `CONFIGURE`, then RMAN automatically releases the preallocated channels.

> **See Also:** "Configuring Automatic Channels" on page 8-10 to learn how to configure automatic channels

Automatic Channel Device Configuration

The `CONFIGURE DEVICE TYPE ... PARALLELISM` command specifies the number of channels that RMAN uses when allocating automatic channels for a specified device type. For example, if you configure parallelism to 3, then RMAN allocates three channels of the default device type whenever it uses automatic channels.

When parallelizing, RMAN always allocates channels in numerical order, beginning with 1 and ending with the parallelism setting. For example, if the device type is `sbt` and parallelization is set to 4, then RMAN allocates as follows:

```
ORA_SBT_TAPE_1
ORA_SBT_TAPE_2
ORA_SBT_TAPE_3
ORA_SBT_TAPE_4
```

You can change a parallelism setting by issuing another `CONFIGURE DEVICE TYPE ... PARALLELISM` command. This example configures `PARALLELISM 2` and then changes it to 3:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
```

The parallelism setting defines the number of channels for a device that RMAN allocates in parallel. It does not have to correspond to the actual number of channels configured for the device. For example, you can manually configure four different `sbt` channels and set `PARALLELISM` for `sbt` to `2`, `1`, or `10`.

You can view the default setting for parallelism by running the `SHOW DEVICE TYPE` command. The default value is followed by a number sign (#). For example:

```
RMAN> SHOW DEVICE TYPE;
RMAN configuration parameters are:
CONFIGURE DEVICE TYPE DISK PARALLELISM 1; # default
```

The following example configures the default device to `sbt` and runs another `SHOW` command:

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO sbt;
new RMAN configuration parameters:
CONFIGURE DEFAULT DEVICE TYPE TO 'sbt';
new RMAN configuration parameters are successfully stored

RMAN> SHOW DEVICE TYPE;
RMAN configuration parameters are:
CONFIGURE DEVICE TYPE SBT PARALLELISM 1; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1; # default
```

See Also: "Configuring a Generic Automatic Channel for a Device Type" on page 8-12

Automatic Channel Default Device Types

Run the `CONFIGURE DEFAULT DEVICE TYPE` command to specify a default device type for automatic channels. For example, you may make backups to tape most of the time and only occasionally make a backup to disk. In this case, configure channels for disk and tape devices, but make the device of `sbt` the default device:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 1; # configure device disk
CONFIGURE DEVICE TYPE sbt PARALLELISM 2; # configure device sbt
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

RMAN only allocates `sbt` channels when you run backup commands. For example, you may issue the following:

```
BACKUP TABLESPACE users;
```

RMAN only allocates channels of type `sbt` during the backup because `sbt` is the default device.

You can override the default device for backups and copies by specifying a different device on the command using the channel. For example, run a backup as follows:

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

If the default device type is `DISK`, then the preceding command overrides this default and uses the `sbt` channel configuration. Note that this command fails unless you have configured the `sbt` device or configured `sbt` channels.

During a restore operation, RMAN allocates all automatic channels according to the parallelism settings configured for each device type. The default device type configuration is irrelevant. For example, if you configure `PARALLELISM` to `3` for the default `sbt` device and `PARALLELISM` to `2` for `DISK`, then RMAN automatically allocates three `sbt` channels and two `DISK` channels during the restore.

> **See Also:** "Changing the Default Device Type" on page 8-11

Automatic Channel Naming Conventions

RMAN uses the following convention for channel naming: `ORA_`*`devicetype_n`*, where *`devicetype`* refers to the user's device type (such as `DISK` or `sbt_tape`) and *`n`* refers to the channel number.

> **Note:** The `sbt` and `sbt_tape` device types are synonymous, but RMAN output always displays `sbt_tape` whether the input is `sbt` or `sbt_tape`.

For example, RMAN names the first `DISK` channel `ORA_DISK_1`, the second `ORA_DISK_2`, and so forth. RMAN names the first `sbt` channel `ORA_SBT_TAPE_1`, the second `ORA_SBT_TAPE_2`, and so forth. When you parallelize channels, RMAN always allocates channels in numerical order, starting with 1 and ending with the parallelism setting (`CONFIGURE DEVICE TYPE ... PARALLELISM` *`n`*), as in this example:

```
ORA_SBT_TAPE_1
ORA_SBT_TAPE_2
ORA_SBT_TAPE_3
```

Automatic channel allocation also applies to maintenance commands. If RMAN allocates an automatic maintenance channel, then it uses the same naming convention as any other automatically allocated channel. If you manually allocate a maintenance channel using `ALLOCATE CHANNEL FOR MAINTENANCE`, then RMAN uses the following convention for channel naming: `ORA_MAINT_`*`devicetype_n`*, where *`devicetype`* refers to the user's device type (for example, `DISK` or `sbt`) and *`n`* refers to the channel number. For example, RMAN uses these names for two manually allocated disk channels:

```
ORA_MAINT_DISK_1
ORA_MAINT_DISK_2
```

Note that if you run the `CONFIGURE DEVICE TYPE` command to configure a device type and do not run `CONFIGURE CHANNEL` for this device type, then RMAN allocates all channels without other channel control options. For example, assume that you configure the `sbt` device and run a backup as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
BACKUP DEVICE TYPE sbt DATABASE;
```

In effect, RMAN does the following:

```
RUN
{
  ALLOCATE CHANNEL ORA_SBT_TAPE_1 DEVICE TYPE sbt;
  BACKUP DATABASE;
}
```

Channel names beginning with the `ORA_` prefix are reserved by RMAN for its own use. You cannot manually allocate a channel with the `ALLOCATE CHANNEL` command and then prefix the channel name with `ORA_`.

Automatic Channel Generic Configurations

The `CONFIGURE CHANNEL DEVICE TYPE` command configures generic settings that are used for all automatic channels of the specified device type. In other words, the command creates a template of settings that RMAN uses for all channels allocated on the device. For example, you can configure disk and tape channels as follows:

```
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='ENV=(NSR_SERVER=bksvr1)';
CONFIGURE CHANNEL DEVICE TYPE DISK RATE 5M FORMAT="?/oradata/%U" MAXOPENFILES=20;
```

Because you do not specify channel numbers for these channels, the channel settings are generic to all automatic channels of the specified type. The configuration acts as a template. For example, if you set `PARALLELISM` for `DISK` to `10`, and the default device type is `DISK`, then RMAN allocates ten disk channels using the settings in the `CONFIGURE CHANNEL DEVICE TYPE DISK` command.

> **See Also:** "Configuring a Generic Automatic Channel for a Device Type" on page 8-12

Automatic Channel Specific Configurations

You can also configure parameters that apply to one specific automatic channel. If you are running in an Oracle Real Application Clusters configuration or using a media manager that requires different settings on each channel, then you may find it useful to configure individual channels.

For example, in an Oracle Real Application Clusters environment you can enter:

```
CONFIGURE DEFAULT DEVICE DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'node1' PARMS='ENV=(NSR_SERVER=bksvr1)';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'node2' PARMS='ENV=(NSR_SERVER=bksvr1)';
```

In this example, RMAN allocates 2 channels for every backup job, ORA_SBT_TAPE_1 and ORA_SBT_TAPE_2. The ORA_SBT_TAPE_1 channel uses the settings for CHANNEL 1 and the ORA_SBT_TAPE_2 channel uses the settings for CHANNEL 2.

You can mix a CONFIGURE CHANNEL command that creates a generic configuration with a CONFIGURE CHANNEL command that creates a specific configuration. A generic automatic channel simply creates a configuration that can be used for any channel that is not explicitly configured. For example, assume that you run these commands:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE = 2M;
CONFIGURE CHANNEL 3 DEVICE TYPE DISK MAXPIECESIZE = 900K;
```

In this scenario, RMAN allocates ORA_DISK_1 and ORA_DISK_2 with option MAXPIECESIZE = 2M, using the settings for the DISK channel with no number. RMAN allocates ORA_DISK_3 with MAXPIECESIZE = 900K because this channel was manually assigned a channel number. RMAN always allocates the number of channels specified in the parallelism parameter.

See Also: "Configuring a Generic Automatic Channel for a Device Type" on page 8-12

Clearing Automatic Channel Settings

You can specify the CLEAR option for any CONFIGURE command. The CLEAR option returns the specified configuration to its default value. Assume you run these commands:

```
CONFIGURE DEVICE TYPE DISK CLEAR;           # returns DISK to default parallelism
CONFIGURE DEFAULT DEVICE TYPE CLEAR;        # returns to default device type of DISK
CONFIGURE CHANNEL DEVICE TYPE sbt CLEAR;    # removes all the options for sbt channel
CONFIGURE CHANNEL 3 DEVICE TYPE DISK CLEAR; # removes all configurations for 3rd channel
```

Each CONFIGURE command removes the user-entered settings and returns the configuration to its default value. To see the default configuration settings, run the SHOW ALL command before entering any configurations of your own. The defaults are prefixed with a number sign (#).

See Also: *Oracle9i Recovery Manager Reference* for the default settings for each CONFIGURE command, and "Showing All RMAN Configuration Settings" on page 17-16

Parallelization for Manually Allocated Channels

If you do not want to use automatic channels, then you can allocate multiple channels manually within a RUN command, thus allowing a single RMAN command to read or write multiple backups or image copies in parallel. Thus, the number of channels that you allocate affects the degree of parallelism within a command. When backing up to tape you should allocate one channel for each physical device, but when backing up to disk you can allocate as many channels as necessary for maximum throughput.

Each manually allocated channel uses a separate connection to the target or auxiliary database. You can specify a different CONNECT string for each channel to connect to different instances of the target database, which is useful in an Oracle Real Application Clusters configuration for distributing work across nodes.

RMAN internally handles parallelization of BACKUP, COPY, and RESTORE commands. You only need to specify:

- Multiple ALLOCATE CHANNEL commands
- The objects that you want to back up, copy, or restore

RMAN executes commands sequentially; that is, it completes the current command before starting the next one. Parallelism is exploited only within the context of a single command. Consequently, to create three backups of a datafile, issue a single BACKUP command specifying all three datafiles rather than three separate BACKUP commands.

The following script uses serialization to create the backups: three separate BACKUP commands are used to back up one file each. Only one channel is active at any one time because only one file is being backed up.

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt;
  ALLOCATE CHANNEL c3 DEVICE TYPE sbt;
  BACKUP DATAFILE 5;
  BACKUP DATAFILE 6;
  BACKUP DATAFILE 7;
}
```

The following statement uses **parallelization** on the same example: one RMAN BACKUP command backs up three datafiles, with all three channels in use. The three channels are **concurrently active**—each server session copies one of the datafiles to a separate tape drive.

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt;
  ALLOCATE CHANNEL c3 DEVICE TYPE sbt;
  BACKUP DATAFILE 5,6,7;
}
```

> **See Also:** *Oracle9i Real Application Clusters Concepts* for information about parallelization in an Oracle Real Application Clusters configuration

Channel Control Options for Manual and Automatic Channels

Whether you allocate channels manually or automatically, you can use channel control commands and options to do the following:

- Control the operating system resources RMAN uses when performing RMAN operations
- Affect the degree of parallelism for a backup or restore (in conjunction with the `FILESPERSET` parameter of the `BACKUP` command)
- Set limits on I/O bandwidth consumption in kilobytes, megabytes, or gigabytes (`ALLOCATE CHANNEL ... RATE`, `CONFIGURE CHANNEL ... RATE`)
- Set limits on the size of backup pieces (the `MAXPIECESIZE` parameter specified on the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands)
- Set limits on the size of backup sets (the `MAXSETSIZE` parameter specified on the `BACKUP` and `CONFIGURE` commands)
- Set limits on the number of concurrently open files (`ALLOCATE CHANNEL ... MAXOPENFILES`, `CONFIGURE CHANNEL ... MAXOPENFILES`)
- Send vendor-specific commands to the media manager (`SEND`)
- Specify vendor-specific parameters for the media manager (`ALLOCATE CHANNEL ... PARMS`, `CONFIGURE CHANNEL ... PARMS`)
- Specify which instance performs the operation (`ALLOCATE CHANNEL ... CONNECT`, `CONFIGURE CHANNEL ... CONNECT`)

On some platforms, the channel allocation and channel control commands specify the name or type of an I/O device to use. On other platforms, they specify which operating system access method or I/O driver to use. Not all platforms support the selection of I/O devices through this interface; on some platforms, I/O device selection is controlled through platform-specific mechanisms.

In Oracle9*i*, the `ALLOCATE CHANNEL` command causes RMAN to contact the media manager whenever the type specified is other than `DISK`. In releases before Oracle9*i*, the `ALLOCATE CHANNEL` command does not cause RMAN to contact the media manager; RMAN does not call the media manager unless a `BACKUP`, `RESTORE`, or `RECOVER` command is issued.

Note: When you specify `DEVICE TYPE DISK` with any version of RMAN, RMAN does not allocate operating system resources other than for the creation of the server session and does not call the media manager.

Because RMAN has a preconfigured automatic `DISK` channel, you do not have to manually allocate a maintenance channel when running `CHANGE ... AVAILABLE`, `CROSSCHECK`, or `DELETE` against a file that is only on disk (that is, an `ARCHIVELOG`, `DATAFILECOPY`, or `CONTROLFILECOPY`). A maintenance channel is useful only for a maintenance task; you cannot use it as an input or output channel for a backup or restore job.

See Also: *Oracle9i Recovery Manager Reference* for `ALLOCATE CHANNEL` syntax, and *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax

Hardware Multiplexing by the Media Manager

Hardware multiplexing occurs when the media manager writes multiple RMAN backups to a single sequential device (such as a tape drive). Oracle Corporation does not recommend hardware multiplexing of RMAN backups.

Backup Sets

When you execute the `BACKUP` command, you create one or more backup sets. This section contains these topics:

- About Backup Sets
- About Proxy Copies
- Storage of Backup Sets
- Backup Set Compression
- Backups of Archived Logs

- Multiplexed Backup Sets
- Duplexed Backup Sets
- Parallelization of Backups
- Backups of Backup Sets

See Also: Chapter 9, "Making Backups and Copies with Recovery Manager" to learn how to make backups, and *Oracle9i Recovery Manager Reference* for information on the BACKUP command

About Backup Sets

A **backup set**, which is a logical object, contains one or more physical **backup pieces**. By default, one backup set contains one backup piece. Backup pieces are operating system files that contain the backed up datafiles, control files, or archived redo logs. For example, you can back up ten datafiles into a single backup set containing a single backup piece (that is, a single output file). You cannot split a file across different backup sets or mix archived logs and datafiles into one backup set.

A backup set is a complete set of backup pieces that make up a full or incremental backup of the objects specified in the BACKUP command. Backup sets are in an RMAN-specific format. An **image copy**, which is a complete image of a single datafile, control file, or archived log, is not in an RMAN-specific format.

You can back up datafiles, control files, archived redo logs, and the current server parameter file. You can also back up another backup set, as when you want to back up a disk backup to tape, or an image copy. For example, you can issue commands such as the following, each of which uses an automatic channel configuration:

```
BACKUP DATABASE;
BACKUP TABLESPACE users, tools;
BACKUP (SPFILE) (CURRENT CONTROL FILE);
BACKUP BACKUPSET 12;
BACKUP DATAFILECOPY '/tmp/system01.dbf';
```

When backing up datafiles, the target database must be mounted or open. If the database is in ARCHIVELOG mode, then the target can be open or closed: you do not need to close the database cleanly. If the database is in NOARCHIVELOG mode, then you *must* close it cleanly before making a backup.

Note: You cannot make a backup of a transported tablespace until after it has been specified read / write.

See Also: "Automatic Channel Generic Configurations" on page 5-8 to learn about automatic channels, and "Backing Up Database Files and Archived Logs with RMAN" on page 9-3 to learn how make backups

About Proxy Copies

A **proxy copy** is a special type of backup in which RMAN turns over control of the data transfer to a media manager that supports this feature. 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 using 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 uses conventional backup sets to perform the backup. An exception occurs when you use the PROXY ONLY option, which causes Oracle to issue an error message when it cannot proxy copy.

Oracle records each proxy-copied file in the control file. RMAN uses this data to resynchronize the recovery catalog. Use the V$PROXY_DATAFILE view to obtain the proxy copy information. Use the CHANGE PROXY command or DELETE PROXY command to change the status of or delete a proxy backup.

Note: The proxy functionality was introduced in Oracle8*i* Release 1 (8.1.5). If a proxy version of RMAN is used with a non-proxy target database, RMAN will *not* use proxy copy to create backup sets. If you make backups using proxy copy and then downgrade Oracle to a non-proxy version, RMAN will not use proxy copy backups when restoring and will issue a warning when the best available file is a proxy copy.

See Also: *Oracle9i Database Reference* for more information about V$PROXY_DATAFILE and V$PROXY_ARCHIVEDLOG

Storage of Backup Sets

RMAN can create backup sets that are written to disk or tertiary storage. If you specify DEVICE TYPE DISK, then you must back up to random access disks. You can make a backup on any device that can store an Oracle datafile: in other words, if the statement CREATE TABLESPACE *tablespace_name* DATAFILE '*filename*' works, then '*filename*' is also a valid backup path name.

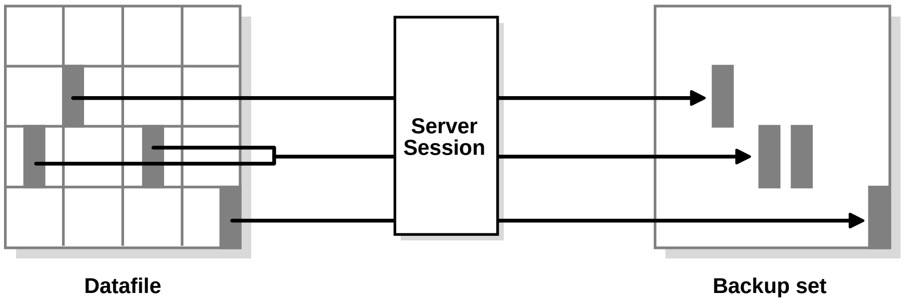
Using a media management system that is available and supported on your operating system, you can write backup sets to sequential output media such as magnetic tape. If you specify a device type such as `sbt` that is other than `DISK`, then you can back up to any media supported by the media management software.

> **See Also:** "Backup Retention Policies" on page 5-49 to learn how to configure a policy that determines which backups are obsolete

Backup Set Compression

As Figure 5–2 illustrates, RMAN performs **compression** on its backups, which means that datafile blocks that have never been used are not backed up. Image copies of a datafile, however, always contain all datafile blocks.

Figure 5–2 Backup Set Compression



> **See Also:** "Disk Buffer Allocation" on page 14-3 and "Tape Buffer Allocation" on page 14-5 to learn how RMAN buffers its backups

Backups of Archived Logs

RMAN provides additional functionality for backups of archived redo logs. This section contains these topics:

- Deletion of Archived Logs After Backups
- Backup Failover for Archived Redo Logs
- Automatic Online Redo Log Switches During Backups of Archived Logs
- Backups of Archived Logs Needing Backups

Deletion of Archived Logs After Backups

Although you cannot archive logs directly to tape, you can use RMAN to back up archived logs from disk to tape. If you specify the `DELETE INPUT` option, then RMAN backs up exactly one copy of each specified log sequence number and then deletes the copy from disk after backing it up.

If you specify the `DELETE ALL INPUT` option, then RMAN backs up exactly one copy of each specified log sequence number and then deletes the copies that match the specified criteria. For example, if you specify the `LIKE` parameter, RMAN deletes whichever logs match the string. RMAN automatically restores needed archived logs from tape to disk during recovery.

Backup Failover for Archived Redo Logs

In releases prior to Oracle9*i*, RMAN only looked in the first archiving destination for archived redo logs when backing them up. In Oracle9*i*, RMAN can perform **archived redo log failover**. RMAN can do the following:

- If at least one log corresponding to a given log sequence and thread is available in any of the archiving destinations, then RMAN backs up the available log.
- If there is a corrupt block in a log that RMAN is accessing, then RMAN searches other directories for a file without corrupt blocks.

RMAN always only backs up one copy of each distinct log sequence number. For example, assume that you archive logs 121 to 124 to two archiving destinations: `/arch1` and `/arch2`. The control file contains archived log records as follows:

Sequence	Filename
121	`/arch1/archive1_121.arc`
121	`/arch2/archive1_121.arc`
122	`/arch1/archive1_122.arc`
122	`/arch2/archive1_122.arc`
123	`/arch1/archive1_123.arc`
123	`/arch2/archive1_123.arc`
124	`/arch1/archive1_124.arc`
124	`/arch2/archive1_124.arc`

However, unknown to RMAN, someone accidentally deletes logs 122 and 124 from the `/arch1` directory. Then, you run the following backup:

```
BACKUP ARCHIVELOG FROM SEQUENCE 121 UNTIL SEQUENCE 125;
```

In this case, RMAN can use the failover feature to back up the archived logs as described in the following table.

Sq.	If RMAN Searches For . . .	Then RMAN Can Find and Back Up . . .
121	`/arch1/archive1_121.arc`	`/arch1/archive1_121.arc`
122	`/arch1/archive1_122.arc` (missing)	`/arch2/archive1_122.arc`
123	`/arch1/archive1_123.arc`	`/arch1/archive1_123.arc`
124	`/arch1/archive1_124.arc` (missing)	`/arch2/archive1_124.arc`

Automatic Online Redo Log Switches During Backups of Archived Logs

At the beginning of every `BACKUP ... ARCHIVELOG` command that does not specify an `UNTIL` clause, RMAN attempts to automatically switch out of and archive the current online redo log. In this way, RMAN can include the current redo log in the backup set.

If the database is open, then at the start of an archived log backup RMAN tries to switch out of and archive the current online log according to these rules:

- RMAN runs `ALTER SYSTEM ARCHIVE LOG CURRENT`.
- If the `UNTIL` clause or `SEQUENCE` parameter is specified, RMAN does not try to switch or archive online logs.

Backups of Archived Logs Needing Backups

When making backups, run the `BACKUP ... PLUS ARCHIVELOG` command to archive online logs as well as back up archived logs. The purpose of this feature is to guarantee that the backed up datafiles can be recovered to a consistent state.

When `PLUS ARCHIVELOG` is specified, RMAN performs the following actions in sequential order:

1. Runs `ALTER SYSTEM ARCHIVE LOG CURRENT`.
2. Runs `BACKUP ARCHIVELOG ALL`.
3. Backs up the files specified in the `BACKUP` command.
4. Runs `ALTER SYSTEM ARCHIVE LOG CURRENT`.

5. Backs up any remaining archived redo logs generated during backup.

You cannot specify `PLUS ARCHIVELOG` when explicitly backing up archived logs, but only when backing up another object such as the database. For example, you can run this command:

```
BACKUP DEVICE TYPE sbt DATABASE PLUS ARCHIVELOG;
```

Note: If backup optimization is enabled, then RMAN skips backups of archived logs that have already been backed up to the allocated device.

See Also:

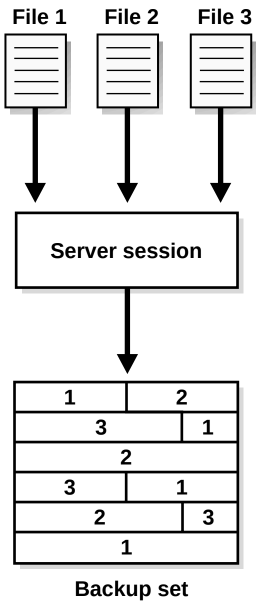
- "Backing Up Logs Using BACKUP ... PLUS ARCHIVELOG" on page 9-13 to learn how to use `PLUS ARCHIVELOG`
- "Backup Retention Policies" on page 5-49
- "Backup Optimization" on page 5-56

Multiplexed Backup Sets

The technique of RMAN **multiplexing** is to simultaneously read files on disks and and then write them into the same backup piece. For example, RMAN can read from two datafiles simultaneously, and then combine the blocks from these datafiles into a single backup piece. Note that multiplexing in this sense is completely different from **duplexing**.

As Figure 5–3 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 files.

Figure 5–3 Datafile Multiplexing



Multiplexing is affected by the factors described in Table 5–1.

Table 5–1 RMAN Multiplexing

Concepts	Definition
Number of files in each backup set	The minimum of these values: FILESPERSET setting, and the number of files read by each channel.
Multiplexing	The minimum of these values: MAXOPENFILES setting, and the number of files in each backup set.

The FILESPERSET parameter determines how many datafiles should be included in each backup set, while MAXOPENFILES defines how many datafiles RMAN can read from simultaneously.

Assume that you are backing up six datafiles with one RMAN channel. If FILESPERSET is 6 and MAXOPENFILES is 1, then the channel includes 6 datafiles in a set but does not multiplex the files because RMAN is not reading from more than one file simultaneously. The channel reads one file at a time and writes to the backup piece. In this case, the degree of multiplexing is 1.

Now, assume that FILESPERSET is 6 and MAXOPENFILES is 3. In this case, the channel can read and write in the following order:

1. Read from datafiles 1, 2, and 3 simultaneously and write to the backup piece
2. Read from datafiles 4, 5 and 6 simultaneously and write to the backup piece

So in this example, the degree of multiplexing is 3 (the lesser of 6 and 3).

When multiplexing files, you can do the following:

- Partition the datafiles into backup sets explicitly by assigning channels to specific datafiles, or let RMAN automatically select a partitioning.
- Keep a high performance sequential output device streaming by including a sufficient number of datafiles in the backup. Keeping the device streaming is important for open database backups in which the backup operation must compete with the online system for I/O bandwidth.
- Include the control file in a datafile backup set. In this case, the control file is written first and its blocks are not multiplexed with datafile blocks.
- Create a backup set containing either datafiles or archived logs, but not both together. You cannot write datafiles and archived logs to the same backup set because the Oracle logical block size of the objects in a multiplexed backup must be the same.

Note that multiplexing too many files can decrease restore performance. If possible, group files that will be restored together into the same backup set. Assume that RMAN backs up seventeen files with FILESPERSET = 64 and MAXOPENFILES = 16. You decide to restore data17.f, which is datafile 17. So, RMAN reads the multiplexed data for the first sixteen files and then starts reading the data for data17.f. In this case, moving to the beginning of the backup of data17.f may take more time than the restore itself.

See Also:

- "Disk Buffer Allocation" on page 14-3 to learn how multiplexing affects allocation of disk buffers during backups
- "Multiplexing Datafiles in a Backup: Example" on page 9-39 to learn how to multiplex backups
- *Oracle9i Recovery Manager Reference* for BACKUP syntax

Duplexed Backup Sets

RMAN provides an efficient way to produce multiple copies of each backup piece in a backup set. This functionality is also known as **duplexing** a backup set.

You can create up to four identical copies of each piece in a backup set by issuing one of the following commands, listed in order of precedence. If multiple commands are in effect simultaneously, then commands higher on the list override commands that are lower on the list.

Command	Command Restriction	Object Restriction
BACKUP COPIES	Only BACKUP command on which COPIES is specified	Any object specified on BACKUP command
SET BACKUP COPIES	All BACKUP commands within RUN block	Any object specified on BACKUP command
CONFIGURE ... BACKUP COPIES	All BACKUP commands	Only datafiles, control files, server parameter files, and archived logs

Note: Control file autobackups on disk are a special case and are never duplexed: RMAN always writes one and only one copy.

You can specify up to 4 values for the FORMAT option. RMAN uses the second, third, and fourth values only when BACKUP COPIES, SET BACKUP COPIES, or CONFIGURE ... BACKUP COPIES is specified. This example creates 3 copies of the backup of datafile 7:

```
BACKUP DEVICE TYPE DISK COPIES 3 DATAFILE 7 FORMAT '/tmp/%U','?/oradata/%U','?/%U';
```

RMAN places the first copy of each backup piece in /tmp, the second in ?/oradata, and the third in the Oracle home. Note that RMAN does not produce 3 backup sets, each with a different unique backup set key. Rather, RMAN produces one backup set with a unique key, and generates 3 identical copies of each backup piece in the set, as shown in this sample LIST output:

```
List of Backup Sets
===================

BS Key  Type LV Size
------- ---- -- ----------
1       Full    64K
  List of Datafiles in backup set 1
```

```
File LV Type Ckp SCN    Ckp Time  Name
---- -- ---- ---------- --------- ----
7       Full 98410      08-FEB-02 /oracle/oradata/trgt/tools01.dbf

Backup Set Copy #1 of backup set 1
Device Type Elapsed Time Completion Time Tag
----------- ------------ --------------- ---
DISK        00:00:01     08-FEB-02       TAG20020208T152314

  List of Backup Pieces for backup set 1 Copy #1
  BP Key  Pc# Status      Piece Name
  ------- --- ----------- ----------
  1       1   AVAILABLE   /tmp/01dg9tb2_1_1

Backup Set Copy #2 of backup set 1
Device Type Elapsed Time Completion Time Tag
----------- ------------ --------------- ---
DISK        00:00:01     08-FEB-02       TAG20020208T152314

  List of Backup Pieces for backup set 1 Copy #2
  BP Key  Pc# Status      Piece Name
  ------- --- ----------- ----------
  2       1   AVAILABLE   /oracle/oradata/01dg9tb2_1_2

Backup Set Copy #3 of backup set 1
Device Type Elapsed Time Completion Time Tag
----------- ------------ --------------- ---
DISK        00:00:01     08-FEB-02       TAG20020208T152314

  List of Backup Pieces for backup set 1 Copy #3
  BP Key  Pc# Status      Piece Name
  ------- --- ----------- ----------
  3       1   AVAILABLE   /oracle/01dg9tb2_1_3
```

When choosing which `FORMAT` value to use for each backup piece, RMAN uses the first format value for copy number 1, the second format value for copy number 2, and so forth. If the number of format values exceeds the number of copies, then the extra formats are not used. If the number of format values is less than the number of copies, then RMAN reuses the format values, starting with the first one.

See Also:

- "Duplexing Backups" on page 9-15 to learn how to duplex backups
- *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax
- *Oracle9i Recovery Manager Reference* for `SET` syntax

Parallelization of Backups

When you configure PARALLELISM to greater than 1 or manually allocate multiple channels, RMAN writes multiple backup sets in parallel. The server sessions divide the work of backing up the specified files.

Note: You cannot stripe a backup set across multiple channels.

By default, RMAN determines which channels should back up which database files. You can use the BACKUP ... CHANNEL command to manually assign a channel to back up specified files. You can also use the FILESPERSET parameter to limit the number of datafiles included in a backup set. This example shows a parallelized backup to disk that uses the default automatic DISK channels:

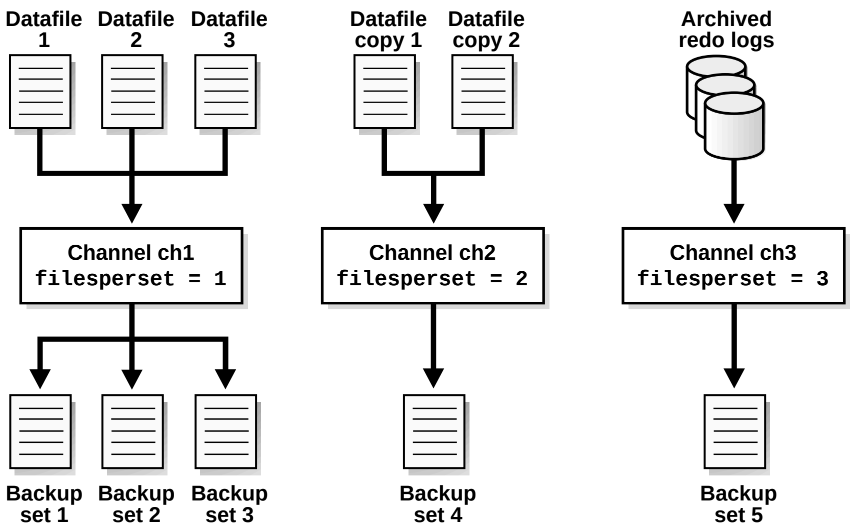
```
BACKUP
  (DATAFILE 1,2,3
   FILESPERSET = 1
   CHANNEL ORA_DISK_1)
  (DATAFILECOPY '/tmp/system01.dbf', '/tmp/tools01.dbf'
   FILESPERSET = 2
   CHANNEL ORA_DISK_2)
  (ARCHIVELOG FROM SEQUENCE 100 UNTIL SEQUENCE 102 THREAD 1
   FILESPERSET = 3
   CHANNEL ORA_DISK_3);
```

You can also manually allocate channels as in the following example:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS="ENV=(BACKUP_SERVER=tape_server1)";
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt PARMS="ENV=(BACKUP_SERVER=tape_server2)";
  ALLOCATE CHANNEL c3 DEVICE TYPE sbt PARMS="ENV=(BACKUP_SERVER=tape_server3)";
  BACKUP
   (DATAFILE 1,2,3
    FILESPERSET = 1
    CHANNEL c1)
   (DATAFILECOPY '/tmp/system01.dbf', '/tmp/tools01.dbf'
     FILESPERSET = 2
     CHANNEL c2)
   (ARCHIVELOG FROM SEQUENCE 100 UNTIL SEQUENCE 102 THREAD 1
     FILESPERSET = 3
     CHANNEL c3);
}
```

Figure 5–4 shows an example of parallelization in which channel ch1 backs up datafiles, channel ch2 backs up datafile copies, and channel ch3 backs up logs.

Figure 5–4 Parallelization of Backups



See Also:

- "Parallelization for Manually Allocated Channels" on page 5-10 for an overview of how allocated channels affect parallelization
- "Determining How Channels Distribute a Backup Workload: Example" on page 9-43 to learn how to parallelize backups
- *Oracle9i Recovery Manager Reference* for reference material on the CHANNEL parameter of the BACKUP command

Backups of Backup Sets

The RMAN BACKUP BACKUPSET command backs up backup sets rather than actual database files. This command supports disk-to-disk or disk-to-tape backups, but not tape-to-tape backups.

The BACKUP BACKUPSET command uses the default disk channel to copy backup sets from disk to disk. To back up from disk to tape, you must either manually allocate a channel of DEVICE TYPE sbt or configure an automatic sbt channel.

Uses for Backups of Backup Sets

The BACKUP BACKUPSET command is a useful way to spread backups among multiple media. For example, you can execute the following BACKUP command weekly as part of the production backup schedule:

```
BACKUP DEVICE TYPE sbt BACKUPSET ALL;
```

Note: Backups to the sbt device that use automatic channels require that you first run the CONFIGURE DEVICE TYPE sbt command.

In this way, you ensure that all your backups exist on both disk and tape. Note that you can also duplex backups of backup sets (except for control file autobackups, which are never duplexed by BACKUP BACKUPSET), as in this example:

```
BACKUP COPIES 2 DEVICE TYPE sbt BACKUPSET ALL;
```

You can also use BACKUP BACKUPSET to manage backup space allocation. For example, assume that you want more recent backups to exist on disk and older backups to exist on tape, but you do not need backups to exist on both disk and tape at the same time. In this case, you can regularly run the following command:

```
BACKUP BACKUPSET COMPLETED BEFORE 'SYSDATE-7' DELETE INPUT;
```

This command backs up backup sets that were created more than a week ago from disk to tape, and then deletes the backup sets from disk. 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.

Backup Optimization When Backing Up Backup Sets

Note that if **backup optimization** is enabled when you issue the command to back up a backup set, and the identical backup set has already been backed up to the same device type, then RMAN skips the backup of that backup set. For example, the following command backs up to tape all backup sets that do not already exist on device type sbt:

```
BACKUP DEVICE TYPE sbt BACKUPSET ALL;
```

Backup Failover When Backing Up Backup Sets

When backing up backup sets, RMAN searches for all available backup copies when the copy that it is trying to back up is either corrupted or missing. This behavior is similar to the behavior of RMAN when backing up archived redo logs that exist in multiple archiving destinations.

For example, assume that backup set 123 contains three backup pieces, and that `BACKUP COPIES 3` was issued so that three copies of each backup piece exist. Each copy of a backup piece is on a different file system. The following table indicates that some of the copies of the pieces are corrupted or missing, while others are intact.

Backup Piece Number	Copy Number of the Piece	Status of Copy
1	1	Corrupted
1	2	Intact
1	3	Corrupted
2	1	Missing
2	2	Corrupted
2	3	Intact
3	1	Intact
3	2	Corrupted
3	3	Missing

Assume that you run the following command:

```
BACKUP BACKUPSET 123;
```

RMAN performs an automatic failover and includes only the uncorrupted copies in its backup set. The following table indicates which copies RMAN includes.

Backup Piece Number	Copy Number of the Piece	Status of Copy
1	2	Intact
2	3	Intact
3	1	Intact

See Also: "Duplexed Backup Sets" on page 5-21, and "Backup Optimization" on page 5-56

Backup Options: Naming, Sizing, and Speed

Recovery Manager provides a number of options to control backups.

This section contains these topics:

- Filenames for Backup Pieces
- Size of Backup Pieces
- Number and Size of Backup Sets: Basic Algorithm
- Number and Size of Backup Sets: Advanced Algorithm
- I/O Read Rate of Backups

Filenames for Backup Pieces

You can either let RMAN determine a unique name for backup pieces or use the `FORMAT` parameter to specify a name. If you do not specify a filename, then RMAN uses the `%U` substitution variable to generate a unique name. For example, enter:

```
BACKUP TABLESPACE users;
```

RMAN automatically generates unique names for the backup pieces.

The `FORMAT` parameter provides substitution variables that you can use to generate unique filenames. For example, you can run a command as follows:

```
BACKUP TABLESPACE users FORMAT = '/tmp/users_%u%p%c';
```

As described in "Duplexed Backup Sets" on page 5-21, you can specify up to four `FORMAT` values. RMAN uses the second, third, and fourth values only when you run `BACKUP COPIES`, `SET BACKUP COPIES`, or `CONFIGURE ... BACKUP COPIES`.

Note: If you use a media manager, check your vendor documentation for restrictions on `FORMAT` such as the size of the name, the naming conventions, and so forth.

See Also: *Oracle9i Recovery Manager Reference* for descriptions of the `FORMAT` parameter and the substitution variables

Size of Backup Pieces

Each backup set contains at least one backup piece. If you do not restrict the backup piece size, then every backup set contains only 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.

For example, restrict the backup piece size for disk backups to 2 GB by configuring an automatic disk channel, and then run a backup as follows:

```
CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE = 2G;
BACKUP DATAFILE 1;
```

A LIST BACKUP command reveals that RMAN created five backup pieces rather than one backup piece to conform to the MAXPIECESIZE size restriction:

```
BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ --------------------
29      Full    9728M      DISK        00:00:35     NOV 02 2001 18:29:26
  List of Datafiles in backup set 29
  File LV Type Ckp SCN    Ckp Time             Name
  ---- -- ---- ---------- -------------------- ----
  1       Full 177590     NOV 02 2001 18:28:51 /oracle/oradata/trgt/system01.dbf

  Backup Set Copy #1 of backup set 29
  Device Type Elapsed Time Completion Time      Tag
  ----------- ------------ -------------------- ---
  DISK        00:00:35     NOV 02 2001 18:29:26 TAG20011102T152701

    List of Backup Pieces for backup set 29 Copy #1
    BP Key  Pc# Status      Piece Name
    ------- --- ----------- ----------
    53      1   AVAILABLE   /oracle/dbs/10d85733_1_1
    54      2   AVAILABLE   /oracle/dbs/10d85733_2_1
    55      3   AVAILABLE   /oracle/dbs/10d85733_3_1
    56      4   AVAILABLE   /oracle/dbs/10d85733_4_1
    57      5   AVAILABLE   /oracle/dbs/10d85733_5_1
```

See Also:

- *Oracle9i Recovery Manager Reference* for ALLOCATE CHANNEL syntax
- *Oracle9i Recovery Manager Reference* for CONFIGURE syntax

Number and Size of Backup Sets: Basic Algorithm

Use the *backupSpec* clause of the BACKUP command to specify the objects that you want to back up as well as specify other options. Each *backupSpec* clause produces at least one backup set. The total number and size of backup sets depends on which algorithm RMAN uses: the **basic algorithm** or the **advanced algorithm**.

RMAN uses the basic algorithm when *any* of the following conditions is true:

- You are backing up files other than datafiles or control files
- The operating system does not provide data about which files are located on which disks, or which nodes in a cluster have affinity with which disks
- You manually set the DISKRATIO parameter of the BACKUP command to 0

If *all* of these conditions are false, then RMAN uses the advanced algorithm described in "Number and Size of Backup Sets: Advanced Algorithm" on page 5-34. Note that you can always force RMAN to use the basic algorithm by setting DISKRATIO=0.

Factors Affecting the Number and Size of Backup Sets

In determining the characteristics of the RMAN backup sets, the basic algorithm is influenced by the following factors:

- The number of *backupSpec* clauses that you specify
- The number of input files specified or implied in each *backupSpec* clause
- The number of channels that you allocate
- The FILESPERSET parameter, which limits the number of files for a backup set
- The MAXSETSIZE parameter (specified on the CONFIGURE and BACKUP commands), which specifies a maximum backup set size

The most important rules in the algorithm for backup set creation are:

- Each allocated channel that performs work in the backup job—that is, each channel that is not idle—generates at least one backup set. By default, this backup set contains one backup piece.

Note: RMAN writes backup pieces sequentially; striping a backup piece across multiple output devices is not supported. For example, RMAN does not simultaneously write half of a backup piece to one disk and the other half to another disk.

- RMAN always tries to divide the backup load so that all allocated channels have roughly the same amount of work to do.
- The maximum number of datafiles in each backup set is determined by the FILESPERSET parameter of the BACKUP command.
- The number of datafiles multiplexed in a backup set is limited by the lesser of FILESPERSET and MAXOPENFILES.
- The maximum size of a backup set is determined by the MAXSETSIZE parameter of the CONFIGURE or BACKUP command.

Using the FILESPERSET Parameter

The FILESPERSET parameter limits the number of files (control files, datafiles, or archived logs) that can go in a backup set. The default value is calculated by RMAN as follows: RMAN compares the value 64 to the rounded-up ratio of number of files divided by the number of channels, and sets FILESPERSET to the lower value. For example, if you back up 70 files with one channel, then RMAN takes 70 divided by 1, compares 70 to 64, and sets FILESPERSET to 64 because it is lower.

The number of backup sets produced by RMAN is the rounded-up ratio of number of datafiles divided by FILESPERSET. For example, if you back up 70 datafiles and FILESPERSET is 64, then RMAN produces 2 backup sets.

RMAN tries to make backup sets roughly the same size as the ratio of total number of blocks divided by total number of sets. The total number of blocks in a backup is equal to the number of blocks in each file that is backed up. For example, if you back up 70 files that each contain 50 blocks, and the number of sets is 2, then RMAN attempts to make each backup set about 3500/2 = 1750 blocks.

Using the Default Value for FILESPERSET: Example Assume the following example in which RMAN backs up eight files with three channels. Because FILESPERSET is not specified, RMAN compares 64 to 3 (8 divided by 3 and rounded up) and sets FILESPERSET to 3. RMAN creates three backup sets and groups the files into the sets so that each set is approximately the same size. An example of the RMAN LIST output follows:

```
List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
6       Full    320K       DISK        00:00:05     08-FEB-02
        BP Key: 17   Status: AVAILABLE   Tag: TAG20020208T153359
        Piece Name: /oracle/dbs/06dg9tv9_1_1
```

```
  List of Datafiles in backup set 6
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  3       Full 98410      08-FEB-02 /oracle/oradata/trgt/cwmlite01.dbf
  4       Full 98410      08-FEB-02 /oracle/oradata/trgt/drsys01.dbf
  6       Full 98410      08-FEB-02 /oracle/oradata/trgt/indx01.dbf
  8       Full 98410      08-FEB-02 /oracle/oradata/trgt/users01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
7       Full    312K       DISK        00:00:07     08-FEB-02
        BP Key: 18   Status: AVAILABLE   Tag: TAG20020208T153359
        Piece Name: /oracle/dbs/07dg9tv9_1_1
  List of Datafiles in backup set 7
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  2       Full 98410      08-FEB-02 /oracle/oradata/trgt/undotbs01.dbf
  5       Full 98410      08-FEB-02 /oracle/oradata/trgt/example01.dbf
  7       Full 98410      08-FEB-02 /oracle/oradata/trgt/tools01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
8       Full    196M       DISK        00:00:36     08-FEB-02
        BP Key: 19   Status: AVAILABLE   Tag: TAG20020208T153359
        Piece Name: /oracle/dbs/08dg9tv9_1_1
  SPFILE Included: Modification time: 08-FEB-02
  List of Datafiles in backup set 8
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 98410      08-FEB-02 /oracle/oradata/trgt/system01.dbf
```

Specifying FILESPERSET: Example If you back up eight datafiles with three channels and specify `FILESPERSET = 2`, then RMAN places no more than two datafiles in each backup set. Consequently, RMAN creates at least four (8 divided by 2) backup sets: it may create more depending on the sizes of the datafiles as well as other factors. An example of the RMAN `LIST` output follows:

```
RMAN> list backup;


List of Backup Sets
===================

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
9       Full    144K       DISK        00:00:03     08-FEB-02
        BP Key: 20   Status: AVAILABLE   Tag: TAG20020208T153550
        Piece Name: /oracle/dbs/0adg9u2n_1_1
  List of Datafiles in backup set 9
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  4       Full 98410      08-FEB-02 /oracle/oradata/trgt/drsys01.dbf
  7       Full 98410      08-FEB-02 /oracle/oradata/trgt/tools01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
10      Full    144K       DISK        00:00:04     08-FEB-02
        BP Key: 21   Status: AVAILABLE   Tag: TAG20020208T153550
        Piece Name: /oracle/dbs/0bdg9u2n_1_1
  List of Datafiles in backup set 10
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  5       Full 98410      08-FEB-02 /oracle/oradata/trgt/example01.dbf
  8       Full 98410      08-FEB-02 /oracle/oradata/trgt/users01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
11      Full    144K       DISK        00:00:06     08-FEB-02
        BP Key: 22   Status: AVAILABLE   Tag: TAG20020208T153550
        Piece Name: /oracle/dbs/09dg9u2n_1_1
  List of Datafiles in backup set 11
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  3       Full 98410      08-FEB-02 /oracle/oradata/trgt/cwmlite01.dbf
  6       Full 98410      08-FEB-02 /oracle/oradata/trgt/indx01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
12      Full    152K       DISK        00:00:02     08-FEB-02
        BP Key: 23   Status: AVAILABLE   Tag: TAG20020208T153550
        Piece Name: /oracle/dbs/0cdg9u2r_1_1
  List of Datafiles in backup set 12
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  2       Full 98410      08-FEB-02 /oracle/oradata/trgt/undotbs01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
13      Full    196M       DISK        00:00:31     08-FEB-02
        BP Key: 24   Status: AVAILABLE   Tag: TAG20020208T153550
        Piece Name: /oracle/dbs/0ddg9u2t_1_1
  SPFILE Included: Modification time: 08-FEB-02
  List of Datafiles in backup set 13
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 98410      08-FEB-02 /oracle/oradata/trgt/system01.dbf
```

For datafile or datafile copy backups, group multiple datafiles into a single backup set to the extent necessary to keep an output tape device streaming, or to prevent the backup from consuming too much bandwidth from a particular datafile.

The fewer the files contained in a backup set, the faster one of them can be restored, because there is less data belonging to other datafiles that must be skipped. For backup sets containing archived logs, group logs from the same time period into a backup set because they will probably need to be restored at the same time.

> **See Also:** *Oracle9i Recovery Manager Reference* to learn the syntax for the *`backupSpec`* clause, and Chapter 14, "Tuning Recovery Manager" to learn about RMAN buffer management

Using the MAXSETSIZE Parameter

Each channel produces at least one backup set. To specify the maximum size of the backup set, use the `MAXSETSIZE` parameter in the `CONFIGURE` or `BACKUP` command. By limiting the overall 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.

You can use the `MAXSETSIZE` parameter to restrict the total size of a backup set to the maximum file size supported by your media manager or operating system. In this way, you do not cause either disk backups to fail because of space restrictions or tape backups to span multiple volumes.

Relationship Between MAXSETSIZE and FILESPERSET The following table compares the `MAXSETSIZE` parameter to the `FILESPERSET` parameter.

Parameter	Meaning	You decide ...	RMAN decides ...
`MAXSETSIZE`	Sets the maximum size in bytes of the backup set without specifying a limit to the number of files in the set.	The maximum size of the backup set.	How many files to put in each set to keep the parameter restriction.
`FILESPERSET`	Sets a limit to the number of files in the backup set without specifying a maximum size in bytes of the set.	The maximum number of files to include in the backup set.	What size to make the backup sets to keep the parameter restriction.

Because `FILESPERSET` has a default value, if you set `MAXSETSIZE` then you must also account for the behavior of `FILESPERSET`. When both parameters are in use:

- The number of backup sets is the greater of the following ratios:
  - Total number of blocks divided by`MAXSETSIZE`
  - Total number of datafiles divided by`FILESPERSET`
- RMAN enforces both the `FILESPERSET` and `MAXSETSIZE` limits. If necessary, then RMAN creates more backup sets than are calculated in the preceding comparison.
- RMAN uses a complex algorithm of best fit so that the majority of backup sets have a size as close to the `MAXSETSIZE` limit with the number of files as close to the `FILESPERSET` limit as possible.

Specifying MAXSETSIZE: Example Assume that you want to back up 50 datafiles, each containing 1000 blocks. To set a maximum backup set size for a database backup to 10 MB, you issue the following command:

```
BACKUP DATABASE MAXSETSIZE = 10M;
```

Because you did not set `FILESPERSET`, RMAN calculates the default value for you, comparing 64 to 50/2 and setting `FILESPERSET = 25`. RMAN compares the following values and chooses the higher value:

- 50000/10000 (total number of blocks/`MAXSETSIZE`) = 5
- 50/25 (total number of files/`FILESPERSET`) = 2

Consequently, RMAN attempts to make five backup sets, with each backup set containing no more than 25 files and totalling no more than 10 MB in size.

Note that if you set `MAXSETSIZE` to a value smaller than the size of the largest input file, you receive the `RMAN-06183` error:

```
RMAN-06183: datafile or datafilecopy larger than MAXSETSIZE: file# 1
            /oracle/oradata/trgt/system01.dbf
```

> **See Also:** *Oracle9i Recovery Manager Reference* for information on the `MAXSETSIZE` parameter

Number and Size of Backup Sets: Advanced Algorithm

The advanced algorithm determines the number and size of backup sets by using the same factors described in "Number and Size of Backup Sets: Basic Algorithm" on page 5-29. The difference is that the advanced algorithm is also influenced by the

DISKRATIO parameter. If DISKRATIO=*n*, then each backup set must read data from at least *n* disk drives. RMAN uses file location information obtained from the database server to determine which datafiles are on which disk drives.

If you set FILESPERSET but not DISKRATIO, then DISKRATIO defaults to the same value as FILESPERSET. If you specify neither parameter, then DISKRATIO defaults to 4. RMAN compares the DISKRATIO value to the actual number of devices involved in the backup and uses the lowest value. For example, if DISKRATIO=4 and the datafiles are located on three disks, then RMAN attempts to distribute the datafiles into three backup sets.

Assume that the database contains 50 datafiles spread across 6 disks, and the operating system is able to deliver this disk contention information to the server. You configure a single sbt channel and then run the following command:

```
BACKUP DATABASE;
```

RMAN uses the advanced algorithm. The basic algorithm indicates that because you did not specify FILESPERSET or MAXSETSIZE, RMAN should produce a single backup set. The advanced algorithm also looks at DISKRATIO, which in this case defaults to 4. Hence, each backup set must contain datafiles from at least four disks. Because RMAN is only producing one backup set containing all datafiles from all six disks, DISKRATIO makes no difference.

Assume that you change the backup command as follows:

```
BACKUP DATABASE FILESPERSET 5;
```

In this case, the basic algorithm produces ten backup sets. The advanced algorithm also factors in DISKRATIO, which here defaults to 5 (the same value as FILESPERSET). Hence, the advanced algorithm dictates that each of the 10 backup sets must contain datafiles from at least 5 of the 6 disks.

Note that the advanced algorithm also recognizes node affinity. Node affinity is only used when you allocate channels on different nodes in a Real Application Clusters configuration, and RMAN detects that it is more efficient to read specific datafiles from one node than another.

I/O Read Rate of Backups

You can limit the speed of a backup by using the RATE option of the ALLOCATE CHANNEL or CONFIGURE CHANNEL commands. The RATE option specifies the maximum number of bytes for each second that RMAN reads on the channel.

For example, you can configure automatic channels to limit channel `c1` reads to 700 KB a second and channel `c2` reads to 1 MB a second:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt RATE 700K;
CONFIGURE CHANNEL 2 DEVICE TYPE sbt RATE 1M;

BACKUP
  (TABLESPACE system, undotbs
     CHANNEL ORA_SBT_TAPE_1)
  (TABLESPACE users, tools, indx
     CHANNEL ORA_SBT_TAPE_2);
```

In effect, the `RATE` option throttles RMAN so that a backup job does not consume excessive bandwidth on the computer.

> **See Also:** "Improving RMAN Backup Performance" on page 14-11 for tips about how to optimize RMAN performance

Backup Types

This section contains these topics:

- About Backup Types
- Full Backups
- Incremental Backups
- Types of Files That RMAN Can Back Up

About Backup Types

With RMAN, you can control the type of backups you produce. As explained in Table 5–2, RMAN backups can be classified in these ways:

- Full or incremental
- Open or closed
- Consistent or inconsistent

Table 5–2 Backup Types

Backup Type	Definition
Full	A backup that is not incremental. A full backup includes all used data blocks in the datafiles. Full backups of control files and archived logs always include all blocks in the files. **Note:** A full backup is different from a whole database backup, which is a backup of all datafiles and the current control file.
Incremental	A backup of datafiles that includes only the blocks that have changed since a previous incremental backup. Incremental backups require a full or incremental level 0 backup to serve as a basis.
Online	A backup of online, read / write datafiles when the database is open. **Note:** Do not put a tablespace in backup mode with the `ALTER TABLESPACE BEGIN BACKUP` statement. RMAN uses a different method to guarantee consistency in online backups.
Closed	A backup of any part of the target database when it is mounted but not open. Closed backups can be consistent or inconsistent.
Consistent	A backup taken when the database is mounted (but not open) and was *not* crashed or shut down with the `ABORT` option prior to mounting. The checkpoint SCNs in the datafile headers match the header information in the control file and none of the datafiles has changes beyond its checkpoint. Consistent backups can be restored without recovery.
Inconsistent	A backup of any part of the target database when ■ It is open. ■ It crashed or a `SHUTDOWN ABORT` was run prior to mounting. An inconsistent backup requires recovery to become consistent.

Full Backups

A **full backup** reads the entire file and copies all blocks into the backup set, only skipping datafile blocks that have never been used. RMAN does not skip blocks when backing up archived redo logs or control files.

Note: A full backup is not the same as a whole database backup: "full" is an indicator that the backup is not incremental.

A full backup has no effect on subsequent incremental backups, which is why it is not considered part of the incremental strategy. In other words, a full backup does not affect which blocks are included in subsequent incremental backups.

RMAN can create and restore full backups of the datafiles, control files, and archived redo logs. Note that backup sets containing archived redo logs are always full backups.

Incremental Backups

This section contains these topics:

About Incremental Backups

An incremental backup reads the entire file and then backs up 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. Note that RMAN can include a control file in an incremental backup set, but the control file is always included in its entirety—no blocks are skipped.

During media recovery, RMAN examines the restored files to determine whether it can recover them with an incremental backup. RMAN always chooses incremental backups over archived logs. Note that RMAN does not need to restore a base incremental backup of a datafile in order to apply incremental backups to the datafile during recovery. For example, you can restore non-incremental image copies of the datafiles in the database, and RMAN can recover these datafiles with incremental backups.

The primary reasons for making an incremental backup are

- To save tape when using a media manager or disk space when making disk backups
- To save network bandwidth when backing up over a network

- When the aggregate tape bandwidth available for tape write I/Os is much less than the aggregate disk bandwidth for disk read I/Os
- To be able to recover changes to objects created with the `NOLOGGING` option (direct load inserts do not log redo, although they do change data blocks and so are captured by incremental backups)
- To reduce backup sizes for `NOARCHIVELOG` databases. Instead of making a whole database backup every time, you can make incremental backups. Note that incremental backups of a `NOARCHIVELOG` database are only legal after a consistent shutdown.

One effective strategy is to make incremental backups to disk and then run `BACKUP BACKUPSET` to copy the backups to a media manager. Then, you do not have the problem of keeping tape streaming that sometimes occurs when making incremental backups directly to tape. Because incrementals are not as big as full backups, you can create them on disk more easily.

Multilevel Incremental Backups

RMAN can create **multilevel incremental backups**. Each incremental level is denoted by an integer, for example, 0, 1, 2, and so forth. A level 0 incremental backup, which is the base for subsequent incremental backups, copies all blocks containing data. The only difference between a level 0 backup and a full backup is that a full backup is never included in an incremental strategy. If no level 0 backup exists when you run a level 1 or higher backup, RMAN makes a level 0 backup automatically to serve as the base.

A level *n* incremental backup in which *n* is greater than zero backs up either:

- All blocks changed after the most recent backup at level *n* or lower (the default type of incremental backup, which is called a **differential backup**)
- All blocks changed after the most recent backup at level *n*-1 or lower (called a **cumulative backup**)

Note: In most circumstances, cumulative backups are preferable to differential backups because fewer incremental backups need to be applied during recovery.

The benefit of performing multilevel incremental backups is that RMAN does not back up all blocks all of the time. Since RMAN needs to read all of the blocks of the datafile, full backups and incremental backups take approximately the same amount of time.

Incremental backups at levels greater than zero only copy blocks that were modified. The size of the backup file depends solely upon the number of blocks modified and the incremental backup level.

How Incremental Backups Work

Each data block in a datafile contains a system change number (SCN), which is the SCN at which the most recent change was made to the block. During an incremental backup, RMAN reads the SCN of each data block in the input file and compares it to the checkpoint SCN of the parent incremental backup. RMAN reads the entire file every time whether or not the blocks have been used.

The parent backup is the backup that RMAN uses for comparing the SCNs. If the current incremental is a differential backup at level *n*, then the parent is the most recent incremental of level *n* or less. If the current incremental is a cumulative backup at level *n*, then the parent is the most recent incremental of level *n*-1 or less. If the SCN in the input data block is greater than or equal to the checkpoint SCN of the parent, then RMAN copies the block.

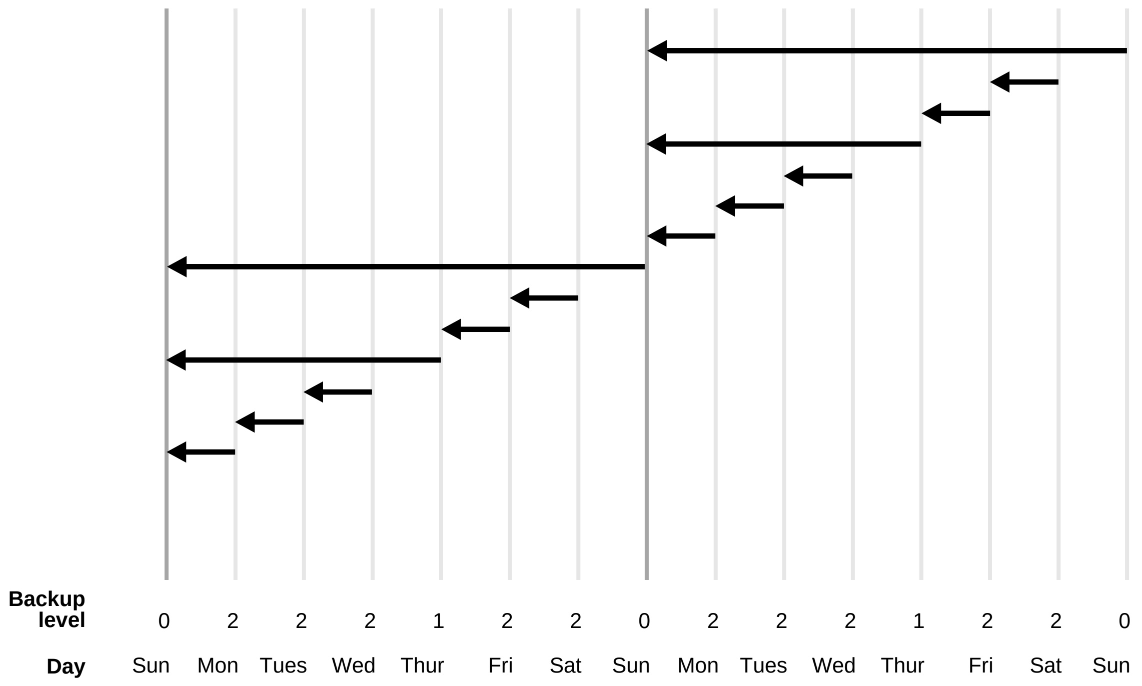
Note that one consequence of this mechanism is that RMAN applies all blocks containing changed data during recovery—even if the change is to an object created with the `NOLOGGING` option. Hence, making incremental backups functions as a safeguard against the loss of changes made to `NOLOGGING` tables.

> **See Also:** *Oracle9i Database Concepts* for more information about `NOLOGGING` mode

Differential Incremental Backups

In a differential level *n* incremental backup, RMAN backs up all blocks that have changed since the most recent backup at level *n* or lower. For example, in a differential level 2 backup, RMAN determines which level 1 or level 2 backup occurred most recently and backs up all blocks modified after that backup. If no level 1 is available, RMAN copies all blocks changed since the base level 0 backup. If no level 0 backup is available, RMAN makes a new base level 0 backup for this file. Incremental backups are differential by default.

Figure 5–5 Differential Incremental Backups (Default)



In the example shown in Figure 5–5, the following occurs:

- Sunday

  An incremental level 0 backup backs up *all* blocks that have ever been in use in this database.

- Monday

  A differential incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n* or less; in this case, the most recent incremental backup at level 2 or less is the level 0 Sunday backup, so only the blocks changed since Sunday will be backed up.

- Tuesday

  A differential incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n* or less; in this case, the most recent incremental backup at level 2 or less is the level 2 Monday backup, so only the blocks changed since Monday will be backed up.

- Wednesday

  A differential incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n* or less; in this case, the most recent incremental backup at level 2 or less is the level 2 Tuesday backup, so only the blocks changed since Tuesday will be backed up.

- Thursday

  A differential incremental level 1 backup backs up all blocks that have changed since the most recent incremental backup at level *n* or less; in this case, the most recent incremental backup at level 1 or less is the level 0 Sunday backup, so all the blocks changed since the Sunday level 0 backup will be backed up.

- Friday

  A differential incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n* or less; in this case, the most recent incremental backup at level 2 or less is the level 1 Thursday backup, so only the blocks changed since the Thursday level 1 backup will be backed up.

- Saturday

  A differential incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n* or less; in this case, the most recent incremental backup at level 2 or less is the level 2 Friday backup, so only the blocks changed since the Friday level 2 backup will be backed up.

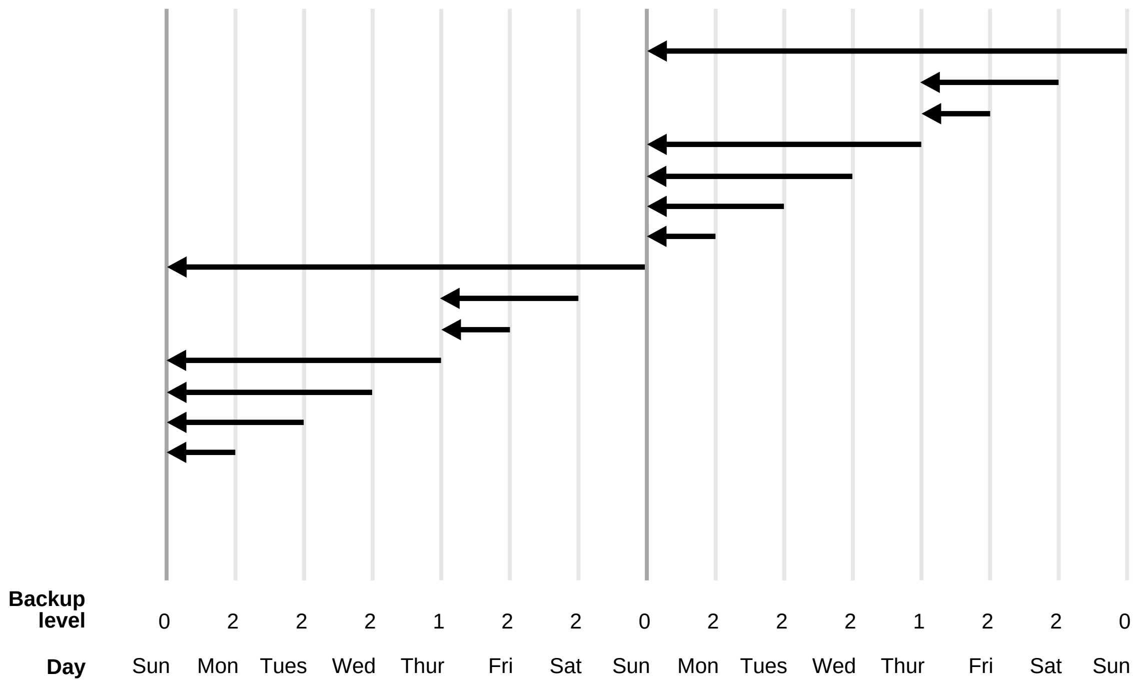
- The cycle is repeated for the next week.

Cumulative Incremental Backups

Oracle provides an option to make cumulative incremental backups at level 1 or greater. In a cumulative level *n* backup, RMAN backs up all the blocks used since the most recent backup at level *n*-1 or lower. For example, in a cumulative level 2 backup, RMAN determines which level 1 backup occurred most recently and copies all blocks changed since that backup. If no level 1 backup is available, RMAN copies all blocks changed since the base level 0 backup.

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, however, because they duplicate the work done by previous backups at the same level.

Figure 5–6 Cumulative Incremental Backups


Backup level
0 2 2 2 1 2 2 0 2 2 2 1 2 2 0
Day
Sun Mon Tues Wed Thur Fri Sat Sun Mon Tues Wed Thur Fri Sat Sun

In the example shown in Figure 5–6, the following occurs:

- Sunday

  An incremental level 0 backup backs up *all* blocks that have ever been in use in this database.

- Monday

  A cumulative incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n*-1 or less; in this case, the most recent incremental backup at level 1 (2 minus 1) or less is the level 0 Sunday backup, so only the blocks changed since Sunday will be backed up.

- Tuesday

  A cumulative incremental level 2 backup occurs. This backs up all blocks that have changed since the most recent incremental backup at level *n*-1 or less; in this case, the most recent incremental backup at level 1 (2 minus 1) or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up. (This backup includes those blocks that were copied on Monday, since this backup is cumulative and includes the blocks copied at backups taken at the same incremental level as the current backup).

- Wednesday

  A cumulative incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n*-1 or less; in this case, the most recent incremental backup at level 1 (2 minus 1) or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up. (This backup includes those blocks that were copied on Monday and Tuesday, since this backup is cumulative and includes the blocks copied at backups taken at the same incremental level as the current backup).

- Thursday

  A cumulative incremental level 1 backup backs up all blocks that have changed since the most recent incremental backup at level *n*-1 or less; in this case, the most recent incremental backup at level 0 (1 minus 1) or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up.

- Friday

  A cumulative incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n*-1 or less; in this case, the most recent incremental backup at level 1 (2 minus 1) or less is the level 1 Thursday backup, so all the blocks changed since Thursday will be backed up.

- Saturday

  A cumulative incremental level 2 backup backs up all blocks that have changed since the most recent incremental backup at level *n*-1 or less; in this case, the most recent incremental backup at level 1 (2 minus 1) or less is the level 1 Thursday backup, so all the blocks changed since Thursday will be backed up.

- The cycle is repeated for the next week.

Incremental Backup Strategy

Choose a backup scheme 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 backup is taken weekly, and a cumulative level 2 is taken daily. In this scheme, you never have to apply more than a day's worth 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 whenever 50% 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 query displays the number of blocks written to a backup set for each datafile with at least 50% 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 > .5
  ORDER BY COMPLETION_TIME;
```

Compare the number of blocks in differential or cumulative backups to a base level 0 backup. For example, if you only create level 1 cumulative backups, then when the most recent level 1 backup is about half of the size of the base level 0 backup, take a new level 0.

See Also: "Making Incremental Backups with RMAN" on page 9-17

Types of Files That RMAN Can Back Up

In a single-instance database RMAN performs backup operations only when an instance has the database mounted or open, because there must be a control file available to record the backup information.

The BACKUP command can back up the following types of files:

- Database, which includes all datafiles as well as the current control file and current server parameter file
- Tablespaces (except for locally-managed temporary tablespaces)
- Current datafiles
- Current control file
- Archived redo logs
- Current server parameter file
- Backup sets

RMAN does not back up the following:

- Online redo logs
- Transported tablespaces before they have been made read/write
- Client-side initialization parameter files or noncurrent server parameter files

See Also:

- *Oracle9i Real Application Clusters Administration* for information about Oracle Real Application Clusters backup constraints
- "Tests and Integrity Checks for Backups" on page 5-65 to learn about the PUT command

Backup Errors

RMAN can handle the two primary types of backup errors: I/O errors and corrupt blocks. Any I/O errors that RMAN encounters when reading files or writing to the backup pieces cause the system to terminate the jobs. For example, if RMAN tries to back up a datafile but the datafile is not on disk, then RMAN terminates the backup.

If an RMAN job produces more than one separate backup set and an error occurs, then RMAN needs to rewrite the backup sets that it was writing at the time of the error. However, it retains any backup sets that it successfully wrote before terminating. The NOT BACKED UP SINCE option of the BACKUP command restarts a backup that partially completed, backing up only files that did not get backed up.

RMAN copies datafile blocks that are already identified as corrupt into the backup. If RMAN encounters datafile blocks that have not already been identified as corrupt, then it writes them to the backup with a reformatted header indicating that the block has media corruption (assuming that SET MAXCORRUPT is not equal to 0 for this datafile and the number of corruptions does not exceed the limit). In either case, Oracle records the address of the corrupt block and the type of corruption in the control file. Access these records through the V$BACKUP_CORRUPTION and V$COPY_CORRUPTION views.

Use the SET MAXCORRUPT command to allow a certain number of previously undetected block corruptions in specified datafiles. If a BACKUP or COPY command detects more than this number of corruptions, then the command terminates. The default limit is zero, meaning that RMAN does not tolerate corrupt blocks.

See Also:

- "Tests and Integrity Checks for Backups" on page 5-65 for more information about fractured and corrupt blocks
- "Restartable Backups" on page 5-61 for more information about the NOT BACKED UP SINCE clause
- *Oracle9i Database Reference* for a description of V$BACKUP_ CORRUPTION
- *Oracle9i Recovery Manager Reference* for reference information on SET MAXCORRUPT

Control File and Server Parameter File Autobackups

If CONFIGURE CONTROLFILE AUTOBACKUP is ON (by default it is OFF), then RMAN automatically backs up the control file and the current server parameter file (if used) in the following circumstances:

- After every BACKUP or COPY command issued at the RMAN prompt.
- Whenever a BACKUP or COPY command within a RUN block is followed by a command that is neither BACKUP nor COPY.
- At the end of every RUN block if the last command in the block was either BACKUP or COPY.
- After database structural changes such as adding a new tablespace, altering the state of a tablespace or datafile (for example, bringing it online), adding a new online redo log, renaming a file, adding a new redo thread, and so forth. This type of autobackup, unlike autobackups that occur in the preceding circumstances, is only on disk. You can run CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK to set a nondefault disk location. Note that the autobackup never causes the associated structural change to fail. For example, if you add a datafile, and if the resulting autobackup fails, then the datafile addition is still successful.

Note: If you use Oracle Enterprise Manager, then you can use the Maintenance wizard to configure the autobackup feature.

The first channel allocated during the backup or copy job creates the autobackup and places it into its own backup set; for post-structural autobackups, the default disk channel makes the backup. RMAN writes the control file and the server parameter file to the same backup piece. After the control file autobackup completes, Oracle writes a message containing the complete path of the backup piece and the device type to the alert log.

As explained by the following table, the RMAN behavior when the BACKUP command includes datafile 1 differs depending on whether CONFIGURE CONTROLFILE AUTOBACKUP is ON or OFF.

CONTROLFILE AUTOBACKUP	BACKUP Command Behavior
ON	If the backup includes datafile 1, then RMAN does *not* automatically include the current control file in the datafile backup set. Instead, RMAN writes the control file and server parameter file to a separate autobackup piece. **Note:** The autobackup occurs regardless of whether the BACKUP or COPY command explicitly includes the current control file, for example, BACKUP DATABASE INCLUDE CURRENT CONTROLFILE.
OFF	If the backup includes datafile 1, then RMAN automatically includes the current control file and server parameter file in the datafile backup set. RMAN does *not* create a separate autobackup piece containing the control file and server parameter file.

The automatic backup of the control file occurs in addition to any backup of the current control file that has been performed during these commands. You can turn the autobackup feature on or off by running the following commands:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
```

The purpose of the control file autobackup is to provide a way to restore the backup repository contained in the control file when the control file is lost and the recovery catalog is either lost or was never used. You do not need a recovery catalog or target control file to restore the control file autobackup. For example, you can issue:

```
RESTORE CONTROLFILE FROM AUTOBACKUP;
```

After you restore and mount the control file, then you can use it to obtain the backup information necessary to restore and recover the database. For example, if the recovery catalog database was destroyed along with the target database, you can create a new catalog and register the target database. The new catalog will be populated with backup information from the restored control file. You can also simply connect to the target instance in `NOCATALOG` mode and restore the database.

The control file autobackup filename has a default format of `%F` for all device types so that RMAN can restore it without a repository. The substitution variable `%F` is defined in the description of the `CONFIGURE` command. You can specify a different format for a device type with the `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT` command. A valid format must include the `%F` variable as part of the string.

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

1. `SET` within `RUN`
2. `SET` at `RMAN` prompt
3. `CONFIGURE`

Note that you can configure the autobackup format even when `CONFIGURE CONTROLFILE AUTOBACKUP` is set to `OFF`, but RMAN does not generate autobackups in this case. For RMAN to make autobackups, you must set `CONFIGURE CONTROLFILE AUTOBACKUP` to `ON`.

See Also:

- "Overriding the Control File Autobackup Format" on page 9-31
- *Oracle9i Recovery Manager Reference* for `BACKUP` syntax
- *Oracle9i Recovery Manager Reference*for `RESTORE` syntax

Backup Retention Policies

This section contains these topics:

- About Backup Retention Policies
- Recovery Window
- Backup Redundancy
- Batch Deletes of Obsolete Backups and Copies
- Backups Exempt from the Retention Policy

About 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 backups and copies of datafiles and control files as obsolete, that is, no longer needed for media recovery, according to criteria that you specify in the `CONFIGURE` command. You can then periodically or regularly issue the `REPORT OBSOLETE` command to view obsolete files and `DELETE OBSOLETE` to delete them.

The retention policy is ongoing. As you produce datafile, control file, and archived log backups over time, RMAN keeps track of which to keep and which are obsolete. RMAN does not automatically delete the files for you, but it does perform the record keeping.

The term *obsolete* does not mean the same as *expired*. A backup or copy is obsolete when `REPORT OBSOLETE` or `DELETE OBSOLETE` determines, based on the user-defined retention policy, that it is not needed for media recovery. A backup or copy expires only when RMAN performs a crosscheck and sees either that the file is missing from disk or that the media manager has returned "Not found" for the file. In short, *obsolete* means "not needed," whereas *expired* means "not found."

Note that from the perspective of a retention policy, a *backup* is a backup or copy of an individual datafile or control file. It does not matter whether the backup is a datafile copy, a proxy copy, or part of another backup set. 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 backups in backup sets, RMAN cannot delete the backup set until all of the individual datafile backups within the backup set are obsolete.

Besides affecting datafile and control file backups, the retention policy affects archived logs and archived log backups. First, RMAN decides which datafile and control file backups are obsolete. Then, RMAN considers as obsolete all archived log backups that are older than the oldest datafile or control file backup that must be retained. This behavior occurs regardless of whether the retention policy is configured for a recovery window or redundancy.

Note: RMAN cannot implement an automatic retention policy if backups are deleted by non-RMAN methods, 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.

You have two mutually exclusive options for implementing an retention policy: specifying a **recovery window**, or specifying **redundancy** (the default type of retention policy). The retention policy commands are as follows:

- `CONFIGURE RETENTION POLICY TO RECOVERY WINDOW`
- `CONFIGURE RETENTION POLICY TO REDUNDANCY`

You can also disable the retention policy completely by running the following command, meaning that RMAN does not consider any backup or copy to be obsolete:

```
CONFIGURE RETENTION POLICY TO NONE;
```

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 this window of time must extend back exactly seven days from the present so that you can restore a backup and recover it to this point. 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 - (SELECT CHECKPOINT_TIME FROM V$DATAFILE) >=  7
```

All backups older than the most recent backup that satisfied this condition are obsolete.

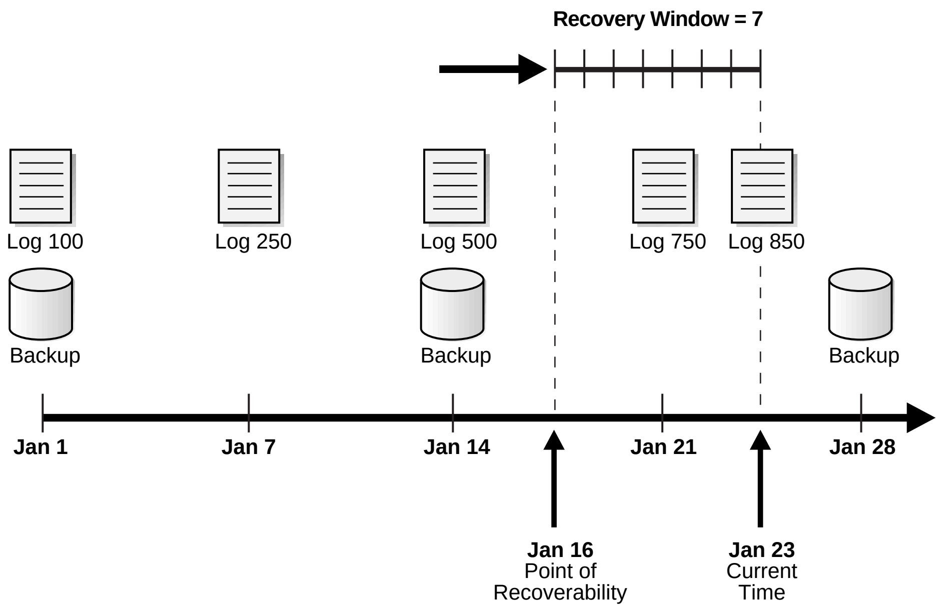
Assume the following retention policy illustrated in Figure 5–7. The retention policy has the following aspects:

- The recovery window is 7 days.
- Database backups are scheduled every two weeks on these days:
  - January 1
  - January 14
  - January 28

  – February 4

- The database runs in `ARCHIVELOG` mode, and archived logs are saved on disk only as long as needed for the retention policy.

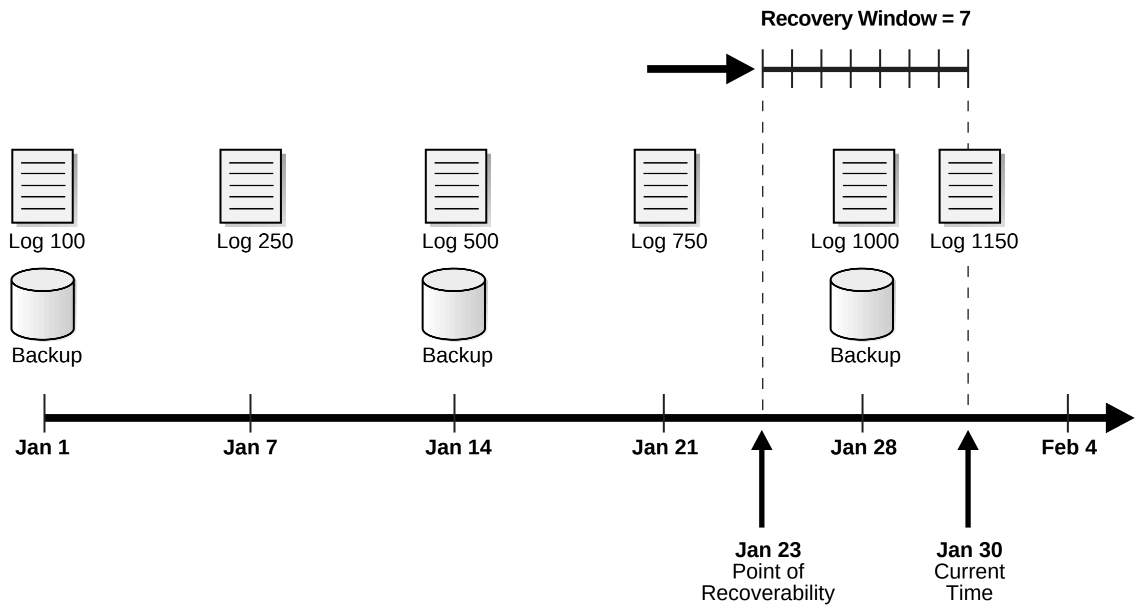
Figure 5–7 Recovery Window, Part 1



As illustrated in Figure 5–7, the current time is January 23 and the point of recoverability is January 16. Hence, the January 14 backup is needed for recovery, as 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 5–8.

Figure 5–8 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 within the window, you must save the January 14 backup as well as all archived redo logs from log sequence 500 to 1150.

See Also: "Configuring the Retention Policy for a Recovery Window" on page 8-20

Backup Redundancy

Besides `RECOVERY WINDOW`, the `CONFIGURE` command has another mechanism for controlling how long backups should be retained: the `REDUNDANCY` parameter. The

REDUNDANCY parameter specifies that any number of backups or copies beyond a specified number need not be retained. For example, you can specify:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
```

Although the recovery window is the best practice for specifying an retention policy, you choose not to specify it because the number of backups that must be kept by the recovery window is not constant and depends on the backup schedule. Use the CONFIGURE RETENTION POLICY TO REDUNDANCY command to implement a retention policy that maintains a constant number of backups of each datafile. The RECOVERY WINDOW option is mutually exclusive with the REDUNDANCY option.

By default, the retention policy is configured to REDUNDANCY = 1 to maintain compatibility with the behavior of REPORT OBSOLETE in earlier RMAN releases. If you run the following command, then RMAN returns the retention policy to the default value:

```
CONFIGURE RETENTION POLICY CLEAR;
```

You can also run the following command to disable the retention policy altogether:

```
CONFIGURE RETENTION POLICY TO NONE;
```

If the retention policy is configured to NONE, then REPORT OBSOLETE and DELETE OBSOLETE do not consider any backups to be obsolete.

See Also: "Configuring the Retention Policy for Redundancy" on page 8-21

Batch Deletes of Obsolete Backups and Copies

Run the REPORT OBSOLETE command to determine which backups and copies are currently obsolete according to the retention policy. For example, if you set the recovery window to 7 days with the CONFIGURE command, then you can run the following command to determine which backups and archived redo logs are obsolete:

```
REPORT OBSOLETE;
```

Note that if you configure the retention policy to NONE, then RMAN does not consider any backups or copies as obsolete. Consequently, RMAN issues an error when you run REPORT OBSOLETE without any other options.

A companion command, DELETE OBSOLETE, deletes the files rendered obsolete by the retention policy. You can run this command to delete all obsolete backups and copies at once. For example, you can run DELETE OBSOLETE in a weekly script.

Executing the DELETE OBSOLETE command ensures that you do not waste disk space by storing unnecessary files.

You can also specify the REDUNDANCY or RECOVERY WINDOW options on the REPORT or DELETE commands. These settings determine what is obsolete when running the commands and override the CONFIGURE RETENTION POLICY settings.

See Also:

- "Reporting on Backups, Copies, and Database Schema" on page 17-10
- "Deleting Backups and Copies" on page 18-4
- *Oracle9i Recovery Manager Reference* for DELETE syntax
- *Oracle9i Recovery Manager Reference* for REPORT syntax

Backups Exempt from the Retention Policy

You may want to store a **long-term backup**, potentially offsite, for much longer than the time dictated by the user's retention policy. For example, you may make a database backup on the first day of every year just to keep a record. You do not intend to use this backup in your backup and recovery strategy, although you may want to restore it at some point for report generation. For example, you may want to see how many customers were in the database 5 years ago.

You do not want to include a long-term backup in the production backup schedule, but you do want to record it in the recovery catalog. A long-term backup should not be included in the retention policy because RMAN would quickly mark it as obsolete, and then a DELETE OBSOLETE command would remove it.

By using the KEEP option in the BACKUP or COPY command (or the KEEP option of the CHANGE command), you exclude the backup from the retention policy. The backup is still a fully valid backup, however, and can be restored just like any other RMAN backup. The NOKEEP option (default) indicates that the backup is not immune from the configured retention policy. You can change a backup between KEEP and NOKEEP status with the CHANGE command.

You can specify the LOGS option to save archived logs for a possible incomplete recovery of the long-term backup. When LOGS is specified, all logs more recent than the backup are kept as long as the backup is kept. In other words, KEEP FOREVER LOGS means that RMAN not consider logs generated after the long-term backup to be obsolete. If you specify NOLOGS, then RMAN does not keep the logs required to recover the backup. Note that in this case the long-term backup must be consistent.

You can either specify that the backup should be kept FOREVER, or specify an end date using the UNTIL clause. If you specify UNTIL, then RMAN will not mark the backup as obsolete until after the UNTIL date has passed.

The following commands are examples of long-term backups:

```
# exempts the backup from retention policy until last day of 2002
BACKUP DATABASE KEEP UNTIL TIME "TO_DATE('31-DEC-2002', 'dd-mon-yyyy')" NOLOGS;

# alters status of backup set 231 from KEEP to NOKEEP
CHANGE BACKUPSET 231 NOKEEP;

# specifies that this backup and the logs required to recover it are indefinitely exempt
# from the retention policy
BACKUP TABLESPACE users KEEP FOREVER NOLOGS;
```

See Also: *Oracle9i Recovery Manager Reference* for CHANGE syntax

Backup Optimization

RMAN's **backup optimization** refers to its intelligent skipping of files during backups when certain conditions are met.

This section contains these topics:

- Backup Optimization Algorithm
- Requirements for Enabling and Disabling Backup Optimization
- Effect of Retention Policies on Backup Optimization

Backup Optimization Algorithm

If you enable backup optimization, then the BACKUP command skips the backup of a file when the identical file has already been backed up to the allocated 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 5–3 Criteria to Determine an Identical File

Type of File	Criteria to Determine an Identical File
Datafile	Same DBID, checkpoint SCN, and RESETLOGS SCN and time. The datafile must be offline-normal, read-only, or closed normally.
Archived redo log	Same thread, sequence number, and RESETLOGS SCN and time.
Backup set	Same backup set recid and stamp.

If RMAN determines that a file is identical and it has already been backed up, then it is a candidate for skipping. However, RMAN must do further checking to determine whether to skip the file, because both the retention policy feature and the backup duplexing feature influence the algorithm that determines whether RMAN has "enough" backups on the specified device type.

Table 5–4 describes the algorithm that backup optimization uses when determining whether to skip the backup of an identical file.

Table 5–4 Backup Optimization Algorithm

For an Identical ...	Backup Optimization Algorithm
Datafile	If you configured a recovery window, then RMAN skips a datafile backup *only if* the latest backup of a file is earlier than the earliest point in the window. If you did not configure a recovery window, then RMAN sets *n*=`1` and searches for values of *n* in this order of precedence (that is, values higher on the list override values lower on the list): **1.** If `CONFIGURE RETENTION POLICY TO REDUNDANCY` *r* is enabled, then RMAN only skips datafiles when *n*=*r*+1 backups exist. **2.** `BACKUP ... COPIES` *n* **3.** `SET BACKUP COPIES` *n* **4.** `CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE ... TO` *n* RMAN skips backup only if at least *n* backups of an identical file exist on the specified device. If RMAN does not skip the backup, then it makes the backup exactly as specified.
Archived log	By default, *n*=`1`. RMAN searches for values of *n* in this order of precedence (that is, values higher on the list override values lower on the list): **1.** `BACKUP ... COPIES` *n* **2.** `SET BACKUP COPIES` *n* **3.** `CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE ... TO` *n* RMAN skips backup only if at least *n* backups of an identical file exist on the specified device. If RMAN does not skip the backup, then it makes the backup exactly as specified.
Backup set	By default, *n*=`1`. RMAN searches for other values of *n* in this order of precedence (that is, values higher on the list override values lower on the list): **1.** `BACKUP ... COPIES` *n* **2.** `SET BACKUP COPIES` *n* RMAN skips backup only if at least *n* backups of an identical file exist on the specified device. If RMAN does not skip the backup, then it makes the backup exactly as specified.

For example, assume that at 9 a.m. you back up three copies of all existing archived logs to tape:

```
BACKUP DEVICE TYPE sbt COPIES 3 ARCHIVELOG ALL;
```

Later, you enable the following configuration setting:

```
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE sbt TO 4;
CONFIGURE BACKUP OPTIMIZATION ON;
```

Then, you run the following archived log backup at noon:

```
BACKUP DEVICE TYPE sbt COPIES 2 ARCHIVELOG ALL;
```

In this case, the `BACKUP ... COPIES` setting overrides the `CONFIGURE ... COPIES` setting, so RMAN sets *n*=`2`. RMAN skips the backup of a log only if at least two copies of the log exist on the `sbt` device. Because three copies of each log exist on `sbt` of all the logs generated before 9 a.m., RMAN skips the backups of these logs. However, RMAN backs up two copies of all logs generated after 9 a.m. because these logs have not yet been backed up to tape.

At this point, three copies of the logs created before 9 a.m. exist on tape, and two copies of the logs created after 9 a.m. exist on tape. Assume that you run the following backup at 3 p.m.:

```
RUN
{
  SET BACKUP COPIES 3;
  BACKUP DEVICE TYPE sbt ARCHIVELOG ALL;
}
```

In this case, RMAN sets *n*=`3` and so will not back up the logs created before 9 a.m. because three copies already exist on tape. However, only two copies of the logs created after 9 a.m. exist on tape, so RMAN does not optimize backups of these logs. Hence, RMAN backs up three copies of the logs created after 9 a.m.

Requirements for Enabling and Disabling Backup Optimization

Backup optimization is enabled when the following conditions are true:

- The `CONFIGURE BACKUP OPTIMIZATION ON` command has been run.
- You run `BACKUP DATABASE`, `BACKUP ARCHIVELOG` with `ALL` or `LIKE` options, or `BACKUP BACKUPSET ALL.`
- Only one type of channel is allocated, that is, you do not mix channels of type `DISK` and `sbt`.

For example, assume that you run these commands:

```
BACKUP DEVICE TYPE sbt DATABASE PLUS ARCHIVELOG;
BACKUP DEVICE TYPE sbt BACKUPSET ALL;
```

If none of these files has changed since the last backup, then RMAN does not back up the files again, nor signal an error if it skips all files specified in the command.

Note: Whether or not backup optimization is enabled, if you run BACKUP ARCHIVELOG and no archived logs exist, then RMAN does not signal an error. Probably no new logs were generated after the previous BACKUP ARCHIVELOG ALL DELETE INPUT command.

To override backup optimization and back up all files whether or not they have changed, specify the FORCE option on the BACKUP command. To disable backup optimization, specify OFF on the CONFIGURE BACKUP OPTIMIZATION command. The CLEAR option returns backup optimization to its default value of OFF.

Effect of Retention Policies on Backup Optimization

The retention policy influences backup optimization. Because the retention policy defaults to REDUNDANCY=1, a retention policy is always in place unless it is explicitly disabled with CONFIGURE RETENTION POLICY TO NONE.

Note: Use caution when enabling backup optimization if you use a media manager that has an expiration policy. The media manager can expire tapes containing backups, and RMAN will not make new backups because of optimization. Run CROSSCHECK periodically to synchronize the repository with the media manager.

Backup Optimization and a Recovery Window

If optimization is enabled, and if a retention policy is configured for a recovery window, then RMAN always backs up datafiles whose most recent backup is older than the recovery window. For example, assume that:

- 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 changed 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 allows 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 allows the media manager to expire the tape containing the superfluous January 3 backup.

Backup Optimization and Redundancy

Assume that you configure a retention policy for redundancy. In this case, RMAN only skips backups of offline or read-only datafiles when there are *r* + 1 backups of the files, where *r* is set in `CONFIGURE RETENTION POLICY TO REDUNDANCY` *r*.

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;
```

So, 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 tablespace `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

Table 5–5 Effect of Redundancy Setting on Backup Optimization

Day	Action	Result	Redundant 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:** "Backing Up Files Using Backup Optimization" on page 9-27, and "Configuring Backup Optimization" on page 8-23

Restartable Backups

Using the **restartable backup** feature, RMAN can back up only those files that have not been backed up since a specified date. This feature is intended for cases when a backup fails partway through and you only want to back up the part of the database that did not finish.

The unit of restartability is a single backup set. If the entire database is backed up into one backup set, and if a backup fails, then the entire backup has to be rerun. If the backup generates multiple backup sets, then the backups that completed successfully do not have to be rerun. For this reason, you can set FILESPERSET to a value much lower than the default so that RMAN limits the number of files that it places in each backup set.

For example, you can back up the database daily as follows:

```
BACKUP DATABASE FILESPERSET = 8;
```

Then, back up all files in the database that were not backed up in the last 24 hours by issuing:

```
BACKUP DATABASE NOT BACKED UP SINCE TIME 'SYSDATE-1';
```

Note that if you do not specify the `SINCE TIME` parameter, then RMAN only backs up files that have never been backed up.

When determining whether a file has been backed up, RMAN compares the `SINCE TIME` date with the completion time of the most recent backup of the file. The completion time for a file in a backup set is the completion time of the entire backup set. In other words, all files in the same backup set have the same completion time. If the `SINCE TIME` is later than the completion time, then RMAN backs up the file.

> **See Also:** "Restarting a Backup After It Partially Completes" on page 9-28 and *Oracle9i Recovery Manager Reference* for `BACKUP` syntax

Image Copies

An **image copy** contains a single datafile, archived redo log file, or control file that you can use as-is to perform recovery. Run the RMAN `COPY` command or an operating system command such as the UNIX `cp` command to create image copies.

An image copy produced with the RMAN `COPY` command is similar to an operating system copy of a single file, except that an Oracle server session produces it. The server session performs additional actions like validating the blocks in the file and recording the copy in the control file. An image copy differs from a backup set because it is not multiplexed, nor is there any additional header or footer control information stored in the copy. RMAN only writes image copies to disk.

This sections contains these topics:

- RMAN Image Copies
- User-Managed Image Copies
- Tags for Backups and Image Copies
- Long-Term RMAN Copies

RMAN Image Copies

Use the RMAN `COPY` command to create an image copy to disk. You cannot use `COPY` to make a copy to a device other than `DISK`. If you run a `RESTORE` command, then by default RMAN restores the image copy of a datafile or control file to its original location.

Note that if you need to replace a current datafile, and if you have an image copy of the datafile, then you do not need to restore the copy. Instead, you can use the `SWITCH` command to point the control file at the copy and update the repository to indicate that the copy has been switched. Issuing the `SWITCH` command in this case is equivalent to issuing the SQL statement `ALTER DATABASE RENAME DATAFILE`. You can then perform recovery on the copy.

RMAN can inspect an existing image copy and enter its metadata into the control file and recovery catalog. This operation is important when the recovery catalog is lost and you must perform disaster recovery. Only image copies and archived logs can be cataloged.

User-Managed Image Copies

Oracle supports image copies created by mechanisms other than RMAN, also known as **user-managed copies** or **operating system copies**. For example, a copy of a datafile that you make with the UNIX `dd` command is an operating system copy. You must catalog operating system copies with RMAN before using them with the `RESTORE` or `SWITCH` commands.

You can create an operating system copy when the database is open or closed. If the database is open and the datafile is online and read/write, then you must place the tablespace in **backup mode**, that is, issue the SQL statement `ALTER TABLESPACE BEGIN BACKUP` before creating the copy.

Caution: If you do not put a tablespace in backup mode before making an online user-managed backup, the backup can contain fractured blocks and become unrecoverable. Refer to "Detection of Logical Block Corruption" on page 5-67.

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 operating system copy, thus making it a candidate for use in a restore operation. You must notify RMAN when the copy is no longer available for restore, however, by using the `CHANGE ... UNCATALOG` command. In this example, after **resilvering the mirror** (not including other copies of the broken mirror), then you must use a `CHANGE ... UNCATALOG` command to update the recovery catalog and indicate that this copy is no longer available.

See Also:

- "Cataloging Archived Logs and User-Managed Copies" on page 18-15 to learn how to catalog copies
- "Making Split Mirror Backups with RMAN" on page 9-18 to learn about split mirror backups
- *Oracle9i Recovery Manager Reference* for `CHANGE` syntax

Tags for Backups and Image Copies

You can assign a user-specified character string called a **tag** to backup sets and image copies (either copies created by RMAN or copies created by an operating system utility). A tag is a case-insensitive symbolic name for a backup set or file copy such as `weekly_backup`. You can specify the tag rather than the filename when executing the `RESTORE` or `CHANGE` command. The maximum length of a tag is 30 characters.

If you do not specify a tag name, then by default RMAN creates a tag for backups and copies (except for control file autobackups) in the format `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 receive the tag `TAG20020208T133437`. The date and time refer to when RMAN started the backup. If multiple backup sets are created by one `BACKUP` command, then each backup piece is assigned the same default tag.

When applied to a backup set, a tag applies to a specific copy of the backup set. If you do not duplex a backup set, that is, make multiple identical copies of it, then a one-to-one correspondence exists between the tag and the backup set. For example, `BACKUP COPIES 1 DATAFILE 7 TAG foo` creates one backup set with tag `foo`. However, you can back up this backup set and give this new copy of the backup set the tag `bar`. So, the backup set has two identical copies: one tagged `foo` and the other tagged `bar`.

Tags do not need to be unique, so multiple backup sets or image copies can have the same tag name, for example, `weekly_backup`. When a tag is not unique, then with respect to a given datafile, the tag refers to the most current suitable file. By default, RMAN selects the most recent backups to restore unless qualified by a tag or a `SET UNTIL` command. The most current suitable backup containing the specified file may not be the most recent backup, as can occur in point-in-time recovery.

For example, if datafile copies are created each Monday evening and are always tagged `mondaypmcopy`, then the tag refers to the most recent copy (assuming that no SET UNTIL command is issued before restore or recovery). Thus, if complete recovery is desired, this command switches datafile 3 to the most recent Monday evening copy:

```
SWITCH DATAFILE 3 TO DATAFILECOPY TAG mondaypmcopy;
```

Tags can indicate the intended purpose or usage of different classes of backups or file copies. For example, datafile copies that are suitable for use in a SWITCH can be tagged differently from file copies that should be used only for RESTORE.

Note: If you specify a tag when specifying input files to a RESTORE or SWITCH command, then RMAN considers *only* backup sets with a matching tag when choosing which backup to use

See Also: *Oracle9i Recovery Manager Reference* for SWITCH syntax, and *Oracle9i Recovery Manager Reference* for RESTORE syntax

Long-Term RMAN Copies

You can use the COPY ... KEEP command to specify that an image copy should be exempt from the current retention policy. The behavior of this command is the same as BACKUP ... KEEP.

See Also: "Backups Exempt from the Retention Policy" on page 5-55 for a description of retention policies and how files can be exempt or non-exempt

Tests and Integrity Checks for Backups

This section contains these topics:

- About Tests and Integrity Checks
- Detection of Physical Block Corruption
- Detection of Logical Block Corruption
- Detection of Fractured Blocks During Open Backups
- Test Backups Using RMAN

About Tests and Integrity Checks

Oracle prohibits any attempts to perform operations that result in unusable backup files or corrupt restored datafiles. By using integrity checks, the Oracle server automatically does the following:

- Ensures that restore operations do not corrupt the database by applying backups from a previous incarnation of the database
- Ensures that incremental backups are applied in the correct order
- Prohibits accessing datafiles that are in the process of being restored or recovered
- Allows only one restore operation for each datafile at a time
- Stores special error checking information in backups to ensure that corrupt backup files are detected

You can use the `BACKUP VALIDATE` command to perform a test run backup without actually producing output files. In this way, you can check your datafiles for possible problems.

Detection of Physical Block Corruption

Because an Oracle server session is performing backup and copy operations, the server session is able to detect many types of physically corrupt blocks. Each new corrupt block not previously encountered in a backup or copy operation is recorded in the control file and in the `alert.log`. By default, error checking for physical corruption is enabled.

RMAN queries corruption information at the completion of a backup and stores it in the recovery catalog and control file. Access this data using the views `V$BACKUP_CORRUPTION` and `V$COPY_CORRUPTION`.

If the server session encounters a datafile block during a backup that has already been identified as corrupt by the database, then the server session copies the corrupt block into the backup and Oracle logs the corruption in the control file as either a logical or media corruption. RMAN copies the block in case the user wants to try to salvage the contents of the block.

If RMAN encounters a datafile block with a corrupt header that has not already been identified as corrupt by the database, then it writes the block to the backup with a reformatted header indicating that the block has media corruption.

Note: RMAN cannot detect all types of block corruption.

Detection of Logical Block Corruption

RMAN tests data and index blocks that pass physical corruption checks for logical corruption, for example, corruption of a row piece or index entry. If RMAN finds logical corruption, it then logs the block in the `alert.log` and server session trace file. By default, error checking for logical corruption is disabled.

Provided the sum of physical and logical corruptions for a file remains below its `MAXCORRUPT` setting, the RMAN command completes and Oracle populates `V$BACKUP_CORRUPTION` and `V$COPY_CORRUPTION` with corrupt block ranges. If `MAXCORRUPT` is exceeded, the command terminates without populating the views.

Note: For `COPY` and `BACKUP` commands the `MAXCORRUPT` parameter sets the total number of physical and logical corruptions permitted in a file.

See Also: *Oracle9i Recovery Manager Reference* for `BACKUP ... MAXCORRUPT` syntax, and *Oracle9i Recovery Manager Reference* for `COPY ... MAXCORRUPT` syntax

Detection of Fractured Blocks During Open Backups

One danger in making online backups is the possibility of inconsistent data within a block. For example, assume that you are backing up block 100 in datafile `users.dbf`. Also, assume that the copy utility reads the entire block while database writer is in the middle of updating the block. In this case, the copy utility may read the old data in the top half of the block and the new data in the bottom top half of the block. In this case, the block is a **fractured block**, meaning that the data contained in this block is not consistent.

When performing open backups *without* using RMAN, you must put tablespaces in **backup mode** in case the operating system reads a block for a backup that is currently being written by the database writer. When not in backup mode, Oracle records only changed bytes in the redo stream, not whole blocks. When a tablespace is in backup mode, Oracle writes the before-image of each changed block in the tablespace to the redo log before modifying it. Then, Oracle also records the changes to the block in the redo log. When you recover using SQL*Plus, Oracle applies the blocks and changes during recovery, so it does not matter that the block in the backup was fractured.

During an RMAN backup, an Oracle server session reads the datafiles, not an operating system utility. The server session reads whole Oracle blocks and determines whether the block is fractured by comparing the header and footer of each block. If the session detects a fractured block, then it rereads the block until it gets a consistent picture of the data. For this and other reasons, RMAN does not require you to place the tablespaces in backup mode, and it is neither desired or correct to place them in backup mode before RMAN is started.

See Also: *Oracle9i User-Managed Backup and Recovery Guide* for information about backup mode

Test Backups Using RMAN

You can run the `BACKUP ... VALIDATE` command to do the following:

- Check datafiles for physical and logical corruption
- Confirm that all database files exist and are in the correct locations

RMAN does not actually produce backup sets, but scans the specified files to determine whether they can be backed up. In this sense, the `BACKUP VALIDATE` command is similar to the `RESTORE VALIDATE` command, except for backups rather than restore jobs.

For example, you can validate that all database files and archived redo logs can be backed up by issuing a command as follows:

```
BACKUP VALIDATE DATABASE ARCHIVELOG ALL;
```

You cannot use the `MAXCORRUPT` or `PROXY` parameters with the `VALIDATE` option.

See Also: *Oracle9i Recovery Manager Reference* for `BACKUP` syntax

6 RMAN Concepts II: Restore, Recovery, and Duplication

This chapter describes the basic concepts involved in using RMAN to restore, recover, and duplicate databases.

This chapter contains these topics:

Restoring Files with RMAN

This section contains these topics:

- About the RESTORE command
- Mechanics of Restore Operations
- File Selection in Restore Operations
- Restore Optimization
- Automatic Location of Backups When Restoring in Real Application Clusters

About the RESTORE command

Use the RMAN `RESTORE` command to restore the following types of files from image copies on disk or backups on disk or other media:

- Database (all datafiles)
- Tablespaces
- Control files
- Archived redo logs
- Server parameter files

Because a backup set is in an Oracle proprietary format, you cannot simply copy it as you would a backup database file created with an operating system utility; you must use the RMAN `RESTORE` command to extract it. In contrast, Oracle can use image copies created by the RMAN `COPY` command without additional processing.

Note: You do not normally restore archived logs because RMAN performs this operation automatically as needed during recovery. You can improve recovery performance, however, by manually restoring backups of archived redo logs that you need for recovery.

See Also: "Restoring and Recovering with RMAN: Overview" on page 10-2 to learn how to restore, and *Oracle9i Recovery Manager Reference* for `RESTORE` syntax and prerequisites

Mechanics of Restore Operations

RMAN automates the procedure for restoring files. You do not need to go into the operating system, locate the backup or copy that you want to use, and manually copy files into the appropriate directories. When you issue a `RESTORE` command, RMAN directs a server session to restore the correct backups and copies to either:

- The default location, overwriting the files with the same name currently there
- A new location, which you can specify with the `SET NEWNAME` command

To restore a datafile, either mount the database or keep it open and take the datafile to be restored offline. When RMAN performs a restore, the `RESTORE` command creates the restored files as datafile copies and records them in the repository. The following table describes the behavior of the `RESTORE`, `SET NEWNAME`, and `SWITCH` commands.

Run SET NEWNAME?	RESTORE Behavior	Run SWITCH?
No	RMAN restores the files to their current path names and immediately removes the repository records for the datafile copies created during the restore.	N/A
Yes	RMAN restores the files to the path names specified by `SET NEWNAME` and does not remove the repository records for the datafile copies created during the restore.	If yes, then RMAN updates the datafile names in the control file to the names of the restored files; if no, then RMAN does not update the filenames in the control file and the restored files become datafile copies.

For example, if you restore datafile `?/oradata/trgt/tools01.dbf` to its default location, then RMAN restores the file `?/oradata/trgt/tools01.dbf` and overwrites any file that it finds with the same filename. If you run a `SET NEWNAME` command before you restore a file, then RMAN creates a datafile copy with the name that you specify. For example, assume that you run the following commands:

```
SET NEWNAME FOR DATAFILE '?/oradata/trgt/tools01.dbf' TO '/tmp/tools01.dbf';
RESTORE DATAFILE '?/oradata/trgt/tools01.dbf';
```

In this case, RMAN creates a datafile copy of `?/oradata/trgt/tools01.dbf` named `/tmp/tools01.dbf` and records it in the repository. To change the name for datafile `?/oradata/trgt/tools01.dbf` to `/tmp/tools01.dbf` in the control file, run a `SWITCH` command so that RMAN considers the restored file as the current database file. For example:

```
SWITCH DATAFILE '/tmp/tools01.dbf' TO DATAFILECOPY '?/oradata/trgt/tools01.dbf';
```

The `SWITCH` command is equivalent to the SQL statement `ALTER DATABASE RENAME FILE`.

See Also: *Oracle9i Recovery Manager Reference* for `SET NEWNAME` syntax, and *Oracle9i Recovery Manager Reference* for `SWITCH` syntax

File Selection in Restore Operations

RMAN uses the repository to select the best available backup sets or image copies for use in the restore operation. It gives preference to image copies rather than backup sets. When multiple choices are available, RMAN uses the most current backup sets or copies, taking into account whether you specified an `UNTIL` clause in the `RESTORE` command.

All specifications of the `RESTORE` command must be satisfied before RMAN restores a backup set or file copy. Unless limited by the `DEVICE TYPE` clause, the `RESTORE` command searches for backups and copies on all device types of configured channels.

If no available backup or copy in the repository satisfies all the specified criteria, then RMAN returns an error during the compilation phase of the restore job. If you manually allocate channels, and the file cannot be restored because no backup sets or datafile copies exist on the device types allocated in the job, then create a new job specifying channels for devices containing the existing backup sets or copies. This problem does not occur when you configure automatic channels.

See Also: "Configuring Automatic Channels" on page 8-10 to learn how to configure automatic channels

Restore Optimization

By default, RMAN does not perform a restore if the file to be restored is in the correct place and its header contains the expected information. In releases prior to Oracle9*i*, RMAN always restored the requested files. In Oracle9*i*, RMAN only restores a file if the header check does not succeed, although you can use the `FORCE` option of the `RESTORE` command to override this behavior and restore the requested files unconditionally.

Note: Restore optimization only checks the datafile header and does not the scan the datafile body for corrupted blocks.

Restore optimization is particularly useful in cases where a restore only partially completes. For example, assume that a full database restore encounters a power failure after all except one of the datafiles has been restored. If you start the same restore again, then RMAN only restores the single datafile that was not restored during the previous attempt.

Automatic Location of Backups When Restoring in Real Application Clusters

RMAN automatically discovers which nodes of an Oracle Real Application Clusters configuration contain the backups, control file copies, or datafile copies that you want to restore. So long as you configured or manually allocated channels that connect to each node in the cluster, RMAN hunts for files on all channels and restores files only from those channels that locate the backup or copy on tape or on a local file system.

For example, assume that you configure a three-node cluster. If a channel connected to node 1 backs up an archived log to a local tape drive, then RMAN does not attempt to use channels connected to node 2 or node 3 to restore this log.

See Also: *Oracle9i Recovery Manager Reference* for description of `RESTORE` behavior in a Real Application Clusters configuration

Datafile Media Recovery with RMAN

The concepts of **datafile media recovery** is the application of online or archived redo logs or incremental backups to a restored datafile in order to update it to the current time or some other specified time. Use the RMAN `RECOVER` command to perform media recovery and apply logs or incremental backups automatically.

This section contains these topics:

- RMAN Media Recovery: Basic Steps
- Mechanics of Recovery: Incremental Backups and Redo Logs
- Incomplete Recovery
- Tablespace Point-in-Time Recovery
- Disaster Recovery with a Control File Autobackup

RMAN Media Recovery: Basic Steps

If possible, make the recovery catalog available to perform the media recovery. If it is not available, then RMAN uses metadata from the target database control file. If both the control file and recovery catalog are lost, then you can still recover the database—assuming that you have backups of the datafiles and at least one autobackup of the control file.

The generic steps for media recovery using RMAN are as follows:

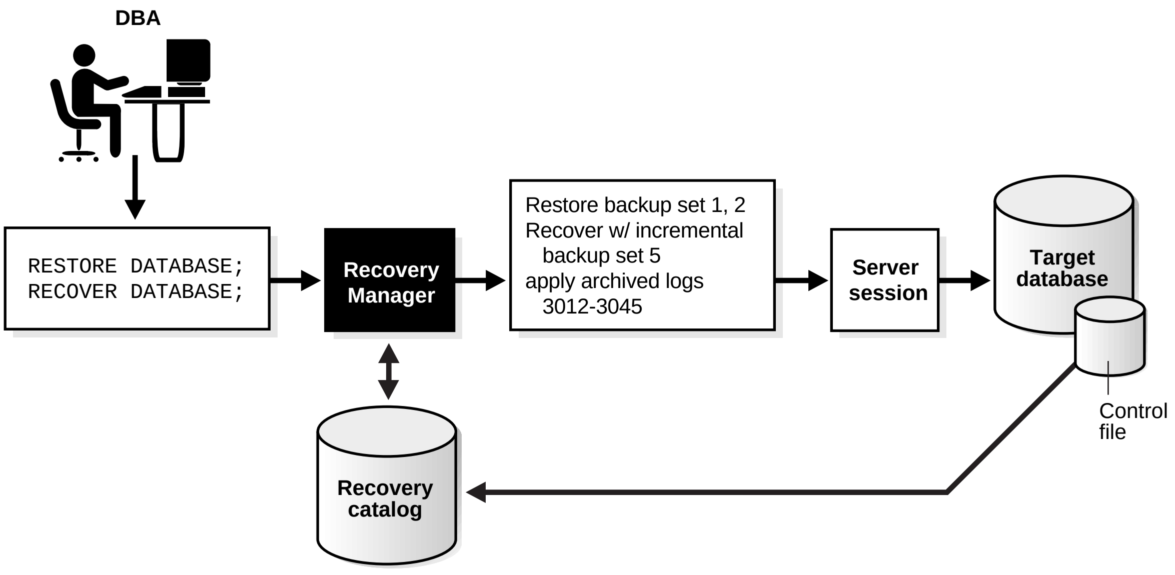
1. Place the database in the appropriate state: mounted or open. For example, mount the database when performing whole database recovery, or open the database when performing online tablespace recovery.
2. To perform incomplete recovery, use the `SET UNTIL` command to specify the time, SCN, or log sequence number at which recovery terminates. Alternatively, specify the `UNTIL` clause on the `RESTORE` and `RECOVER` commands.
3. Restore the necessary files using the `RESTORE` command.
4. Recover the datafiles using the `RECOVER` command.
5. Place the database in its normal state. For example, open it or bring recovered tablespaces online.

Figure 6–1 illustrates an example of RMAN media recovery. The DBA runs the following commands:

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

RMAN then queries the repository, which in this example is a recovery catalog. The recovery catalog obtains its metadata from the target database control file. RMAN then decides which backup sets to restore, and which incremental backups and archived logs to use for recovery. A server session on the target database instance performs the actual work of restore and recovery.

Figure 6–1 Performing RMAN Media Recovery



See Also:

- Chapter 10, "Restoring and Recovering with Recovery Manager" for detailed restore and recovery procedures
- "Control File and Server Parameter File Autobackups" on page 5-47
- *Oracle9i Recovery Manager Reference* for `RESTORE` syntax
- *Oracle9i Recovery Manager Reference* for `RECOVER` syntax

Mechanics of Recovery: Incremental Backups and Redo Logs

If RMAN has a choice between applying an incremental backup or applying redo to the restored datafiles, then it always chooses to use an incremental backup. If overlapping levels of incremental backup are available, then RMAN automatically chooses the one covering the longest period of time.

Note that RMAN does not need to apply incremental backups to a restored level 0 incremental backup: it can also apply archived logs. RMAN simply restores the datafiles that it needs from available backups and copies, and then applies incremental backups to the datafiles if it can and if not applies logs.

How RMAN Searches for Archived Redo Logs During Recovery

If RMAN cannot find an incremental backup, then it looks in the repository for the names of archived redo logs to use for recovery. Oracle records an archived log in the control file whenever one of the following occurs:

- The archiver process archives a redo log
- RMAN restores an archived log
- The RMAN `COPY` command copies a log
- The RMAN `CATALOG` command catalogs a user-managed backup of an archived log

RMAN propagates archived log data into the recovery catalog during resynchronization, classifying archived logs as image copies. You can view the log information through:

- The `LIST` command
- The `V$ARCHIVED_LOG` control file view
- The `RC_ARCHIVED_LOG` recovery catalog view

During recovery, RMAN looks for the needed logs using the filenames specified in the `V$ARCHIVED_LOG` view. If the logs were created in multiple destinations or were generated by the `COPY`, `CATALOG`, or `RESTORE` commands, then multiple, identical copies of each log sequence number exist on disk. RMAN does not have a preference for one copy over another during recovery: all copies of a log sequence number listed as `AVAILABLE` are candidates. In a sense, RMAN is blind to the fact that the logs were generated in different destinations or in different ways.

If the RMAN repository indicates that a log has been deleted or uncataloged, then RMAN ceases to consider it as available for recovery. For example, assume that the database archives log 100 to directories `/dest1` and `/dest2`. The RMAN repository indicates that `/dest1/log100.arc` and `/dest2/log100.arc` exist. If you delete `/dest1/log100.arc` with the `DELETE` command, then the repository indicates that only `/dest2/log100.arc` is available for recovery.

If the RMAN repository indicates that no copies of a needed log sequence number exist on disk, then RMAN looks in backups and restores archived redo logs as needed to perform the media recovery. By default, RMAN restores the archived redo logs to the first local archiving destination specified in the initialization parameter file. You can run the SET ARCHIVELOG DESTINATION command to specify a different restore location. If you specify the DELETE ARCHIVELOG option on RECOVER, then RMAN deletes the archived logs after restoring and applying them. If you also specify MAXSIZE *integer* on the RECOVER command, then RMAN staggers the restores so that they consume no more than *integer* amount of disk space at a time.

RMAN Behavior When the Repository Is Not Synchronized

If an archived log is deleted from disk and the repository does not reflect this fact, then RMAN does not perform automatic failover during recovery. For example, if the repository indicates that /dest1/log100.arc is on disk when in fact this log was deleted using an operating system command, and RMAN attempts to apply this log file during recovery, then recovery terminates with an error. RMAN does not automatically attempt to apply other copies of log 100 that are listed as available in the repository.

This situation can sometimes occur when you delete an archived log using an operating system utility and then fail to run a CROSSCHECK to synchronize the repository. If you run a CROSSCHECK so that the repository is synchronized, then recovery can proceed by applying available copies of the log or restoring a backup of the log if no disk copies are available.

See Also: *Oracle9i Recovery Manager Reference* for SET syntax

Incomplete Recovery

RMAN can perform either complete or incomplete recovery. You can specify a time, SCN, or log sequence number as a limit for incomplete recovery with the SET UNTIL command or with an UNTIL clause specified directory on the RESTORE and RECOVER commands. The easiest method is run the SET UNTIL command before issuing the RESTORE and RECOVER commands. After performing incomplete recovery, you must open the database with the RESETLOGS option.

See Also: *Oracle9i Recovery Manager Reference* for the UNTIL clause syntax

Tablespace Point-in-Time Recovery

Recovery Manager automated Tablespace Point-in-Time Recovery (TSPITR) enables you to recover one or more tablespaces to a point in time that is different from that of the rest of the database. RMAN TSPITR is most useful in these cases:

- To recover from an erroneous drop or truncate table operation
- To recover a table that has become logically corrupted
- To recover from an incorrect batch job or other DML statement that has affected only a subset of the database
- In cases where there are multiple logical schemas in separate tablespaces of one physical database, and where one schema must be recovered to a point different from that of the rest of the physical database
- For VLDBs (very large databases), even if a full database point-in-time recovery would suffice, you might choose to do tablespace point-in-time recovery rather than restore the whole database from a backup and perform a complete database roll forward

Similar to a table export, RMAN TSPITR enables you to recover a consistent data set; however, the data set is the entire tablespace rather than a single object.

> **See Also:** Chapter 11, "Performing RMAN Tablespace Point-in-Time Recovery" to learn how to perform TSPITR using RMAN

Disaster Recovery with a Control File Autobackup

Assume that you lose both the target database and the recovery catalog. All that you have remaining is a tape with RMAN backups of the target database and archived redo logs. Can you still recover the database? Yes, assuming that you enabled the control file autobackup feature. In a disaster recovery situation, RMAN can determine the name of a control file autobackup even without a repository available. You can then restore this control file, mount the database, and perform media recovery.

> **See Also:** "Performing Disaster Recovery" on page 10-35, and "Enabling and Disabling the Control File Autobackup" on page 8-18

Block Media Recovery with RMAN

This section contains these topics:

- About Block Media Recovery
- When Block Media Recovery Should Be Used
- Block Media Recovery When Redo Is Missing

About Block Media Recovery

Although datafile media recovery is the principal form of recovery, you can also use the RMAN BLOCKRECOVER command to perform **block media recovery**. Block media recovery recovers an individual corrupt datablock or set of datablocks within a datafile. In cases when a small number of blocks require media recovery, you can selectively restore and recover damaged blocks rather than whole datafiles.

Block media recovery provides the several advantages over datafile media recovery. For example, block media recovery

- Lowers the **Mean Time to Recovery (MTTR)** because only blocks needing recovery are restored and only necessary corrupt blocks undergo recovery. Block media recovery minimizes redo application time and avoids I/O overhead during recovery.
- Allows affected datafiles to remain online during recovery of the blocks. Without block-level recovery, if even a single block is corrupt you must restore a backup of the entire datafile and apply all redo generated for that file after the backup was created.

Note these restrictions of block media recovery:

- You can only perform block media recovery with Recovery Manager. No SQL*Plus recovery interface is available.
- You can only perform complete recovery of individual blocks. In other words, you cannot stop recovery before all redo has been applied to the block.
- You can only recover blocks marked media corrupt. The V$DATABASE_BLOCK_CORRUPTION view indicates which blocks in a file were marked corrupt since the most recent BACKUP, BACKUP ... VALIDATE, or COPY command was run against the file.
- You must have a full RMAN backup. Incremental backups are not allowed.
- Blocks that are marked media corrupt are not accessible to users until recovery is complete. Any attempt to use a block undergoing media recovery results in an error message indicating that the block is media corrupt.

> **See Also:** "Recovering Datablocks with RMAN" on page 10-38 and *Oracle9i Recovery Manager Reference* for BLOCKRECOVER syntax

When Block Media Recovery Should Be Used

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. Block media recovery is not a replacement for traditional datafile media recovery, but a supplement to it.

In most cases, Oracle marks a block as media corrupt, invalidates the block in the instances (or all enabled instances in an Oracle Real Application Clusters configuration), 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. This corrupt status effectively takes the block offline in all database instances and prevents user access during recovery.

Block media recovery is most useful for data losses that affect specific 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. Typically, these types of block corruption are reported in these locations:

- Oracle 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'
```

You can then specify the corrupt blocks in the `BLOCKRECOVER` command as follows:

```
BLOCKRECOVER DATAFILE 7 BLOCK 3 DATAFILE 2 BLOCK 235;
```

Block Media Recovery When Redo Is Missing

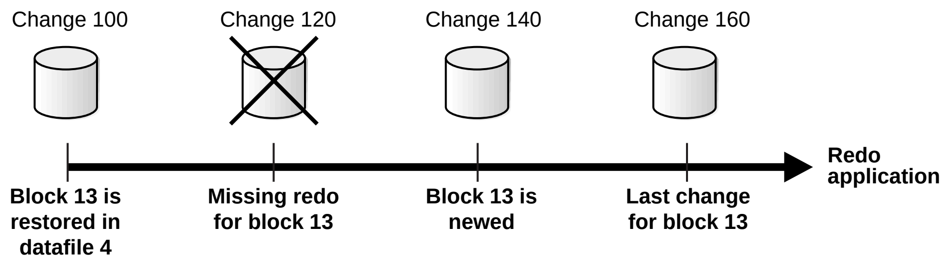
Like datafile media recovery, block media recovery cannot survive a missing or inaccessible archived log. Nevertheless, block media recovery can survive gaps in the redo stream if the missing or corrupt redo records do not affect the blocks being recovered. Whereas datafile recovery requires an unbroken 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, Oracle can new a block when users delete all the rows recorded in the block or drop a table.

Assume that media recovery is performed on block 13 as depicted in Figure 6–2.

Figure 6–2 Performing RMAN Media Recovery



After block recovery begins, RMAN discovers that change 120 is missing. RMAN does not terminate 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 `EMPLOYEE` stored in block 13. At this point, Oracle formats block 13 as a new block. Because the redo for block 13 in change 120 related to the `EMPLOYEE` table, and the `EMPLOYEE` table was dropped in change 140, RMAN can skip this missing change and apply the redo between changes 140 and 160.

Database Duplication with RMAN

Use the RMAN `DUPLICATE` command to create a copy of the target database that you can use for testing. The command restores backups and copies of the primary database's files and creates a new database.

As part of the duplicating operation, RMAN manages the following:

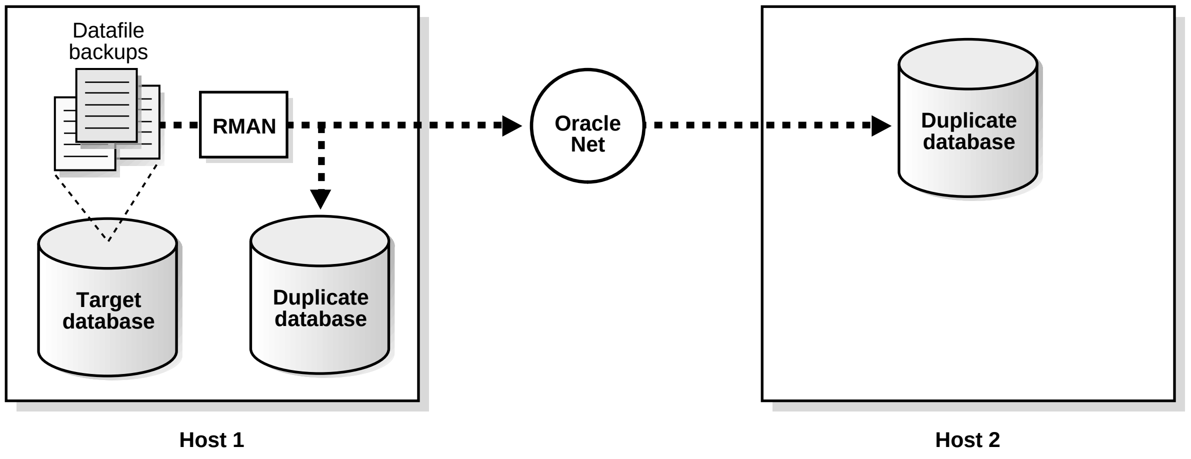
- Restores the target datafiles into the duplicate database and performs incomplete recovery using all available archived log and incremental backups.
- Opens the duplicate database with the `RESETLOGS` option after incomplete recovery to create the online redo logs.
- Generates a new, unique database identifier for the duplicate database.

Note also the following features of RMAN duplication. You can:

- Skip read-only tablespaces with the `SKIP READONLY` clause (read-only tablespaces are included by default). You can also exclude any tablespace with the `SKIP TABLESPACE` clause so long as it is not the `SYSTEM` tablespace and does not contain rollback or undo data. If you omit tablespaces, then you can add them later.
- Create your duplicate database in a new host. If the same directory structure is available, then you can use the `NOFILENAMECHECK` option and reuse the target datafile filenames for the duplicate datafiles.
- Create your duplicate database by using the `SET UNTIL` command or `UNTIL` clause of the `DUPLICATE` command to recover it to a noncurrent time. By default, the `DUPLICATE` command creates the database using the most recent backups of the target database and then performs recovery to the most recent consistent point contained in the incremental and archived redo log backups.
- Use the duplicate database without a recovery catalog.
- Register the duplicate database in the same recovery catalog as the target database. This option is possible because the duplicate database receives a new database identifier during duplication. If you copy the target database using operating system utilities, then the database identifier of the copied database remains the same so you cannot register it in the same recovery catalog.

Figure 6–3 illustrates a case of database duplication. In this example, RMAN creates two duplicate database by using one set of datafile backups: one database on the local host and one database a remote host.

Figure 6–3 Creating a Duplicate Database from Backups



The method you use to duplicate your database depends on whether you are creating your duplicate database on the same or a different host and whether the duplicate directory structure is the same as your target database directory structure. For example, in some cases you can keep the same directory structure and filenames in your duplicate database, while other times you must rename the files.

> **See Also:** Chapter 12, "Duplicating a Database with Recovery Manager" to learn how to make a duplicate database, and *Oracle9i Recovery Manager Reference* for `DUPLICATE` command syntax.

Standby Database Creation with RMAN

You can use the Recovery Manager `DUPLICATE` command to create a standby database. RMAN automates the following steps of the creation procedure:

1. Restores the standby control file.
2. Restores the primary datafile backups and copies.
3. Optionally, RMAN recovers the standby database (after the control file has been mounted) up to the specified time or to the latest archived redo log generated.
4. RMAN leaves the database mounted so that the user can activate it, place it in manual or managed recovery mode, or open it in read-only mode.

RMAN cannot fully automate creation of the standby database because you must manually create an initialization parameter file for the standby database, start the standby instance without mounting the control file, and perform any Oracle Net setup required before performing the creation of the standby. Also, you must have RMAN backups of all datafiles available as well as a control file backup that is usable as a standby control file.

RMAN can back up the standby database and archived redo logs. These standby backups are fully interchangeable with primary backups. In other words, you can restore a backup of a standby datafile to the primary database, and you can restore a backup of a primary datafile to the standby database.

See Also:

- Chapter 13, "Creating a Standby Database with Recovery Manager" to learn how to create a standby database with RMAN
- "Backing Up Files at a Standby Database Site with RMAN" on page 9-20 to learn how to back up a standby database
- *Oracle9i Data Guard Concepts and Administration* to learn how to create a standby database without using RMAN

7

RMAN Concepts III: Maintenance

This chapter describes the basic concepts involved in using the Recovery Manager (RMAN) utility.

This chapter contains these topics:

See Also: Chapter 18, "Performing Maintenance with Recovery Manager" to learn how to perform RMAN maintenance jobs

RMAN Reporting Functionality

The RMAN repository, which is either a recovery catalog or the target database control file, contains a wealth of metadata about backups and copies as well as other useful things such as database schema and configuration settings. You can use RMAN commands `LIST`, `REPORT`, and `SHOW` to access this repository information.

This section contains these topics:

- LIST Command Output
- REPORT Command Output
- SHOW Command Output
- PRINT SCRIPT Command Output

LIST Command Output

The `LIST` command queries the RMAN repository (recovery catalog or control file) and produces a record of its contents. Use this command to obtain data about:

- RMAN-generated files, that is, backup sets, proxy copies, and image copies
- Specified objects contained in the RMAN-generated files, that is, archived logs, datafiles, and control files
- Incarnations of a specified database or of all databases known to the recovery catalog

RMAN records the output to either standard output or the message log, but not to both at the same time. You can also control how the output is organized as well as the level of detail in the output. Running the `LIST` command is usually preferable to querying `V$` or recovery catalog views.

The `LIST` command displays the same files that the `CROSSCHECK` and `DELETE` commands operate on. Consequently, you can issue `LIST` to see what is in the repository, and then run `CROSSCHECK` to ensure that these files exist on disk or tape.

See Also:

- "Listing RMAN Backups, Copies, and Database Incarnations" on page 17-3 to learn how to generate lists
- "Querying the Recovery Catalog Views" on page 17-23 and "Querying the Recovery Catalog Views" on page 17-23 to learn how to use views as an alternative to `LIST`
- *Oracle9i Recovery Manager Reference* for `LIST` command syntax
- *Oracle9i Recovery Manager Reference* for `LOG` command-line syntax

REPORT Command Output

This section contains these topics:

- About RMAN Reports
- Reports of Obsolete Backups
- Reports of Orphaned Backups

About RMAN Reports

RMAN reports are intended to provide analysis of your backup and recovery situation. An RMAN report can answer questions such as:

- Which datafiles need a backup?
- Which backups and copies are obsolete because they are redundant or because they are not needed for recovery within a recovery window?
- Which datafiles are not recoverable because of unrecoverable operations performed on them?
- What is the current physical schema of the database or what was it at some previous time?
- Which backups are **orphaned**, that is, unusable in a restore operation, because they belong to incarnations of the database that are not direct predecessors of the current incarnation?

Run the `REPORT NEED BACKUP` and `REPORT UNRECOVERABLE` commands regularly to ensure that the necessary backups are available to perform media recovery, as well as to ensure that you can perform media recovery within a reasonable amount of time. Also, run REPORT OBSOLETE regularly so that you can delete unnecessary files and conserve disk space.

Note: A datafile that does not have a backup is not considered unrecoverable. You can recover such datafiles through the CREATE DATAFILE statement if redo logs starting from when the file was created still exist.

Reports of Obsolete Backups

The REPORT OBSOLETE command displays backups and copies of datafiles, control files, and archived redo logs that can be deleted because they are no longer needed. You can define what makes a file obsolete in the following mutually exclusive ways:

Parameter	Meaning
REDUNDANCY *integer*	At least *integer* more recent backups or image copies of this file already exist.
RECOVERY WINDOW *integer*	The backup or copy is not needed for recovery to a random point within the recovery window of *integer* days. For each datafile, one backup that is older than the recovery window must exist. In other words, one backup of each datafile must satisfy the condition SYSDATE - CHECKPOINT_TIME >= RECOVERY WINDOW. All backups older than the most recent backup that satisfies this condition are obsolete.

In addition to obsolete datafile backups, RMAN reports obsolete archived logs and archived log backups. Regardless of which parameter is specified, RMAN uses this setting to determine which backups and copies of datafiles are no longer needed, which in turn determines when archived logs (and backups of archived logs) are no longer needed. The creation of a datafile is considered as a backup when deciding which logs to keep.

You can specify the REDUNDANCY and RECOVERY WINDOW parameter options in the following places:

- The REPORT OBSOLETE command with the REDUNDANCY or RECOVERY WINDOW clauses
- The CONFIGURE RETENTION POLICY command with the REDUNDANCY or RECOVERY WINDOW clauses

When generating a report, the REPORT OBSOLETE options override the CONFIGURE RETENTION POLICY options. For example, if you configure a retention policy to a recovery window of 7 days, and then run REPORT OBSOLETE REDUNDANCY 2, the REPORT OBSOLETE command displays backups and copies that are obsolete because they have REDUNDANCY 2. Note that the configured retention policy is still in effect: the REPORT command simply displays what is obsolete given the options that specified in the command. If you run REPORT OBSOLETE and specify neither the REDUNDANCY nor RECOVERY WINDOW options, then RMAN displays what is obsolete given the configured retention policy.

If you disable the retention policy, that is, if you run CONFIGURE RETENTION POLICY TO NONE, then RMAN does not consider any backups or copies as obsolete. If you then run REPORT OBSOLETE with no other options, then RMAN issues an error message because no retention policy is configured to mark files as obsolete.

Note: An obsolete backup differs from an expired backup. An obsolete backup is no longer needed according to the user's retention policy. An expired backup is a backup that the CROSSCHECK command fails to find on the specified media device.

See Also: *Oracle9i Recovery Manager Reference* for CONFIGURE command syntax

Reports of Orphaned Backups

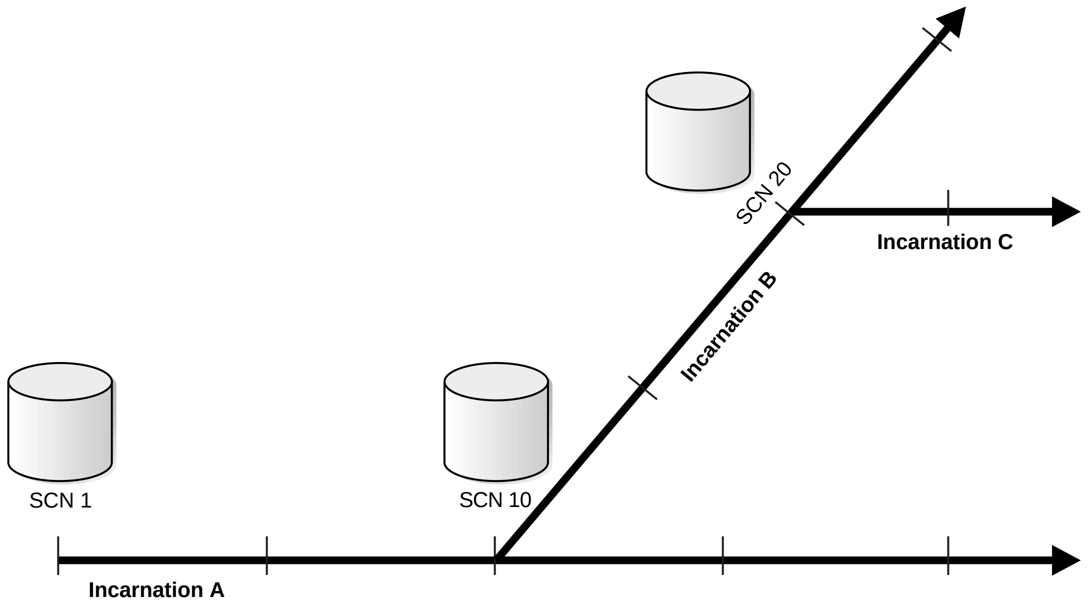
The REPORT OBSOLETE ORPHAN command displays **orphaned backups**. Orphaned backups are backups that are unusable because they belong to incarnations of the database that are not direct ancestors of the current incarnation.

A new incarnation of a database occurs whenever you open a database with the RESETLOGS option, which resets the current redo log to log sequence 1. The RESETLOGS option causes Oracle to write a new RESETLOGS SCN into the control file and datafile headers. To prevent possible redo corruption, archived redo logs from a database with one RESETLOGS SCN cannot be applied to a database with a different RESETLOGS SCN. Although the database is the same, that is, it has the same DBID, the database can go through different incarnations depending on how often a RESETLOGS operation is performed.

RMAN can reset a database to a previous incarnation. For example, if you perform a RESETLOGS operation on database prod1 on Tuesday, you can use RMAN to reset the current database incarnation to Monday's incarnation and restore backups and archived redo logs associated with it. For this reason, backups from previous incarnations are in some cases still usable.

The situation can become complicated when a database goes through multiple incarnations. In such cases, some backups can become unusable, that is, orphaned. You should delete orphaned backups to conserve disk or tape space. The incarnation scenario depicted in Figure 7–1 shows a database that goes through three incarnations.

Figure 7–1 Orphaned Backups



Incarnation A of the database started at SCN 1. At SCN 10, assume that you performed a `RESETLOGS` operation and created incarnation B. At SCN 20, you performed another `RESETLOGS` operation on incarnation B and created a new incarnation C.

The following table explains which backups are orphans depending on which incarnation is current.

Current Incarnation	Usable Backups (Nonorphaned)	Orphaned Backups
Incarnation A	All backups from incarnation A	All backups from incarnations B and C
Incarnation B	■ All backups from incarnation A prior to SCN 10 ■ All backups from incarnation B	■ Backups from incarnation A after SCN 10. ■ All backups from incarnation C
Incarnation C	■ All backups from incarnation A prior to SCN 10 ■ All backups from incarnation B prior to SCN 20 ■ All backups from incarnation C	■ All backups from incarnation A after SCN 10 ■ All backups from incarnation B after SCN 20

See Also: "Reporting on Backups, Copies, and Database Schema" on page 17-10 to learn how to generate reports, and *Oracle9i Recovery Manager Reference* for `REPORT` syntax

SHOW Command Output

The `SHOW` command can display any configuration set by the `CONFIGURE` command. For example, to display the `CONFIGURE CHANNEL` settings, run `SHOW CHANNEL`. You can also run `SHOW ALL` to display all current configurations.

See Also: "Showing RMAN Configuration Settings" on page 17-15 to learn how to show RMAN configurations, and *Oracle9i Recovery Manager Reference* for `SHOW` syntax

PRINT SCRIPT Command Output

The `PRINT SCRIPT` command displays the text of a specified RMAN script stored in the recovery catalog. Note that this command does not display the list of all the scripts stored in the catalog. To see the list of all catalog scripts, query the recovery catalog view `RC_STORED_SCRIPT_LINE`.

See Also: "Printing Scripts Stored in the Recovery Catalog" on page 17-21 to learn how to display stored scripts, and *Oracle9i Recovery Manager Reference* for `PRINT SCRIPT` syntax

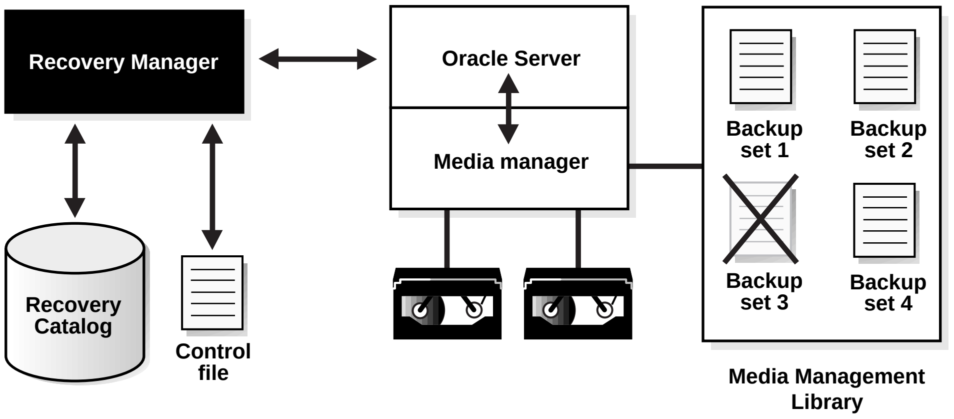
Crosschecks of RMAN Backups and Copies

Sometimes backups and copies disappear from disk, or tapes in the media management library become unavailable. For example, someone may inadvertently delete backup pieces from disk, or one of the tapes used by the media manager may become corrupted.

To ensure that data about backup sets and image copies in the recovery catalog or control file is synchronized with actual files on disk or in the media management catalog, perform a **crosscheck**. The `CROSSCHECK` command operates only on files that are recorded in the RMAN repository.

Figure 7–2 illustrates a crosscheck of the media manager. RMAN queries the repository (recovery catalog or target database control file) 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 7–2 Crosschecks



Crosschecks are useful because they can

- Update outdated information about backups and copies that disappeared from disk or tape or became corrupted
- Update the repository if you delete archived redo logs or other files using operating system commands

Use the crosscheck feature to check the status of a backup or copy on disk or tape. If the backup or copy 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.

Backup sets, backup pieces, and copies can have the status `AVAILABLE`, `EXPIRED`, or `UNAVAILABLE`. You can view the status information in the output of the `LIST` command and the recovery catalog views.

Note that you can issue the `DELETE EXPIRED` command to delete all expired backups and copies. 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.

Note: The `CROSSCHECK` command does *not* delete operating system files or remove repository records. You must use the `DELETE` command for these operations.

See Also:

- "Crosschecking Backups and Copies" on page 18-2 to learn how to perform crosschecks
- *Oracle9i Recovery Manager Reference* for CROSSCHECK syntax and a description of the possible status values
- *Oracle9i Recovery Manager Reference* for DELETE syntax

Deletion of RMAN Backups and Copies

This section contains these topics:

- About RMAN Deletions
- Summary of RMAN Deletion Methods
- Removal of Backups and Copies with the DELETE Command
- Behavior of DELETE Command When the Repository and Media Do Not Correspond
- Removal of Backups and Copies with the BACKUP ... DELETE INPUT Command

About RMAN Deletions

Every RMAN backup or copy produces a corresponding record in the control file and, if used, recovery catalog. For example, if you generate a full database backup set, then you can view the record for this backup set in the V$BACKUP_SET control file view. If you use a recovery catalog, then you can also see the record in the RC_BACKUP_SET catalog view.

The V$ control file views and catalog tables differ in the way that they store information, and this affects how RMAN handles repository records. The RMAN information stored in the control file is not stored in an actual table, although the V$ views display it in tabular format. The data is stored in an internal data structure in the circular reuse section of the control file. In contrast, the recovery catalog data is stored in actual database tables.

When you use an RMAN command to delete a backup or copy, RMAN performs the following steps:

- Removes the physical file from the operating system
- Updates the backup or copy records in the control file to status DELETED

- Removes the backup or copy records from the catalog tables (if a catalog is used)

Because of the way that control file data is stored, RMAN cannot delete the record from the control file, only update it to DELETED status. However, because the catalog tables are ordinary database tables, RMAN removes rows from them.

Summary of RMAN Deletion Methods

Table 7–1 describes the functionality of the various RMAN deletion commands and scripts. None of these commands or scripts requires a recovery catalog.

Table 7–1 Maintenance Commands and Scripts **(Page 1 of 2)**

Command or Script	Purpose
`DELETE`	To delete physical backups and image copies, update the control file records to status `DELETED`, and remove their records from the catalog (if a catalog is used). You can specify that `DELETE` should remove backups or copies that are `EXPIRED` or `OBSOLETE`. If you run `DELETE EXPIRED` on an object that exists, RMAN issues a warning and does not delete the object. You can override this behavior and delete the object by running `DELETE FORCE`.
`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 copy that it backs up. If you specify `ALL`, then RMAN deletes all copies of the specified logs that it finds in the `V$ARCHIVED_LOG` view.
`CHANGE ... UNCATALOG`	To delete recovery catalog records for specified backups and copies and change their control file records to status `DELETED`. Note that the `CHANGE ... UNCATALOG` command does not delete files from the operating system.

Table 7–1 Maintenance Commands and Scripts **(Page 2 of 2)**

Command or Script	Purpose
`prgrmanc.sql`	To remove all records of backups or copies with status `DELETED` from the recovery catalog. Prior to release 8.1.6, RMAN sometimes updated recovery catalog records to `DELETED` status rather than removing them entirely (as it does in the current release). **Note:** If the version of your recovery catalog is the same version as the current release of RMAN, then the only way that the recovery catalog could have records marked `DELETED` is if the catalog has been upgraded from a catalog prior to release 8.1.6, or the catalog was resynchronized from a backup control file created prior to release 8.1.6.

See Also: "Crosschecks of RMAN Backups and Copies" on page 7-8

Removal of Backups and Copies with the DELETE Command

The `DELETE` command can remove any file that the `LIST` and `CROSSCHECK` commands can operate on. For example, you can delete backup sets, archived redo logs, and datafile copies. The `DELETE` command removes both the physical file and the catalog record for the file.

Advantage of Using DELETE Instead of Operating System Commands

Always use `DELETE` rather than an operating system utility to remove RMAN backups and copies. If you do not, then the RMAN repository is not synchronized with what exists on the file system or on tape. If you remove files using an operating system utility, you can

- Run `CROSSCHECK` to change the status of these files to `EXPIRED` and then run `DELETE EXPIRED`
- Run `DELETE` on the files that you deleted with the operating system utility
- Run `CHANGE ... UNCATALOG` to remove the catalog records

Deletion of Obsolete Backups and Copies

The `DELETE OBSOLETE` command provides a convenient way to physically delete backups and copies that are no longer needed, as well as remove their catalog records. It uses the same `REDUNDANCY`, `RECOVERY WINDOW`, and `ORPHAN` options as the `REPORT OBSOLETE` command.

If you have configured a retention policy, then you can run `DELETE OBSOLETE` periodically to delete all backups and copies considered obsolete by this policy. For example, you can run `DELETE OBSOLETE` as a script every night using a scheduling utility. In this way, you keep your disk drives uncluttered by removing all unnecessary files.

See Also: "Reports of Obsolete Backups" on page 7-4

Deletion of Expired Backups

The `CROSSCHECK` command marks a backup or copy as expired when it cannot locate it at the location to which it was backed up. In short, *expired* means "not found." You can view the expired backups and copies by running the `CROSSCHECK` command as in the following example:

```
RMAN> CROSSCHECK BACKUP;

crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=0ad8d32i_1_1 recid=10 stamp=445025363
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-1334876723-20011105-00 recid=11 stamp=445025367
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=0cd8d361_1_1 recid=12 stamp=445025473
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-1334876723-20011105-01 recid=13 stamp=445025475
Crosschecked 4 objects
```

The `DELETE EXPIRED` command removes the recovery catalog records for expired backups and copies, and updates their control file records to status `DELETED`.

This command is especially useful if you inadvertently delete RMAN backups or archived logs from disk using an operating system utility. If you do this, then the RMAN repository is not synchronized with the physical reality of what is on disk. By running the `CROSSCHECK` command, RMAN marks the backups and copies that it cannot find as expired. Then, you can run `DELETE EXPIRED` to remove the records for these files.

Deletion of Archived Redo Logs That Are Already Backed Up

You may want to delete files such as archived logs only if they have been backed up a specified number of times to tape. The `DELETE` command supports this behavior. The following example deletes all backups and copies (archived logs are considered as copies) that have already been backed up at least two times to tape:

```
DELETE ARCHIVELOG ALL BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

Behavior of DELETE Command When the Repository and Media Do Not Correspond

The repository record for an object can sometimes fail to reflect the physical status of the object. For example, you back up a log to disk and then use an operating system utility to delete the object. If you do not run the `CROSSCHECK` command to update the repository, and if you then run `DELETE` against the object, then the repository shows that the object is `AVAILABLE` while the object is in fact missing. The following table indicates the behavior of `DELETE` in such situations.

Repository Status	Physical Status	Behavior of DELETE Command
`AVAILABLE`	Not found on media	Does not delete the object and reports the list of mismatched objects at the end of the job. RMAN does not update the repository status.
`EXPIRED`	Found on media	Does not delete the object and reports the list of mismatched objects at the end of the job. RMAN does not update the repository status.
`UNAVAILABLE`	Any	Removes repository record and deletes object if it exists. All I/O errors are ignored.
`FORCE`	Any	Removes repository record and deletes file if it exists. All I/O errors are ignored. RMAN displays the number of objects deleted at the end of the job.

Removal of Backups and Copies with the BACKUP ... DELETE INPUT Command

The `BACKUP ... DELETE INPUT` command can delete archived redo logs, datafile copies, and backup sets after backing them up. This functionality is especially useful when backing up archived logs on disk to tape. RMAN backs up one and only one copy of each log sequence number, and then deletes the file that it backs up. For example, assume that you issue:

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

In this command, RMAN backs up one copy of each log for each available sequence number, and then deletes only the file that it actually backs up.

If you specify the `DELETE ALL INPUT` option, then RMAN deletes whichever files match the criteria that you specify, even if there are several files of the same log sequence number. For example, assume that you archive to 3 different directories. Then, you issue this command:

```
BACKUP ARCHIVELOG ALL FROM SEQUENCE 1200 DELETE ALL INPUT;
```

In this case, RMAN backs up only one copy of each log sequence between 1200 and the most recent sequence, but deletes all logs with these sequence numbers contained in the three archive destinations.

The archived log failover feature means that RMAN searches every enabled archiving destination for good copies of a log sequence number. For example, assume that `/log1` and `/log2` are the only enabled archiving destinations, and that they contain the same sequence number. You run this command:

```
BACKUP ARCHIVELOG FROM SEQUENCE 123 DELETE ALL INPUT;
```

RMAN can start reading from any enabled archiving directory. For example, assume RMAN starts in directory `/log1` and finds `log_123.f` there. Then, if RMAN discovers that `log_124.f` is corrupt, it searches in `/log2` for a good copy of this log. Because `DELETE ALL INPUT` is specified, RMAN deletes all copies of logs on disk of sequence 123 and higher.

See Also:

- "Deleting Backups and Copies" on page 18-4 to learn how to delete backups and copies
- "Exempting a Backup or Copy from the Retention Policy" on page 18-14 to learn how to use the `CHANGE ... UNCATALOG` command
- *Oracle9i Recovery Manager Reference* for `CHANGE` syntax
- *Oracle9i Recovery Manager Reference* for `DELETE` syntax

Allocation of Multiple Channels for RMAN Maintenance Commands

You can use maintenance commands such as `CROSSCHECK` or `DELETE` on all configured device types. If you have automatic channels configured for all devices on which you have made backups, then you can simply run `CROSSCHECK` or `DELETE` at the RMAN prompt as in the following example (sample output included):

```
CROSSCHECK BACKUP;

allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: sid=14 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=/ade/lashdown_seeddb/oracle/dbs/c-1337987235-20011210-01 recid=6
```

```
stamp=448109988
Crosschecked 1 objects

crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=01dbb78r_1_1 recid=1 stamp=448109852
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=02dbb791_1_1 recid=2 stamp=448109857
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=03dbb7af_1_1 recid=3 stamp=448109904
Crosschecked 3 objects
```

RMAN uses the default configured `DISK` channel as well as a channel for the configured `sbt` device. RMAN looks for each backup on all channels that have the same device type as the channel used to create the backup. The multiple-channel crosscheck feature is designed for the following scenarios:

- To allow crosschecking or deleting of all backup pieces or proxy copies, both on disk and tape, with a single command
- To make crosschecking and deleting work correctly in an Oracle Real Application Clusters configuration in which each backup piece or proxy copy exists only on one node

To perform maintenance on files on only one device type, you can specify the device type on the maintenance command itself. For example:

```
DELETE EXPIRED BACKUP DEVICE TYPE sbt;
```

If you do not have an automatic channel configured, then you can manually allocate a maintenance channel. For example, you can run commands as follows:

```
ALLOCATE CHANNEL FOR MAINTENANCE TYPE sbt;
CROSSCHECK BACKUP;
```

See Also:

- "Crosschecking and Deleting on Multiple RMAN Channels" on page 18-8 to learn how to allocate multiple maintenance channels
- *Oracle9i Recovery Manager Reference* for `ALLOCATE CHANNEL FOR MAINTENANCE` syntax

Changes to Availability of RMAN Backups and Copies

RMAN can update the repository to show backups and copies as available or unavailable. For example, you may have several backups on a tape drive that is being upgraded or replaced. You can use the `CHANGE ... UNAVAILABLE` command to mark these backups and copies as unavailable for the duration of the maintenance on the drive. RMAN does not consider `UNAVAILABLE` backups and copies for its backup and recovery operations.

When the maintenance is complete, you can issue the `CHANGE ... AVAILABLE` command to inform RMAN that these backups and copies are now available again. After setting the files back to status `AVAILABLE`, you can also run a `CROSSCHECK` to double-check that RMAN can access the files.

See Also:

- "Changing the Availability Status of a Backup or Copy Record" on page 18-12
- *Oracle9i Recovery Manager Reference* for `CHANGE` syntax

Changes to Retention Status of RMAN Backups and Copies

RMAN can change the status of backups and copies by using the `KEEP` or `NOKEEP` options. The `KEEP` option exempts a backup or copy from the current retention policy either indefinitely or until the specified `UNTIL` time. RMAN does not mark the files as obsolete even if they would be considered obsolete under the retention policy. Such backups are called **long-term backups**. The `NOKEEP` option removes the exempt status of the files.

The `CHANGE ... KEEP` or `CHANGE ... NOKEEP` commands alter the retention status of a backup or copy. For example, this command allows the retention policy to mark backup sets with the tag `SAVE_BACKUP` as obsolete:

```
CHANGE BACKUPSET TAG save_backup NOKEEP;
```

See Also:

- "Exempting a Backup or Copy from the Retention Policy" on page 18-14
- *Oracle9i Recovery Manager Reference* for `CHANGE` syntax

Part III

Performing Backup and Recovery with Recovery Manager

Part III describes how to use the RMAN utility to perform backup and recovery operations as well as create duplicate and standby databases. It also explains how to tune RMAN and troubleshoot typical problems. This part contains these chapters:

8 Configuring the Recovery Manager Environment

This chapter describes how to perform setup and configuration tasks. This chapter contains these topics:

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 Oracle. A media manager is not an Oracle product and must be obtained from a third-party vendor. If you choose to use RMAN with a media manager, then you must obtain all product-specific information from the vendor.

This section describes the generic steps for configuring RMAN for use with a media manager. The actual steps depend on the media management product that you install and the platform on which you are running Oracle.

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

1. Prerequisites for Using a Media Manager with RMAN
2. How Oracle Interacts with the Media Manager
3. Integrating RMAN with the Media Manager: Basic Steps
4. Testing Whether the Media Manager Library Is Integrated Correctly
5. Configuring Automatic Channels for Use with a Media Manager

> **See Also:** "Media Management" on page 4-16 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 media manager, you must install it and make sure that RMAN can communicate with it. Instructions for this procedure should be available in the media manager vendor's software documentation.

Depending on the product that you are installing, the following basic steps apply:

1. Install and configure the media management software on the target host or production network. No RMAN integration is required at this stage.
2. Ensure that you can make non-RMAN backups of operating system files on the target database host. This step makes later troubleshooting much easier. Refer to your media management documentation to learn how to back up files to the media manager.
3. Obtain and install the third-party media management module for integration with the Oracle server. This module must contain the library that Oracle loads when accessing the media manager.

If you have not performed the preceding steps, then do not proceed with the media management configuration.

How Oracle Interacts with the Media Manager

To make backups to a media manager, RMAN must communicate with a media management library. The procedure for using RMAN with a media manager in Oracle9*i* has not changed from previous releases.

The loading of a media manager library is operating system specific. When you RMAN allocates channels, it uses the following algorithm when deciding which library to load:

1. Attempts to load the library indicated by the `SBT_LIBRARY` parameter in the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command. If the `SBT_LIBRARY` parameter is not specified, then Oracle proceeds to the next step.
2. Attempts to load the default media management library. The filename of the default library is operating system specific. On UNIX, the library filename is `$ORACLE_HOME/lib/libobk.so`, with the extension name varying according to platform: `.so`, `.sl`, `.a`, and so forth. On Windows NT the library is named `%ORACLE_HOME%\bin\orasbt.dll`.

   Note: The default media management library file is *not* part of the standard Oracle installation as in previous releases, so you can only create it by installing third-party media management software.

3. If Oracle is unable to locate the default library in the previous step, then RMAN issues an `ORA-27211` error and exits.

   Whenever channel allocation fails, Oracle writes a trace file to the `USER_DUMP_DEST` 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
```

Integrating RMAN with the Media Manager: Basic Steps

After you install the media management software, the media management library should already be integrated with the Oracle server. You should not have to perform further integration. However, you may choose to follow the procedure in this section to ensure that the media manager is integrated correctly.

Because the specifics of your media management integration with Oracle depends on both the product and the operating system, this section cannot cover all possible cases. This section describes the basic procedures for naming the media management library correctly on UNIX and Windows NT so that Oracle can load it.

Integrating RMAN with a Media Manager on UNIX: Basic Steps

On UNIX, Oracle accesses the media management library through the UNIX shared library `libobk.so`. This file must exist somewhere in the system path. It is highly recommended that you place `libobk.so` in `$ORACLE_HOME/lib`, which is where Oracle searches first.

You do not need to start or shut down the instance when installing the media management library.

To integrate the media manager on UNIX:

1. If an old `libobk.so` symbolic link already exists in `$ORACLE_HOME/lib`, then remove it *before* installing the media manager. For example:

   ```
   % rm $ORACLE_HOME/lib/libobk.so
   ```

2. After installation, check your media management vendor documentation to determine where the media management library is installed. For example, suppose that the library is installed as `/vendor/lib/oracle_lib.so`.
3. Either change the name of the installed media management library to `$ORACLE_HOME/lib/libobk.so`, or created a symbolic link to the library called `libobk.so`. For example, you can create a symbolic link to the library as follows:

   ```
   % ln -s /vendor/lib/oracle_lib.so $ORACLE_HOME/lib/libobk.so
   ```

   Alternatively, you can simply change the name of the library to `libobk.so`. For example:

   ```
   % mv /vendor/lib/oracle_lib.so $ORACLE_HOME/lib/libobk.so
   ```

Integrating RMAN with a Media Manager on Windows NT: Basic Steps

On Windows NT, Oracle accesses the media management library through the library `orasbt.dll`. This file must exist somewhere in the system path. Typically, the file is located in the `%ORACLE_HOME%\bin` folder of the Oracle home.

Note that you do not need to start or shut down the instance when installing the media management library.

To integrate the media manager with RMAN on Windows NT:

1. If an `orasbt.dll` already exists in the system path, then remove it *before* installing the media manager. For example:

   ```
   D:\> del %ORACLE_HOME%\bin\orasbt.dll
   ```

2. After installation, check your media management vendor documentation to determine where the media management library is installed. For example, suppose that the library is installed as `D:\vendor\lib\oracle_lib.dll`.
3. If the installed library is not called `orasbt.dll`, then rename the installed media management library to `%ORACLE_HOME%\bin\orasbt.dll`. For example, you can copy the installed library as follows:

   ```
   D:\> copy D:\vendor\lib\oracle_lib.dll %ORACLE_HOME%\bin\orasbt.dll
   ```

Note that the `orasbt.dll` file does not have to be in the `%ORACLE_HOME\bin` folder as long as the folder containing the library is in the system `PATH` variable setting. To see the `PATH` variable setting, choose Start > Settings > Control Panel > System > Environment.

See Also:

- Your operating system specific Oracle documentation and the documentation supplied by your media vendor for instructions on how to achieve media manager integration on your platform
- "After Installation of Media Manager, RMAN Channel Allocation Fails: Scenario" on page 15-25 for troubleshooting scenarios involving media manager problems

Testing Whether the Media Manager Library Is Integrated Correctly

After you have confirmed that the Oracle server can load the media management library, test to make sure that RMAN can back up to the media manager. This test occurs in the steps described in the following scenarios:

- Configuring Media Management Software for RMAN Backups
- Testing a Channel Allocation on the Media Manager
- Testing a Backup to the Media Manager

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, configure classes, and so forth.

You should determine which `PARMS` settings are needed for the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` commands as well as the recommended `FORMAT` string for the `BACKUP` command (if needed). The `PARMS` parameter sends instructions to the media manager. For example, the following vendor-specific `PARMS` setting instructs the media manager to back up to a volume pool called `oracle_tapes`:

```
PARMS='ENV=(NSR_DATA_VOLUME_POOL=oracle_tapes)'
```

Refer to your third-party vendor documentation for the appropriate settings.

Note that you can use the substitution variables provided by RMAN to generate unique backup piece names when writing backups to a media manager. A backup piece name is determined by the `FORMAT` string specified in the `BACKUP` command, the `CONFIGURE CHANNEL` command, or the `ALLOCATE CHANNEL` command.

If you do not specify the `FORMAT` parameter, then RMAN automatically generates a unique filename using the `%U` substitution variable. The media manager considers the backup piece name as the filename of the backup file, so this name must be unique in the media manager catalog.

Note: Some media managers only support a 14-character backup piece name, and some require special `FORMAT` strings. Refer to your media management documentation to determine the string character limit for the media manager.

Some media managers have limits on the maximum size of files that they can back up or restore. File size is an issue in those situations in which RMAN multiplexes multiple datafiles into one output file, but the backup piece size is in excess of the size that the media manager or file system is able to store.

To avoid problems, see your media management documentation for operational limits on file sizes. Ensure that the files written by RMAN do not exceed the limits. To limit backup piece sizes, use the parameter `MAXPIECESIZE`, which you can set in the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands. Refer to the `*.rcv` scripts in the `demo` directory on your system, which is located in an operating system specific location (`$ORACLE_HOME/rdbms` on UNIX) for an example.

See Also:

- *Oracle9i Recovery Manager Reference* for `ALLOCATE CHANNEL` syntax
- *Oracle9i Recovery Manager Reference* for channel control options
- *Oracle9i Recovery Manager Reference* for the complete list of variables allowable in the `BACKUP` command

Testing a Channel Allocation on the Media Manager

After integrating the media management library with Oracle, test whether RMAN is able to load the library.

To test channel allocation on the media manager:

1. Start RMAN and connect to the target database. For example, enter:

   ```
   % rman TARGET /
   ```

2. Run the `ALLOCATE CHANNEL` command with the `PARMS` required by your media management software. For example, run this command:

   ```
   RUN
   {
     ALLOCATE CHANNEL c1 DEVICE TYPE sbt
       PARMS='ENV=(NSR_SERVER=tape_srv,NSR_GROUP=oracle_tapes)';
   }
   ```

If you do not receive an error message, then Oracle successfully loaded the shared library. However, channel allocation can fail with the `ORA-27211` error:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of allocate command on c1 channel at 11/30/2001 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
```

The `ORA-27211` error indicates that Oracle was not able to load the media management library that you installed. In this case, you must check your media management installation to make sure that the library is correctly installed and redo the steps from the section "Integrating RMAN with a Media Manager on Windows NT: Basic Steps" on page 8-4.

If you do not receive ORA-27211, then the library is loaded. For any other errors, check the trace file in USER_DUMP_DEST directory for more information.

> **See Also:** "After Installation of Media Manager, RMAN Channel Allocation Fails: Scenario" on page 15-25 for a troubleshooting scenario

Testing a Backup to the Media Manager

After testing a channel allocation on the media manager, make a test backup. For example, to test whether your backup goes successfully to tape, you might run the following command:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS='ENV=(NSR_SERVER=tape_srv,NSR_GROUP=oracle_tapes)';
  BACKUP DATAFILE 1;
}
```

The specifics of your PARMS and FORMAT settings depend on the media management software that you are using. The following table illustrates possible outcomes and suggests possible responses in case of a failure.

Case	Response
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 "Integrating RMAN with the Media Manager: Basic Steps" on page 8-3 are correctly done. Refer to "Backup Job Is Hanging: Scenario" on page 15-27 if the problem persists.
The backup fails with ORA-19511 or ORA-70*xxx* error	This error indicates that the media management software is not correctly configured. Ensure that the steps in "Integrating RMAN with the Media Manager: Basic Steps" on page 8-3 are correctly done. Also, ensure that you have the correct PARMS and FORMAT strings required by your media management software.

Case	Response
The backup succeeds.	In this case, you are ready to use RMAN to make `sbt` backups.

> **See Also:** "Testing the Media Management API" on page 15-9 and "RMAN Troubleshooting Scenarios" on page 15-24 for more information about troubleshooting RMAN with a media manager

Configuring Automatic Channels for Use with a Media Manager

This section describes how to configure automatic channels specifically for use with a media manager. To gain an overview of what automatic channels are and how they are used, refer to the section "Configuring Automatic Channels" on page 8-10. The following setup procedure references the sections in "Configuring Automatic Channels" where it is appropriate.

To configure automatic channels for use with a media manager:

1. Configure a generic channel of `DEVICE TYPE sbt` as described in "Configuring a Generic Automatic Channel for a Device Type" on page 8-12. In the configuration type all parameters you tested in the section "Testing a Backup to the Media Manager" on page 8-8. For example, assume that your media vendor requires `PARMS` settings as follows:

```
CONFIGURE CHANNEL DEVICE TYPE sbt
  PARMS='ENV=(NSR_SERVER=tape_svr, NSR_CLIENT=oracleclnt, NSR_GROUP=oracle_tapes)'
  FORMAT "BACKUP_%U";
```

2. After configuring the channel, test the backup with the following command:

```
BACKUP DEVICE TYPE sbt DATAFILE 1;
```

3. Check your configuration by running the following command:

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

4. Configure the default device to `sbt` as described in "Changing the Default Device Type" on page 8-11. By configuring the device to `sbt`, RMAN will automatically send all backups to the media manager. For example:

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

5. After configuring the default device, make a test backup to determine whether it is really going to the media manager:

   ```
   BACKUP DATAFILE 1;
   ```

6. Check your configuration by running the following command:

   ```
   SHOW DEFAULT DEVICE TYPE;
   ```

7. If you use more than one media manager, then you must specify the channel parallelism as described in "Configuring Parallelism for Automatic Channels" on page 8-11. Assume that you want to back up to your media manager using two tape drives in parallel. In this case, you can run the following commands:

   ```
   CONFIGURE DEVICE TYPE sbt PARALELLISM 2;       # two channels (tape drives)
   BACKUP DATABASE; # backup goes to tape but in two streams in parallel (two tapes)
   ```

Configuring Automatic Channels

This section contains these topics:

- Configuring Automatic Channels: Overview
- Changing the Default Device Type
- Configuring Parallelism for Automatic Channels
- Configuring a Generic Automatic Channel for a Device Type
- Configuring a Specific Channel for a Device Type
- Clearing Channel and Device Settings
- Configuring the Control File and Server Parameter File Autobackup

See Also: "RMAN Automatic and Manual Channel Allocation" on page 5-2 for a conceptual overview of automatic and manual channels, and *Oracle9i Recovery Manager Reference* for syntax

Configuring Automatic Channels: Overview

You can save persistent configuration information such as channel parameters, parallelism, and the default device type in the RMAN repository. Hence, you do not have to manually allocate channels for each backup. Instead, you can configure automatic channels for use in backup, restore, recovery, and maintenance jobs.

You can always override automatic channels by using ALLOCATE CHANNEL to allocate channels manually. The automatic channel feature is mutually exclusive with the manual channel feature: RMAN uses one or the other for every job.

By default, RMAN has preconfigured a disk channel so that you can back up to disk without doing any manual configuration. Hence, if you are backing up to disk rather than to a media manager, you can immediately begin backing up to disk. The only change you may want to parallelize the channels, as described in "Configuring Parallelism for Automatic Channels" on page 8-11.

Changing the Default Device Type

The default device is the device that RMAN uses on the BACKUP command when you do not explicitly allocate channels. For example, if the default device is sbt (that is, the media manager), then the BACKUP DATABASE command backs up to the configured sbt device. A default device type is required so that RMAN knows which device it should use when you do not manually allocate a channel. You cannot set the default device type to an unspecified value such as NULL.

The preconfigured default device type is DISK. To change the default device type to sbt, you must run this command:

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

Change the default back to DISK by running either of the following commands:

```
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
CONFIGURE DEFAULT DEVICE TYPE CLEAR; # returns to default of DISK
```

The following example sets the default device type to sbt and then runs a backup:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
BACKUP DATABASE; # backup goes to sbt
```

Configuring Parallelism for Automatic Channels

By default, channel parallelism for each configured device is set to 1. Run the CONFIGURE DEVICE TYPE ... PARALLELISM *integer* command to specify the number of channels that RMAN should allocate for each job on a specified device type, where *integer* stands for a positive integer less than 255. For example, to allocate three channels for each job on the DISK device, run this command:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
```

You can change PARALLELISM for a device type simply by running a new command. The following example sets parallelism to 3, changes parallelism from 3 to 4, then changes it to 2:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
CONFIGURE DEVICE TYPE DISK PARALLELISM 4;
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
```

Assume that you want to back up to your media manager using two tape drives in parallel. You run the following commands:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;    # make the tape drive the default backup device
CONFIGURE DEVICE TYPE sbt PARALELLISM 2;    # configure two channels (tape drives)
BACKUP DATABASE; # backup goes to tape but in two parallel streams (two tapes)
```

In this case, each configured sbt channel will back up roughly half the total data.

> **See Also:** "Parallelization for Manually Allocated Channels" on page 5-10

Configuring a Generic Automatic Channel for a Device Type

By default, RMAN automatically allocates a preconfigured DISK channel without any options. However, you may use a media manager that requires special options (PARMS, FORMAT, MAXPIECESIZE, and so forth) or you may want to change the default DISK setting. By configuring channels, you define which parameters are used when RMAN automatically allocate channels.

Use the CONFIGURE CHANNEL command to configure automatic channel options for the available device types: DISK and sbt. You can use the same options for CONFIGURE CHANNEL that you use for ALLOCATE CHANNEL, and you must specify at least one of these options. For example, you can configure generic disk and tape channels as in this example:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT = '?/bkup_%U';
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='ENV=(NSR_SERVER=tape_svr,NSR_CLIENT=oracleclnt,
                                               NSR_GROUP=oracle_tapes)';
```

To configure a **generic channel**, that is, a template that is used for all parallelized channels, do not assign a number for the channel. If you set the parallelism for a device, and then make the device default, then RMAN uses the same channel configuration for each parallelized channel.

To configure new generic channel settings for a specified device type, simply run a new command for the device type. The following example configures the default DISK channel to MAXPIECESIZE 2G, then erases this setting and sets a FORMAT:

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

Configured Channels and the Default Device Type

The automatic channel that RMAN allocates for its backups depends on the default device type. If the default device type is DISK, then RMAN uses the DISK channel only. If the default device type is sbt, then RMAN uses the sbt channel only. RMAN cannot automatically allocate channels in backup jobs for multiple device types simultaneously.

This example creates a configuration so that all backups go to two tapes in parallel. Additionally, the media management software requires additional parameters: ENV=(NSR_DATA_VOLUME_POOL=oracle_tapes) and FORMAT of %U_backup.

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;        # by default, backup goes to media manager
CONFIGURE DEVICE TYPE sbt PARALELLISM 2;     # two tapes in parallel
CONFIGURE CHANNEL DEVICE TYPE                # sets parameters for all channels
  PARMS 'ENV=(NSR_DATA_VOLUME_POOL=oracle_tapes)' FORMAT '%U_backup';
BACKUP DATABASE;                             # backs up database
```

Manually Overriding Configured Channels

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

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

In this case, RMAN uses the channel that you manually allocated within the RUN block—and only this channel. None of the CONFIGURE DEVICE TYPE, CONFIGURE DEFAULT DEVICE, or CONFIGURE CHANNEL settings apply to this job.

See Also:

- "Backup Sets" on page 5-12 to learn about automatic channels
- *Oracle9i Recovery Manager Reference* for ALLOCATE syntax
- *Oracle9i Recovery Manager Reference* for CONFIGURE syntax

Configuring a Specific Channel for a Device Type

Besides configuring a generic channel for a device, you can also configure one or more specific channels for each device type by manually assigning your own channel numbers to the channels. 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.

Use specific channels when you want you want to alter the configuration for a channel. Typically, you should only need to do this when running an Oracle Real Application Clusters configuration and when using a media manager with multiple tape drives requiring different `PARMS` settings.

Configuring Specific Channels: Examples

For example, assume that you have two tape drives and want one tape drive to use tapes from the first pool and the second tape drive to use tapes from second tape pool. You run the following commands:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;    # backup goes to sbt
CONFIGURE DEVICE TYPE sbt PARALELLISM 2; # two tapes used in parallel
# configure first stream to go to data volume pool named first_pool
CONFIGURE CHANNEL 1 DEVICE TYPE sbt PARMS 'ENV=(NSR_DATA_VOLUME_POOL=first_pool)';
# configure second stream to go to data volume pool named second_pool
CONFIGURE CHANNEL 2 DEVICE TYPE sbt PARMS 'ENV=(NSR_DATA_VOLUME_POOL=second_pool)';
BACKUP DATABASE; # first stream goes to 'first_pool' and second to 'second_pool'
```

In this example, you want to back up to two different disks because not enough space exists on a single disk. So, you do the following:

```
CONFIGURE DEFAULT DEVICE TYPE TO disk;          # backup goes to disk
CONFIGURE DEVICE TYPE sbt PARALELLISM 2;        # two channels used in in parallel
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT '/disk1/%U' # first channel goes to disk1
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT '/disk2/%U' # second channel goes to disk2
BACKUP DATABASE; # backup - first channel goes to disk1 and second to disk2
```

Mixing Generic and Specific Channels

When parallelizing, RMAN always allocates channels beginning with `CHANNEL 1` and ending with channel number equal to the parallelism setting. Hence, RMAN uses a specific configuration for a given channel if you have configured it; otherwise, it uses a generic configuration.

Assume you enter the following channel configuration:

```
# disk channel configuration
CONFIGURE DEVICE TYPE DISK PARALLELISM 4;
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT = '/tmp/backup_%U';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK MAXPIECESIZE = 2M;
CONFIGURE CHANNEL 4 DEVICE TYPE DISK MAXPIECESIZE = 4M;

# sbt channel configuration
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='SBT_LIBRARY=oracle.disksbt,
                                         ENV=(BACKUP_DIR=?/oradata)';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt PARMS='SBT_LIBRARY=oracle.disksbt,
                                           ENV=(BACKUP_DIR=/tmp)';
```

The following table illustrates the channel names and channel settings that RMAN allocates when the default device is `DISK` and parallelism for `DISK` is set to `4`.

Channel Name	Setting
ORA_DISK_1	FORMAT = '/tmp/backup_%U'
ORA_DISK_2	MAXPIECESIZE = 2M
ORA_DISK_3	FORMAT = '/tmp/backup_%U'
ORA_DISK_4	MAXPIECESIZE = 4M

The following table illustrates the channel names and channel settings that RMAN allocates when the default device is `sbt` and parallelism for `sbt` is set to `3`.

Channel Name	Setting
ORA_SBT_TAPE_1	PARMS='ENV=(BACKUP_DIR=/disk2/oracle/backup)'
ORA_SBT_TAPE_2	PARMS='ENV=(BACKUP_DIR=/disk2/oracle/backup)'
ORA_SBT_TAPE_3	PARMS='ENV=(BACKUP_DIR=/store)'

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. You can set parallelism to any value.

RMAN always numbers parallel channels starting with `1` and ending with the parallelism setting. For example, if the default device is `sbt` and parallelism for `sbt` is set to `3`, then RMAN names the channels as follows:

```
ORA_SBT_TAPE_1
ORA_SBT_TAPE_2
ORA_SBT_TAPE_3
```

Note that 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 manual channels if you have configured them and generic channels if you have not.

> **See Also:** "Automatic Channel Specific Configurations" on page 5-8 for concepts about manually numbered channels, and "Configuring Specific Channels: Examples" on page 8-14

Clearing Channel and Device Settings

To clear a configuration is to return it to its default settings. You can clear channel and device settings by using these commands:

- `CONFIGURE DEVICE TYPE ... CLEAR`
- `CONFIGURE DEFAULT DEVICE TYPE CLEAR`
- `CONFIGURE CHANNEL DEVICE TYPE ... CLEAR`
- `CONFIGURE CHANNEL n DEVICE TYPE ... CLEAR` (where *n* is an integer)

Each `CONFIGURE ... CLEAR` command clears only itself. For example, `CONFIGURE DEVICE TYPE ... CLEAR` does not clear `CONFIGURE DEFAULT DEVICE TYPE`. The `CONFIGURE DEVICE TYPE ... CLEAR` command removes the configuration for the specified device type and returns it to the default (`PARALLELISM 1`).

> **Note:** You cannot specify any other options when clearing a device type.

The `CONFIGURE DEFAULT DEVICE TYPE ... CLEAR` command clears the configured default device and returns it to `DISK` (the default setting).

The `CONFIGURE CHANNEL DEVICE TYPE ... CLEAR` command erases the channel configuration for the specified device type. RMAN does not change the parallelism setting for the device type because `PARALLELISM` is specified through a separate `CONFIGURE` command.

If you have manually assigned numbers to automatic channels, then clear the options for these channels individually by specifying the channel number in

CONFIGURE CHANNEL *n* DEVICE TYPE ... CLEAR. For example, assume that you run the following:

```
CONFIGURE CHANNEL DEVICE TYPE DISK MAXOPENFILES = 10 MAXPIECESIZE = 1800K;
CONFIGURE CHANNEL 3 DEVICE TYPE DISK FORMAT = /tmp/%U;
CONFIGURE CHANNEL 3 DEVICE TYPE DISK CLEAR;
```

In this case, RMAN clears the settings for CHANNEL 3, but leaves the settings for the generic DISK channel (the channel with no number manually assigned) intact.

See Also: "Clearing Automatic Channel Settings" on page 5-9

Configuring the Control File and Server Parameter File Autobackup

RMAN can automatically back up the control file and server parameter file whenever the database structure metadata in the control file changes and whenever a backup or copy record is added.

If CONFIGURE CONTROLFILE AUTOBACKUP is ON (by default it is OFF), then RMAN automatically backs up the control file and the current server parameter file (if used) in the following circumstances:

- After every BACKUP or COPY command issued at the RMAN prompt.
- Whenever a BACKUP or COPY command within a RUN block is followed by a command that is neither BACKUP nor COPY.
- At the end of every RUN block if the last command in the block was either BACKUP or COPY.
- After database structural changes such as adding a new tablespace, altering the state of a tablespace or datafile (for example, bringing it online), adding a new online redo log, renaming a file, adding a new redo thread, and so forth. This type of autobackup, unlike autobackups that occur in the preceding circumstances, is only on disk. You can run CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK to set a nondefault disk location. Note that the autobackup never causes the associated structural change to fail. For example, if you add a datafile, and if the resulting autobackup fails, then the datafile addition is still successful.

Note: If you use Oracle Enterprise Manager, then you can use the Maintenance wizard to configure the autobackup feature.

The first channel allocated during the backup or copy job creates the autobackup and places it into its own backup set; for post-structural autobackups, the default disk channel makes the backup. RMAN writes the control file and the server parameter file to the same backup piece. After the control file autobackup completes, Oracle writes a message containing the complete path of the backup piece and the device type to the alert log.

As explained by the following table, the RMAN behavior when the `BACKUP` command includes datafile `1` differs depending on whether `CONFIGURE CONTROLFILE AUTOBACKUP` is `ON` or `OFF`.

CONTROLFILE AUTOBACKUP	BACKUP Command Behavior
`ON`	If the backup includes datafile `1`, then RMAN does *not* automatically include the current control file in the datafile backup set. Instead, RMAN writes the control file and server parameter file to a separate autobackup piece. **Note:** The autobackup occurs regardless of whether the `BACKUP` or `COPY` command explicitly includes the current control file, for example, `BACKUP DATABASE INCLUDE CURRENT CONTROLFILE`.
`OFF`	If the backup includes datafile `1`, then RMAN automatically includes the current control file and server parameter file in the datafile backup set. RMAN does *not* create a separate autobackup piece containing the control file and server parameter file.

The purpose of the autobackup is to enable RMAN to recover the database even if the current control file, recovery catalog, and server parameter file are inaccessible. Because the autobackup uses 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, use the RMAN repository in the mounted control file to restore the datafiles.

Enabling and Disabling the Control File Autobackup

You can enable the autobackup feature by running this command:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

You can disable the feature by running this command:

```
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
```

Configuring the Control File Autobackup Format

By default, the format of the autobackup file for all configured devices is the substitution variable `%F`. This variable format translates into *`c-IIIIIIIIII-YYYYMMDD-QQ`*, where:

- *`IIIIIIIIII`* stands for the DBID. The DBID is a unique numerical identifier for the database. The DBID is printed in decimal so that it can be easily associated with the target database.
- *`YYYYMMDD`* is a time stamp in the Gregorian calendar of the day the backup is generated
- *`QQ`* is the sequence in hexadecimal number that starts with `00` and has a maximum of `FF` (256)

You can change the default format by using the following command, where *`deviceSpecifier`* is any valid device such as `DISK` or `sbt`, and `'`*`string`*`'` contains the variable `%F` 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:

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

Note: The format string *must* contain the `%F` variable. If the format string does not contain the `%F` variable, then RMAN issues an error.

You can return the default autobackup format to `%F` by running this command, where *`deviceSpecifier`* is any valid device such as `DISK` or `sbt`:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE deviceSpecifier CLEAR;
```

The `SET CONTROLFILE AUTOBACKUP FORMAT` command, which you can specify either within a `RUN` block 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 `RUN`)
2. `SET CONTROLFILE AUTOBACKUP FORMAT` (at `RMAN` prompt)
3. `CONFIGURE CONTROLFILE AUTOBACKUP FORMAT`

See Also:

- "Control File and Server Parameter File Autobackups" on page 5-2 for a conceptual overview
- *Oracle9i Recovery Manager Reference* for CONFIGURE syntax
- *Oracle9i Recovery Manager Reference* for SET syntax

Configuring the Backup Retention Policy

A retention policy specifies when RMAN should consider the backups and copies it creates as obsolete. Before attempting to configure the policy, read the conceptual explanation in "Backup Retention Policies" on page 5-49.

Use the CONFIGURE command to set the retention policy. If you use Oracle Enterprise Manager, then you can also set the policy with the Maintenance wizard.

This section contains these topics:

- Configuring the Retention Policy for a Recovery Window
- Configuring the Retention Policy for Redundancy
- Disabling the Retention Policy
- Returning the Retention Policy to its Default Setting

See Also:

- *Oracle9i Recovery Manager Reference* for CONFIGURE syntax
- *Oracle Enterprise Manager Administrator's Guide* to learn about RMAN restore and recovery

Configuring the Retention Policy for a Recovery Window

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 backup or copy as obsolete if it falls within the recovery window. Additionally, RMAN retains all archived logs and incremental backups that are needed to recover to a random point within the window.

Run the CONFIGURE command at the RMAN prompt. This example ensures that you can recover the database to any point within the last 7 days:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

RMAN does not automatically delete backups and copies rendered obsolete by the recovery window. Instead, RMAN shows them as OBSOLETE in the REPORT OBSOLETE output and deletes them if you run DELETE OBSOLETE.

To change the retention policy setting, run the CONFIGURE RETENTION POLICY command with a new setting. For example:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 4 DAYS;
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 11 DAYS;
```

See Also: "Recovery Window" on page 5-51 for a conceptual introduction to the recover window

Configuring the Retention Policy for Redundancy

The REDUNDANCY parameter of the CONFIGURE RETENTION POLICY command specifies how many backups and copies of each datafile and control file that RMAN should keep. In other words, if the number of backups and copies for a specific datafile or control file exceeds the REDUNDANCY setting, then RMAN considers the extra backups and copies as obsolete.

As you produce more backups, RMAN keeps track of which ones to retain and which are obsolete. Additionally, RMAN retains all archived logs and incremental backups that are needed to recover the nonobsolete backups.

Assume that you make a backup of a specific datafile on Monday, Tuesday, Wednesday, and Thursday. Hence, you have four backups of the datafile. If REDUNDANCY is 2, then the Monday and Tuesday backups are obsolete. If you produce another backup on Friday, then the Wednesday backup also becomes obsolete.

Run the CONFIGURE RETENTION POLICY command at the RMAN prompt, as in the following example:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
```

To change the setting, run the command with a new REDUNDANCY setting. For example, run the following:

```
CONFIGURE RETENTION POLICY TO REDUNDANCY 4;
CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
```

See Also: "Backup Redundancy" on page 5-53 for a conceptual introduction to backup redundancy

Disabling the Retention Policy

To disable the retention policy, run this command:

```
CONFIGURE RETENTION POLICY TO NONE;
```

This command means that RMAN does not consider any backup or copy as obsolete. Consequently, if you run `REPORT OBSOLETE` or `DELETE OBSOLETE` with no other options, RMAN issues an error because no retention policy exists to determine which backups and copies are obsolete.

Note: Configuring the retention policy to `NONE` is not the same as clearing it. Clearing it returns it to its default setting, whereas `NONE` disables it completely.

Returning the Retention Policy to its Default Setting

To clear the retention policy, that is, return the retention policy to its default setting, run this command:

```
CONFIGURE RETENTION POLICY CLEAR;
```

This command returns the retention policy to its default setting, which is `REDUNDANCY = 1`.

Configuring the Maximum Size of Backup Sets

The `CONFIGURE MAXSETSIZE` command limits the size of backup sets created on a channel. The `CONFIGURE` settings apply to any channel, whether manually or automatically allocated, when the `BACKUP` command is run.

You can set `MAXSETSIZE` in bytes (default), kilobytes (`K`), megabytes (`M`), and gigabytes (`G`). The default value is given in bytes and is rounded down to the lowest kilobyte value. For example, if you set the maximum set size to 2000, then RMAN rounds down this value to 1 kilobyte (1024 bytes). If you set the maximum set size to 2049, then RMAN rounds down this value to 2 kilobytes (2048 bytes).

You can specify `MAXSETSIZE` on the following commands, listed in descending order of precedence (that is, the higher overriding the lower):

1. `BACKUP`
2. `CONFIGURE`

An example illustrates the hierarchy of settings. Assume that you issue the following commands at the RMAN prompt:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(NSR_DATA_VOLUME_POOL=first_pool)';
CONFIGURE MAXSETSIZE TO 7500K;
BACKUP TABLESPACE users;
BACKUP TABLESPACE tools MAXSETSIZE 10M;
```

Note these consequences of RMAN behavior:

- The backup of the `hr` tablespace uses the automatic `sbt` channel and the default `MAXSETSIZE` setting of `7500K`.
- The backup of the `orders` tablespace uses the `MAXSETSIZE` setting of `10M` used in the `BACKUP` command.

See Also: *Oracle9i Recovery Manager Reference* for `BACKUP` syntax

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. Before configuring optimization, read the discussion of the backup optimization algorithm in "Backup Optimization" on page 5-56.

Note that backup optimization applies only to these commands:

- `BACKUP DATABASE`
- `BACKUP ARCHIVELOG` with `ALL` or `LIKE` options
- `BACKUP BACKUPSET ALL`

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;
```

By default, backup optimization is configured to `OFF`. To enable backup optimization, run the following command:

```
CONFIGURE BACKUP OPTIMIZATION ON;
```

To disable backup optimization, run the following command:

```
CONFIGURE BACKUP OPTIMIZATION OFF;
```

To clear the current backup optimization setting, that is, return backup optimization to its default setting of OFF, run this command:

```
CONFIGURE BACKUP OPTIMIZATION CLEAR;
```

Note: If you use Oracle Enterprise Manager, then you can use the Maintenance wizard to configure backup optimization.

See Also:

- "Backup Optimization" on page 5-56 for the complete criteria that determine whether a file is identical and the conditions under which backup optimization is operative
- "Backing Up Files Using Backup Optimization" for examples of how to optimize RMAN backups

Configuring the Number of Backup Copies

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 feature is known as **duplexing**. The CONFIGURE settings applies only to backups of datafiles (which includes the current control file) and archived redo logs.

Note: Control file autobackups on disk are a special case and are *never* duplexed: RMAN always creates one and only one copy.

If you backup to sbt, than you have to set initialization parameter BACKUP_TAPE_IO_SLAVES=true. If you do not, then RMAN signals the following error:

```
RMAN-10035: exception raised in RPC: ORA-19565: BACKUP_TAPE_IO_SLAVES not enabled when
            duplexing to sequential devices
```

To configure the number of backup copies, specify an integer. The following examples show possible configurations:

```
# Makes 2 copies of every datafile and control file backup (autobackups excluded) to disk
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;
```

If you use the duplexing feature in conjunction with multiple FORMAT strings, then you can name each individual backup copy. For example, assume that you configure BACKUP COPIES to 3. Then, you can issue:

```
BACKUP DATABASE FORMAT '/tmp/%U', '?/dbs/%U', '?/oradata/%U';
```

RMAN generates 3 identical copies of each backup piece in the backup set, and names each piece according to the specified FORMAT string: the first copy is placed in the /tmp directory, the second in the ?/dbs directory, and the third in the ?/oradata directory. Note that you can specify the FORMAT string on the BACKUP, CONFIGURE CHANNEL, and ALLOCATE CHANNEL commands.

To return a BACKUP COPIES configuration to its default value, run the same CONFIGURE command with the CLEAR option, as in this 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 SET BACKUP COPIES commands.

See Also:

- "Duplexed Backup Sets" on page 5-21 for concepts
- *Oracle9i Recovery Manager Reference* for BACKUP syntax
- *Oracle9i Recovery Manager Reference* for CONFIGURE syntax
- *Oracle9i Recovery Manager Reference* for SET syntax

Configuring Tablespaces for Exclusion from Whole Database Backups

You can CONFIGURE EXLUDE FOR TABLESPACE to exempt the specified tablespace from the BACKUP DATABASE command. The exclusion condition applies to any datafiles that you add to this tablespace in the future.

This tablespace exclusion feature is useful when you do not want to make a 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.

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:

```
BACKUP DATABASE NOEXCLUDE;    # backs up the whole database, including cwmlite and example
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.

Note: If you use Oracle Enterprise Manager, then you can use the Maintenance wizard to exclude tablespaces from backups.

See Also:

- *Oracle9i Recovery Manager Reference* for `BACKUP` syntax
- *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax
- *Oracle Enterprise Manager Administrator's Guide* to learn about the Maintenance wizard

Setting Globalization Support Environment Variables for RMAN

This section describes globalization support variable settings in RMAN and contains these topics:

- Setting NLS_DATE_FORMAT and NLS_LANG
- Specifying the Database Character Set

Setting NLS_DATE_FORMAT and NLS_LANG

Before invoking RMAN, 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'
```

Specifying the Database Character Set

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. To avoid this problem, set the NLS_LANG to specify the target database's character set. For example, if the character set is WE8DEC, you can set the NLS_LANG parameter as follows:

```
NLS_LANG=american_america.we8dec
```

Note: You must set both NLS_LANG and NLS_DATE_FORMAT for NLS_DATE_FORMAT to be used.

See Also: *Oracle9i Database Reference* for more information about the NLS_LANG and NLS_DATE_FORMAT parameters, and *Oracle9i Database Globalization Support Guide*

Configuring the Snapshot Control File Location

When RMAN needs to resynchronize from a read-consistent version of the control file, it creates a temporary **snapshot control file**. RMAN needs a snapshot control file only when resynchronizing with the recovery catalog or when making a backup of the current control file.

The default value for the snapshot control file is platform-specific and depends on the Oracle home. For example, the default filename on some UNIX platforms in Oracle9*i* is `$ORACLE_HOME/dbs/snapcf_@.f`.

In general, you should only need to set the control file location when You are upgrading to the current release from a release earlier than 8.1.7. In these earlier releases, the default location for the snapshot control file was not dependent on the Oracle home, whereas in the current release the default location is dependent on the Oracle home.

Setting the Location of the Snapshot Control File to the Default Value

In some cases, you may want to set the snapshot control file to the default location. Run the `CONFIGURE SNAPSHOT CONTROLFILE LOCATION CLEAR` command. The behavior of the default value is described in the following table.

If you ...	Then ...
Create a new database in the current release	The snapshot control file location uses the `DEFAULT` value. In this case, the default snapshot control file location changes if you change the Oracle home.
Upgrade to the current release from a release prior to 8.1.7	The snapshot control file location is not set to the `DEFAULT` value. Instead, RMAN uses the snapshot location that is already stored in the control file. In this case, the snapshot control file location does not change if you change the Oracle home. To set the snapshot control file location to the default value, run `CONFIGURE SNAPSHOT CONTROLFILE NAME CLEAR`.

You can see the current snapshot location by running the `SHOW` command. This example shows a snapshot location that is default:

```
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 '/tmp/snapcf_trgt.f'; # default
```

Setting the Location of the Snapshot Control File to a Nondefault Value

Use the `CONFIGURE SNAPSHOT CONTROLFILE NAME TO '`*`filename`*`'` command to change the name of the snapshot control file to a nondefault value. 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. An Oracle Real Application Clusters configuration does not require the snapshot control file to be shared across all instances, but the snapshot controlfile name must be set to a location where any instance can create one. Each instance is permitted to create the snapshot control file in its own file system as it needs one. The following example sets the snapshot control file name to a raw device:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/dev/vgd_1_0/rlvt5';
```

Note that if one RMAN job is already backing up the control file while another needs to create a new snapshot control file, you may see the following message:

```
waiting for snapshot controlfile enqueue
```

Under normal circumstances, a job that must wait for the control file enqueue waits for a brief interval and then successfully retrieves the enqueue. Recovery Manager makes up to five attempts to get the enqueue and then fails the job. The conflict is usually caused when two jobs are both backing up the control file, and the job that first starts backing up the control file waits for service from the media manager.

> **See Also:** "Backup Fails Because of Control File Enqueue: Scenario" on page 15-33, and *Oracle9i Recovery Manager Reference* for `CONFIGURE` syntax

Setting Up RMAN for Use with a Shared Server

RMAN cannot connect to the target database through a shared server dispatcher: it requires a dedicated server process. Nevertheless, you can connect specified sessions to dedicated servers, even when the target is configured for a shared server.

To ensure that RMAN does not connect to a dispatcher when the 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 the 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)(port1521))
    (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)(port1521))
       (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, to connect to a dedicated session you can issue:

   ```
   SQL> CONNECT SYS/oracle@inst1_ded
   Connected.
   SQL> SELECT SERVER FROM V$SESSION WHERE SID = (SELECT DISTINCT SID FROM V$MYSTAT);

   SERVER
   ---------
   DEDICATED
   1 row selected.
   ```

   To connect to a shared server session, you can issue:

   ```
   SQL> CONNECT SYS/oracle@inst1_shs AS SYSDBA
   Connected.
   SQL> SELECT SERVER FROM V$SESSION WHERE SID = (SELECT DISTINCT SID FROM V$MYSTAT);

   SERVER
   ---------
   SHARED
   1 row selected.
   ```

3. Connect to the target database (and optionally the recovery catalog) with the dedicated service name. For example, enter:

```
% rman TARGET SYS/oracle@inst1_ded CATALOG rman/cat@catdb
```

See Also: Your operating system-specific Oracle documentation and your *Oracle9i Net Services Reference Guide* for a complete description of Oracle Net connect string syntax

Setting Up a Recovery Catalog

A recovery catalog is not necessary when using RMAN. To use a recovery catalog, you need to create a schema in a recovery catalog database. The catalog will be located in the default tablespace of the scheme. Note that SYS cannot be the owner of the catalog.

Create the recovery catalog schema in a different host, on different disks, and in a different database from the target database you will be backing up. If you do not, then the benefits of using a recovery catalog are lost if you lose the database and need to restore.

Caution: Ensure that the recovery catalog and target databases do *not* reside on the same disks. If they do and you lose one database, you will probably lose the other.

The basic procedure for setting up a recovery catalog, which is described in complete detail in Chapter 16, "Managing the Recovery Manager Repository", is as follows:

1. Create the user who will own the recovery catalog schema in the recovery catalog database. For example, start SQL*Plus and run:

```
-- connect as SYS to recovery catalog database
CONNECT SYS/oracle@catdb AS SYSDBA

-- create user that will own catalog tables
CREATE USER rman IDENTIFIED by cat
  TEMRORARY TABLESPACE temp
  DEFAULT TABLESPACE tools QUOTA UNLIMITED ON tools;
```

2. Grant the recovery catalog owner privilege and any other desired privileges. For example, in SQL*Plus run:

```
GRANT RECOVERY_CATALOG_OWNER TO rman;
GRANT CONNECT, RESOURCE TO rman;
```

3. Start RMAN and create the recovery catalog schema itself with the `CREATE CATALOG` command. For example:

   ```
   % rman TARGET / CATALOG rman/cat@catdb
   RMAN> CREATE CATALOG
   ```

4. Register the target databases in the recovery catalog with the `REGISTER DATABASE` command. For example:

   ```
   RMAN> REGISTER DATABASE;
   ```

   See Also: "Creating the Recovery Catalog" on page 16-2, and "Registering a Database in the Recovery Catalog" on page 16-5

9

Making Backups and Copies with Recovery Manager

This chapter describes how to use Recovery Manager to manage backup and copy operations. This chapter contains these topics:

Configuring and Allocating Channels for Use in Backup and Copy Jobs

You have the following mutually exclusive options for executing backup and copy jobs:

- Configure automatic channels with the `CONFIGURE` command, and then issue `BACKUP` and `COPY` commands at the RMAN prompt or within a `RUN` block
- Allocate channels manually and issue `BACKUP` and `COPY` commands within a `RUN` block

The easiest way to make backups is to configure automatic channels. For example, so long as you have already configured an `sbt` device type, you can configure a default `sbt` channel as follows (note that the `PARMS` value is vendor-specific):

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(NSR_SERVER=bksvr1)';
```

Then, you can back up the database to tape at the RMAN prompt as follows:

```
BACKUP DATABASE;
```

RMAN preconfigures a `DISK` channel for you, so you can make disk backups using automatic channels without performing any configuration whatsoever.

The other method is to allocate channels manually within a run job. For example, this command allocates multiple disk channels and then backs up the database and archived redo logs:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL ch2 DEVICE TYPE DISK;
  ALLOCATE CHANNEL ch3 DEVICE TYPE DISK;
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

This example manually allocates an `sbt` channel (with a vendor-specific `PARMS` value) and backs up a datafile copy:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt PARMS 'ENV=(NSR_SERVER=bksvr1)';
  BACKUP DATAFILECOPY '/tmp/system01.dbf';
}
```

For the most part, the procedures in this chapter assume that you have configured automatic channels.

Backing Up Database Files and Archived Logs with RMAN

This section contains these topics:

- About RMAN Backups
- Making Consistent and Inconsistent Backups with RMAN
- Making Whole Database Backups with RMAN
- Backing Up Tablespaces with RMAN
- Backing Up Datafiles and Datafile Copies with RMAN
- Backing Up Control Files with RMAN
- Duplexing Backups
- Backing Up Archived Redo Logs with RMAN

See Also:

- "Backup Sets" on page 5-12 for an overview of RMAN backups
- *Oracle9i Recovery Manager Reference* for BACKUP command syntax
- "Backup Types" on page 5-36 for a discussion of the various RMAN backup types

About RMAN Backups

Run backups of any of the following objects with the RMAN BACKUP command when the database is either mounted or open:

- Primary or Standby Database (all datafiles and current control file)
- Tablespace
- Datafile (current or image copy)
- Archived redo log
- Control file (current or image copy)
- Server parameter file (currently in use only)
- Backup set

Although the BACKUP command works on datafiles, archived redo log files, control files, and server parameter files, the database depends on other files for production operation. You should back up other important files on the operating system either with a third-party backup product. You should back up files needed for the operating system to run as well as networking files, password files, and files in the Oracle home.

The BACKUP command backs up database files into one or more backup sets on disk or tape. You can set parameters for the BACKUP command to specify the filenames for the backup pieces, the number of files to go in each set, and which channel should operate on each input file.

You can make RMAN backups when the database is open or closed. Closed backups can be **consistent** or **inconsistent**, depending on how the database was shut down. RMAN backup are further divided into **full** and **incremental** backups. Full backups are nonincremental, that is, every used block is backed up.

About RMAN Backups with Oracle Enterprise Manager

If you use Oracle Enterprise Manager, then you can use Backup wizard instead of the command-line interface. You can perform the following RMAN tasks through the Backup Wizard:

- **Back up a database using a predefined strategy** if you want to back up your entire database without having to make too many decisions
- **Back up a database with a customized strategy** where you can decide to select a full or incremental backup, to select an online or offline backup, to override the backup and retention policy set in the database, or whether or not to obsolete backups and copies during the process.
- **Back up individual files such as tablespaces or datafiles** with various options such as choosing to delete obsolete backups after the backup, selecting a full or an incremental backup, choosing an online or offline mode of backup, or choosing to override the backup and retention policy set in the database.
- **Back up archive logs**, specifying the date and time for the first and last archive logs to be backed up and whether you want to delete the input logs (from the primary archiving destination only) automatically after the backup completes
- **Copy a datafile** to use for recovery later
- **View the current backup and retention policy settings** in the target database

Using the Oracle Enterprise Manager's Create Backup Configuration property sheets, you can edit an existing backup configuration or create other backup configurations. A backup configuration is automatically created for each target database by Enterprise Manager. In the backup configuration, you can specify the following options:

- **Create a configuration that backs up to disk** using a backup set
- **Create a configuration that backs up to tape** using a backup set
- **Create a configuration that uses an image copy**
- **Set the limits for any backup or copy operation**
- **Specify to have Media Management Software take over the backup and recovery of your files** instead of using RMAN
- **Set storage parameters for the current backup set**, overriding the Recovery Manager default settings that the Recovery Manager has calculated for datafiles or archivelogs
- **Register the database with the recovery catalog**. A recovery catalog is created if you specify for the Enterprise Manager repository to be located in a local database.

See Also:

- "Backup Sets" on page 5-12 for an overview of RMAN backups
- "Backup Types" on page 5-36 for a discussion of the various RMAN backup types
- *Oracle9i Recovery Manager Reference* for a complete description of `BACKUP` command syntax
- *Oracle Enterprise Manager Administrator's Guide* to learn about making RMAN backups

Making Consistent and Inconsistent Backups with RMAN

Consistent backups can be restored without recovery. To make a consistent backup, the database:

- Must be mounted, but not open
- Must not have suffered an instance failure or closed with `SHUTDOWN ABORT` the last time it was open

If these conditions are not met, then the backup is inconsistent. An inconsistent backup requires media recovery when it is restored, but is otherwise just as valid as a consistent backup.

You can use SQL*Plus or RMAN to start up and shutdown the database. The following example connects to the target database, shuts it down cleanly, and then mounts it in preparation for a backup:

```
% rman TARGET /
RMAN> SHUTDOWN IMMEDIATE # closes database consistently
RMAN> STARTUP MOUNT # uses SPFILE
```

Making Whole Database Backups with RMAN

If you can afford to close the primary database, then take closed, consistent backups of the whole database. If you cannot shut down the database, then the only option is to make a backup while the database is open.

> **See Also:** *Oracle9i Data Guard Concepts and Administration* for more information about backing up standby databases

To make a whole database backup:

1. After starting RMAN, run the `BACKUP DATABASE` command at the RMAN prompt. This example backs up all the datafiles as well as the control file and server parameter file (if used). It does not specify a `FORMAT` parameter, so RMAN gives each backup piece a unique name automatically and stores it in the port-specific default location (`$ORACLE_HOME/dbs` on UNIX):

   ```
   BACKUP DATABASE;  # uses automatic channels to make backup
   SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT'; # switches logs and archives all logs
   ```

   Optionally, use the `FORMAT` parameter to specify a filename for the backup piece. For example, enter:

   ```
   BACKUP DATABASE FORMAT '/tmp/%U';  # %U generates a unique filename
   ```

   Optionally, use the `TAG` parameter to specify a backup tag. For example, enter:

   ```
   BACKUP DATABASE TAG = 'weekly_backup';   # gives the backup a tag identifier
   ```

   Note that RMAN assigns a default tag to backups. Refer to the `BACKUP ... TAG` description in *Oracle9i Recovery Manager Reference* for the default format.

2. Issue a `LIST BACKUP OF DATABASE` command to see a listing of backup sets and pieces.

Backing Up Tablespaces with RMAN

Back up tablespaces when the database is either open or closed. Note that all open database backups are always inconsistent. Do not issue ALTER DATABASE BEGIN BACKUP before making an online tablespace backup.

To back up a tablespace:

1. After starting RMAN, run the BACKUP TABLESPACE command at the RMAN prompt. This example backs up three tablespaces, using the FILESPERSET parameter to specify that no more than three datafiles go in each backup set:

   ```
   BACKUP FILESPERSET = 3 TABLESPACE system, users, tools;
   ```

2. Run a LIST BACKUP OF TABLESPACE command to see a listing of backup sets and pieces. For example:

   ```
   LIST BACKUP OF system, users, tools;
   ```

Backing Up Datafiles and Datafile Copies with RMAN

Back up datafiles and datafile copies when the database is either open or closed. Note that all open database backups are always inconsistent.

Backing Up Datafiles

Use the BACKUP DATAFILE command to back up individual datafiles. You can specify the datafiles by name or number.

To back up a datafile:

1. After starting RMAN, run the BACKUP DATAFILE command at the RMAN prompt. This example backs up datafiles 1 through 4 as well as an image copy of a datafile:

   ```
   BACKUP DATAFILE 1,2,3,4 FILESPERSET 3 DATAFILECOPY '/tmp/system01.dbf';
   ```

   If CONFIGURE CONTROLFILE TO AUTOBACKUP is OFF, then the current control file and server parameter file are automatically included in the datafile backup set. If ON, then RMAN writes these files to a separate autobackup piece.

2. Issue a LIST BACKUP OF DATAFILE command to see a listing of backup sets and pieces. For example:

   ```
   LIST BACKUP OF DATAFILE 1,2,3,4;
   ```

Backing Up Datafile Copies

Use the BACKUP DATAFILECOPY command to back up datafile copies. Datafile copies exist on disk only.

To back up a datafile copy:

1. After starting RMAN, run the BACKUP DATAFILECOPY command at the RMAN prompt. This example backs up datafile /tmp/system01.dbf to tape:

   ```
   BACKUP DATAFILECOPY '/tmp/system01.dbf';
   ```

2. Issue a LIST BACKUP OF DATAFILECOPY command to see a listing of backup sets and pieces. For example:

   ```
   LIST BACKUP OF DATAFILE 1;
   ```

Backing Up Control Files with RMAN

You can make backups of the control file when the database is open or closed. RMAN uses a snapshot control file to ensure a read-consistent version.

If CONFIGURE CONTROLFILE AUTOBACKUP is ON (by default it is OFF), then RMAN automatically backs up the control file and the current server parameter file (if used) in the following circumstances:

- After every BACKUP or COPY command issued at the RMAN prompt.
- Whenever a BACKUP or COPY command within a RUN block is followed by a command that is neither BACKUP nor COPY.
- At the end of every RUN block if the last command in the block was either BACKUP or COPY.
- After database structural changes such as adding a new tablespace, altering the state of a tablespace or datafile (for example, bringing it online), adding a new online redo log, renaming a file, adding a new redo thread, and so forth. This type of autobackup, unlike autobackups that occur in the preceding circumstances, is only on disk. You can run CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK to set a nondefault disk location. Note that the autobackup never causes the associated structural change to fail. For example, if you add a datafile, and if the resulting autobackup fails, then the datafile addition is still successful.

Note: If you use Oracle Enterprise Manager, then you can use the Maintenance wizard to configure the autobackup feature.

The first channel allocated during the backup or copy job creates the autobackup and places it into its own backup set; for post-structural autobackups, the default disk channel makes the backup. RMAN writes the control file and the server parameter file to the same backup piece. After the control file autobackup completes, Oracle writes a message containing the complete path of the backup piece and the device type to the alert log.

As explained by the following table, the RMAN behavior when the `BACKUP` command includes datafile `1` differs depending on whether `CONFIGURE CONTROLFILE AUTOBACKUP` is `ON` or `OFF`.

CONTROLFILE AUTOBACKUP	BACKUP Command Behavior
`ON`	If the backup includes datafile `1`, then RMAN does *not* automatically include the current control file in the datafile backup set. Instead, RMAN writes the control file and server parameter file to a separate autobackup piece. **Note:** The autobackup occurs regardless of whether the `BACKUP` or `COPY` command explicitly includes the current control file, for example, `BACKUP DATABASE INCLUDE CURRENT CONTROLFILE`.
`OFF`	If the backup includes datafile `1`, then RMAN automatically includes the current control file and server parameter file in the datafile backup set. RMAN does *not* create a separate autobackup piece containing the control file and server parameter file.

Consequently, whenever you back up a file or set of files, RMAN automatically makes a control file backup afterward. This control file backup contains metadata about the previous backup, which is crucial in a disaster recovery situation.

If the autobackup feature is not set, then you must manually back up the control file in one of the following ways:

- Back up datafile `1`, which automatically includes the control file in the backup
- 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

Note: A manual backup of the control file is not the same as a control file autobackup. In manual control file backups, only metadata for backups within the current RMAN session is contained in the control file backup, and the backed up control file cannot be automatically restored.

Backing Up the Control File Manually

This procedure creates a control file backup by using the `BACKUP CURRENT CONTROLFILE` command.

To back up the current control file manually:

1. After starting RMAN, run the `BACKUP CURRENT CONTROLFILE` command at the RMAN prompt. This example backs up the current control file to tape and assign a tag:

   ```
   BACKUP CURRENT CONTROLFILE TAG = mondaypmbackup;
   ```

   Note that if the autobackup feature is enabled, then RMAN makes two control file backups in this example: the explicit control file backup (`BACKUP CURRENT CONTROLFILE`) and the subsequent control file and server parameter file autobackup.

2. Optionally, issue a `LIST` command to display backup sets and pieces. For example:

   ```
   LIST BACKUP OF CONTROLFILE;
   ```

Including the Control File in a Backup Set

This procedure creates a control file backup by including the control file in the backup of another object.

To include the current control file in another backup:

Specify the `INCLUDE CURRENT CONTROLFILE` option after specifying the backup object. For example, this command backs up tablespace `users` and includes the current control file in the backup:

```
BACKUP TABLESPACE users INCLUDE CURRENT CONTROLFILE;
```

Note that if the autobackup feature is enabled, then RMAN also generates an autobackup of the control file after the `BACKUP TABLESPACE` command completes.

Backing Up a Control File Copy

This procedure creates a control file backup by using the `BACKUP CONTROLFILECOPY` command.

To back up a control file copy:

1. Ensure that the database is mounted or open.
2. After starting RMAN, run the `BACKUP CONTROLFILECOPY` command at the RMAN prompt. This example creates the control file copy `'/tmp/control01.ctl'` and then backs it up:

   ```
   COPY CURRENT CONTROLFILE TO '/tmp/control01.ctl';
   BACKUP CONTROLFILECOPY '/tmp/control01.ctl';
   ```

3. Optionally, issue a `LIST BACKUP OF CONTROLFILE` command to see a listing of backup sets and pieces. For example:

   ```
   LIST BACKUP OF CONTROLFILE;
   ```

Backing Up Server Parameter Files with RMAN

As explained in "Backing Up Control Files with RMAN" on page 9-8, RMAN automatically backs up the current server parameter file in certain circumstances. You can also use the `BACKUP SPFILE` command to back up the server parameter file explicitly. For example:

```
BACKUP COPIES 2 DEVICE TYPE sbt SPFILE;
```

Note that the backup is only by the server parameter file currently in use by the instance. If the instance is started with a client-side initialization parameter file, then RMAN does not back up anything.

Backing Up Archived Redo Logs with RMAN

Archived redo logs are the key to successful media recovery. Back them up regularly. You can back up logs by issuing `BACKUP ARCHIVELOG` or by backing up datafiles and control files and specifying `BACKUP ... PLUS ARCHIVELOG`.

Note that when specific conditions are met, RMAN attempts to switch out of and archive the current online redo logs when you back up archived redo logs.

> **See Also:** "Automatic Online Redo Log Switches During Backups of Archived Logs" on page 5-17 for the conditions under which RMAN attempts to switch out of the current online log

Backing Up Logs Using BACKUP ARCHIVELOG

To back up archived logs, run BACKUP ARCHIVELOG with the desired filtering options. If you archive to multiple locations, RMAN does not put multiple copies of the same log sequence number into the same backup set. The BACKUP ARCHIVELOG ALL command backs up exactly one copy of each distinct log sequence number.

You can specify the DELETE INPUT option in the BACKUP command, which deletes the archived logs after backing them up. Thus, you can back up archived logs to tape and clear disk space of old logs in one step. RMAN only deletes the specific copy of the archived redo log that it backs up, and then deletes only this copy.

If you specify the DELETE ALL INPUT option, then RMAN makes a backup of each specified log sequence number, but deletes logs from all enabled archiving destinations. For example, assume that you archive to /arc_dest1, /arc_dest2, and /arc_dest3, and you run the following command:

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

RMAN backs up only one copy of each log sequence number that it finds in these directories, and then deletes all copies of all logs that it finds in these directories. If you had specified DELETE INPUT rather than DELETE ALL INPUT, then RMAN would only delete the specific disk copy that it backed up (for example, only the logs in /arc_dest1).

Note that if you issue BACKUP ARCHIVELOG ALL or BACKUP ARCHIVELOG LIKE '...', and if no archived redo logs exist, then RMAN does not signal an error.

See Also:

- "Backing Up and Deleting Multiple Copies of an Archived Redo Log: Example" on page 9-41 for a scenario involving a backup of logs located in multiple destinations
- "Backing Up Archived Logs in a Real Application Clusters Database with RMAN" on page 9-14 to learn about backing up in Oracle Real Application Clusters configuration

To back up archived redo logs using BACKUP ARCHIVELOG:

1. After starting RMAN, run the BACKUP ARCHIVELOG command at the RMAN prompt. This example uses a configured channel to back up one copy of each log sequence number to tape and delete all the copies on disk:

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

Note that depending on the conditions (refer to "Automatic Online Redo Log Switches During Backups of Archived Logs" on page 5-17), RMAN attempts to switch out of archive the current redo log as well.

You can also specify a range of archived redo logs by time, SCN, or log sequence number. The following example backs up all logs that could be used to recover from a point 30 days ago to a point 7 days ago:

```
BACKUP ARCHIVELOG FROM TIME 'SYSDATE-30' UNTIL TIME 'SYSDATE-7';
```

Note that because you specified UNTIL TIME, RMAN does not automatically switch out of archive the current online redo log.

2. Issue a LIST BACKUP OF ARCHIVELOG ALL command to see a listing of backup sets and pieces. For example:

```
LIST BACKUP OF ARCHIVELOG ALL;
```

Backing Up Logs Using BACKUP ... PLUS ARCHIVELOG

When backing up objects other than archived logs, you can specify BACKUP ... PLUS ARCHIVELOG to include backups of archived logs. The command performs the following sequential actions:

1. Runs the ALTER SYSTEM ARCHIVE LOG CURRENT command.
2. Runs BACKUP ARCHIVELOG ALL. Note that if backup optimization is enabled (refer to "Backing Up Files Using Backup Optimization" on page 9-27), then RMAN skips logs that it has already backed up to the specified device.
3. Backs up files specified in BACKUP command.
4. Runs the ALTER SYSTEM ARCHIVE LOG CURRENT command.
5. Backs up any remaining archived logs generated during the backup.

In this way, you guarantee that the datafiles that you are backing up are recoverable to a consistent state.

To back up archived redo logs using BACKUP ... PLUS ARCHIVELOG:

1. Ensure that the database is mounted or open.
2. After starting RMAN, run the BACKUP command at the RMAN prompt and specify PLUS ARCHIVELOG. This example backs up the database and all archived redo logs:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

3. Issue a `LIST BACKUP` command to see a listing of backup sets and pieces. For example:

   ```
   LIST BACKUP OF DATABASE SUMMARY;
   LIST BACKUP OF ARCHIVELOG SUMMARY;
   ```

Backing Up Archived Logs in a Real Application Clusters Database with RMAN

Backing up archived redo logs in an Oracle Real Application Clusters environment poses special problems. To illustrate, assume that you are *not* using NFS on UNIX or mapped drives on Windows to mount remote directories, and that you have configured the cluster as follows:

- Node 1 archives to its local file system `/d1/arc_dest`
- Node 2 archives to its local file system `/d2/arc_dest`
- Node 3 archives to its local file system `/d3/arc_dest`

Note that if you do not use NFS or mapped drives, then each node must archive to a directory with a different path, or the backup of the logs will fail.

In this example, the control file can have records of archived logs that look something like the following

```
/d1/arc_dest/log100_1.arc
/d1/arc_dest/log101_1.arc
/d3/arc_dest/log143_3.arc
/d3/arc_dest/log144_3.arc
/d2/arc_dest/log55_2.arc
/d2/arc_dest/log56_2.arc
/d2/arc_dest/log57_2.arc
/d3/arc_dest/log145_3.arc
/d3/arc_dest/log146_3.arc
/d2/arc_dest/log58_2.arc
/d1/arc_dest/log102_1.arc
/d2/arc_dest/log59_2.arc
/d1/arc_dest/log103_1.arc
/d1/arc_dest/log104_1.arc
```

Assume that you start RMAN on node 2 and run the following command:

```
RUN
{
  ALLOCATE CHANNEL node2 DEVICE TYPE sbt;
  BACKUP ARCHIVELOG ALL.
}
```

In this case, the channel can only see the logs in `/d2/arc_dest`, causing the job to fail because it cannot find the `/d1` or `/d3` logs.

Refer to *Oracle9i Real Application Clusters Administration* to learn how to configure your environment to avoid this situation. For example, you can implement a cluster file system (CFS), or you can use NFS or mapped drives to mount directories so that when RMAN connects to any node, it can access the logs generated by all nodes. However, if you do not configure your environment for remote access, then you can solve the problem of backing up logs simply by specifying which channel should back up which logs.

To back up archived redo logs in an Oracle Real Application Clusters configuration:

1. Ensure that the database is mounted or open.
2. Ensure that a channel is allocated for each node of the Oracle Real Application Clusters database. The following example shows three configured `sbt` channels, one for each node:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE to sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'SYS/oracle@node1'
    PARMS 'ENV=(NSR_SERVER=bksvr1)'; # channel 1 is for first node
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'SYS/oracle@node2'
    PARMS 'ENV=(NSR_SERVER=bksvr2)'; # channel 2 is for second node
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'SYS/oracle@node3'
    PARMS 'ENV=(NSR_SERVER=bksvr3)'; # channel 3 is for third node
```

3. Back up the logs. For example:

```
BACKUP ARCHIVELOG ALL;
```

See Also:

- *Oracle9i Recovery Manager Reference* for `BACKUP` syntax and a description of the `LIKE` parameter
- "Backing Up in an Oracle Real Application Clusters Environment: Example" on page 9-44 for a scenario involving backing up archived logs in an Oracle Real Application Clusters environment

Duplexing Backups

Prudence suggests making multiple copies of backups to protect against disaster, media damage, or human error. RMAN can make up to four copies of a backup set simultaneously, each an exact duplicate of the others. A copy of a 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`).

In most cases, the easiest method is to use `BACKUP COPIES` or `CONFIGURE ... BACKUP COPIES` to duplex backups. Note that there is little value in creating multiple copies on the same physical media. 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 will automatically put each copy onto a separate medium (for example, a separate tape).

Note: You must set the `BACKUP_TAPE_IO_SLAVES` initialization parameter to `TRUE` in order to perform duplexed backups to an `sbt` device. Otherwise, an error is signaled. RMAN uses as many spawned processes as needed for the number of backup copies you request.

Duplexing Backups with CONFIGURE BACKUP COPIES

The `CONFIGURE ... BACKUP COPIES` command specifies the number of identical backups that you want to create on the specified device type. This command applies only to datafiles (which includes current control files as well as control file autobackups) and archived logs. You must have automatic channels configured.

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. This example configures duplexing for datafiles and archived logs on tape as well as duplexing for datafiles (but not archived logs) on disk:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/save1/%U', '/save2/%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. Duplex the backup. The following command backs up the database and archived logs to tape, making two copies of each datafile and archived redo log:

```
BACKUP DATABASE PLUS ARCHIVELOG; # uses default sbt channel
```

The following command backs up the database to disk, placing one copy in the `/save1` directory and the other in the `/save2` directory:

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

3. Issue a `LIST BACKUP` command to see a listing of backup sets and pieces (the `#Copies` column shows the degree of duplexing). For example, enter:

   ```
   LIST BACKUP SUMMARY;
   ```

Duplexing Backups with BACKUP COPIES

The `COPIES` option of the `BACKUP` command overrides every other `COPIES` or `DUPLEX` setting.

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` directory:

   ```
   BACKUP COPIES 3 INCREMENTAL LEVEL = 0 DATABASE;
   ```

   Note that because you specified `COPIES` on the `BACKUP` command itself, RMAN makes three copies 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 degree of duplexing). For example, enter:

   ```
   LIST BACKUP SUMMARY;
   ```

Making Incremental Backups with RMAN

You can make consistent or inconsistent incremental backups of the database or individual tablespaces or datafiles. If the database is in `NOARCHIVELOG` mode, then you can only make consistent incremental backups, so the database must be closed cleanly. In `ARCHIVELOG` mode the database can be open or closed.

To make an incremental backup:

1. After starting RMAN, run the `BACKUP INCREMENTAL` command at the RMAN prompt. This example makes a level 0 backup:

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

This example makes a differential level 1 backup of the SYSTEM tablespace and datafile sales.f; 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
  DATAFILE '?/oradata/trgt/tools01.dbf';
```

This example makes a cumulative level 2 backup of the tablespace users; it will only back up those data blocks changed since the most recent level 1 or level 0 backup:

```
BACKUP INCREMENTAL LEVEL = 2 CUMULATIVE TABLESPACE users;
```

2. Optionally, issue a LIST BACKUP command to see a listing of backup sets and pieces. For example:

```
LIST BACKUP OF DATABASE;
```

Making Split Mirror Backups with RMAN

Many sites keep an backup of the database stored on disk in case a failure occurs on the primary database or an incorrect user action such as a DROP TABLE requires incomplete 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 an datafile backup on disk is to use disk mirroring. For example, you can use the operating system to 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 operations. 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 following procedure shows how to make a split mirror backup with the optional SUSPEND/RESUME functionality. The SUSPEND/RESUME feature is not required for split mirror backups in most cases, although it is necessary if your system requires the database cache to be free of dirty buffers before the volume can be split.

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. For example, to place tablespace users in backup mode, run the following commands:

   ```
   % rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
   RMAN> SQL 'ALTER TABLESPACE users BEGIN BACKUP';
   ```

2. Suspend the I/Os if your mirroring software or hardware requires it. For example, enter the following SQL statement:

   ```
   RMAN> SQL 'ALTER SYSTEM SUSPEND';
   ```

3. Split the mirrors for the underlying datafiles contained in these tablespaces. Refer to *Oracle9i User-Managed Backup and Recovery Guide* for more information about splitting mirrors.
4. Take the database out of the suspended state:

   ```
   RMAN> SQL 'ALTER SYSTEM RESUME';
   ```

5. Take the tablespaces out of backup mode. For example, enter:

   ```
   RMAN> SQL 'ALTER TABLESPACE users END BACKUP';
   ```

6. Start an RMAN session and then catalog the user-managed mirror copies as datafile copies with the CATALOG command. For example, enter:

   ```
   RMAN> CATALOG DATAFILECOPY '/dk2/oradata/trgt/users01.dbf';  # catalog split mirror
   ```

7. Back up the datafile copies. For example, assuming that you have configured automatic channels, run the BACKUP DATAFILECOPY command at the prompt:

   ```
   RMAN> 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:

   ```
   RMAN> CHANGE DATAFILECOPY '/dk2/oradata/trgt/users01.dbf' UNCATALOG;
   ```

9. Resilver the split mirror for the affected datafiles.

See Also: *Oracle9i SQL Reference* for ALTER SYSTEM SUSPEND syntax

Backing Up Files at a Standby Database Site with RMAN

This section contains these topics:

- About RMAN Backups of Standby Database Datafiles and Archived Logs
- Restrictions and Usage Notes When Making RMAN Standby Database Backups
- Interpreting the RC_ARCHIVED_LOG View
- Determining When to Back Up Standby Database Archived Redo Logs
- Backing Up a Standby Database with RMAN

See Also: Chapter 13, "Creating a Standby Database with Recovery Manager" for an overview of how RMAN interacts with a standby database

About RMAN Backups of Standby Database Datafiles and Archived Logs

RMAN can back up the standby database and its associated archived redo logs. Standby backups of datafiles and archived redo logs are fully interchangeable with primary database backups. In other words, you can run the RESTORE command to restore a backup of a standby datafile to the primary database, and you can restore a backup of a primary datafile to the standby database. The standby control file and primary control file, however, are *not* interchangeable.

Backing up standby files is often better than backing up the production files, for the following reasons:

- Because the standby database is not the production database, a standby backup does not interfere with transactions or batch jobs in the production database. Hence, you can use the standby database as a backup host without interfering with the production system.
- If the standby and primary databases are on separate hosts, then standby backup operations do not consume CPU cycles, allocate memory, or consume other resources on the production host.

Both the primary database and standby database should use the same recovery catalog. Even though these databases share the same DBID, RMAN is able to differentiate the standby database from the primary. Note that you do not need to register the standby database in the catalog if the primary is already registered: simply connect to the standby database and run the `BACKUP` command.

If you activate a standby database using `ALTER DATABASE ACTIVATE STANDBY DATABASE`, then the standby database becomes the new primary database. Because a `RESETLOGS` must be performed at standby activation, RMAN creates a new incarnation record for the new primary database. Backups of this new incarnation of the primary database are not different from backups of the primary database after a `RESETLOGS` operation.

> **See Also:** "Opening the Database After Media Recovery" in the chapter "Performing Recovery Using SQL and SQL*Plus" from *Oracle9i User-Managed Backup and Recovery Guide* to learn about `RESETLOGS` operations

Restrictions and Usage Notes When Making RMAN Standby Database Backups

Note these restrictions when making backups of a standby database:

- For the standby database backups to be usable for restore jobs at the primary site, you must be connected to the recovery catalog when backing up the standby database or resynchronize the standby database shortly after the backup. This step is necessary is because there is no way for the primary database to know about the standby backups unless these backup records are stored in the recovery catalog.
- You cannot back up the standby control file being used by the standby database.
- You cannot make an image copy or non-RMAN backup of the standby control file and then use it to restore the primary database.
- You cannot connect to the standby database and then use the `DUPLICATE` command to create another standby database. To create another standby database, connect to the primary database and run `DUPLICATE`.
- You should not attempt to register the standby database in the recovery catalog.
- When you back up the standby database, you must connect to the standby database with the `TARGET` keyword (not the `AUXILIARY` keyword). Essentially, the standby database is "substituting" for the primary database during the backup.

> **See Also:** *Oracle9i Data Guard Concepts and Administration* for more information about backing up standby databases

Interpreting the RC_ARCHIVED_LOG View

If you are making archived log backups on the standby site, then ensure that all necessary archived logs are available on the primary site in the event of a failure. The situation can be confusing because archived logs can be in any of the following locations:

- In the archived log destination directories on the primary site
- Backed up at the primary site and then deleted from the archived log destination directories on the primary site
- Backed up at the standby site and then deleted from the archived log destination directories on the primary site

The recovery catalog view `RC_ARCHIVED_LOG` indicates when an archived log is located at the primary site and when it is at the standby site. The archived logs information in `RC_ARCHIVED_LOG` is important because you need to know when you must back up a log or copy it to the primary site from the standby site.

For example, assume that you start SQL*Plus, connect to the recovery catalog database as the recovery catalog schema owner, and then run this query:

```
SELECT SEQUENCE#, IS_STANDBY
FROM RC_ARCHIVED_LOG;

   SEQUENCE# IS_
  ---------- ---
         113 YES
         114 NO
         115 NO
         116 YES
         116 NO
```

The `IS_STANDBY` column indicates whether the log is located at the standby site (`YES`) or at the primary site (`NO`). If the same log sequence number has `IS_STANDBY` set to both `YES` and `NO`, then the log is located at both the standby and primary sites. For example, sequence number 116 has both a `YES` and `NO` value for `IS_STANDBY`, so it is at the primary and standby sites.

Determining When to Back Up Standby Database Archived Redo Logs

If you are making all your backups at the standby site, then you must ensure that you have backed up all the archived logs generated by the primary database. You have two methods for determining whether you need to back up a standby database archived log so that RMAN can use it for recovery:

- Running `LIST` command
- Querying `RC_ARCHIVED_LOG`

Using the LIST Command to Determine When to Back Up Standby Logs

Use the `LIST BACKUP OF ARCHIVELOG ALL` command to determine which logs RMAN has backed up.

To determine whether a log backup is needed by using the LIST command:

1. Query the recovery catalog to determine the locations of the archived redo logs. For example, issue:

```
SELECT SEQUENCE#, IS_STANDBY
FROM RC_ARCHIVED_LOG;

 SEQUENCE# IS_
---------- ---
       113 YES
       114  NO
       115  NO
       116  NO
```

   This output indicates that log sequence 113 is at the standby site but not at the primary site, and archived logs 114 through 116 are at the primary site but not the standby site.

2. Determine which logs are backed up by connecting to the recovery catalog and running a `LIST BACKUP` command in RMAN. For example:

```
LIST BACKUP OF ARCHIVELOG ALL;

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ----------------------
319     4          394624547  0  394624546  5          11-APR-00

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    320     1   1   AVAILABLE   11-APR-00              /vobs/oracle/dbs/05boavh2_1_1

    List of Archived Logs Included
    Thrd Seq     Low SCN    Next SCN   Low Time        Next Time
    ---- ------- ---------- ---------- --------------- ---------------
    1    116     95153      95156      07-APR-00       07-APR-00
```

   This output shows that RMAN has backed up archived log 116, but has not backed up archived log 113. Because log 113 exists only at the standby site, you should either back up this log or copy it to the primary site.

Querying RC_ARCHIVED_LOG to Determine When to Back Up Standby Logs

You can query the recovery catalog to determine which logs RMAN has backed up.

To determine whether a log backup is needed by querying the catalog:

1. Query the RC_ARCHIVED_LOG recovery catalog view to determine whether all archived logs necessary for recovery are on disk. For example, issue the following query, where *first_log_needed_for_recovery* is the sequence number of the log that begins recovery and *expected_num_of_logs* is the number of logs that should be applied during complete recovery:

   ```
   SELECT 1 FROM RC_ARCHIVED_LOG
   WHERE SEQUENCE# >= first_log_needed_for_recovery
   AND IS_STANDBY='NO'
   AND STATUS='A'
   HAVING COUNT(*) = expected_num_of_logs;
   ```

   If the query returns no rows, then you do not have all logs necessary for complete recovery on disk. If the query does return rows, then you do have the necessary logs for complete recovery on disk.

2. Query the RC_BACKUP_REDOLOG view to determine whether you have backups of the logs necessary for complete recovery. For example, issue the following query, where *first_log_needed_for_recovery* is the sequence number of the log that begins recovery and *expected_num_of_logs* is the number of logs that should be applied during complete recovery:

   ```
   SELECT 1 FROM RC_BACKUP_REDOLOG
   WHERE SEQUENCE# >= first_log_needed_for_recovery
   AND STATUS='A'
   HAVING COUNT(DISTINCT SEQUENCE#) = expected_num_of_logs;
   ```

   If the query returns no rows, then you do not have backups of all logs necessary for complete recovery. If the query does return rows, then you do have backups of all logs necessary for complete recovery.

Backing Up a Standby Database with RMAN

Use the RMAN BACKUP command to back up the standby database. A backup of the standby database is exactly the same as a backup of the primary database, except that the backup takes place on the standby site. The primary database has no influence on the backup of the standby database. Note that when you connect to the standby database to perform the backup, you connect using the TARGET keyword and not the AUXILIARY keyword.

As the following table shows, whether the standby database backup is consistent or inconsistent depends on the state of the standby database when the backup is made. Only a consistent backup can be restored without performing media recovery.

Standby Database Status	Backup Status
Shutdown cleanly and then mounted (but not placed in recovery mode)	Consistent
Mounted after instance failure or `SHUTDOWN ABORT`	Inconsistent
Manual recovery mode	Inconsistent
Managed recovery mode	Inconsistent
Read-only mode	Inconsistent

To make a whole database backup of a standby database:

1. To make a consistent backup of the standby database, make sure that the last shutdown of the standby database was clean and that it was not placed in recovery mode after that time, and then mount the control file. For example:

   ```
   sqlplus SYS/oracle@sbdb1 <<EOF
   SHUTDOWN IMMEDIATE
   STARTUP NOMMOUNT PFILE=initSTANDBY.ora
   ALTER DATABASE MOUNT STANDBY DATABASE;
   EOF
   ```

   You can back up the standby database when it is in any other mode, but the backups will be inconsistent.

2. Start RMAN and connect to the standby database with the `TARGET` keyword (not the `AUXILIARY` keyword) and the recovery catalog database. You *must* be connected to the recovery catalog. For example, enter:

   ```
   % rman TARGET SYS/oracle@sbdb1 CATALOG rman/cat@catdb
   ```

3. If do not have automatic channels configured, then manually allocate one or more channels of type `DISK` or `sbt`. Note that you are connected to the standby host, so the backups are made by server sessions on the standby (not the primary) host.

   This example backs up all the standby datafiles as well as the control file and archived logs by using automatic channels:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

You can use the FORMAT parameter to specify a filename for the backup piece. For example, enter:

```
BACKUP DATABASE FORMAT '/tmp/standby_%U';  # %U generates a unique filename
```

You can specify TAG to give a tag to the backup. For example, enter:

```
BACKUP DATABASE TAG = 'weekly_standby_backup';   # gives the standby backup a tag
```

RMAN assigns a default tag to backups. Refer to the BACKUP ... TAG description in *Oracle9i Recovery Manager Reference* for the default format.

4. If desired, issue a LIST command to see a listing of backup sets and pieces.

Backing Up Backup Sets with RMAN

Use the BACKUP BACKUPSET command to back up backup sets rather than database files. This command is 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 deallocating the space on disk

Note: You cannot duplex control file autobackups when running BACKUP BACKUPSET. RMAN always makes one and only one copy on the specified media.

To back up backup sets from disk to tape:

1. Assuming that you have configured an automatic sbt channel, issue the BACKUP BACKUPSET command at the RMAN prompt. This example allocates the default disk channel and the configured sbt channel to back up all backup sets to tape:

   ```
   BACKUP DEVICE DEVICE TYPE sbt BACKUPSET ALL;
   ```

   This example backs up all disk backup sets to tape and then deletes the input disk backups:

   ```
   BACKUP DEVICE TYPE sbt BACKUPSET ALL DELETE INPUT;
   ```

2. Issue a LIST command to see a listing of backup sets and pieces.

Restarting and Optimizing RMAN Backups

RMAN supports two distinct features by which it can back up only those files that require backups: **restartable backups** and **backup optimization**.

With the restartable backup feature, RMAN backs up only those files that were not backed up after a specified date. For example, by specifying the `NOT BACKED UP SINCE TIME` clause, you can direct RMAN to back up only those files that were not backed up within the last day.

With backup optimization, the `BACKUP` command skips the backup of a file if the identical file has already been backed up to the allocated device type. To override this behavior and back up all files whether or not they have changed, specify the `FORCE` option on the `BACKUP` command. To enable or disable backup optimization, specify `ON` or `OFF` on the `CONFIGURE BACKUP OPTIMIZATION` command.

Additionally, a third feature can archive unarchived online logs as well as back up archived logs (refer to "Backing Up Logs Using BACKUP ... PLUS ARCHIVELOG" on page 9-13).

> **See Also:** "Backup Optimization" on page 5-56 for a conceptual overview of optimization, and "Restartable Backups" on page 5-61 for a conceptual overview of restartable backups

Backing Up Files Using Backup Optimization

For backup optimization to be enabled, you must `CONFIGURE BACKUP OPTIMIZATION` to `ON`. Backup optimization is `OFF` by default.

To optimize a backup:

1. If you have not already enabled backup optimization, then enable it by running the `CONFIGURE OPTIMIZATION` command. For example, enter:

```
CONFIGURE OPTIMIZATION ON;
```

2. Back up the desired files. For example, this command uses a preconfigured channel to back up two copies of the database and archived logs to disk:

```
BACKUP DEVICE TYPE DISK COPIES 2 DATABASE PLUS ARCHIVELOG;
```

The following example backs up logs to an `sbt` device:

```
BACKUP DEVICE TYPE sbt ARCHIVELOG ALL;
```

Depending on the retention policy and backup duplexing settings, RMAN backs up only those files that changed after the last backup. RMAN does not signal an error when it skips a file, even if it skips all files.

See Also: "Backup Optimization" on page 5-56 for a conceptual overview of optimization and backup retention policies

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 you do not specify the `SINCE` 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:

Specify a valid date in the `SINCE TIME` parameter. For example, this command 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;
```

Performing a Backup Validation with RMAN

You can use the `VALIDATE` keyword of the `BACKUP` command to do the following:

- Check datafiles for physical and logical corruption
- Confirm that all database files exist and are in the correct locations

RMAN does not actually produce backup sets, but rather scans the specified files to determine whether they can be backed up and are not corrupted. In this sense, the `BACKUP VALIDATE` command is similar to the `RESTORE VALIDATE` command, except for backups rather than restore jobs. If the backup validation discovers corrupt blocks, then RMAN updates the `V$DATABASE_BLOCK_CORRUPTION` view with rows describing the corruptions. After a corrupt block is repaired, the row identifying this block is deleted from the view.

For example, you can validate that all database files and archived redo logs can be backed up by running a command as follows:

```
BACKUP VALIDATE DATABASE ARCHIVELOG ALL;
```

RMAN displays the same output that it would if it were really backing up the files. If RMAN cannot validate the backup of one or more of the files, then it displays 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 08/29/2001 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
```

You cannot use the `MAXCORRUPT` or `PROXY` parameters with the `VALIDATE` option.

See Also:

- *Oracle9i Recovery Manager Reference* for `BACKUP` syntax
- "Recovering Blocks Listed in V$DATABASE_BLOCK_CORRUPTION" on page 10-40 to learn how to repair corrupt blocks discovered by `BACKUP ... VALIDATE`

Copying Files with RMAN

In many cases, making a copy is better than making a backup, because copies are not in an RMAN-specific format and hence are suitable for use without any additional processing. In contrast, you must process a backup set with a `RESTORE` command before it is usable. So, you can perform media recovery on a datafile copy, but not directly on a backup set, even if it contains only one datafile and is composed of a single backup piece.

Note: You cannot make incremental copies, although you can use the `LEVEL` parameter to make a copy serve as a basis for subsequent incremental backup sets.

Use the `COPY` command to create image copies. RMAN always writes the output file to disk. You can copy the following types of files:

- Datafiles (current or copies)
- Archived redo logs
- Control files (current or copies)

Copying the Whole Database

There is no COPY DATABASE command to correspond to the BACKUP DATABASE command, so you must copy the datafiles individually. Run the REPORT SCHEMA command to determine the filenames of the datafiles.

To make consistent copies of all database files:

1. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. For example, enter:

   ```
   SHUTDOWN IMMEDIATE
   STARTUP MOUNT
   ```

   If the database is open, or if it is mounted but *not* closed cleanly when last opened, then the backup will be inconsistent.

2. Generate a report of the current database schema:

   ```
   REPORT SCHEMA;

   Report of database schema
   File K-bytes    Tablespace           RB segs Datafile Name
   ---- ---------- -------------------- ------- -------------------
   1        307200 SYSTEM               ***     /oracle/oradata/trgt/system01.dbf
   2         20480 UNDOTBS              ***     /oracle/oradata/trgt/undotbs01.dbf
   3         10240 CWMLITE              ***     /oracle/oradata/trgt/cwmlite01.dbf
   4         10240 DRSYS                ***     /oracle/oradata/trgt/drsys01.dbf
   5         10240 EXAMPLE              ***     /oracle/oradata/trgt/example01.dbf
   6         10240 INDX                 ***     /oracle/oradata/trgt/indx01.dbf
   7         10240 TOOLS                ***     /oracle/oradata/trgt/tools01.dbf
   8         10240 USERS                ***     /oracle/oradata/trgt/users01.dbf
   ```

3. Assuming that you have configured automatic channels, issue the COPY command at the RMAN prompt. Copy all of the datafiles and include the current control file. For example, enter the following at the RMAN prompt:

   ```
   COPY
     DATAFILE 1 TO '/tmp/system01.dbf',
     DATAFILE 2 TO '/tmp/undotbs01.dbf',
     DATAFILE 3 TO '/tmp/cwmlite01.dbf',
     DATAFILE 4 TO '/tmp/drsys01.dbf',
     DATAFILE 5 TO '/tmp/example01.dbf',
     DATAFILE 6 TO '/tmp/indx01.dbf',
     DATAFILE 7 TO '/tmp/tools01.dbf',
     DATAFILE 8 TO '/tmp/users01.dbf',
     CURRENT CONTROLFILE TO '/tmp/control01.ctl';
   ```

4. Issue a `LIST COPY OF DATABASE` command to see a listing of image copies. For example:

   ```
   LIST COPY OF DATABASE;
   ```

Copying Selected Datafile Copies and Archived Redo Logs

Besides copying datafiles and control files, you can copy other copies and archived redo logs (and archived redo log copies).

To copy datafiles, archived redo logs, and control files:

1. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. For example, enter:

   ```
   SHUTDOWN IMMEDIATE
   STARTUP MOUNT
   ```

   If the database is open, or if it is mounted but *not* closed cleanly when last opened, then the backup will be inconsistent.

2. Assuming that you have configured automatic channels, issue the `COPY` command at the RMAN prompt. Copy the desired datafiles, archived redo logs, and control files. For example, enter:

   ```
   COPY
     DATAFILECOPY '/tmp/system01.dbf' TO '/save/system01.dbf',
     DATAFILECOPY TAG = 'weekly_df8_copy' TO '/tmp/users01.dbf',
     # copy archived redo logs
     ARCHIVELOG '?/oradata/trgt/arch/archive1_1.dbf' TO '/tmp/archive1_1.dbf',
     ARCHIVELOG '?/oradata/trgt/arch/archive1_2.dbf' TO '/tmp/archive1_2.dbf'',
     # copy a control file copy
     CONTROLFILECOPY '/tmp/control01.ctl' TO '/save/control01.ctl';
   ```

3. Issue a `LIST COPY` command to see a listing of image copies.

Overriding the Control File Autobackup Format

As explained in "Control File and Server Parameter File Autobackups" on page 5-47, if the autobackup feature is enabled then RMAN automatically backs up the control file after `BACKUP` or `COPY` commands.

The section "Configuring the Control File Autobackup Format" on page 8-19 describes the default format for the control file autobackup. You can also use the `CONFIGURE` command to change the default autobackup format to a new value.

Note that you must include the %F format in the string, or RMAN signals an error. This format change is persistent across all RMAN sessions. For example, you can change the default as follows:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'c_%F';
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '?/oradata/%F.bck';
```

You can override the configured location for the control file backup in an RMAN session by using the SET command to specify the directory and possibly a prefix and suffix. For example, you can set the following:

```
SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'controlfile_%F';
RUN { SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/tmp/%F.bck'; }
```

The order of precedence of control file autobackup format commands is:

1. SET CONTROLFILE AUTOBACKUP FORMAT within RUN
2. SET CONTROLFILE AUTOBACKUP FORMAT at the RMAN prompt
3. CONFIGURE CONTROLFILE AUTOBACKUP FORMAT

See Also:

- "Resynchronizing the Recovery Catalog" on page 16-11 for an overview of RMAN resynchronization using the snapshot control file
- *Oracle9i Recovery Manager Reference* for CONFIGURE SNAPSHOT CONTROLFILE command syntax
- "Backup Fails Because of Control File Enqueue: Scenario" on page 15-33 for a scenario involving a backup that fails because of an enqueue

RMAN Backup and Copy Examples

This section contains these topics:

- Specifying the Device Type on the BACKUP Command: Example
- Skipping Tablespaces when Backing Up a Database: Example
- Restarting a Backup: Example
- Spreading a Backup Across Multiple Disk Drives: Example
- Backing Up a Large Database to Multiple File Systems: Example

Specifying the Device Type on the BACKUP Command: Example

Assume that you configure an automatic `sbt` channel as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1; # configure device
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='...'; # configure options for channels
CONFIGURE DEFAULT DEVICE TYPE to sbt; # set default device type
```

Assume that you want to back up the database to disk and use the default configured `DISK` channel. You can specify that the `BACKUP` command should use a `DISK` channel as follows:

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

To back up the database to the `sbt` device run this command:

```
BACKUP DATABASE;
```

Skipping Tablespaces when Backing Up a Database: Example

The following example assumes that the database is running in ARCHIVELOG mode and that you have an automatic sbt channel configured as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='ENV=(NSR_DATA_VOLUME_POOL=BackupPool)';
```

To back up the database while skipping offline and read-only tablespaces, you can run the following command:

```
BACKUP DATABASE
  SKIP READONLY
  SKIP OFFLINE;
```

You need to back up a read-only tablespace only once after it has been made read-only. You can use the SKIP READONLY option to skip read-only datafiles. If you use the SKIP OFFLINE option, then the BACKUP command does not attempt to access offline datafiles. Use this option if the offline datafiles are not available.

Another way to persistently skip tablespaces across RMAN sessions is to issue the CONFIGURE EXCLUDE command for each tablespace that you always want to skip. For example, you may always want to skip the example tablespace, which has been made read-only. You can then issue:

```
CONFIGURE EXCLUDE FOR TABLESPACE example;
```

Then, whenever you run BACKUP DATABASE, RMAN skips this tablespace. You do not have to specify a SKIP clause on the BACKUP command. You can override this behavior and include the earnings tablespace as follows:

```
BACKUP DATABASE NOEXCLUDE;
```

Restarting a Backup: Example

Assume that you back up the database and archived logs every night to tape by running this command:

```
BACKUP FILESPERSET 2 DATABASE PLUS ARCHIVELOG;
```

This command limits each backup set to two datafiles, so it produces multiple backup sets. Assume that the media management device fails halfway through the backup and is then restarted. The next day you discover that only half the backup sets completed. In this case, you can run this command in the evening:

```
BACKUP
  # Note that the NOT BACKED UP SINCE clause should be placed immediately after the BACKUP
  # keyword or after each individual backupSpec clause
  NOT BACKED UP SINCE TIME 'SYSDATE-1'
  FILESPERSET 2
  DATABASE PLUS ARCHIVELOG;
```

RMAN backs up only files that were not backed up during in the previous 24 hours. When RMAN finds out that particular file is already backed up it displays output similar to the following:

```
RMAN-06501: skipping datafile 1; already backed up on MAY 02 2001 18:10:00
RMAN-06501: skipping datafile 2; already backed up on MAY 02 2001 18:09:45
RMAN-06501: skipping datafile 3; already backed up on MAY 02 2001 18:09:45
```

Spreading a Backup Across Multiple Disk Drives: Example

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 if you need 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:

```
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 DATABASE;
}
```

You can accomplish the same result by configuring automatic 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 1st ch
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT '/disk2/%d_backups/%U'; # configure 2nd ch
CONFIGURE CHANNEL 3 DEVICE TYPE DISK FORMAT '/disk3/%d_backups/%U'; # configure 3rd ch
BACKUP DATABASE;
```

If you specify a nonexistent directory, RMAN displays output such as the following:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of backup command on ORA_DISK_1 channel at 08/29/2001 14:36:04
ORA-19504: failed to create file "foo/0cd2momi_1_1"
ORA-27040: skgfrcre: create error, unable to create file
SVR4 Error: 2: No such file or directory
```

Backing Up a Large Database to Multiple File Systems: Example

In this scenario, you have a 40 GB database that you want to back up to disk. Because RMAN can only write one backup piece on a raw disk device, you decide to spread the backup across file systems. You decide to back up to four file systems and make each backup set roughly the same size: 10 GB. You want each backup piece to be no more than 2 GB so that each backup set contains five backup pieces.

You decide to use the FORMAT parameter of the CONFIGURE CHANNEL command so that each channel will write to a different file system. You use conversion variables to guarantee unique names for the backup pieces. For example, the following commands configure channels across four file systems (/fs1, /fs2, /fs3, /fs4) and group the datafiles so that each backup set is about the same size.

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 4;  # first, configure the device for parallelism 4
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT='/fs1/%u.%p' MAXPIECESIZE 2G;
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT='/fs2/%u.%p' MAXPIECESIZE 2G;
CONFIGURE CHANNEL 3 DEVICE TYPE DISK FORMAT='/fs3/%u.%p' MAXPIECESIZE 2G;
CONFIGURE CHANNEL 4 DEVICE TYPE DISK FORMAT='/fs4/%u.%p' MAXPIECESIZE 2G;
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
```

Then, you can run this command every night to generate four backup sets, each in a different directory and each approximately the same size:

```
BACKUP DATABASE;
```

You can also back up the backup sets from disk to four different tapes from a tape pool by setting PARALLELISM=4 for the sbt device (and specifying the appropriate vendor-specific PARMS for the sbt channel), as in the following example:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 4;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='...';
BACKUP DEVICE TYPE sbt BACKUPSET ALL DELETE INPUT;
```

Specifying the Size of Backup Sets: Example

When making backups, RMAN divides the total number of files requiring backups by the number of allocated channels to calculate the number of files to place in each backup set. Use the FILESPERSET and MAXSETSIZE parameters to override this calculation and specify how many files should go in each backup set.

Controlling the Size of Backup Sets with FILESPERSET

When you specify the FILESPERSET parameter, RMAN compares the FILESPERSET value to the automatically calculated value (number of files for each allocated channel) and takes the lower of the two values, thereby ensuring that all channels are used. For example, if you are backing up twelve datafiles with three channels, and set FILESPERSET=2, RMAN puts two datafiles into each backup rather than four.

If the number of files specified or implied by the combined *backupSpec* clauses is greater than FILESPERSET, for example, if eight total files need backing up when FILESPERSET=4, then RMAN creates multiple backup sets to maintain the correct ratio of files for each backup set.

If you do not specify FILESPERSET, then RMAN compares the calculated value (number of files divided by number of allocated channels) to the default value of 64 and takes the lower of the two values, again ensuring that all channels are used. The default value of 64 is high for most applications: specify a lower value or use the MAXSETSIZE parameter to limit the size of a backup set.

RMAN always attempts to create enough backup sets so that all allocated channels have work to do. An exception to the rule occurs when there are more channels than files to back up. For example, if RMAN backs up one datafile when three channels are allocated, then two channels are idle.

This example configures disk parallelism to 4:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 4;
```

This script runs a backup specifying that no more than 3 datafiles go in any one backup set, and no more than 16 archived logs in any one backup set:

```
BACKUP
{
  DATABASE
    FILESPERSET = 3
  ARCHIVELOG ALL
    FILESPERSET = 16;
}
```

Controlling the Size of Backup Sets with MAXSETSIZE

The MAXSETSIZE parameter 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 sets to this size.

You can use MAXSETSIZE to limit the size of backup sets so that the database is divided among more than one backup set. Otherwise, if the backup fails partway through, then you must start the database backup from scratch. If you configure MAXSETSIZE so that you generate multiple backup sets, however, then if the backup fails partway through, you can use the restartable backup feature to back up only those files that were not backed up during the previous attempt.

The MAXSETSIZE parameter is easier to use than FILESPERSET when you make backups of archived redo logs. This example configures a tape device, then backs up archived redo logs to tape, limiting the size to 100 MB so that if the backup fails partway through, it can be restarted:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
BACKUP MAXSETSIZE = 100M ARCHIVELOG ALL;
```

This example accomplishes the same result with CONFIGURE MAXSETSIZE:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE MAXSETSIZE = 100M;
BACKUP ARCHIVELOG ALL;
```

Note that if you specify a MAXSETSIZE value that is smaller that the smallest file that you are backing up, then 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 08/29/2001 14:40:33
RMAN-06182: archive log larger than SETSIZE: thread 1 seq 1
            /oracle/oradata/trgt/arch/archive1_1.dbf
```

Limiting the Size of Backup Pieces: Example

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 MAXPIECESIZE parameter of the CONFIGURE CHANNEL or ALLOCATE CHANNEL command to limit the size of backup pieces.

For example, to limit the backup file size to 2000 MB or less, you can configure the automatic DISK channel as follows and then run BACKUP DATABASE:

```
CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE 2000M;  # max file size for backup pieces
BACKUP DATABASE;
```

Note that in version 2.0 of the media management API, media vendors can specify the maximum size of a backup piece, causing RMAN to comply with this restriction automatically.

Multiplexing Datafiles in a Backup: Example

Assume you need to back up a database called `trgt`. The following conditions exist in the database environment:

- You have three tape drives available for the backup.
- The database contains 1000 relatively small datafiles.

You do not want a large number of backup sets, so you set `FILESPERSET=64`. However, you do not want to multiplex more than four files because this value is sufficient to keep the tape drive streaming. So, you set `MAXOPENFILES=4`.

Assume that you have configured three `sbt` channels as follows (using the `PARMS` setting appropriate required by each media management device):

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;   # parallelize to 3 tape drives
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt MAXOPENFILES 4 PARMS='...';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt MAXOPENFILES 4 PARMS='...';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt MAXOPENFILES 4 PARMS='...';
```

Then, you can create the following backup script:

```
CREATE SCRIPT TRGT_FULL
{
   BACKUP FILESPERSET 4
     DATABASE FORMAT 'TRGT.FULL.%d.%s.%p';
}
```

Alternatively, you can create a script that allocates channels manually:

```
CREATE SCRIPT TRGT_FULL
{
   ALLOCATE CHANNEL t1 DEVICE TYPE sbt MAXOPENFILES 4 PARMS='...';
   ALLOCATE CHANNEL t2 DEVICE TYPE sbt MAXOPENFILES 4 PARMS='...';
   ALLOCATE CHANNEL t3 DEVICE TYPE sbt MAXOPENFILES 4 PARMS='...';
   BACKUP FILESPERSET 64
     DATABASE FORMAT 'TRGT.FULL.%d.%s.%p';
}
```

You can then run the script as follows:

```
RUN { EXECUTE SCRIPT TRGT_FULL; }
```

This script backs up the whole database, including all datafiles and the control file. Because there are 1001 files to be backed up (1000 datafiles and a control file) and a maximum of four files for each backup set, Oracle creates 16 backup sets. Because `MAXOPENFILES` is 4, each channel reads from 4 datafiles simultaneously. The backup piece filenames have the following format, where `db_name` is the database name, `set_num` is the backup set number, and `piece_num` is the piece number:

```
TRGT.FULL.db_name.set_num.piece_num
```

For example, a backup piece could have the following filename:

```
TRGT.FULL.trgt.3.1
```

If no backup sets have been recorded in the recovery catalog before this job, then `set_num` will range from one through seven and `piece_num` will be one or more.

Backing Up Archived Redo Logs in a Failover Scenario: Example

Assume that you set your initialization parameters so that you archive to the following local destinations:

```
LOG_ARCHIVE_DEST_1 = 'LOCATION=/disk1/arch/'
LOG_ARCHIVE_DEST_2 = 'LOCATION=/disk2/arch/'
LOG_ARCHIVE_DEST_3 = 'LOCATION=/disk3/arch/'
```

Each directory contains the same set of logs, starting with log sequence 1 and ending at log sequence 400. Unknown to you, a user inadvertently deletes logs 300 through 400 from `/disk1/arch` and logs 350 through 400 from `/disk2/arch`. You run this backup command:

```
BACKUP ARCHIVELOG
  FROM SEQUENCE 288 UNTIL SEQUENCE 388
  THREAD 1
  DELETE INPUT;
```

RMAN begins backing up logs starting with log sequence 288. If the copy of log 300 that was deleted from `/disk1/arch` is the one that RMAN attempts to back up, then RMAN checks the repository to determine whether other copies of this log sequence exist, and backs up the log in either `/disk2/arch` or `/disk3/arch`. Hence, because a copy of each log in sequence 288 through 388 is located in at least one of the three directories, RMAN can back up all the specified logs.

Backing Up Archived Logs Needed to Recover an Online Backup: Example

Assume that you back up database `trgt` while it is open. You want to back up only those archived redo logs required to recover this online backup. How do you determine which logs to back up?

To determine the archived logs needed for recovery of an online backup:

1. Start SQL*Plus and archive all unarchived logs, including the current log:

   ```
   ALTER SYSTEM ARCHIVE LOG CURRENT;
   ```

2. Query `V$LOG` to determine the log sequence number of the current redo log, as in the following example (which includes output):

   ```
   SELECT SEQUENCE#
   FROM V$LOG
   WHERE STATUS = 'CURRENT';

    SEQUENCE#
   ----------
         9100
   ```

3. Start RMAN and make an online backup of the database. For example, enter:

   ```
   BACKUP DATABASE;
   ```

4. Archive all unarchived logs, including the current log:

   ```
   SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
   ```

5. In SQL*Plus, query `V$LOG` to determine the log sequence number of the current redo log:

   ```
   SELECT SEQUENCE#
   FROM V$LOG
   WHERE STATUS = 'CURRENT';

    SEQUENCE#
   ----------
         9112
   ```

6. Back up the logs beginning with the first sequence number that you queried, and ending with the last sequence number minus 1. The log before the current log is the most recent archived log. For example, if the first query returned 9100, then start at 9100. If the second query returned 9112, then end at 9111.

   For example, issue the following to back up the necessary archived logs:

   ```
   BACKUP ARCHIVELOG FROM SEQUENCE 9100 UNTIL SEQUENCE 9111;
   ```

Backing Up and Deleting Multiple Copies of an Archived Redo Log: Example

In this scenario, you set initialization parameters so that you automatically archive redo logs to two directories: `?/oradata/trgt/arch/dest_1` and

?/oradata/trgt/arch/dest_2. Therefore, you have two identical copies of the archived redo log for each log sequence number. You decide to back up each copy of the archived redo logs and then delete the originals.

The easiest solution in this case is to use the DELETE ALL INPUT option means that RMAN deletes all logs that match the ARCHIVELOG criteria. Hence, it can remove all logs from both ?/oradata/trgt/arch/dest_1 and ?/oradata/trgt/arch/dest_2.

For example, run the following command to back up all logs that could be used to recover from a point 10 days ago, and then delete all logs within the specified time range from disk:

```
BACKUP DEVICE TYPE sbt
  ARCHIVELOG ALL FROM TIME 'SYSDATE-10'
  DELETE ALL INPUT;
```

Performing Differential Incremental Backups: Example

A differential incremental backup contains only blocks that have been changed since the most recent backup at the same level or lower. The first incremental backup must be a level 0 backup that contains all used blocks. The following is a level 0 base backup:

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

An incremental backup at level 1 or higher will contain all blocks changed since the most recent level 1 backup. If no previous level 1 backup is available, then RMAN copies all blocks changed since the base level 0 backup. The following is a level 1 backup of the database:

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

If you add a new datafile or tablespace to the database, then make a level 0 backup before making another incremental backup. Otherwise, the incremental backup of the tablespace or the database fails because RMAN does not find a parent backup for the new datafiles. The following is a level 0 backup of a single tablespace:

```
BACKUP INCREMENTAL LEVEL 0 TABLESPACE users2;
```

Note that you can perform incremental backups in NOARCHIVELOG mode, but the backups must be consistent. Hence, you cannot take online incremental backups.

Performing Cumulative Incremental Backups: Example

A cumulative incremental backup at level *n* contains only blocks that have been changed since the most recent backup at level *n* - 1 or lower. Cumulative backups require more storage space than differential backups, but they are preferable during a restore operation because only one backup for a given level is needed. Note that the first incremental backup must be a level 0 backup that contains all used blocks.

A cumulative backup at level 2 will contain all blocks changed since the most recent level 1 backup, copying all blocks changed since the base level 0 backup only if a previous level 1 is unavailable. In contrast to a cumulative backup, a differential backup at level 2 will determine which level 1 *or* level 2 backup occurred most recently and copy all blocks changed since that backup.

```
BACKUP INCREMENTAL LEVEL 2 CUMULATIVE DATABASE; # blocks changed since level 0 or level 1
```

Determining How Channels Distribute a Backup Workload: Example

When you create multiple backup sets and allocate multiple channels, RMAN automatically writes multiple backup sets in parallel. The allocated server sessions share the work of backing up the specified datafiles, control files, and archived redo logs. Note that you cannot stripe a single backup set across multiple channels.

RMAN automatically assigns a backup set to a device. You can use the `CHANNEL` parameter so that RMAN writes all backup sets for a *`backupSpec`* to a specific channel.

For example, this example parallelizes the backup operation by specifying which channels RMAN should back up to disk and which to `sbt`:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE DISK FORMAT = '/backup/df/%U';
  ALLOCATE CHANNEL ch2 DEVICE TYPE DISK FORMAT = '/backup/cf/%U';
  ALLOCATE CHANNEL ch3 DEVICE TYPE sbt;
  BACKUP
    # channel ch1 backs up datafiles to /backup/df directory
    DATAFILE 1,2,3,4
    CHANNEL ch1
    # channel ch2 backs up control file copy to /backup/cf directory
    CONTROLFILECOPY '/tmp/control01.ctl'
    CHANNEL ch2
    # channel ch3 backs up archived redo logs to tape
    ARCHIVELOG FROM TIME 'SYSDATE-14'
    CHANNEL ch3;
}
```

Note that you cannot back up to `DISK` and `sbt` at the same time using automatic channels: you must manually allocate them.

Backing Up in NOARCHIVELOG Mode: Example

This script puts the database into the correct mode for a consistent, whole database backup and then backs up the database. Note that the script performs a shutdown, startup, shutdown, and then startup again before creating multiple copies of the backup:

```
# Shut down the database cleanly using immediate priority. This type of shutdown lets
# current calls to the database complete, but prevents further logons or calls.
# If the database is not up now, you will get a message saying so but RMAN will not
# treat this situation as an error.
SHUTDOWN IMMEDIATE;

# Start up the database in case it suffered instance failure or was closed with SHUTDOWN
# ABORT before starting this script. The scripts performs crash recovery if it is needed.
# Oracle uses the default init.ora file. Alternatively, use this form: STARTUP FORCE DBA
# pfile=filename. Use the DBA option because you are going to shut down again right
# away and do not want to let users in during the short interval. Use the FORCE
# option because it cannot hurt and might help in certain situations.
STARTUP FORCE DBA;
SHUTDOWN IMMEDIATE;

# The database is cleanly closed and is now ready for a consistent backup. RMAN requires
# that the database be started and mounted to perform a backup.
STARTUP MOUNT;

# this example uses automatic channels to make the backup
BACKUP COPIES 2 INCREMENTAL LEVEL 0 FILESPERSET 5 DATABASE;

# Now that the backup is complete, open the database.
ALTER DATABASE OPEN;
```

Note that you can skip tablespaces, but any skipped tablespace that has not been offline or read-only since its last backup will be lost if the database has to be restored from a backup. When backing up to disk, make sure that the destination (file system or raw device) has enough free space.

Backing Up in an Oracle Real Application Clusters Environment: Example

Assume that `/node1` is a local directory accessible by node 1 of an Oracle Real Application Clusters configuration and `/node2` is a local directory accessible by node 2. The automatic channel configuration is as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
# the different CONNECT strings in the channels cause RMAN to enable
# the autlocation feature
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'SYS/oracle@node_1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'SYS/oracle@node_2';
```

Because in this scenario node 1 is more powerful than node 2, you want node 1 to back up the largest tablespaces. The following script distributes datafile and archived redo log backups across the two nodes:

```
BACKUP FILESPERSET 1
  (TABLESPACE system, tools, users, undotbs
   CHANNEL ORA_SBT_TAPE_1)
  (TABLESPACE cwmlite, drsys, example, indx
   CHANNEL ORA_SBT_TAPE_2);
BACKUP FILESPERSET 20
  ARCHIVELOG ALL;
```

Because of the autolocation feature, the channel connected to node 1 backs up only the archived logs readable on node 1, and the channel connected to node 2 backs up only the archived logs readable on node2.

> **See Also:** *Oracle9i Real Application Clusters Administration* for information about Oracle Real Application Clusters backups

Cataloging Operating System Copies: Example

You can use operating system utilities to make datafile copies and then catalog them in the recovery catalog. Note that you can only catalog disk copies. Because the format of backup pieces is proprietary, operating system utilities cannot write backups readable by RMAN.

You must make the datafile copies by means of user-managed methods. If the database is open and the datafile is online, then issue `ALTER TABLESPACE ... BEGIN BACKUP`. For example, the resulting image copy can be cataloged:

```
CATALOG DATAFILECOPY '/tmp/users01.dbf';
```

Note that 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 08/29/2001 14:44:34
ORA-19563: datafile copy header validation failed for file /tmp/tools01.dbf
```

Keeping a Long-Term Backup: Example

If you configure a retention policy, then you may want to exclude specified backups from this policy. For example, you may want to archive a consistent backup of the database once a year to serve as a historical record. This long-term backups does not function as a backup that you may perform recovery on, but an archived snapshot of data at a particular time.

To exempt a backup from the retention policy, specify the `KEEP` option on the `BACKUP` command. You can also specify `LOGS` or `NOLOGS` to indicate whether RMAN should save archived logs for possible recovery of this backup. If you specify `NOLOGS`, then the backup must be consistent.

This example keeps the backup of the database indefinitely and does not save archived logs needed to recover it:

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;  # put database in consistent state
BACKUP DATABASE KEEP FOREVER NOLOGS TAG 'db_archive_1'; # make long-term consistent backup

# mark backup as unavailable in the repository so that RMAN does not attempt to restore it
# unless explicitly specified on the RESTORE command
CHANGE BACKUP TAG 'db_archive_1' UNAVAILABLE;
SQL 'ALTER DATABASE OPEN';
```

Optimizing Backups: Examples

Run the `CONFIGURE BACKUP OPTIMIZATION` command to enable backup optimization. When specific conditions are met (described in "Backup Optimization Algorithm" on page 5-56), RMAN skips backups of files that are identical to files that are already backed up.

Assume that you configure optimization and a retention policy as follows:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 4 DAYS;
```

Optimizing a Database Backup: Example

Then, you run this 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 the read-only datafile `?/oradata/trgt/history01.dbf` every night, so long as one backup set containing this file exists within the recovery window.

For example, if the most recent backup of the datafiles was on Sunday, and the point of recoverability (that is, the earliest date in the recovery window) is on Saturday, then RMAN skips the datafiles when you run the Wednesday backup. On Friday, the point of recoverability is now Monday, so the Sunday backup is now outside the window. Hence, the Friday backup does not skip the datafiles.

Optimizing a Daily Archived Log Backup to a Single Tape: Example

Assume that you want to back up all the archived logs every night. However, you do not want to have multiple copies of each log sequence number. So, you configure backup optimization to `ON`, then run this command in a script every night 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 Tapes: Example

In this example, you back up logs that are not already on tape to one tape pool, then back up the same logs to a second tape pool. Finally, you delete old logs.

For the first step, perform the one-time configuration:

```
# configure backup optimization
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

Then, run the following script at the same time every night to back up the logs generated during the previous day to two separate tape pools:

```
# The following command will back up just the archived logs that are not on tape. The
# first copies are saved to the tapes from the pool "archivelog_pool_1"
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS='NSR_DATA_VOLUME_POOL=ARCHIVELOG_POOL_1';
  BACKUP CHANNEL 'ORA_SBT_TAPE_1' ARCHIVELOG ALL;
}
# Make one more copy of the archived logs and save them to tapes from a different pool
RUN
{
  ALLOCATE CHANNEL c2 DEVICE TYPE sbt
    PARMS='NSR_DATA_VOLUME_POOL=ARCHIVELOG_POOL_2';
  BACKUP CHANNEL 'ORA_SBT_TAPE_2' ARCHIVELOG
    FROM TIME 'SYSDATE-1'
    UNTIL TIME 'SYSDATE';  # specify UNTIL so that RMAN does not archive current log
}
# Delete old logs - for example, delete logs created within the last week.
DELETE ARCHIVELOG ALL COMPLETED AFTER 'SYSDATE-7';
```

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. However, you are worried about tape failure, 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.

First, perform a one-time configuration:

```
# configure backup optimization
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFUALT DEVICE TYPE TO sbt;
# configure a default channel that sends the backups to the tape pool called "first_copy"
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='ENV=(NSR_DATA_VOLUME_POOL=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" pool that have not already been backed up:

```
BACKUP ARCHIVELOG ALL;
```

Every Friday evening you create an additional backup of all archived logs in a different tape pool. Also, 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:

```
BACKUP ARCHIVELOG ALL; # backs up logs not already on tape to pool "first_copy"

RUN
{
  # manually allocate a channel so that you bypass backup optimization for this job.
  # specify that the backup run by this channel should go to the pool "second_copy."
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
      PARMS='ENV=(NSR_DATA_VOLUME_POOL=second_copy)';
  BACKUP ARCHIVELOG UNTIL TIME 'SYSDATE' # use UNTIL clause so RMAN does not switch logs
    NOT BACKED UP 2 TIMES                # back up only logs without 2 backups on tape
    TAG SECOND_COPY;                     # specify TAG for convenience
}

# 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 Friday script creates a second copy of all archived logs in the "second_copy" tape pool. After the backup, you can send the tape from the pool "second_copy" to the vault or lock it in your desk. You should use this tape backup only if the primary tape from pool "first_copy" is damaged. Because the secondary tape is in a secure place, 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;
```

Handling Errors During Backups and Copies: Example

By default a checksum is calculated for every block read from a datafile and stored in the backup or image copy. If you use the NOCHECKSUM option, then checksums are not calculated. If the block already contains a checksum, however, then the checksum is validated and stored in the backup. If the validation fails, then the block is marked corrupt in the backup.

The SET MAXCORRUPT FOR DATAFILE command sets how many corrupt blocks in a datafile that BACKUP or COPY will tolerate. If a datafile has more corrupt blocks than specified by the MAXCORRUPT parameter, the command terminates. If you specify the CHECK LOGICAL option, RMAN checks for logical and physical corruption.

By default, the BACKUP command terminates when it cannot access a datafile. You can specify parameters to prevent termination, as listed in the following table.

If you specify the option ...	Then RMAN skips...
SKIP INACCESSIBLE	Inaccessible datafiles. A datafile is only considered inaccessible if it cannot be read. 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 OFFLINE	Offline datafiles.
SKIP READONLY	Datafiles in read-only tablespaces.

The following example uses an automatic channel to back up the database, and sets the corruption level for the datafile in the SYSTEM tablespace:

```
RUN
{
  SET MAXCORRUPT FOR DATAFILE 1 TO 0;
  BACKUP DATABASE
```

```
    SKIP INACCESSIBLE
    SKIP READONLY
    SKIP OFFLINE;
}
```

10 Restoring and Recovering with Recovery Manager

This chapter describes how to use Recovery Manager to perform restore and recovery operations. This chapter contains these topics:

See Also: "Monitoring RMAN Job Progress" on page 15-19 to learn how to monitor restore and recovery operations

Restoring and Recovering with RMAN: Overview

Typically, you restore and recover a database or subset of a database in the following cases:

- A media failure has damaged some or all control files or datafiles.
- You want to recover the database to a point before a user error, such as a dropped table, occurred.

If you want to restore a version of the database for testing purposes, then run the `DUPLICATE` rather than the `RESTORE` command.

> **See Also:** Chapter 12, "Duplicating a Database with Recovery Manager" to learn how to duplicate a database

Generic Procedure for Media Recovery

The basic procedure for performing restore and recovery with RMAN is as follows:

1. Determine which database files require recovery.
2. Place the database in the state appropriate for the type of recovery that you are performing. For example, if you are recovering all datafiles, then mount the database. If you are recovering a single tablespace or datafile, then you can keep the database open and take the tablespace or datafile offline.
3. Restore the necessary files using the `RESTORE` command.
4. Recover the restored files using the `RECOVER` command.
5. Place the database in its normal state. For example, open the database if it is closed, or bring all recovered files online if they are offline.

Because so many possible restore and recover scenarios exist, the actual recovery procedure that you should follow differs from case to case.

Note that if you use Oracle Enterprise Manager, then you can use the Recovery wizard instead of running the `RESTORE` and `RECOVER` commands through the RMAN command-line interface. You can perform the following RMAN restore and recovery tasks through the Recovery wizard:

- **restore and recover the entire database** to the current or a noncurrent time
- **restore and recover tablespaces or datafiles** to the latest time or from the latest backup or an older backup

- **restore the control file** from a controlfile autobackup when the database is in the NOMOUNT state and there are no backup configurations which use a recovery catalog
- **restore archived logs** by time, by SCN, or by log sequence.
- **recover datablocks** by using a corruption list, datafiles, or tablespaces

See Also: *Oracle Enterprise Manager Administrator's Guide* to learn about RMAN restore and recovery

Differences Among Restore and Recovery Scenarios

Before performing recovery, identify the conditions under which you will perform the recovery. The recovery procedure differs depending on whether:

- The current control file is available
- You are using a recovery catalog
- The restore host is the same as the original target host
- The restored database files are named the same as the target database files
- The target database runs in an Oracle Real Application Clusters configuration
- You want to recover to the current time or to a noncurrent time
- You are recovering the whole database
- You have a backup made after the most recent `RESETLOGS`
- You are recovering whole datafiles rather than a limited number of corrupt data blocks

The section "Performing Basic RMAN Media Recovery" on page 10-5 describes a typical recovery scenario. This scenario is typical in that sense that is serves as the generic template for media recovery. Obviously, this generic template can cover only some of the possible restore scenarios. The sections in this chapter other than "Performing Basic RMAN Media Recovery" on page 10-5 describe variations. Use Table 10–1 to determine which sections you should refer to when your recovery scenario differs from the generic procedure.

Table 10–1 Differences Among Recovery Scenarios

Question	If yes, then see ...	If no, then see ...
Is the current control file available?	"Performing Basic RMAN Media Recovery" on page 10-5	"Performing Recovery with a Backup Control File" on page 10-13
Is the current server parameter file available?	"Performing Basic RMAN Media Recovery" on page 10-5	"Restoring the Server Parameter File" on page 10-13
Are you using a recovery catalog?	"Performing Basic RMAN Media Recovery" on page 10-5	"Performing Basic RMAN Media Recovery" on page 10-5. If using a backup control file and no catalog, refer to "Performing Recovery with a Backup Control File and No Recovery Catalog" on page 10-16.
Is the restore host the same as the target host?	"Performing Basic RMAN Media Recovery" on page 10-5	"Restoring the Database to a New Host" on page 10-23
Will the restored database files have the same name as the original database files?	"Performing Basic RMAN Media Recovery" on page 10-5	"Restoring Files to a New Location" on page 10-19
Is the target database in an Oracle Real Application Cluster?	"Recovering an Oracle Real Application Clusters Database" on page 10-25	"Performing Basic RMAN Media Recovery" on page 10-5
Are you performing complete recovery?	"Performing Complete Restore and Recovery" on page 10-8	"Performing Incomplete Restore and Recovery" on page 10-10 to recover the whole database, or Chapter 11, "Performing RMAN Tablespace Point-in-Time Recovery" for point-in-time recovery of an individual tablespace

Table 10–1 Differences Among Recovery Scenarios

Question	If yes, then see ...	If no, then see ...
Are you recovering the whole database?	"Restoring and Recovering the Whole Database in the Default Location" on page 10-8	"Restoring and Recovering a Subset of the Database" on page 10-9 if you are performing complete recovery of a database subset, or Chapter 11, "Performing RMAN Tablespace Point-in-Time Recovery" for point-in-time recovery of a database subset
Were the necessary backups performed after the most recent database `RESETLOGS`?	"Performing Basic RMAN Media Recovery" on page 10-5	"Recovering Through a RESETLOGS Operation with RMAN" on page 10-26
Do you need to recover whole datafiles rather than a few corrupt blocks?	"Performing Basic RMAN Media Recovery" on page 10-5	"Recovering Datablocks with RMAN" on page 10-38

Performing Basic RMAN Media Recovery

This section contains these topics:

- About Basic Media Recovery
- Preparing for Media Recovery
- Performing Complete Restore and Recovery
- Performing Incomplete Restore and Recovery

About Basic Media Recovery

All the procedures in this section assume the following:

- The current control file is available, and you are using either a recovery catalog or the current control file as the RMAN repository.
- The restore node is the same as the target node, and the restored datafiles will have the same filenames as the original target datafiles.
- The target database is not running in an Oracle Real Application Clusters configuration.
- You are performing either complete or incomplete recovery of whole datafiles (not individual data blocks). If performing incomplete recovery, then you are

recovering the whole database. If performing complete recovery, then you are recovering either the whole database or a database subset.

Use the RESTORE and RECOVER commands to perform the recovery. The RESTORE command restores backups from disk or a media manager, but restores image copies only from disk.

Note: In Oracle9*i*, unlike in previous RMAN releases, the RESTORE command is optimized: it only restores files if the datafile header does not contain the expected information. In other words, if a file does not need to be restored, then RMAN does not restore it. You can override this behavior by specifying the FORCE option.

If you have automatic channels configured, then RMAN allocates all channels configured for the available device types according to their parallelism settings. For example, if you configure two sbt channels and set parallelism to 1, and if you set parallelism for DISK channels to 3, then RMAN automatically allocates one sbt channel and three DISK channels. For a restore, RMAN allocates all configured channels unless the DEVICE TYPE option restricts the device type from which RMAN restores.

If you are manually allocating channels, then allocate the appropriate DISK or sbt channel when restoring files. If the appropriate device type is not allocated, then RMAN may not be able to find a backup set or copy to restore, and the RESTORE command will fail.

Preparing for Media Recovery

When and how to recover depends on the state of the database and the location of its datafiles. If possible, query fixed views to obtain the needed information.

To determine whether media recovery is necessary:

1. Start SQL*Plus and connect to the target database. For example, issue the following to connect to trgt:

   ```
   % sqlplus 'SYS/oracle@trgt AS SYSDBA'
   ```

2. Determine the status of the database by executing the following SQL query:

   ```
   SELECT STATUS FROM V$INSTANCE;

   STATUS
   -------
   ```

```
OPEN
```

If the status is OPEN, then the database is open, but it is still possible that you need to restore or recover some tablespaces and their datafiles.

3. Check the recovery and error columns of the V$DATAFILE_HEADER view. These columns indicate the status of datafiles. Execute the following SQL script to check the datafile headers and respond according to the table that follows this example:

```
COL FILE# FORMAT 999
COL STATUS FORMAT A7
COL ERROR FORMAT A10
COL TABLESPACE_NAME FORMAT A10
COL NAME FORMAT A30

SELECT FILE#, STATUS, ERROR, RECOVER, TABLESPACE_NAME, NAME
FROM V$DATAFILE_HEADER
/
```

ERROR column	RECOVER column	Solution
NULL	NO	Unless the error is caused by a temporary hardware or operating system problem, restore the datafile or switch to a copy of that datafile.
NULL	YES	Recover the datafile. The RECOVER command first applies any suitable incremental backups and then applies redo logs. RMAN restores incremental backups and archived redo logs as needed.
not NULL		Unless the error is caused by a temporary hardware or operating system problem, restore the datafile or switch to a copy of that datafile.

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, Oracle reports no error if the datafile contains corrupt data blocks but its header block is intact.

4. Optionally, you can also query V$DATAFILE and V$TABLESPACE to obtain the tablespace names for the datafiles requiring recovery as well as status and error information. For example, run the following SQL*Plus script:

```
COL DF# FORMAT 999
COL DF_NAME FORMAT A30
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#
/
```

This script produces output similar to the following:

```
 DF# DF_NAME                         TBSP_NA STATUS  ERROR       CHANGE#       TIME
---- ------------------------------- ------- ------- ---------- --------- ----------
   7 /oracle/oradata/trgt/tools01.dbf  TOOLS OFFLINE    OFFLINE         0
                                                        NORMAL
```

If media recovery is required, then the ERROR column indicates the problem that makes recovery necessary.

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

Performing Complete Restore and Recovery

After determining which datafiles require recovery, you can either restore all datafiles in the database or only a subset of datafiles.

This section contains these topics:

- Restoring and Recovering the Whole Database in the Default Location
- Restoring and Recovering a Subset of the Database

Restoring and Recovering the Whole Database in the Default Location

In this scenario, you have a current control file but all datafiles are damaged or lost. You must restore and recover the whole database.

To restore and recover the database when the current control file is available:

1. After connecting to the target database and, optionally, the recovery catalog database, make sure the database is mounted.

   ```
   STARTUP MOUNT
   ```

2. Do the following:

   a. If automatic channels are *not* configured, then manually allocate one or more channels. Run `SHOW ALL` to see the current configuration.

   b. Restore the database using the `RESTORE` command, and recover it using the `RECOVER` command.

   This example performs recovery using automatic channels and skips the read-only `history` tablespace:

```
RESTORE DATABASE;
RECOVER DATABASE
  # optionally, delete logs restored for recovery and limit disk space used
  DELETE ARCHIVELOG MAXSIZE 1M
  SKIP TABLESPACE history;  # optionally, skip the recovery of some tablespaces
```

3. Examine the output to see if recovery was successful. If so, open the database:

```
ALTER DATABASE OPEN;
```

Restoring and Recovering a Subset of the Database

In this scenario, some but not all of the datafiles are damaged. You have already determined which tablespaces are affected by following the procedure in "Preparing for Media Recovery" on page 10-6.

The following procedure assumes that the database is open, so you must take the tablespaces to be recovered offline.

To recover a tablespace to the default location:

1. After connecting to the target database and, optionally, the recovery catalog database, make sure the database is mounted or open. For example, run:

```
STARTUP MOUNT
```

2. Restore and recover the affected datafiles. Do the following:

   a. If automatic channels are *not* configured, then manually allocate one or more channels. Run `SHOW ALL` to see the current configuration.

   b. If the tablespaces needing recovery are not already offline, then take them offline using `ALTER TABLESPACE ... OFFLINE IMMEDIATE`.

   c. Restore the tablespace or datafile with the `RESTORE` command, and recover it with the `RECOVER` command.

   This example restores and recovers the `users` tablespace:

```
SQL 'ALTER TABLESPACE users OFFLINE IMMEDIATE';
RESTORE TABLESPACE users;
```

```
RECOVER TABLESPACE users;
```

3. If the recovery was successful, then bring the tablespace online:

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

Performing Incomplete Restore and Recovery

This section contains these topics:

- About Incomplete Recovery
- Performing Incomplete Recovery with a Current Control File

About Incomplete Recovery

RMAN can perform recovery of the whole database to a specified noncurrent time, SCN, or log sequence number. This type of recovery is called **incomplete recovery** because it does not completely use all of the available redo. Incomplete recovery of the whole database is also called **database point-in-time recovery (DBPITR)**.

Incomplete recovery of the database requires you to open the database with the RESETLOGS option. Using this option gives the online redo logs a new time stamp and SCN, thereby eliminating the possibility of corrupting datafiles by the application of obsolete archived redo logs. Note that you have to recover *all* datafiles: you cannot recover some datafiles before the RESETLOGS and others after the RESETLOGS. In fact, Oracle prevents you from resetting the logs if a datafile is offline. The only exception is if the datafile is offline normal or read-only. You can bring files in read-only or offline normal tablespaces online after the RESETLOGS because they do not need any redo applied to them.

The easiest way to perform DBPITR is to use the SET UNTIL command (rather than specifying the UNTIL clause on the RESTORE and RECOVER commands individually) because it sets the desired time for any subsequent RESTORE, SWITCH, and RECOVER commands in the same RUN job. Note that if you specify a SET UNTIL command after a RESTORE and before a RECOVER, you may not be able to recover the database to the point in time required because the restored files may already have time stamps more recent than the set time. Hence, it is recommended that you specify the SET UNTIL command *before* the RESTORE command.

Performing Incomplete Recovery with a Current Control File

The database must be closed to perform database point-in-time recovery. Note that if you are recovering to a time, you should set the time format environment variables before invoking RMAN (refer to "Setting Globalization Support

Environment Variables for RMAN" on page 8-27). The following are sample Globalization Support settings:

```
NLS_LANG = american_america.us7ascii
NLS_DATE_FORMAT="Mon DD YYYY HH24:MI:SS"
```

To recover the database until a specified time, SCN, or log sequence number:

1. After connecting to the target database and, optionally, the recovery catalog database, ensure that the database is mounted. If the database is open, shut it down and then mount it:

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

2. Determine the time, SCN, or log sequence that should end recovery. 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 will lose all changes to the database made after that time.

   You can also examine the `alert.log` to find the SCN of an event and recover to a prior SCN. Alternatively, you can determine the log sequence number that contains the recovery termination SCN, and then recover through that log. For example, query `V$LOG_HISTORY` to view the logs that you have archived.

   ```
   RECID      STAMP      THREAD#    SEQUENCE#  FIRST_CHAN FIRST_TIM NEXT_CHANG
   ---------- ---------- ---------- ---------- ---------- --------- ----------
            1  344890611          1          1      20037 24-SEP-01      20043
            2  344890615          1          2      20043 24-SEP-01      20045
            3  344890618          1          3      20045 24-SEP-01      20046
   ```

3. Perform the following operations within a `RUN` command:

   a. Set the end recovery time, SCN, or log sequence. 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 one or more channels.

   c. Restore and recover the database.

   The following example performs an incomplete recovery until November 15 at 9 a.m.

   ```
   RUN
   {
     SET UNTIL TIME 'Nov 15 2001 09:00:00';
     # SET UNTIL SCN 1000;       # alternatively, you can specify SCN
   ```

```
  # SET UNTIL SEQUENCE 9923;  # alternatively, you can specify log sequence number
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

4. If recovery was successful, then open the database and reset the online logs:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

5. It is recommended that you immediately back up the database, preferably with the database mounted (to avoid possible data loss in an open database). Because the database is a new incarnation, the backups made before the `RESETLOGS` are not easily usable. For example, run the following to back up the database:

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

Restoring the Server Parameter File

RMAN can restore the server parameter file either to the default location or to a nondefault location. Also, RMAN can restore the server parameter file as a server parameter file or as a client-side initialization parameter file.

Note the following restrictions and usage notes 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 Solaris).

To restore the server parameter file:

1. Connect to the target database and, optionally, the recovery catalog database. For example, run:

   ```
   % rman TARGET / CATALOG rman/cat@catdb
   ```

2. If you are connected to a catalog, and if the target database `DB_NAME` of the target database is unique in the catalog, then skip this step. Otherwise, set the DBID of the target database. For example:

```
SET DBID 676549873;
```

3. Shut down the instance and restart it without mounting. Because the server parameter file is lost, RMAN starts the instance with a dummy parameter file. For example:

   ```
   STARTUP FORCE NOMOUNT;
   ```

4. Restore the server parameter file. If restoring to the default location, then simply run:

   ```
   RESTORE SPFILE; # if you are using a catalog
   RESTORE SPFILE FROM AUTOBACKUP; # if in NOCATALOG mode
   ```

   If restoring to a nondefault location, then you could run commands as in the following example:

   ```
   RESTORE SPFILE TO '/tmp/spfileTEMP.ora'; # if you are using a catalog
   RESTORE SPFILE TO '/tmp/spfileTEMP.ora' FROM AUTOBACKUP; # if in NOCATALOG mode
   ```

   You can restore the server parameter file as a client-side initialization parameter file with the `TO PFILE` *`'filename'`* clause as in the following example:

   ```
   RESTORE SPFILE TO PFILE '/tmp/initTEMP.ora';
   ```

5. Restart the instance with the restored file. If restarting with a server parameter file in a nondefault location, then create a new client-side initialization parameter file with the single 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:

   ```
   HOST 'echo "SPFILE=/tmp/spfileTEMP.ora" > /tmp/init.ora';
   STARTUP FORCE PFILE=/tmp/init.ora; # starts instance with /tmp/spfileTEMP.ora
   ```

   If you restored the server parameter file as a client-side initialization parameter file, then simply specify the path name of this restored file. For example:

   ```
   STARTUP FORCE PFILE=/tmp/pfileTEMP.ora; # starts instance with /tmp/pfileTEMP.ora
   ```

Performing 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 before you can perform recovery. The procedure differs depending on whether you use a catalog. Note these usage notes and restrictions that apply to both cases:

- You must run the RECOVER command after restoring a backup control file, even if no datafiles have been restored.
- After restoring a backup control file, entries for tempfiles in locally-managed temporary tablespaces are removed. Hence, you must add new tempfiles to these tablespaces after you open with the RESETLOGS option. If you do not, then Oracle can display the following error for when attempting to sort: `ORA-25153: Temporary Tablespace is Empty`.
- You must open the database with the RESETLOGS option after performing either complete *or* incomplete recovery with a backup control file.
- If the online redo logs are inaccessible, then you must perform incomplete recovery to an SCN before the earliest SCN in the online redo logs. This limitation is necessary because RMAN does not back up online logs.
- Starting with Oracle9*i*, RMAN automatically searches in specific locations for online and archived redo logs during recovery that are not recorded in the RMAN repository, and catalogs any that it finds. RMAN attempts to find a valid archived log in any of the current archiving destinations with the current log format. The current format is specified in the initialization parameter file used to start the instance (or all instances in a Real Application Clusters). Similarly, RMAN attempts to find the online redo logs by using the filenames as specified in the control file.

  If RMAN is not able to automatically catalog a needed online or archived log, which can happen 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 reports errors similar to the following:

  ```
  RMAN-00571: ===========================================================
  RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
  RMAN-00571: ===========================================================
  RMAN-03002: failure of recover command at 08/29/2001 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 logs to the repository so that recovery can proceed. The cataloging procedure is described in "Cataloging Archived Logs and User-Managed Copies" on page 18-15.

This section contains these topics:

- Performing Recovery with a Backup Control File and a Recovery Catalog
- Performing Recovery with a Backup Control File and No Recovery Catalog

Performing Recovery with a Backup Control File and a Recovery Catalog

If you use a recovery catalog and have a backup control file available, this procedure does not differ substantially from "Performing Complete Restore and Recovery" on page 10-8. The procedure in this section assumes that you are restoring the control file to its default location. If you must restore the control file to a new location, then refer to "Restoring Control Files to a New Location" on page 10-20 for instructions.

When you perform a restore operation using a backup control file and you use a catalog, RMAN automatically adjusts the control file to reflect the structure of the restored backup.

The following procedure assumes that you do *not* have more than one target database registered in the catalog with the same name. If multiple target databases *are* registered with the same name, then you must specify the DBID with the `SET DBID` command so that RMAN knows which control file to restore. The DBID is the unique numerical identifier for a database.

> **See Also:** "Performing Recovery with a Backup Control File and No Recovery Catalog" on page 10-16 to learn how to set the DBID

To recover the database with a backup control file and a recovery catalog:

1. After connecting to the target database and recovery catalog database, start the instance without mounting the database:

   ```
   STARTUP NOMOUNT
   ```

2. Restore the backup control file, then restore and recover the database. Do the following:

   a. Run the `RESTORE CONTROLFILE` command to restore the control file to all default locations specified in the `CONTROL_FILES` initialization parameter. To restore a control file from an older backup, you can run `SET UNTIL` or specify the `UNTIL` clause on the `RESTORE CONTROLFILE` command.

   b. Mount the database using the restored control file.

   c. Optionally, run a `SET UNTIL` command for incomplete recovery. Note that you can also specify the `UNTIL` clause on the `RESTORE` and `RECOVER` commands.

   d. Restore and recover the database as described in "Performing Basic RMAN Media Recovery" on page 10-5.

This example restores the control file to its default location, then restores and completely recovers the database:

```
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
RESTORE DATABASE;
RECOVER DATABASE;
```

3. If recovery was successful, then open the database and reset the online logs:

```
ALTER DATABASE OPEN RESETLOGS;
```

4. If the database uses locally-managed temporary tablespaces, then add new tempfiles to these tablespaces. For example:

```
SQL "ALTER TABLESPACE temp ADD TEMPFILE ''?/oradata/trgt/temp01.dbf'' REUSE";
```

5. It is recommended that you immediately back up the database, preferably with the database mounted (to avoid possible data loss in an open database). Because the database is a new incarnation, the backups made prior to the `RESETLOGS` are not easily usable. For example, run the following to back up the database:

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

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. Assuming that you enabled the control file autobackup feature for the target database, you can restore an autobackup of the control file. Because the autobackup uses a default 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 (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 server records the location of every autobackup in the alert log.

Because you are not connected to a recovery catalog, the control file must have a record of all needed backups. If any backups are not listed in the control file, then RMAN cannot restore them. If datafile copies are located on disk but are not in the control file, however, then you can add them to the control file repository with the `CATALOG` command. This cataloging procedure is described in "Cataloging Archived Logs and User-Managed Copies" on page 18-15.

Because the repository is not available when you restore the control file, run the `SET DBID` command to identify the target database. You should only run the `SET DBID` command in the following specialized circumstances:

- You are not connected to a recovery catalog and want to restore the control file or server parameter file.
- You are connected to a recovery catalog want to restore the control file, but the database name is not unique in the recovery catalog.
- The server parameter file is lost and you want to restore it.

To recover the database with an autobackup of the control file without a recovery catalog:

1. Start RMAN and connect to the target database. For example, run:

   ```
   CONNECT TARGET /
   ```

2. Start the target 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 the target. You can also obtain it by running `LIST`, querying the catalog, or looking at the filenames of control file autobackup. (refer to "Restoring When Multiple Databases in the Catalog Share the Same Name: Example" on page 10-43). For example, run:

   ```
   SET DBID 676549873;
   ```

4. Restore the autobackup control file, then perform recovery. Do the following:

   a. Optionally, specify the most recent backup time stamp that RMAN can use when searching for a control file autobackup to restore.

   b. If a nondefault format was used to create the control file, then specify a nondefault format for the restore of the control file.

   c. If the channel that created the control file autobackup was device type `sbt`, then you must allocate one or more `sbt` channels. Because no repository is available, you cannot use automatic channels. If the autobackup was created on a disk channel, however, then you do not need to manually allocate a channel.

d. Restore the autobackup of the control file, optionally setting the maximum number of days backward that RMAN can search (up to 366) and the initial sequence number that it should use in its search for the first day.

e. Mount the database. Note that because the repository is now available, any automatic channels that you configured are also available.

f. If the online logs are inaccessible, then restore and recover the database as described in "Performing Incomplete Restore and Recovery" on page 10-10. You must terminate recovery by setting the `UNTIL` clause to a time, log sequence, or SCN before the online redo logs. If the online logs are usable, then restore and recover the database as described in "Performing Complete Restore and Recovery" on page 10-10.

In this example, the online redo logs have been lost. This example limits the restore of the control file autobackup, then performs recovery of the database to log sequence 13243, which is the most recent archived log:

```
RUN
{
  # Optionally, set upper limit for eligible time stamps of control file backups
  # SET UNTIL TIME '09/10/2000 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; # manually allocate one or more channels
  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;
}
# uses automatic channels configured in restored control file
RESTORE DATABASE UNTIL SEQUENCE 13243;
RECOVER DATABASE UNTIL SEQUENCE 13243; # recovers to most recent archived log
```

5. If recovery was successful, then open the database and reset the online logs:

```
ALTER DATABASE OPEN RESETLOGS;
```

6. It is recommended that you immediately back up the database, preferably with the database mounted (to avoid possible data loss in an open database). Because the database is a new incarnation, the backups made before the `RESETLOGS` are not easily usable. For example, enter:

```
SHUTDOWN IMMEDIATE
```

```
STARTUP MOUNT
BACKUP DATABASE;
ALTER DATABASE OPEN;
```

Restoring Files to a New Location

In a recovery scenario, you may be unable to restore some or all database files or archived logs to their original locations. For example, if your machine uses two disk drives and the second drive fails, then you may be forced to restore the datafiles from the second drive to the first.

This section contains these topics:

- Restoring Datafiles to a New Location
- Restoring Control Files to a New Location
- Restoring Archived Redo Logs to a New Location

Restoring Datafiles to a New Location

If you cannot restore datafiles to the default location, then follow the generic procedure in "Restoring and Recovering a Subset of the Database" on page 10-9, except run a `SET NEWNAME` command to rename each datafile before performing the restore. RMAN restores each datafile to its `NEWNAME` location rather than its original location.

After restoring the datafiles but before recovering them, run a `SWITCH` command to permanently change the filenames of the datafiles renamed by `SET NEWNAME` commands. The `SWITCH` command is equivalent to the SQL statement `ALTER DATABASE RENAME FILE`. If you run `SWITCH DATAFILE ALL`, then all datafiles for which a `SET NEWNAME` has been issued in this job are switched to their new name.

See Also: *Oracle9i Recovery Manager Reference* for `SWITCH` syntax

To restore a tablespace to a new location and then recover it:

1. After connecting to the target database and, optionally, the recovery catalog database, do the following:
   a. Take the tablespaces requiring recovery offline.
   b. Specify new filenames for the datafiles in the offline tablespaces.
   c. Restore the datafiles to the new location.

d. Point the control file to the restored datafiles.

This example restores the datafiles in tablespace `users` and `tools` to a new location, then performs recovery:

```
RUN
{
  SQL 'ALTER TABLESPACE users OFFLINE IMMEDIATE';
  SQL 'ALTER TABLESPACE tools OFFLINE IMMEDIATE';
  # restore the datafile to a new location
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/users01.dbf' TO '/tmp/users01.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/tools01.dbf' TO '/tmp/tools01.dbf';
  RESTORE TABLESPACE users, tools;
  SWITCH DATAFILE ALL;   # point control file to new filenames
  RECOVER TABLESPACE users, tools;
}
```

2. If recovery is successful, then bring the tablespaces online:

```
SQL 'ALTER TABLESPACE users ONLINE';
SQL 'ALTER TABLESPACE tools ONLINE';
```

Restoring Control Files to a New Location

If a media failure damages the control file and you do not have multiplexed copies, then you must restore a backup. Specify a new name with `RESTORE CONTROLFILE TO '`*`filename`*`'` when restoring a control file to a new location. The default location is the first location specified in the `CONTROL_FILES` initialization parameter. If the filename already exists, then Oracle overwrites the file.

After restoring the control file to a new location, run the `RESTORE CONTROLFILE FROM '`*`filename`*`'` command to copy it to all `CONTROL_FILES` destinations. The `RESTORE CONTROLFILE FROM '`*`filename`*`'` command is equivalent to running multiple `COPY CONTROLFILE` commands. Note that in case a media failure permanently damages some of the `CONTROL_FILES` locations, you can edit the server parameter file before starting the instance to specify new `CONTROL_FILES` locations.

To restore the control file to a new location:

1. After connecting to the target database and optionally the recovery catalog, start the database without mounting it:

```
STARTUP NOMOUNT
```

2. Restore and mount the control file. Do the following:

a. Optionally, run a SET UNTIL command to restore a control file created before a specified date.

b. Restore the backup control file to a temporary location.

c. Restore the control file from the restored location to all locations specified in the CONTROL_FILES parameter of the parameter file.

d. Mount the database using the newly restored control file.

```
RUN
{
  # To restore a control file created before a certain date, issue the following
  # SET command using a valid date for 'date_string'. You can also specify an SCN
  # or log sequence number.
  # SET UNTIL TIME = 'date_string';
  RESTORE CONTROLFILE TO '/tmp/control01.ctl'; # restore to new location
  # replicate the control file manually to CONTROL_FILES locations
  RESTORE CONTROLFILE FROM '/tmp/control01.ctl';
  STARTUP MOUNT;
}
```

3. Perform media recovery as described in "Performing Recovery with a Backup Control File" on page 10-13.

See Also: *Oracle9i Recovery Manager Reference* for RESTORE syntax

Restoring Archived Redo Logs to a New Location

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 combine in a port-specific fashion to derive the name of the restored archived log.

You can override the default names using 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:

1. After connecting to the target database and, optionally, the recovery catalog database, make sure the database is started, mounted, or open. For example, run:

```
STARTUP MOUNT
```

2. Perform the following operations within a `RUN` command:

   a. Specify the new location for the restored archived redo logs using `SET ARCHIVELOG DESTINATION`.

   b. Restore the archived redo logs.

   This example 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
  .
  .
  .
}
```

   You can also specify multiple restore destinations for archived logs, although you cannot specify these destinations simultaneously. For example, you can do the following:

```
RUN
{
  # Set a new location for logs 1 through 10.
  SET ARCHIVELOG DESTINATION TO '/tmp';
  RESTORE ARCHIVELOG FROM SEQUENCE 1 UNTIL SEQUENCE 10;
  # Set a new location for logs 11 through 20.
  SET ARCHIVELOG DESTINATION TO '?/oradata';
  RESTORE ARCHIVELOG FROM SEQUENCE 11 UNTIL SEQUENCE 20;
  # Set a new location for logs 21 through 30.
  SET ARCHIVELOG DESTINATION TO '?/dbs';
  RESTORE ARCHIVELOG FROM SEQUENCE 21 UNTIL SEQUENCE 30;
  # restore and recover datafiles as needed
  .
  .
  .
}
```

   Note that if you restore archived redo logs to multiple locations, then you only need to issue a single `RECOVER` command. RMAN finds the restored archived logs automatically and applies them to the datafiles.

Restoring the Database to a New Host

This section contains these topics:

- About Restoring a Database to a New Host
- Specifying Filenames When Restoring to a New Host

About Restoring a Database to a New Host

Various scenarios are possible when restoring a database to a new host. For example, you may want to:

- Create a duplicate version of the production database for testing or other purposes, while retaining the production database on the original host.
- Test the restore of the production database to a new host, while retaining the production database on the original host.
- Move the location of the production database to a new host (possibly because the original host has failed).

To create a duplicate database for testing while maintaining the original database, use the `DUPLICATE` command instead of the `RESTORE` command (refer to Chapter 12, "Duplicating a Database with Recovery Manager"). RMAN automatically creates a unique database identifier for the duplicate database. This chapter covers the use of the `RESTORE` command only.

To test the restore of a database to a new host or to move the database to a new host, run the `RESTORE` command. If you perform a test restore only, then you should do the following to prevent overwriting the target records in the recovery catalog:

- Run RMAN in the default `NOCATALOG` mode when restoring the datafiles.
- If you must use a recovery catalog because the control file is not large enough to contain all of the backups that you need to restore, then export the catalog and import it into a different schema or database and use the copied recovery catalog for the test restore. Otherwise, the catalog considers the restored database as the current target database.

Table 10–2 describes the impact on the RMAN repository when you are restoring or duplicating to a new host.

Table 10–2 Restoring and Duplicating to a New Host

Command	Catalog?	Affect on Repository
`RESTORE`	yes	If you run `SWITCH` commands after the restore, then RMAN considers the restored database as the target. If you do not run `SWITCH` commands, then RMAN views the restored datafiles as image copies that are candidates for future restore jobs.
`RESTORE`	no	If you run `SWITCH` commands after the restore, then RMAN considers the restored database as the target database. If you do not run `SWITCH` commands, then the restore operation has no effect on the repository.
`DUPLICATE`	yes	Generates a new DBID for the duplicate database, which you must manually register in the catalog. After registration, the repository has records of two distinct databases: the target and the duplicate.
`DUPLICATE`	no	Generates a new DBID for the duplicate database. The repository in the target control file is unaffected.

Specifying Filenames When Restoring to a New Host

The basic procedure for performing incomplete recovery on a new host does not differ substantially from incomplete recovery on the original host. The principal issue is whether the path names of the database files on the new host are going to be the same as the path names of the files on the primary host.

The following table indicates which restore procedure you should use depending on the situation.

Path Names of Restored Files	Procedure
Same as the target database path names	"Performing Incomplete Restore and Recovery" on page 10-10
Different from target database path names	"Restoring Files to a New Location" on page 10-19

Note the following when restoring to a new host:

- Make the target initialization parameter file accessible on the new host by copying it from the old host to a new host using an operating system utility.
- 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.

- You cannot use RMAN to restore disk backups or image copies created on one host to a new host. Nevertheless, you can transfer the files with an operating system utility. If the files are in the same location in the new host, then you do not need to recatalog them. If you transfer the files to new location, then use the `CATALOG` command to update the RMAN repository with the new filenames and use the `CHANGE ... UNCATALOG` command to uncatalog the old filenames.
- Because the restored database will not have the online redo logs of the production database, perform incomplete recovery up to the lowest SCN of the most recently archived log in each thread and then open with the `RESETLOGS` option. Obtain the SCN for recovery termination by finding the lowest SCN among the most recent archived logs for each thread.

  Start SQL*Plus and use the following query to determine the necessary SCN:

  ```
  SELECT MIN(SCN)
  FROM (SELECT MAX(NEXT_CHANGE#) SCN
        FROM V$ARCHIVED_LOG
        GROUP BY THREAD#);
  ```

Recovering an Oracle Real Application Clusters Database

The basic procedure for recovering an Oracle Real Application Clusters database does not differ substantially from recovering a non-Oracle Real Application Clusters database as described in "Performing Basic RMAN Media Recovery" on page 10-5. The main difference is that you must either configure or manually allocate channels that connect to each node of the cluster.

Because RMAN performs **autolocation** when restoring backups, datafile copies, and control file copies, it knows which channels should restore the files on each node. For example, if you create datafile copy `df1.copy` on node 2, then only the channel allocated on node 2 attempts to restore this file. Autolocation is enabled whenever the allocated channels have different `PARMS` or `CONNECT` settings.

To restore a database in an Oracle Real Application Clusters configuration:

1. After connecting to the target database and, optionally, the recovery catalog database, make sure the database is mounted. If the database is open, then shut it down and then mount it:

   ```
   SHUTDOWN IMMEDIATE
   STARTUP MOUNT
   ```

2. Perform the following operations within a RUN command:

   a. If automatic channels are not configured, then manually allocate channels for each node.

   b. Restore and recover the database.

   In this example, automatic channels are configured as follows:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'SYS/oracle@node_1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'SYS/oracle@node_2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'SYS/oracle@node_3';
```

   The following command performs complete restore and recovery:

```
RUN
{
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

> **See Also:** *Oracle9i Recovery Manager Reference* to learn about the RESTORE command, and *Oracle9i Real Application Clusters Administration* to learn about recovery in a Real Application Clusters environment

Recovering Through a RESETLOGS Operation with RMAN

This section contains these topics:

- About Recovering Through a RESETLOGS Operation with RMAN
- Requirements for Media Recovery Through a RESETLOGS operation
- Recovering Through a RESETLOGS: Basic Steps
- Recovering Through a RESETLOGS: Scenario

About Recovering Through a RESETLOGS Operation with RMAN

You must reset the online redo logs after performing recovery with a backup control file or performing incomplete recovery. After you open a database with the RESETLOGS option, you cannot typically use the backups of the database made before the RESETLOGS was performed. Oracle Corporation strongly recommends that you take a full database backup after resetting the online redo logs.

If you experience media failure after a RESETLOGS but before performing a backup, then you may be forced to perform media recovery on a backup taken prior to the last database RESETLOGS. You should not rely on the procedure to perform media recovery through RESETLOGS as a normal recovery strategy. Rather, you should resort to it only in a worst-case scenario, such as when no backups are available from after the RESETLOGS.

Requirements for Media Recovery Through a RESETLOGS operation

After you have determined that it is necessary to perform a media recovery through a RESETLOGS, you need the following:

- A control file backup created before the RESETLOGS
- Either a current control file, or a backup control file created after the RESETLOGS
- The **RESETLOGS SCN**. The RESETLOGS SCN is the SCN at which the database was opened with the RESETLOGS option. Obtain this SCN through one of the following:
  - Examining the `alert_`*`SID`*`.log` from the time of the RESETLOGS
  - Mounting a control file created after the RESETLOGS and querying the V$DATABASE control file view for RESETLOGS_CHANGE#. Then, subtract 1 from this value.
  - Run `LIST INCARNATION` and subtract 1 from the `Reset SCN` value.
- Backups of all datafiles requiring recovery from before the last RESETLOGS
- Archived redo logs up to the RESETLOGS SCN
- Archived logs from after the RESETLOGS, up to the end point of media recovery

It is impossible to perform media recovery through RESETLOGS if no backup or copy of the control file from after the RESETLOGS is available.

Recovering Through a RESETLOGS: Basic Steps

Recovery through a RESETLOGS occurs in two sequential phases discussed in this section:

1. Recovery Before RESETLOGS: Basic Steps
2. Recovery After RESETLOGS: Basic Steps

Recovery Before RESETLOGS: Basic Steps

In the first phase, you restore the database and recover it to the state it was in immediately prior to the RESETLOGS command.

1. Obtain the RESETLOGS SCN using one of these methods:
   - Running a LIST INCARNATION command and subtracting 1 from the value in the Reset SCN column. Also, note down the database incarnation keys of all incarnations as listed in the Inc Key column.
   - Examine the alert_*SID*.log from the time of the RESETLOGS, and search for the word RESETLOGS. Look for a line such as this one: RESETLOGS after incomplete recovery UNTIL CHANGE 1234. Use this value as-is.
   - By using a control file from after the RESETLOGS (either the current control file or a backup made after opening RESETLOGS), run the following query:

     ```
     SELECT (RESETLOGS_CHANGE#)-1 FROM V$DATABASE;
     ```

2. Shut down the database with the ABORT option.
3. If the current control files are not lost, then copy them to a different location. You will need them again in a later step. Do not make the copy with RMAN, but use operating system commands after the database has been shut down.
4. Start RMAN and connect to the target database and recovery catalog.
5. Start the target instance without mounting the database.
6. Run the LIST INCARNATION command to obtain the incarnation key of the prior incarnation (if do not already have it). If you have already registered the database incarnation made after the RESETLOGS with the RESET DATABASE command, then run the RESET DATABASE TO INCARNATION *inc_key* command to allow RMAN to restore backups that were taken before the RESETLOGS, where *inc_key* is the key of the prior incarnation.
7. Execute a RUN command with the following subcommands:

   a. Issue a SET UNTIL SCN *resetlogs_scn* command, where *resetlogs_scn* is the SCN obtained from the first step.

   b. After allocating all necessary channels (if you do not have automatic channels configured), restore a backup of the control file created before the RESETLOGS and then mount it.

   c. Restore datafile backups of files needing recovery from backups created before the RESETLOGS. Only restore datafiles that require recovery

   d. If some datafiles do not need to be recovered, or if you have backups of these datafiles from after the RESETLOGS, then take them offline by running the SQL statement ALTER DATABASE DATAFILE ... OFFLINE.

   e. Start SQL*Plus and query V$DATAFILE to make sure that all datafiles are pointing to valid locations and only files that need to be recovered are online.

   f. Perform an incomplete media recovery of the database using RECOVER DATABASE.

8. Shut down the database with the IMMEDIATE option.

Recovery After RESETLOGS: Basic Steps

After the previous phase is complete, follow this procedure.

1. Restore either of the following:

   - The current control files from the saved location in step 3 of "Recovery Before RESETLOGS: Basic Steps" on page 10-28

   - A backup control file copy from after the RESETLOGS if all copies of the current control file are lost (note that using a backup control file here requires another OPEN RESETLOGS after recovery completes)

2. Start the database instance without mounting the database.

3. Reset the database to the most recent incarnation. Obtain the incarnation key from the LIST INCARNATION output.

4. Mount the database.

5. Recover the database to the desired point with the RECOVER command.

6. If you took datafiles offline before recovery, then bring them online again.

7. Open the database after media recovery is complete, specifying RESETLOGS only if you used a backup control file or performed incomplete recovery. Note that a RESETLOGS operation erases all changes from the online redo logs, so if you open RESETLOGS then you cannot restart this recovery procedure and recover to the same point.

8. Back up the database to avoid performing this procedure in the future.

Recovering Through a RESETLOGS: Scenario

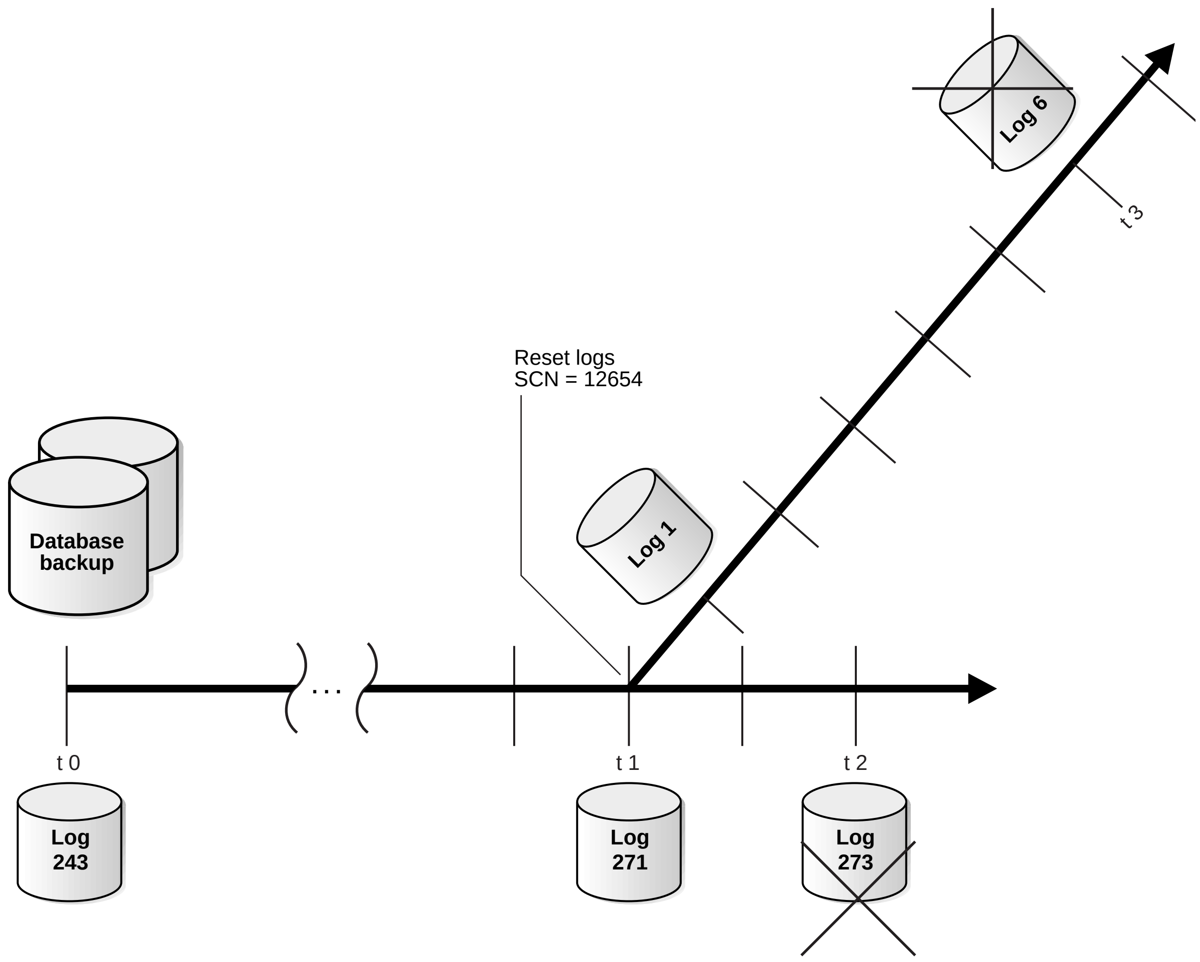
This scenario assumes that you have a database `trgt` that you want to recover through a `RESETLOGS` operation. Assume that `trgt` contains the database files described in Table 10–3.

Table 10–3 Database Files in trgt

FILE	NAME	TABLESPACE
Control file 1	`?/oradata/trgt/control01.ctl`	
Control file 2	`?/oradata/trgt/control02.ctl`	
Control file 3	`?/oradata/trgt/control03.ctl`	
Datafile 1	`?/oradata/trgt/system01.dbf`	`SYSTEM`
Datafile 2	`?/oradata/trgt/undotbs01.dbf`	`undotbs`
Datafile 3	`?/oradata/trgt/cwmlite01.dbf`	`cwmlite`
Datafile 4	`?/oradata/trgt/drsys01.dbf`	`drsys`
Datafile 5	`?/oradata/trgt/example01.dbf`	`example`
Datafile 6	`?/oradata/trgt/indx01.dbf`	`indx`
Datafile 7	`?/oradata/trgt/tools01.dbf`	`tools`
Datafile 8	`?/oradata/trgt/users01.dbf`	`users`
Online redo log 1	`?/oradata/trgt/redo01.log`	
Online redo log 2	`?/oradata/trgt/redo02.log`	
Online redo log 3	`?/oradata/trgt/redo03.log`	

For our case study, assume the scenario depicted in Figure 10–1.

Figure 10–1   Recovery Through RESETLOGS



Assume that the following sequence of events occurs:

1. You back up `trgt` at time t0.
2. At time t2 a media failure results in some of the active log file members being damaged.
3. You restore the t0 backup and perform incomplete recovery to time t1.
4. You open the `trgt` database with the `RESETLOGS` option.

5. After you opened RESETLOGS, the log sequence was reset to 1 and redo was generated until time t3 at which time another media failure results in the loss of datafile system01.dbf.

The alert.log after the RESETLOGS for trgt appears as follows:

```
Starting ORACLE instance
alter database  mount
Successful mount of redo thread 1.
Tue. Nov  7 15:39:41 2001
Completed: alter database  mount
Tue Nov  7 15:39:43 2001
ALTER DATABASE RECOVER database until time 'Nov 07 2001 15:37:54' using backup controlfile
Media Recovery Start
Media Recovery Log
ORA-279 signaled during: ALTER DATABASE RECOVER   database  until time 'Nov...
Tue Nov  7 15:39:43 2000
ALTER DATABASE RECOVER    CONTINUE DEFAULT
Media Recovery Log /oracle/oradata/trgt/arch/archive1_271.dbf
Incomplete recovery done UNTIL CHANGE 12654
Media Recovery Complete
Completed: ALTER DATABASE RECOVER    CONTINUE DEFAULT
Tue Nov  7 15:39:44 2000
alter database open resetlogs
RESETLOGS after incomplete recovery UNTIL CHANGE 12654
```

Recovery of trgt Before the RESETLOGS

In the first phase, you restore the control file and system01.dbf datafile of trgt and recover the database to the state it was in immediately prior to opening with the RESETLOGS option.

1. Because the current control file is accessible, query the V$DATABASE view to obtain the RESETLOGS SCN. For example, enter:

   ```
   SQL> SELECT RESETLOGS_CHANGE# FROM V$DATABASE;
   ```

   Alternatively, if you have a copy of the alert.log from when the RESETLOGS occurred, then look for a line such as this one:

   ```
   RESETLOGS after incomplete recovery UNTIL CHANGE 12654.
   ```

2. Abort the target instance with the following statement:

   ```
   SQL> SHUTDOWN ABORT
   ```

3. Because the current control files are not lost or inaccessible, copy them to a new location. For example, enter:

```
% cp $ORACLE_HOME/oradata/trgt/control01.ctl /tmp/control01.ctl
% cp $ORACLE_HOME/oradata/trgt/control02.ctl /tmp/control02.ctl
% cp $ORACLE_HOME/oradata/trgt/control03.ctl /tmp/control03.ctl
```

4. Start RMAN and connect to the target database and, optionally, the recovery catalog database, as in this example:

```
% rman TARGET / CATALOG rman/cat@catdb
```

5. Start the target database instance without mounting it, as in this example:

```
STARTUP NOMOUNT
```

6. Because the incarnation after the `RESETLOGS` was registered with RMAN, obtain the primary key of the prior incarnation by running a `LIST INCARNATION` command, and then reset RMAN to this incarnation. For example, run the following command, where *`inc_key`* is the key obtained from the `LIST` output:

```
LIST INCARNATION OF DATABASE trgt;
RESET DATABASE TO INCARNATION inc_key;
```

7. Execute a `RUN` command with the following subcommands:

   a. Set the SCN for recovery termination using the value obtained from the first step of the procedure.

   b. If automatic channels are not configured, then manually allocate at least one channel.

   c. Restore a control file created before the `RESETLOGS`, and then mount the control file.

   d. Restore datafile backups of files needing recovery from backups created before the `RESETLOGS`. Only restore `system01.dbf`, because this is the only datafile requiring recovery.

   e. Take all datafiles that do not need to be recovered offline by running `SQL 'ALTER DATABASE DATAFILE ... OFFLINE'`.

   f. Start SQL*Plus and query `V$DATAFILE` to ensure that all datafiles are pointing to valid locations and only files requiring recovery are online.

   g. Recover the database.

   h. Shut down the database with the `IMMEDIATE` option.

For example, run the following command:

```
RUN
{
  SET UNTIL SCN 12654;
  RESTORE CONTROLFILE;
  ALTER DATABASE MOUNT;
  RESTORE DATAFILE system01.dbf;
  SQL 'ALTER DATABASE DATAFILE ?/oradata/trgt/undotbs01.dbf,
       ?/oradata/trgt/cwmlite01.dbf, ?/oradata/trgt/drsys01.dbf,
       ?/oradata/trgt/example01.dbf, ?/oradata/trgt/indx01.dbf,
       ?/oradata/trgt/tools01.dbf, ?/oradata/trgt/users01.dbf OFFLINE';
  HOST; # check V$DATAFILE to make sure everything is OK
  RECOVER DATABASE;
  SHUTDOWN IMMEDIATE;
}
```

Recovery of trgt After RESETLOGS

1. Use an operating system utility to restore the control file that you saved in step 3 of the previous phase. For example, enter:

```
% cp /tmp/control01.ctl $ORACLE_HOME/oradata/trgt/control01.ctl
% cp /tmp/control02.ctl $ORACLE_HOME/oradata/trgt/control02.ctl
% cp /tmp/control03.ctl $ORACLE_HOME/oradata/trgt/control03.ctl
```

2. After connecting to the target database and, optionally, the recovery catalog database, start the database. For example, enter:

```
STARTUP NOMOUNT
```

3. Reset the database to the current incarnation key (to obtain the key, run a `LIST INCARNATION` command). For example, enter the following, where *`curr_inc_key`* is the current incarnation key:

```
RESET DATABASE TO INCARNATION curr_inc_key;
```

4. Mount the database. For example:

```
ALTER DATABASE MOUNT;
```

5. Execute a `RUN` command with the following subcommands:

   a. Optionally, if you want to perform incomplete recovery, then set the end time, SCN, or log sequence number.

   b. If you do not have automatic channels configured, then allocate at least one channel.

   c. Recover the database.

For example, run the following command to perform complete recovery:

```
RECOVER DATAFILE system01.dbf;
```

6. Bring online all tablespaces that you took offline before the recovery. For example:

   ```
   SQL 'ALTER DATABASE DATAFILE ?/oradata/trgt/undotbs01.dbf,
        ?/oradata/trgt/cwmlite01.dbf, ?/oradata/trgt/drsys01.dbf,
        ?/oradata/trgt/example01.dbf, ?/oradata/trgt/indx01.dbf,
        ?/oradata/trgt/tools01.dbf, ?/oradata/trgt/users01.dbf ONLINE';
   ```

7. Open database using `ALTER DATABASE OPEN` or `ALTER DATABASE OPEN RESETLOGS` depending on complete or incomplete recovery. For example:

   ```
   ALTER DATABASE OPEN;
   ```

8. It is recommended that you immediately back up the database, preferably with the database mounted (to avoid possible data loss in an open database). Because the database is a new incarnation, the backups made before the `RESETLOGS` are not easily usable. For example, enter the following to make a new database backup:

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

Performing Disaster Recovery

If you are in a disaster recovery scenario, then presumably you have lost the target database, the recovery catalog database, all control files, all online redo logs, and all parameter files. At minimum, you must have backups of some datafiles, some archived redo logs generated after the time of the backup, and at least one autobackup of the control file.

> **See Also:** "Control File and Server Parameter File Autobackups" on page 5-47

The basic procedure for disaster recovery is found in "Performing Recovery with a Backup Control File" on page 10-13, with an additional first step of restoring an autobackup of the server parameter file. After the instance is started, you can restore an autobackup of the control file, mount it, then restore and recover the datafiles. Because you are restoring to a new host, you should review the considerations described in "Restoring the Database to a New Host" on page 10-23.

The following scenario restores and recovers the database to the most recently available archived log, which in this example is log 1124 in thread 1. It assumes that:

- You are restoring the database to a new host with the same directory structure.
- 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.

In this scenario, you perform the following steps:

1. If possible, restore all relevant network files such as `tnsnames.ora` and `listener.ora` by means of operating system utilities.
2. Start RMAN and connect to the target database. If you do not have the Oracle Net files, then connect through operating system authentication.
3. Specify the DBID for the target database with the `SET DBID` command, as described in "Performing Recovery with a Backup Control File and No Recovery Catalog" on page 10-16.
4. Run the `STARTUP NOMOUNT` command. RMAN attempts to start the instance with a dummy server parameter file.
5. Allocate a channel to the media manager and then run the `RESTORE SPFILE FROM AUTOBACKUP` command.
6. Run `STARTUP FORCE NOMOUNT` mode so that the instance is restarted with the restored server parameter file.
7. Allocate a channel to the media manager and then restore a control file autobackup (refer to"Performing Recovery with a Backup Control File and No Recovery Catalog" on page 10-16).
8. Mount the restored control file.
9. Catalog any archived logs not recorded in the repository with the `CATALOG` command (refer to"Cataloging Archived Logs and User-Managed Copies" on page 18-15).
10. 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 (refer to"Restoring Files to a New Location" on page 10-19).
11. Recover the datafiles. RMAN stops recovery when it reaches the log sequence number specified.

12. Open the database in RESETLOGS mode. Only complete this last step if you are certain that no other archived logs can be applied.

Note: Oracle Corporation recommends that you back up the database after the RESETLOGS operation (not shown in the example).

```
# Start RMAN and connect to the target database
% rman TARGET SYS/oracle@trgt

# set the DBID for the target database
SET DBID 676549873;
STARTUP FORCE NOMOUNT;  # rman starts instance with dummy parameter file
RUN
{
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt;
  RESTORE SPFILE FROM AUTOBACKUP;
}
STARTUP FORCE NOMOUNT;

RUN
{
  # manually allocate a channel to the media manager
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt;
  # Restore an 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.
  SET UNTIL SEQUENCE 1124 THREAD 1;
  ALTER DATABASE MOUNT;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
ALTER DATABASE OPEN RESETLOGS; # Reset the online logs after recovery completes
```

The following example of the RUN command shows the same scenario except with new filenames for the restored datafiles:

```
RUN
{
  #  If you need to restore the files to new locations, tell Recovery Manager
  #  to do this using 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;
  SET UNTIL SEQUENCE 124 THREAD 1;
  ALTER DATABASE MOUNT;
  RESTORE DATABASE;
  SWITCH DATAFILE ALL;   #  Update the control file with new location of the datafiles.
  RECOVER DATABASE;
}
ALTER DATABASE OPEN RESETLOGS;
```

Recovering Datablocks with RMAN

The BLOCKRECOVER command can restore and recover individual datablocks within a datafile. This procedure is useful when a trace file or standard output reveals that a small number of blocks within a datafile are corrupt. Block media recovery is not useful in cases where the extent of data loss or corruption is not known; in this case, use datafile recovery instead.

This section contains these topics:

- Recovering Datablocks By Using All Available Backups
- Recovering Datablocks Using Selected Backups
- Recovering Blocks Listed in V$DATABASE_BLOCK_CORRUPTION

See Also:

- "Block Media Recovery with RMAN" on page 6-10 for an overview of block media recovery,
- *Oracle9i Recovery Manager Reference* for BLOCKRECOVER syntax
- *Oracle9i Database Reference* for V$DATABASE_BLOCK_CORRUPTION syntax

Recovering Datablocks By Using All Available Backups

In this scenario, you identify the blocks that require recovery and then use any available backup to perform the restore and recovery of these blocks.

To recover datablocks using all available backups:

1. Obtain the datafile numbers and block numbers for the corrupted blocks. Typically, you obtain this output from the standard output, the alert.log,

trace files, or a media management interface. For example, you may see the following in a trace file:

```
ORA-01578: ORACLE data block corrupted (file # 8, block # 13)
ORA-01110: data file 8: '/oracle/oradata/trgt/users01.dbf'
ORA-01578: ORACLE data block corrupted (file # 2, block # 19)
ORA-01110: data file 2: '/oracle/oradata/trgt/undotbs01.dbf'
```

2. Assuming that you have preallocated automatic channels, run the `BLOCKRECOVER` command at the RMAN prompt, specifying the file and block numbers for the corrupted blocks as in the following example:

```
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 19;
```

Recovering Datablocks Using Selected Backups

In this scenario, you identify the blocks that require recovery, and then use only selected backups to perform the restore and recovery of these blocks.

To recover datablocks while limiting the type of backup:

1. Obtain the datafile numbers and block numbers for the corrupted blocks. Typically, you obtain this output from the standard output, the `alert.log`, trace files, or a media management interface. For example, you may see the following in a trace file:

```
ORA-01578: ORACLE data block corrupted (file # 8, block # 13)
ORA-01110: data file 8: '/oracle/oradata/trgt/users01.dbf'
ORA-01578: ORACLE data block corrupted (file # 2, block # 19)
ORA-01110: data file 2: '/oracle/oradata/trgt/undotbs01.dbf'
```

2. Assuming that you have preallocated automatic channels, issue the `BLOCKRECOVER` command at the RMAN prompt, specifying the file and block numbers for the corrupted blocks and limiting the backup candidates by means of the available options. For example, you can specify what type of backup should be used to restore the blocks:

```
# restore from backupset
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 19 FROM BACKUPSET;
# restore from datafile image copy
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 19 FROM DATAFILECOPY;
```

You can indicate the backup by specifying a tag:

```
# restore from backupset with tag "mondayam"
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 199 FROM TAG = mondayam;
```

You can limit the backup candidates to those made before a certain point:

```
# restore using backups made before one week ago
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 19 RESTORE UNTIL 'SYSDATE-7';
# restore using backups made before SCN 100
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 19 RESTORE UNTIL SCN 100;
# restore using backups made before log sequence 7024
BLOCKRECOVER DATAFILE 8 BLOCK 13 DATAFILE 2 BLOCK 19 RESTORE UNTIL SEQUENCE 7024;
```

If you limit the restore of datablocks with the UNTIL clause, then RMAN must perform more recovery on the blocks. Also, the recovery phase must scan all logs for changes to the specified blocks. Hence, do not limit recovery unless necessary.

Recovering Blocks Listed in V$DATABASE_BLOCK_CORRUPTION

The V$DATABASE_BLOCK_CORRUPTION view indicates which blocks in a datafile were marked corrupt since the most recent BACKUP, BACKUP ... VALIDATE, or COPY command was run. After a corrupt block is repaired, the row identifying this block is deleted from the view.

You can check for logical corruption in the database by running the BACKUP (with or without VALIDATE option) and COPY commands with the CHECK LOGICAL command. If RMAN finds corrupt blocks, and if the number of blocks is below the MAXCORRUPT setting, then it populates V$DATABASE_BLOCK_CORRUPTION. A historical record of block corruptions in RMAN backups and copies is kept in V$BACKUP_CORRUPTION and V$COPY_CORRUPTION.

In this scenario, you identify the blocks that require recovery by querying V$DATABASE_BLOCK_CORRUPTION, and then instruct RMAN to recover all blocks listed in this view by means of the CORRUPTION LIST keyword.

To recover datablocks while limiting the type of backup:

1. Query V$DATABASE_BLOCK_CORRUPTION to determine whether corrupt blocks exist in the most recent backups and copies of the datafiles:

   ```
   SQL> SELECT * FROM V$DATABASE_BLOCK_CORRUPTION;
   ```

2. Assuming that you have preallocated automatic channels, recover all blocks marked corrupt in V$DATABASE_BLOCK_CORRUPTION by running the BLOCKRECOVER CORRUPTION LIST command. For example, this command restores blocks from backup sets created more than 10 days ago:

   ```
   BLOCKRECOVER CORRUPTION LIST
     FROM BACKUPSET
     RESTORE UNTIL TIME 'SYSDATE-10';
   ```

Validating the Restore of Backups and Copies

A restore **validation** executes a restore test run without actually restoring the files. You can test the restore of either the entire database or individual tablespaces, datafiles, or control files. The `RESTORE ... VALIDATE` and `VALIDATE BACKUPSET` commands test whether you can restore backups or copies. You have these options:

- `RESTORE ... VALIDATE` tests whether RMAN can restore a specific object from a backup or copy. RMAN chooses which backups or copies to use.
- `VALIDATE BACKUPSET` tests the validity of a backup set that you specify.

See Also: *Oracle9i Recovery Manager Reference* for `RESTORE` syntax and *Oracle9i Recovery Manager Reference* for `VALIDATE` syntax

To let RMAN choose which backup sets or copies to validate:

To perform the validation, the database can be mounted or open. You do *not* have to take datafiles offline when validating them.

1. Validate the restore of the backup sets and copies. This example validates the restore of the backup control file, `SYSTEM` tablespace, and all archived logs:

```
RESTORE CONTROLFILE VALIDATE;
RESTORE TABLESPACE SYSTEM VALIDATE;
RESTORE ARCHIVELOG ALL VALIDATE;
```

2. Check the output. If you see an error message stack and then the following message, then you do not have a backup or copy of one of the files that you are validating:

```
RMAN-06026: some targets not found - aborting restore
```

If you see an error message stack and output similar to the following, for example, then there is a problem with the restore of the specified file:

```
RMAN-03009: failure of restore command on c1 channel at 12-DEC-01 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
```

If you do not see an error stack, then RMAN successfully validated the files.

To specify which backup sets to validate:

1. If you do not need to validate the whole database, then find the backup sets and copies that you want to validate by running `LIST` commands, noting primary keys:

   ```
   LIST BACKUPSET;
   LIST COPY;
   ```

2. Validate the restore of the backup sets. This example validates the restore of backup set 1121:

   ```
   VALIDATE BACKUPSET 1121;
   ```

3. Check the output. If you see the `validation complete` message then RMAN successfully validated the restore of the specified backup set. For example:

   ```
   using channel ORA_DISK_1
   channel ORA_DISK_1: starting validation of archive log backupset
   channel ORA_DISK_1: restored backup piece 1
   piece handle=/oracle/dbs/0mdg9v8l_1_1 tag=TAG20020208T155604 params=NULL
   channel ORA_DISK_1: validation complete
   ```

RMAN Restore and Recovery Examples

This section contains these topics:

- Restoring Datafile Copies to a New Host: Example
- Restoring When Multiple Databases in the Catalog Share the Same Name: Example
- Recovering a Backup Made Before a RESETLOGS: Example
- Recovering a Database in NOARCHIVELOG Mode: Example
- Recovering a Lost Datafile Without a Backup: Example
- Transporting a Tablespace Backup to a Different Database with RMAN: Example

Restoring Datafile Copies to a New Host: Example

To move the database to a new host by means of datafile copies, you must transfer the copies manually to the new machine. This example assumes that you are using a recovery catalog.

1. After connecting to the target database and recovery catalog, run a `LIST` command to see a listing of datafile copies and their associated primary keys, as in the following example:

   ```
   LIST COPY;
   ```

2. Copy the datafile copies to the new host by means of an operating system utility. For example, in UNIX:

   ```
   % cp -r /tmp/*dbf /net/new_host/oracle/oradata/trgt
   ```

3. Start RMAN and then uncatalog the datafile copies on the old host. For example, enter:

   ```
   CHANGE COPY OF DATAFILE 1,2,3,4,5,6,7,8 UNCATALOG;
   ```

4. Catalog the datafile copies, using their new filenames. For example, run:

   ```
   CATALOG DATAFILECOPY
     '?/oradata/trgt/system01.dbf', '?/oradata/trgt/undotbs01.dbf',
     '?/oradata/trgt/cwmlite01.dbf', '?/oradata/trgt/drsys01.dbf',
     '?/oradata/trgt/example01.dbf', '?/oradata/trgt/indx01.dbf',
     '?/oradata/trgt/tools01.dbf', '?/oradata/trgt/users01.dbf';
   ```

5. Perform the restore and recovery operation described in "Restoring the Database to a New Host" on page 10-23.

Restoring When Multiple Databases in the Catalog Share the Same Name: Example

As explained in the description for `SET DBID` in *Oracle9i Recovery Manager Reference*, you must run the `SET DBID` command to restore the control file when the target database is not mounted and multiple databases registered in the recovery catalog share the same name. In this case, do the following steps in order:

1. Start RMAN and connect to the target database.
2. Run the `STARTUP FORCE NOMOUNT` command.
3. Run the `SET DBID` command to distinguish this connected target database from other target databases that have the same name.
4. Run the `RESTORE CONTROLFILE` command. After restoring the control file, you can mount the database to restore the rest of the database.

This procedure avoids the `RMAN-20005` message when you attempt to restore the control file. This message occurs because more than one target database has the same name, so RMAN requires the unique DBID to distinguishes the databases from one another.

Obtaining the DBID of a Database You Need to Restore

If you have saved the RMAN output, then refer to this information to determine the database identifier, because RMAN automatically provides it whenever you connect to the database:

```
% rman TARGET /

Recovery Manager: Release 9.2.0.0.0

connected to target database: RMAN (DBID=1231209694)
```

If you have not saved the RMAN output and need the DBID value of a database for a restore operation, then obtain it by querying the `RC_DATABASE` or `RC_DATABASE_INCARNATION` recovery catalog views.

Because the names of the databases that are registered in the recovery catalog are presumed nonunique in this scenario, you must use some other unique piece of information to determine the correct DBID. If you know the filename of a datafile or online redo log associated with the database you wish to restore, and this filename is unique across all databases registered in the recovery catalog, then substitute this fully specified filename for *`filename_of_log_or_df`* in the following queries. Determine the DBID by performing one of the following queries:

```
SELECT DISTINCT DB_ID
FROM DB, DBINC, DFATT
WHERE DB.DB_KEY = DBINC.DB_KEY
  AND DBINC.DBINC_KEY = DFATT.DBINC_KEY
  AND DFATT.FNAME = 'filename_of_log_or_df';

SELECT DISTINCT DB_ID
FROM DB, DBINC, ORL
WHERE DB.DB_KEY = DBINC.DB_KEY
  AND DBINC.DBINC_KEY = ORL.DBINC_KEY
  AND ORL.FNAME = 'filename_of_log_or_df';
```

Restoring a Backup Control File By Using the DBID

To set the DBID, connect RMAN to the target database and run the following `SET` command, where *`target_dbid`* is the value you obtained from the previous step:

```
SET DBID = target_dbid;
```

To restore the control file to its default location and then mount it, run:

```
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
```

To restore and recover the database, run:

```
RESTORE DATABASE;
RECOVER DATABASE
  # optionally, delete logs restored for recovery and limit disk space used
  DELETE ARCHIVELOG MAXSIZE 2M;
```

Recovering a Backup Made Before a RESETLOGS: Example

Assume the following situation:

- You run RMAN with a recovery catalog.
- You made a backup of `trgt` on January 2, 2001.
- You performed incomplete recovery on this database and opened it with the `RESETLOGS` option on January 10, 2001. A new database incarnation was created.

On January 25, you discover that you need crucial data that was dropped from the database at 8:00 a.m. on January 8, 2001. You decide to reset `trgt` to the prior incarnation, restore the January 2 backup, and recover to 7:55 a.m. on January 8.

Note: It is not possible to restore one datafile of a previous incarnation while the current database is in a different incarnation—you must restore the whole database.

To recover the database by means of a backup from the old incarnation:

1. You obtain the primary key of the previous incarnation by executing a `LIST` command:

```
# obtain primary key of old incarnation
LIST INCARNATION OF DATABASE trgt;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         TRGT      1224038686  NO     1            02-JAN-01
1       582       TRGT      1224038686  YES    59727        10-JAN-01
```

2. Make sure the database is started but not mounted:

```
SHUTDOWN ABORT
STARTUP NOMOUNT
```

3. Reset the incarnation to the primary key that you just obtained:

```
# reset database to old incarnation
RESET DATABASE TO INCARNATION 2;
```

4. Recover the database, performing the following operations in the `RUN` command:

   - Set the end time for recovery to the time just before the loss of the data.
   - If automatic channels are not configured, then manually allocate one or more channels.
   - Restore the control file and mount it.
   - Restore and recover the database.

   For example, run the following commands:

```
RUN
{
  SET UNTIL TIME 'Jan 8 2001 07:55:00'; # set time to just before data was lost
  RESTORE CONTROLFILE;
  ALTER DATABASE MOUNT; # mount database after restoring control file
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

5. If recovery is successful, then reset the online redo logs:

```
ALTER DATABASE OPEN RESETLOGS; # this command automatically resets the database
                               # so that this incarnation is the new incarnation
```

Recovering a Database in NOARCHIVELOG Mode: Example

You can recover a database running in `NOARCHIVELOG` mode by means of incremental backups. Note that the incremental backups must be consistent, so you cannot make backups of the database when it is open.

Assume the following scenario:

- You run database `trgt` in `NOARCHIVELOG` mode.
- You use a recovery catalog.
- You shut down the database consistently and make a level 0 backup of database `trgt` to tape on Sunday afternoon.
- You shut down the database consistently and make a level 1 differential incremental backup to tape at 3:00 a.m. on Wednesday and Friday.

- The database has a media failure on Saturday, destroying half of the datafiles as well as the online redo logs.

In this case, you must perform an incomplete media recovery until Friday, since that is the date of the most recent incremental backup. RMAN uses the level 0 Sunday backup as well as the Wednesday and Friday level 1 backups.

Because the online redo logs are lost, you must specify the NOREDO option in the RECOVER command. You must also specify NOREDO if the online logs are available but the redo cannot be applied to the incrementals. If you do not specify NOREDO, then RMAN searches for redo logs after applying the Friday incremental backup, and issues an error message when it does not find them. If the online logs had been available, then you could have run RECOVER DATABASE without specifying NOREDO.

After connecting to trgt and the catalog database, recover the database using the following command:

```
STARTUP FORCE MOUNT;
RESTORE CONTROLFILE;  # restore control file from consistent backup
ALTER DATABASE MOUNT;
RESTORE DATABASE;  # restore datafiles from consistent backup
RECOVER DATABASE NOREDO;  # specify NOREDO because online redo logs are lost
ALTER DATABASE OPEN RESETLOGS;
```

In this scenario, all changes generated between the Friday incremental backup and the Saturday failure are not applied.

Recovering a Lost Datafile Without a Backup: Example

In this scenario, the following sequence of events occurs:

1. You make a whole database backup of your ARCHIVELOG mode database.
2. You create a tablespace containing a single datafile called ?/oradata/trgt/history01.dbf.
3. You populate the newly created datafile with data.
4. You archive all the active online redo logs.
5. Someone accidentally deletes ?/oradata/trgt/history01.dbf from the operating system before you have a chance to back it up.

Are you prevented from recovering the data in the lost datafile because you have no backup of the file? No. You can recover the data by creating a new datafile with the same filename as the lost datafile, then run the `RECOVER` command to apply the redo for this file.

For example, start RMAN, connect to the target database, and then run the following statements at the RMAN prompt:

```
# take the missing datafile offline
# note that SQL statement is bounded by double quotes, but the datafile name has two
# individual single quotes both before and after it
SQL "ALTER DATABASE DATAFILE
    '' ?/oradata/trgt/history01.dbf '' OFFLINE";
# create a new datafile with the same name as the missing datafile
SQL "ALTER DATABASE CREATE DATAFILE
    '' ?/oradata/trgt/history01.dbf '' ";
# recover the newly created datafile
RECOVER DATAFILE '?/oradata/trgt/history01.dbf';
# bring the recovered datafile back online
SQL "ALTER DATABASE DATAFILE
    '' ?/oradata/trgt/history01.dbf '' ONLINE";
```

Transporting a Tablespace Backup to a Different Database with RMAN: Example

You can use the transportable tablespace feature to copy a tablespace from one database to another database. As described in *Oracle9i Database Administrator's Guide*, the basic method for transporting tablespaces does not make use of RMAN. Nevertheless, if you use RMAN to back up your target database, then you can also use RMAN to transport backups of a tablespace from one database into another.

In the following procedure, assume that:

- You are transporting a backup of tablespace `users` from database `trgta`, located on computer `hosta`, to database `trgtb`, located on computer `hostb`
- You want to transport a backup of the `users` tablespace, which has one datafile
- You want to name the restored backup of the `users` datafile as `/net/hostb/oracle/oradata/trgtb/users01.dbf` in database `trgtb`
- You have recoverable backups of the following files from database `trgta`:
  - All datafiles in the tablespaces that you are transporting (in this example, the `users` tablespace)
  - Datafiles in the `SYSTEM` tablespace
  - Datafiles with rollback or undo segments
  - A control file that contains the metadata for the preceding datafile backups

- The backups of `trgta` are accessible by `hostb` through a tape device.

  See Also: *Oracle9i Database Administrator's Guide* to learn how to transport a tablespace

To transport a tablespace into a different database:

1. Create an auxiliary instance on `hostb` according to the instructions in the "Preparing the Auxiliary Instance for Duplication: Basic Steps" on page 12-9.
2. Connect RMAN to the auxiliary instance as if it were a new target instance. For example:

   ```
   rman TARGET SYS/oracle@auxdb CATALOG rman/rman@catdb
   ```

3. Restore the control file to a temporary location, then mount the control file and exit the session. For example:

   ```
   RESTORE CONTROLFILE TO '/net/hostb/tmp/cf.f';
   STARTUP FORCE MOUNT;
   EXIT
   ```

4. Reconnect RMAN to the same auxiliary instance in `NOCATALOG` mode, then restore and recover the auxiliary database. Perform the following steps:

   a. Specify the noncurrent time, SCN, or archived log to which you want to recover the tablespace. You cannot recover the tablespace to the current time. Use the specified `UNTIL` time to indicate which backup of the tablespace that you want to restore.

   b. If the restored control file does not included configured channels, then manually allocate a channel to the device containing the backups.

   c. Run `SET NEWNAME` to specify temporary filenames for the `SYSTEM` datafiles and the datafiles containing rollback or undo segments.

   d. Run `SET NEWNAME` to specify the filenames in the `trgtb` database that will be used by the datafiles in the transported tablespace.

   e. Restore and recover the tablespaces.

   For example, run the following commands:

   ```
   % rman TARGET SYS/oracle@auxdb NOCATALOG

   RUN
   {
     SET UNTIL ARCHIVELOG 1243 THREAD 1;  # set the end recovery log
     ALLOCATE CHANNEL c1 DEVICE TYPE sbt; # allocate a channel if not configured
   ```

```
  # specify temporary name for SYSTEM datafile
  SET NEWNAME FOR DATAFILE 1 TO '/net/hostb/tmp/df1.dbf';
  # specify temporary names for datafiles with undo or rollback segments
  SET NEWNAME FOR DATAFILE 2 TO '/net/hostb/tmp/df2.dbf';
  # specify names for datafiles to be plugged into trgtb database
  SET NEWNAME FOR DATAFILE 8 TO '/net/hostb/oracle/oradata/trgtb/users01.dbf';
  # restore and recover the datafiles
  RESTORE DATAFILE 1, 2, 8;
  RECOVER DATAFILE 1, 2, 8;
}
```

5. Take all auxiliary tablespaces offline except the tablespaces that you recovered in the preceding step. For example:

   ```
   SQL 'ALTER TABLESPACE cwmlite, drsys, example, indx, tools OFFLINE IMMEDIATE';
   ```

6. Open the auxiliary database with the `RESETLOGS` option. For example:

   ```
   ALTER DATABASE OPEN RESETLOGS;
   ```

7. Make the tablespace that you are transporting read-only. For example:

   ```
   SQL 'ALTER TABLESPACE users READ ONLY';
   ```

8. Export the metadata from the transported tablespace as described in "Step 2: Generate a Transportable Tablespace Set" in *Oracle9i Database Administrator's Guide*. For example:

   ```
   exp TRANSPORT_TABLESPACE=y TABLESPACES=(users)
           TRIGGERS=y CONSTRAINTS=n GRANTS=n FILE=expdat.dmp
   ```

9. Shut down the auxiliary instance, and then delete all auxiliary files except the datafiles in the transported tablespace. For example:

   ```
   sqlplus SYS/oracle@auxdb <<EOF
   SHUTDOWN ABORT
   EXIT
   EOF
   rm /net/hostb/tmp/*
   ```

10. Import the metadata from the transported tablespace as described in "Step 4: Plug In the Tablespace Set" in *Oracle9i Database Administrator's Guide*. For example:

    ```
    imp TRANSPORT_TABLESPACE=y FILE=expdat.dmp
      DATAFILES=('/net/hostb/oracle/oradata/trgtb/users01.dbf')
      TABLESPACES=(users) TTS_OWNERS=(usera)
      FROMUSER=(usera) TOUSER=(userb)
    ```

11

Performing RMAN Tablespace Point-in-Time Recovery

This chapter describes how to use Recovery Manager (RMAN) to perform tablespace point-in-time recovery (TSPITR). This chapter contains these topics:

Introduction to RMAN TSPITR

Recovery Manager (RMAN) automated tablespace point-in-time recovery (TSPITR) enables you to quickly recover one or more tablespaces to a time that is different from that of the rest of the database.

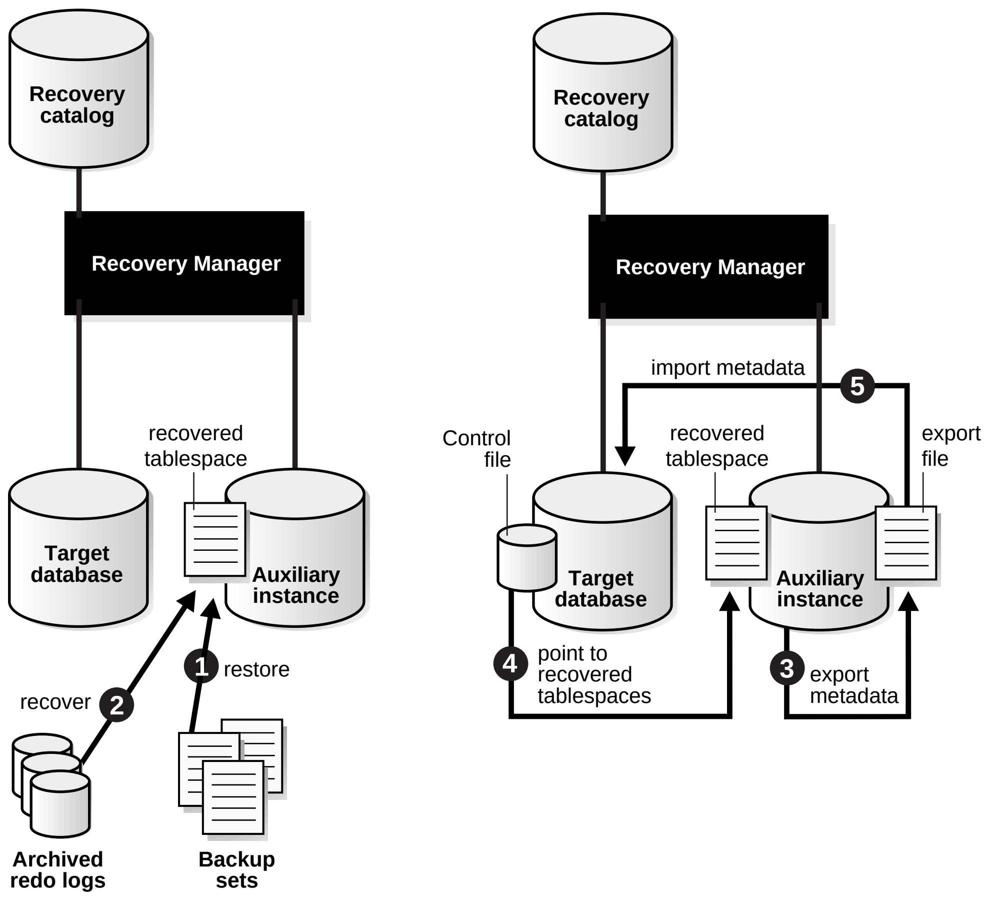
RMAN TSPITR is most useful for recovering the following:

- An erroneous `DROP TABLE` or `TRUNCATE TABLE` statement
- A table that has become logically corrupted
- An incorrect batch job or other DML statement that has affected only a subset of the database
- A logical schema to a point different from the rest of the physical database when multiple schemas exist in separate tablespaces of one physical database

Like a table export, RMAN TSPITR enables you to recover a consistent data set; however, the data set is the entire tablespace rather than one object. As Figure 11–1 illustrates, RMAN does the following:

1. Restores the specified tablespace backups
2. Recovers the specified tablespaces
3. Exports metadata from the auxiliary instance
4. Points the target database control file to the newly recovered datafiles
5. Imports metadata into the target database

Figure 11–1 RMAN TSPITR



Glossary of TSPITR Terminology

Familiarize yourself with the following terms and abbreviations, which are used throughout this chapter:

TSPITR
Tablespace point-in-time recovery

Auxiliary Instance
The auxiliary instance used to recover the backup tablespaces. The database created by TSPITR never has independent existence: it is only an intermediate work area.

Recovery Set
Tablespaces in the target database requiring TSPITR to be performed on them. For example, if you need to recover tablespace `users` to a noncurrent time, then `users` is the tablespace in the recovery set.

Auxiliary Set
Any other files required for TSPITR, including:

- Backup control file
- `SYSTEM` tablespace
- Datafiles containing rollback or undo segments
- Temporary tablespace (optional). A small space is required by Export for sort operations.
- Online redo logs of the auxiliary database.

> **See Also:** "Responding to Unsuccessful RMAN TSPITR" on page 11-19 for more information on sort space issues

Planning for RMAN TSPITR

RMAN TSPITR requires careful planning. Before proceeding, read this chapter thoroughly.

This section covers the following topics:

- Performing TSPITR Without a Recovery Catalog
- Understanding General Restrictions
- Managing Data Relationships

> **Note:** Many of the limitations and planning steps in this chapter can also be found in *Oracle9i User-Managed Backup and Recovery Guide*; however, differences in limitations and planning exist. These differences are explicitly noted.

Performing TSPITR Without a Recovery Catalog

You can perform RMAN TSPITR either with or without a recovery catalog. If you do *not* use a recovery catalog, then note these restrictions:

- The rollback or undo segments at the time of the TSPITR are needed in 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 rollback or undo segments have changed since that time, then TSPITR fails when the recovery tries to write to a nonexistent file.
- TSPITR to a time that is too old may not succeed if Oracle has reused the control file records for needed copies and backups.
- Assume that you run TSPITR on a tablespace, and then bring the tablespace online at time *t*. Backups of the tablespace created before time *t* are no longer usable for recovery with a current control file. Thus, 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*.

  The reason for the additional restrictions in `NOCATALOG` mode is that 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 can no longer use a backup taken prior to time *t*. However, if you restore a control file backed up prior to *t*, then you can perform incomplete recovery of the whole database to any time less than or equal to *t*.

Understanding General Restrictions

When performing RMAN TSPITR, you *cannot* do the following:

- Run the target and auxiliary databases on separate computers. However, the target and auxiliary databases can be in a cluster configuration that uses shared disks.
- Recover dropped tablespaces.
- Recover a tablespace that has been dropped and re-created with the same name.
- Remove a datafile that has been added to a tablespace. If the file was added after the point to which RMAN is recovering, then the file will still be part of the tablespace (and will be empty) after RMAN TSPITR is complete.

- Issue DML statements on the auxiliary instance—the auxiliary instance is a temporary instance used for recovery only.
- Assume that you perform TSPITR on a tablespace when connected to a recovery catalog, and then bring the tablespace back online at time *t*. You can use backups created before time *t* to recover any tablespace or the whole database up to time *t*, but not later. Hence, you should immediately back up the tablespace after performing TSPITR.
- Recover optimizer statistics for objects that have had statistics calculated on them; recalculate statistics after performing TSPITR.
- Place any of the following objects within the recovery set:
  - Replicated master tables.
  - Partial tables. For example, if you perform RMAN TSPITR on partitioned tables and spread partitions across multiple tablespaces, then RMAN returns an error message during the export phase.
  - Tables without their constraints or constraints without their tables.
  - Tables with VARRAY columns.
  - Tables with nested tables.
  - Tables with external files.
  - Snapshot logs and snapshot tables.
  - Tablespaces containing undo or rollback segments.
  - Objects owned by SYS (including rollback segments).

Managing Data Relationships

TSPITR provides views that can detect any data relationships between objects in the recovery set and objects in the rest of the database. TSPITR cannot successfully complete unless these relationships are managed, either by removing or suspending the relationship or by including the related object within the recovery set.

> **See Also:** *Oracle9i User-Managed Backup and Recovery Guide.* to learn how to research and resolve dependency issues.

Determining Whether Objects Will Be Lost

When RMAN TSPITR is performed on a tablespace, any objects created after the recovery time are lost. To see which objects will be lost, query the `TS_PITR_`

OBJECTS_TO_BE_DROPPED view on the primary database. The contents of the view are described in Table 11–1.

Table 11–1 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.

When querying this view, supply all the elements of the date field, otherwise the default setting is used. Also, use the TO_CHAR and TO_DATE functions. For example, with a recovery set consisting of users and tools, and a recovery point in time of '2001-06-02:07:03:11', issue the following statement:

```
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('01-JUN-02:07:03:11','YY-MON-DD:HH24:MI:SS')
ORDER BY TABLESPACE_NAME, CREATION_TIME;
```

See Also: *Oracle9i Database Reference* for more information about the TS_PITR_OBJECTS_TO_BE_DROPPED view

Researching and Resolving Dependencies on the Primary Database

Query the TS_PITR_CHECK view to identify relationships between objects that overlap 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 retrace these relationships after completing TSPITR.

Supply a four-line predicate detailing the recovery set tablespace to query the TS_PITR_CHECK view. For example, with a recovery set consisting of tools and users, the SELECT statement against TS_PITR_CHECK would be as follows:

```
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 following query:

```
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 following output when `TS_PITR_CHECK` is queried against tablespaces `users` and `tools` (assuming appropriate formatting):

```
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
```

The table `SYSTEM.tp` has a partitioned index `tpind` that consists of two partitions, `ip1` in tablespace `id1` and `ip2` in tablespace `id2`. Either drop `tpind` or include `id1` and `id2` in the recovery set.

See Also: *Oracle9i Database Reference* for more information about the `TS_PITR_CHECK` view

Choosing a Naming Method for Files in the Auxiliary and Recovery Sets

This section contains these topics:

- Overview of Datafile Naming in RMAN TSPITR
- Using SET NEWNAME to Name Files in the Auxiliary and Recovery Sets
- Using Datafile Copies in the Auxiliary and Recovery Sets
- Using Initialization Parameters to Name the Auxiliary Set Files

Overview of Datafile Naming in RMAN TSPITR

Before you begin setting up the auxiliary instance to perform the TSPITR, you must decide on a naming method for the datafiles in the auxiliary set. If the names for files in the auxiliary set are not different from the filenames in use by the target database, then RMAN signals an error during TSPITR and exits.

For example, if you are creating the auxiliary database in the `/tmp` directory, then the absolute path name of the auxiliary datafiles must be prefixed by `/tmp`. These auxiliary files are intended *only* for performing the TSPTIR, so you should delete them immediately afterward. If filenames are not converted in the auxiliary set, then RMAN signals an error during TSPITR and exits.

Optionally, you may also choose to rename the datafiles in the recovery set tablespaces on the target (not the auxiliary) database. For example, if you are performing TSPITR on datafile `?/oradata/trgt/users01.dbf` in the `users` tablespace, you may decide to rename it to `/disk2/datafiles/users01.dbf`. This operation is equivalent to an `ALTER DATABASE RENAME FILE` for specified files in the target database.

Table 11–1 describes the commands and parameters used to name datafiles in the auxiliary and recovery sets during TSPITR. The order of precedence in the following table goes top to bottom, so `SET NEWNAME` takes precedence over `CONFIGURE AUXNAME` and `DB_FILE_NAME_CONVERT`.

Table 11–2 Datafile Naming Methods

Order of Precedence	Command/Parameter	Can Name Files in Auxiliary Set?	Can Rename Files in Recovery Set?
1	`SET NEWNAME`	Yes	Yes
2	`CONFIGURE AUXNAME`	Yes	Yes
3	`DB_FILE_NAME_CONVERT`	Yes	No

Using SET NEWNAME to Name Files in the Auxiliary and Recovery Sets

You can specify a new name for any datafiles in the auxiliary set with the RMAN command `SET NEWNAME`. RMAN uses this new name as the temporary location in which to restore and recover the datafile. This new name also overrides the setting in the `DB_FILE_NAME_CONVERT` parameter in the initialization parameter file, if this parameter happens to be set.

You can also use `SET NEWNAME` to rename datafiles in recovery set tablespaces. If you specify a new name, then the new filenames replace the original filenames in the target control file. When setting new filenames, RMAN does not check for conflicts between datafile names at the auxiliary and target databases. Any conflicts result in an RMAN error during TSPITR.

Using SET NEWNAME to Name Files: Example

For example, assume that the auxiliary set contains the following datafiles:

- `?/oradata/trgt/system01.dbf` of the `SYSTEM` tablespace
- `?/oradata/trgt/undotbs01.dbf` of the `undotbs` tablespace

The recovery set contains the following datafiles:

- `?/oradata/trgt/users01.dbf` of the `users` tablespace
- `?/oradata/trgt/tools01.dbf` of the `tools` tablespace

You want to create the auxiliary database in the `/tmp` directory. Also, you decide to rename the datafile in `tools` to `/private1/tools01.dbf`, but leave the datafile in the `users` tablespace with its original name.

In this case, you can run the following command to run TSPITR on `tools` and `users`, causing `?/oradata/trgt/tools01.dbf` to be renamed to `?/dbs/tools01.dbf` on the target database:

```
RUN
{
  # set newnames for auxiliary set datafiles
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/system01.dbf' TO '/tmp/system01.dbf';
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/undotbs01.dbf' TO '/tmp/undotbs01.dbf';
  # rename one recovery set datafile
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/tools01.dbf' TO '/disk1/tools01.dbf';
  RECOVER TABLESPACE tools, users UNTIL SEQUENCE 1034 THREAD 1;
}
```

Using Datafile Copies in the Auxiliary and Recovery Sets

Using a datafile copy on disk is much faster than restoring a datafile. Hence, you may wish to use an appropriate copy of a datafile in the recovery or auxiliary set instead of restoring and recovering a datafile.

RMAN TSPITR can use a datafile copy if the following conditions are met:

1. The datafile copy name is registered in the recovery catalog as the auxiliary name of the corresponding datafile with the following command (where *`filename`* is the datafile name or number, and *`aux_datafile_name`* is the datafile auxiliary name):

   ```
   CONFIGURE AUXNAME FOR DATAFILE FILENAME TO auxiliary_datafile_name;
   ```

2. The datafile copy was made before the time specified in the `UNTIL` clause with the following RMAN command (where *`'filename'`* is the datafile filename):

   ```
   COPY DATAFILE 'filename' TO AUXNAME;
   ```

If RMAN uses a datafile copy and TSPITR completes successfully, then the *`auxiliary_datafile_name`* is removed from the recovery catalog, and updated to status `DELETED` in the control file. The original datafile at the target is replaced by this datafile copy after RMAN TSPITR completes.

Note: RMAN does not use a datafile copy if you use `SET NEWNAME` for the same datafile.

Using CONFIGURE AUXNAME to Name Files: Example

For example, assume that the auxiliary set contains the following datafiles:

- `?/oradata/trgt/system01.dbf` of the `SYSTEM` tablespace
- `?/oradata/trgt/undotbs01.dbf` of the `undotbs` tablespace

The recovery set contains the following datafiles:

- `?/oradata/trgt/users01.dbf` of the `users` tablespace
- `?/oradata/trgt/tools01.dbf` of the `tools` tablespace

Also, assume that you ran the following commands at a time before the end time of the desired TSPTIR:

```
CONFIGURE AUXNAME FOR DATAFILE '?/oradata/trgt/system01.dbf' TO '/copy/system01.dbf';
CONFIGURE AUXNAME FOR DATAFILE '?/oradata/trgt/undotbs01.dbf' TO '/copy/undotbs01.dbf';
CONFIGURE AUXNAME FOR DATAFILE '?/oradata/trgt/tools01.dbf' TO '?/dbs/tools01.dbf';

COPY DATAFILE '?/oradata/trgt/system01.dbf' TO AUXNAME;
COPY DATAFILE '?/oradata/trgt/undotbs01.dbf' TO AUXNAME;
COPY DATAFILE '?/oradata/trgt/tools01.dbf' TO AUXNAME;
```

In this case, you can run the following command to run TSPITR on `tools` and `users`, causing `?/oradata/trgt/tools01.dbf` to be renamed to `?/dbs/tools01.dbf` on the target database:

```
RECOVER TABLESPACE tools, users UNTIL SEQUENCE 1034 THREAD 1;
```

Using Initialization Parameters to Name the Auxiliary Set Files

You can use the `DB_FILE_NAME_CONVERT` parameter in the auxiliary initialization parameter file to name auxiliary set—but not recovery set—datafiles.If neither a new name nor auxiliary name is set for a datafile in an auxiliary set tablespace, then RMAN can use the converted filename specified in the auxiliary database control file to perform the restore and recovery. RMAN checks for conflicts between datafile names at the auxiliary and target databases. Any conflicts result in an error.

If neither a new name or auxiliary name is set for a datafile in a recovery set tablespace, or if the file at the auxiliary name is unusable, then RMAN uses the original location of the datafile.

See Also: "Task 2: Create a Parameter File for the Auxiliary Instance" on page 11-14 for an example of an auxiliary initialization parameter file

Using Initialization Parameters to Name Files: Example For example, assume that the auxiliary set contains the following files:

- `?/oradata/trgt/system01.dbf` of the `SYSTEM` tablespace
- `?/oradata/trgt/undotbs01.dbf` of the `undotbs` tablespace

The recovery set contains the following files:

- `?/oradata/trgt/users01.dbf` of the `users` tablespace
- `?/oradata/trgt/tools01.dbf` of the `tools` tablespace

You want to create the auxiliary database in the `/tmp` directory. Also, you decide to rename the datafile in `tools` to `/private1/tools01.dbf`, but leave the datafile in the `users` tablespace with its original name.

In this case, you can set the following parameter in the auxiliary initialization parameter file to name the auxiliary set files:

```
DB_FILE_NAME_CONVERT=('/oradata/trgt','/tmp') # captures all auxiliary set files
```

In this case, you can run the following command to run TSPITR on `tools` and `users`, causing `?/oradata/trgt/tools01.dbf` to be renamed to `?/dbs/tools01.dbf` on the target database:

```
RUN
{
  # rename one recovery set datafile
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/tools01.dbf' TO '?/dbs/tools01.dbf';
  RECOVER TABLESPACE tools, users UNTIL SEQUENCE 1034 THREAD 1;
}
```

Preparing the Auxiliary Instance for RMAN TSPITR

Satisfy the following requirements discussed in this section before performing RMAN TSPITR:

- Task 1: Create an Oracle Password File for the Auxiliary Instance
- Task 2: Create a Parameter File for the Auxiliary Instance
- Task 3: Ensure Oracle Net Connectivity to the Auxiliary Instance

- Task 4: Start the Auxiliary Instance
- Task 5: Start the Recovery Manager Command-Line Interface

Task 1: Create an Oracle Password File for the Auxiliary Instance

For instructions on how to create and maintain Oracle password files, refer to the *Oracle9i Database Administrator's Guide*.

Task 2: Create a Parameter File for the Auxiliary Instance

Create an initialization parameter file for the auxiliary instance and set the parameters described in Table 11–3.

Table 11–3 Initialization Parameters in the Auxiliary Instance

Parameter	**Specify**
DB_NAME	The same name as the target database.
LOCK_NAME_SPACE	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.
DB_FILE_NAME_CONVERT	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$DATAFILE.NAME, and ensure that the conversion pattern matches the format of the filename displayed in the view. **Note:** Some platforms do not support ending the patterns in a forward or backward slash (/ or \\). **See Also**: "Choosing a Naming Method for Files in the Auxiliary and Recovery Sets" on page 11-9
LOG_FILE_NAME_CONVERT	Patterns to convert filenames for the online redo logs of the auxiliary database. Obtain the online redo log filenames by querying V$LOGFILE.MEMBER, and ensure that the conversion pattern matches the format of the filename displayed in the view. **Note:** Some platforms do not support ending the patterns in a forward or backward slash (\\ or /).

Table 11–3 Initialization Parameters in the Auxiliary Instance

Parameter	**Specify**
CONTROL_FILES	A different value from the CONTROL_FILES parameter in the target parameter file.
REMOTE_LOGIN_PASSWORDFILE	Set to EXCLUSIVE when connecting to the auxiliary instance by means of a password file.
COMPATIBLE	The same value as the parameter in the target database.
DB_BLOCK_SIZE	If this initialization parameter is set in the target database, then it must be set to the same value in the auxiliary instance.

Set other parameters as needed, including the parameters that allow you to connect as SYSDBA through Oracle Net.

Following are examples of the initialization parameter settings for the auxiliary instance:

```
DB_NAME=trgt
LOCK_NAME_SPACE=_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 = 9.0.1
DB_BLOCK_SIZE=8192
```

Note: After setting these parameters, ensure that you do not overwrite the initialization settings for the production files at the target database.

See Also: "Perform TSPITR again, following the instructions in "Performing RMAN TSPITR" on page 11-17." on page 11-20 for details about DB_FILE_NAME_CONVERT, and *Oracle9i Net Services Administrator's Guide* for more information about Oracle Net

Task 3: Ensure 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 connection to the auxiliary instance.

Task 4: Start the Auxiliary Instance

Before beginning RMAN TSPITR, use SQL*Plus to connect to the auxiliary instance and start it in `NOMOUNT` mode (specifying a parameter file if necessary). For example:

```
SQL> CONNECT SYS/oracle@aux AS SYSDBA
SQL> STARTUP NOMOUNT PFILE='/tmp/initAUX.ora'
```

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.

Task 5: Start the Recovery Manager Command-Line Interface

Use one of the following methods discussed in this section to start the RMAN command-line interface:

- Connecting from the Operating System Command Line
- Connecting from the RMAN Prompt

Connecting from the Operating System Command Line

To connect to the auxiliary instance, target instance, and optional recovery catalog, supply the following information when starting RMAN:

```
% rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb AUXILIARY SYS/oracle@aux
```

The following table describes the variables used in this RMAN connection.

Variable	Meaning
`SYS`	User with `SYSDBA` privilege
`oracle`	The password for connecting as `SYSDBA` specified in the target database's `orapwd` file
`trgt`	The net service name for the target database
`rman`	Owner of the recovery catalog

Variable	Meaning
cat	The password for user RMAN specified in the recovery catalog's orapwd file
catdb	The net service name for the recovery catalog database
aux	The net service name for the auxiliary database.

Connecting from the RMAN Prompt

You can start the RMAN command-line interface without a connection to the auxiliary instance, and then use the CONNECT command at the RMAN prompt. This example connects in the default NOCATALOG mode:

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

To connect to a catalog, run the CONNECT CATALOG command:

```
RMAN> CONNECT CATALOG rman/cat@catdb
```

Performing RMAN TSPITR

After completing all planning requirements, recover the recovery set tablespaces, specifying the end point of recovery. You do not have to take the tablespaces offline first because RMAN does it automatically. The following command performs TSPITR on the users tablespace until log sequence 13:

```
RECOVER TABLESPACE users UNTIL SEQUENCE 13 THREAD 1;
```

If no auxiliary device configuration is specified, and if RMAN needs to automatically allocate auxiliary channels, then RMAN uses the target database device configuration. You do not need to configure auxiliary channels unless they require different parameters from the target channels.

The following example assumes that you do *not* have automatic channels configured and so must manually allocate auxiliary channels:

```
# manually allocate at least one auxiliary channel
RUN
{
  ALLOCATE AUXILIARY CHANNEL c1 DEVICE TYPE DISK;
  RECOVER TABLESPACE users UNTIL SEQUENCE 13 THREAD 1;
}
```

Note: If you specify UNTIL TIME, then the format for the time should use the same format as the NLS_DATE_FORMAT parameter.

RMAN automatically performs the following steps during TSPITR:

1. Takes the tablespaces to be recovered offline.
2. Restores the datafiles to the auxiliary instance.
3. Recovers the restored datafiles to the specified time.
4. Opens the auxiliary database with the RESETLOGS option.
5. Exports the dictionary metadata about objects in the recovered tablespaces—the DDL to create the objects along with pointers to the physical locations of those in the recovered datafiles—to the target database.
6. Shuts down the auxiliary instance.
7. Issues SWITCH commands so that the target control file now points to the datafiles in the recovery set that were just recovered at the auxiliary database.
8. Imports the dictionary metadata that was exported from the auxiliary instance, allowing the recovered objects to be accessed.

Note: RMAN attempts to find datafile copies instead of restoring the datafiles being recovered. If it finds none, then it performs a restore operation and does not execute a switch. If you have configured names for the datafiles with the CONFIGURE AUXNAME command, and suitable datafile copies exist in those AUXNAME locations, then RMAN optimizes away the restore and performs a switch to the recovered AUXNAME datafile copy.

Preparing the Target Database for Use After RMAN TSPITR

The tablespaces in the recovery set remain offline until after RMAN TSPITR completes successfully.

To prepare the target database for reuse after TSPITR:

1. Start RMAN and connect to the target database. For example, run:

   ```
   % rman TARGET SYS/oracle@trgt
   ```

2. Make backups of tablespaces in the recovery set before bringing these tablespaces online. For example, makes a new backup of tablespace `users`:

   ```
   BACKUP TABLESPACE users;
   ```

   > **Note:** If you are running in `NOCATALOG` mode, then you cannot perform TSPITR on a tablespace and then perform it again on the same tablespace to a time equal to or *before* the TSPITR time. However, after TSPITR completes and you make a new backup of the tablespace, you can perform TSPITR on the tablespace to any time *after* the backup.

3. Bring the recovery set tablespaces online. For example, enter:

   ```
   SQL "ALTER TABLESPACE users ONLINE";
   ```

4. Connect to the auxiliary instance with SQL*Plus and shut down the auxiliary instance. For example:

   ```
   sqlplus 'SYS/oracle@aux AS SYSDBA'<<EOF
   SHUTDOWN ABORT;
   EXIT;
   EOF
   ```

5. Delete the following files from the operating system:

   - Auxiliary set datafiles restored to temporary locations during TSPITR
   - Auxiliary database control files
   - Auxiliary database redo log files

Responding to Unsuccessful RMAN TSPITR

A variety of problems can cause TSPITR to fail. For example, if there is a conflict between the target database and the converted filename, then you have to shut down the auxiliary instance, correct the converted datafile name, issue a `STARTUP NOMOUNT`, and then run RMAN TSPITR again.

Another possible cause for failure is a lack of sufficient sort space for the Export utility. In this case, you need to edit the `recover.bsq` file (on UNIX, it is located in `$ORACLE_HOME/rdbms/admin`). This file contains the following:

```
#
```

```
# tsiptr_7: do the incomplete recovery and resetlogs.  This member is used once.
#
define tspitr_7
<<<
# make the control file point at the restored datafiles, then recover them
recover clone database tablespace &1&;
alter clone database open resetlogs;
# PLUG HERE the creation of a temporary tablespace if export fails due to lack of
# temporary space.
# For example in Unix these two lines would do that:
#sql clone "create tablespace aux_tspitr_tmp
#           datafile ''/tmp/aux_tspitr_tmp.dbf'' size 500K";
}
>>>
```

Remove the '#' symbols from the last two lines of comments and modify the statement to create a temporary tablespace. Retry the TSPITR operation, increasing the size of the tablespace until the export operation succeeds. If TSPITR is unsuccessful for some reason, then use the following procedure.

To respond to unsuccessful TSPITR:

1. If RMAN TSPITR is unsuccessful, then shut down the auxiliary instance:

   ```
   SHUTDOWN ABORT;
   ```

2. Identify and correct the error.
3. Start the auxiliary instance without mounting it. For example, enter:

   ```
   STARTUP NOMOUNT PFILE=initAUX.ora;
   ```

4. Perform TSPITR again, following the instructions in "Performing RMAN TSPITR" on page 11-17.

12 Duplicating a Database with Recovery Manager

This chapter describes how to use the DUPLICATE command to create a duplicate database for testing purposes. This chapter contains these topics:

- Creating a Duplicate Database: Overview
- Generating Files for the Duplicate Database
- Preparing the Auxiliary Instance for Duplication: Basic Steps
- Creating a Duplicate Database on a Local or Remote Host
- Database Duplication Examples

See Also: *Oracle9i Data Guard Concepts and Administration* to learn how to create a standby database using the DUPLICATE command

Creating a Duplicate Database: Overview

You can use the RMAN `DUPLICATE` command to create a **duplicate database** from target database backups while still retaining the original target database. The duplicate database can be either identical to the original database or contain only a subset of the original tablespaces. A duplicate database is a copy of the target database that you can run independently for a variety of purposes. For example, you can use it to:

- Test backup and recovery procedures
- Export data such as a table that was inadvertently dropped from the production database, and then importing 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 restore and recovery scenarios while the production database on `host1` continues as usual.

A duplicate database is distinct from a standby database, although both types of databases are created with the `DUPLICATE` command. A standby database is a copy of the primary database that you can update continually or periodically by using archived logs from the primary database. If the primary database is damaged or destroyed, then you can perform failover to the standby database and effectively transform it into 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:** Chapter 13, "Creating a Standby Database with Recovery Manager" to learn how to create a standby database with the `DUPLICATE` command

How Recovery Manager Duplicates a Database

To prepare for database duplication, you must first create an **auxiliary instance** as described in "Preparing the Auxiliary Instance for Duplication: Basic Steps" on page 12-9. For the duplication to work, you must connect RMAN to both the target (primary) database and an auxiliary instance started in `NOMOUNT` mode.

You must have at least one auxiliary channel allocated on the auxiliary instance. The principal work of the duplication is performed by the auxiliary channel, which starts a server session on the duplicate host. This channel then restores the necessary backups of the primary database, uses them to create the duplicate database, and initiates recovery.

So long as RMAN is able to connect to the primary and duplicate instances, the RMAN client can run on any machine. However, all backups, copies of datafiles, and archived logs used for creating and recovering the duplicate database must be accessible by the server session on the duplicate host. If the duplicate host is different from the primary host, then you must make backups and copies on the primary host disks available to the remote node with the same full path name as in the primary database. You can accomplish this goal in two ways:

- Using an operating system utility to move the backup pieces, image copies, and archived logs from the primary host to the remote host to an identical path
- Using NFS or shared disks and making sure that the same path is accessible in the remote host

In the case of tape backups, you must make the tapes containing the backups accessible to the remote node, either by physically moving the tape to the remote host or by using a network tape server.

As part of the duplicating operation, RMAN manages the following:

- Restores the target datafiles to the duplicate database and performs incomplete recovery by using all available incremental backups and archived logs
- Shuts down and starts the auxiliary database (refer to "Task 4: Start the Auxiliary Instance" on page 12-11 for issues relating to client-side versus server-side initialization parameter files)
- Opens the duplicate database with the `RESETLOGS` option after incomplete recovery to create the online redo logs (except when running `DUPLICATE ... FOR STANDBY`, in which case RMAN does not open the database)
- Generates a new, unique DBID for the duplicate database (except when you create a *standby* database with `DUPLICATE ... FOR STANDBY`, in which case RMAN does *not* create a unique DBID)

During duplication, RMAN *must* perform incomplete recovery because the online redo logs in the target 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 target database.

See Also: Chapter 13, "Creating a Standby Database with Recovery Manager" to learn how to create a standby database with RMAN

Database Duplication Options

When duplicating a database, you can do the following:

- Run the `DUPLICATE` command with or without a recovery catalog
- Skip read-only tablespaces with the `SKIP READONLY` clause. Read-only tablespaces are included by default. If you omit them, then you can add them later.
- Exclude tablespaces from the duplicate database with the `SKIP TABLESPACE` clause. You cannot skip the `SYSTEM` tablespace or tablespaces containing rollback or undo segments.
- Create the duplicate database in a new host. If the directory structure is the same on the new host, then you can specify the `NOFILENAMECHECK` option and reuse the target datafile filenames for the duplicate datafiles.
- Use the `SET UNTIL` command or `DUPLICATE` command with the `UNTIL` clause when creating the duplicate database to recover it to a noncurrent time. By default, the `DUPLICATE` command creates the database by using the most recent backups of the target database and then performs recovery to the most recent consistent point contained in the incremental backups and archived logs.
- Register the duplicate database in the same recovery catalog as the target database. This option is possible because RMAN gives the duplicate database a new DBID during duplication.

> **Note:** If you copy the target database by means of operating system utilities, 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 *Oracle9i Database Utilities*).

- Set the duplicate database `DB_NAME` differently from the target database `DB_NAME` in some cases. If the duplicate database exists in the same Oracle home as the target, then the `DB_NAME` initialization parameter must be different. If the duplicate database resides in a different Oracle home (on the same machine or on another machine), then its `DB_NAME` setting just has to be different from other database names in the same Oracle home on the duplicate host.

Obeying Duplication Prerequisites and Restrictions

RMAN duplication involves a number of prerequisites, restrictions, and caveats. Review the restrictions section of the `DUPLICATE` syntax entry before attempting duplication (see *Oracle9i Recovery Manager Reference* for `DUPLICATE` syntax).

> **See Also:** *Oracle9i Recovery Manager Reference* for `DUPLICATE` syntax

Generating Files for the Duplicate Database

When duplicating a database, RMAN creates the required database files. This section describes these stages of file creation:

- Creating the Duplicate Control Files
- Creating the Duplicate Online Redo Logs
- Naming the Duplicate Datafiles

Creating the Duplicate Control Files

The `DUPLICATE` command creates the control files by using the names listed in the initialization parameter file of the auxiliary instance. When choosing names for the duplicate database control files, make sure that you set the initialization parameter settings correctly so that you do not overwrite the production files at the target database.

Creating the Duplicate Online Redo Logs

Table 12–1 lists the options for creating the names of the duplicate online redo logs. The options appear in the order of precedence.

Table 12–1 Order of Precedence for Redo Log Filename Creation

Order	Method	Result
1	Specify the `LOGFILE` clause of `DUPLICATE` command.	Creates redo logs as specified.

Table 12–1 Order of Precedence for Redo Log Filename Creation

Order	Method	Result
2	Set `LOG_FILE_NAME_CONVERT` initialization parameter.	Transforms target filenames, for example, from `log_*` to `duplog_*`. Note that you can specify multiple conversion pairs. This parameter allows the redo log to exist as long as the size matches, because it uses the `REUSE` parameter when creating the logs.
3	Do none of the preceding steps.	Makes the duplicate filenames the same as the target filenames. You must specify the `NOFILENAMECHECK` option when using this method and the duplicate database should be in a different host.

The order of precedence determines how RMAN renames the online redo logs. For example, if you specify both the `LOGFILE` clause and the `LOG_FILE_NAME_CONVERT` parameter, then RMAN uses the `LOGFILE` clause. If you specify neither of the first two options, then RMAN uses the original target redo log filenames for the duplicate database files.

Caution: If the target and duplicate databases are in the same host, then do not use the name of an online redo log currently in use by the target database. Also, do not use the name of an online log currently in use by the target database if the duplicate database is in a different host and `NOFILENAMECHECK` is not used.

Naming the Duplicate Datafiles

To have different filenames for the duplicate datafiles, you must use parameters or commands to specify them. Table 12–2 lists the options for renaming datafiles. The options appear in the order of precedence.

Table 12–2 Order of Precedence for Datafile Filename Creation

Order	Method	Result
1	Issue `SET NEWNAME` command.	Creates new datafile filenames. You must reissue this command each time you rename files.

Table 12–2 Order of Precedence for Datafile Filename Creation

Order	Method	Result
2	Issue CONFIGURE AUXNAME command.	Creates new datafile filenames. This setting stays in effect until disabled with a CONFIGURE AUXNAME ... CLEAR command.
3	Set DB_FILE_NAME_CONVERT initialization parameter.	Transforms target filenames, for example, from /oracle/ to /dup/oracle/. Note that you can specify multiple conversion pairs. You can use this parameter for those files not renamed with either SET NEWNAME and CONFIGURE AUXNAME.
4	Do none of the preceding steps.	Reuses the target filenames. You must specify the NOFILENAMECHECK option when using this method and the duplicate database should be in a different host.

The order of precedence determines how RMAN names the files. For example, if you specify all the commands and the initialization parameter, then RMAN uses SET NEWNAME. If you run CONFIGURE AUXNAME and DB_FILE_NAME_CONVERT, then RMAN uses CONFIGURE AUXNAME. If you do not specify any of the first three options, then RMAN uses the original target filenames for the duplicate file.

Skipping Read-Only Tablespaces

When you specify SKIP READONLY, RMAN does not duplicate the datafiles of read-only tablespaces. After duplication is complete, you can query the views in the duplicate database described in Table 12–3 and Table 12–4 to determine whether datafiles are read-only. The STATUS and ENABLED columns are the key to determining the current status of the duplicate datafile.

Table 12–3 Read-Only Tablespaces in V$DATAFILE View on Duplicate Database

In the column ...	The value is ...
STATUS	OFFLINE
ENABLED	READ ONLY
NAME	MISSINGxxx

Table 12–4 Read-Only Tablespaces in Data Dictionary Views on Duplicate Database

View	In the column ...	The value is ...
DBA_DATA_FILES	STATUS	AVAILABLE
DBA_TABLESPACES	STATUS	READ ONLY

Skipping OFFLINE NORMAL Tablespaces

When tablespaces are taken offline with the OFFLINE NORMAL option, RMAN does not duplicate the datafiles of these tablespaces. After duplication, you can manually add or drop these tablespaces. Query the views in the duplicate database described in Table 12–5 and Table 12–6 to determine whether datafiles are offline. The STATUS and ENABLED columns are the key to determining the current status of the datafile.

Table 12–5 Offline Tablespaces in V$ Views on Duplicate Database

In the column ...	The value is ...
STATUS	OFFLINE
ENABLED	DISABLED
NAME	MISSINGxxx

Table 12–6 Offline Tablespaces in Data Dictionary Views on Duplicate Database

View	In the column ...	The value is ...
DBA_DATA_FILES	STATUS	AVAILABLE
DBA_TABLESPACES	STATUS	OFFLINE

When you take a tablespace offline with the IMMEDIATE option, RMAN duplicates rather than skips the tablespace because this tablespace requires recovery. As with online tablespaces, RMAN requires a valid backup for duplication.

Preventing Filename Checking

It is possible for a CONFIGURE AUXNAME, SET NEWNAME, or DB_FILE_NAME_CONVERT to generate a name that is already in use in the target database. In this case, specify NOFILENAMECHECK to avoid an error message. For example, assume that the host A database has two files: datafile 1 is named /oracle/data/file1.f and datafile 2 is named /oracle/data/file2.f. When duplicating to host B, you use a configured channel to duplicate as follows:

```
RUN
{
  SET NEWNAME FOR DATAFILE 1 TO /oracle/data/file2.f; # rename datafile 1 as file2.f
  SET NEWNAME FOR DATAFILE 2 TO /oracle/data/file1.f; # rename datafile 2 as file1.f
  DUPLICATE TARGET DATABASE TO newdb;
}
```

Even though you issued `SET NEWNAME` commands for all the datafiles, the `DUPLICATE` command fails because the duplicate filenames are still in use in the target database. Although datafile `1` in the target is not using `/oracle/data/file2.f`, and datafile `2` in the target is not using `/oracle/data/file1.f`, the target filename is used by one of the duplicate datafiles and so you must specify `NOFILENAMECHECK` to avoid an error.

Note: Only use `DB_FILE_NAME_CONVERT` by itself, that is, without also using either `SET NEWNAME` or `CONFIGURE AUXNAME`, if all the filenames are converted by this parameter. In Oracle9*i*, you can specify multiple conversion pairs with this initialization parameter. For example, you can set the parameter to convert `/dsk1/dbs` to `/sby1/dbs` and `/dsk2/df` to `/sby2/dbs`.

Preparing the Auxiliary Instance for Duplication: Basic Steps

Perform these tasks before performing RMAN duplication:

- Task 1: Create an Oracle Password File for the Auxiliary Instance
- Task 2: Ensure Oracle Net Connectivity to the Auxiliary Instance
- Task 3: Create an Initialization Parameter File for the Auxiliary Instance
- Task 4: Start the Auxiliary Instance
- Task 5: Mount or Open the Target Database
- Task 6: Make Sure You Have the Necessary Backups and Archived Redo Logs
- Task 7: Allocate Auxiliary Channels if Automatic Channels Are Not Configured

Task 1: Create an Oracle Password File for the Auxiliary Instance

For instructions on how to create and maintain Oracle password files, refer to *Oracle9i Database Administrator's Guide*.

Task 2: Ensure Oracle Net Connectivity to the Auxiliary Instance

The auxiliary instance must be accessible through Oracle Net. Before proceeding, start SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

Task 3: Create an Initialization Parameter File for the Auxiliary Instance

Create a client-side initialization parameter file for the auxiliary instance, and set at least the parameters described in the following table.

Parameter	You must specify:
DB_NAME	The same name used in the DUPLICATE command. You must set the DB_NAME parameter in the duplicate parameter file to the same database name specified in the DUPLICATE command. You cannot use the same database name for the target and duplicate when the duplicate is in the same Oracle home as the target. If the duplicate is in a different Oracle home from the target, then its DB_NAME just has to differ from other database names in that same Oracle home.
CONTROL_FILES	Refer to "Creating the Duplicate Control Files" on page 12-5.

Optionally, set the parameters described in the following table.

Parameter	You must specify:
DB_FILE_NAME_CONVERT	Refer to "Naming the Duplicate Datafiles" on page 12-6.
LOG_FILE_NAME_CONVERT	Refer to "Creating the Duplicate Online Redo Logs" on page 12-5.

Set other 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 special attention to all parameters specifying path names. Verify that all paths are accessible on the host where the database is being duplicated.

Following are examples of the initialization parameter settings for the duplicate database:

```
DB_NAME=newdb
CONTROL_FILES=(/dup/oracle/oradata/trgt/control01.ctl,
               /dup/oracle/oradata/trgt/control02.ctl)
DB_FILE_NAME_CONVERT=(/oracle/oradata/trgt/,/dup/oracle/oradata/trgt/)
LOG_FILE_NAME_CONVERT=(/oracle/oradata/trgt/redo,/dup/oracle/oradata/trgt/redo)
```

After you create the client-side initialization parameter file, you can run the CREATE SPFILE command from SQL*Plus to create a server-side initialization parameter file. You can run this command before or after instance startup. For example, you can create a server-side parameter file in the default location as follows, specifying the filename of the client-side initialization parameter file in the FROM clause:

```
CREATE SPFILE FROM PFILE='/tmp/initDUPDB.ora';
```

A server-side parameter file in the default location is an advantage when duplicating a database because you do not need to specify the PFILE parameter on the DUPLICATE command. Because RMAN shuts down and restarts the auxiliary instance as part of the duplication process, you must tell RMAN which client-side file to use if you use a client-side parameter file. It is highly recommended that you create a server-side parameter file for use in database duplication.

Task 4: Start the Auxiliary Instance

Before beginning RMAN duplication, use SQL*Plus to connect to the auxiliary instance and start it in NOMOUNT mode (specifying a client-side parameter file if necessary). In this example, oracle is the password for the user with SYSDBA authority and aux is the net service name for the auxiliary instance:

```
CONNECT SYS/oracle@aux AS SYSDBA
-- start instance with the server parameter file
STARTUP FORCE NOMOUNT
```

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 process. Hence, it is a good idea to create a server-side initialization parameter file for the auxiliary instance in the default location.

If you do *not* have a server-side initialization parameter file for the auxiliary instance in the default location, then you must specify the client-side initialization parameter file with the PFILE parameter on the DUPLICATE command. The client-side parameter file for the auxiliary instance must reside on the same host as the RMAN executable used to perform the duplication.

Task 5: Mount or Open the Target Database

Before beginning RMAN duplication, connect SQL*Plus to the target database and mount or open tit—specifying a client-side parameter file if necessary—if it is not already mounted or open. For example, enter:

```
-- connect to target database
CONNECT SYS/oracle@trgt
-- mount or open target database
STARTUP
```

Task 6: Make Sure You Have the Necessary Backups and Archived Redo Logs

Make sure backups all target datafiles are available on the duplicate host. If you do not have backups of everything, then the duplicate operation fails. The database backup does not have to be a whole database backup: you can use a mix of full and incremental backups of individual datafiles.

Make sure that you meet either of the following conditions:

- You have backups of all the archived redo logs necessary to recover to the desired time, SCN, or log sequence number. These backups must be accessible by the node where the duplicate database is going to be created. For example, if you back up through a media manager, then ensure that the duplicate host can restore objects backed up on the target host.
- The archived redo logs are accessible on the node where the duplicate database is going to be created. Note that you can copy logs to the duplicate host manually or use NFS to mount log directories on the target host.

Task 7: Allocate Auxiliary Channels if Automatic Channels Are Not Configured

Start RMAN with a connection to the target database, the duplicate database, and (if you use one) the recovery catalog database. You can start the RMAN executable on any host so long as it can connect to all the instances. Note that if the duplicate instance requires a client-side initialization parameter file, then this file must exist on the same host that runs the RMAN executable.

In this example, a connection is established to three databases, all through the use of net service names:

```
% rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb AUXILIARY SYS/oracle@aux
```

If you do not have automatic channels configured, then before issuing the `DUPLICATE` command, manually allocate at least one auxiliary channel within the same `RUN` command. The channel type (`DISK` or `sbt`) must match the media where the backups of the target 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 for the operation.

```
RUN
{
  # to manually allocate a channel of type sbt issue:
  ALLOCATE AUXILIARY CHANNEL ch1 DEVICE TYPE sbt;

  # to manually allocate three auxiliary channels for disk issue (specifying whatever
  # channel id that you want):
  ALLOCATE AUXILIARY CHANNEL aux1 DEVICE TYPE DISK;
  ALLOCATE AUXILIARY CHANNEL aux2 DEVICE TYPE DISK;
  ALLOCATE AUXILIARY CHANNEL aux3 DEVICE TYPE DISK;
  .
  .
  .
  DUPLICATE ...
}
```

Note: If you configure automatic channels, then RMAN can use configured channels for duplication even if they do not specify the `AUXILIARY` option. Nevertheless, if the auxiliary channels need some special parameters (for example, to point to a different media management subsystem), then you can configure an automatic channel with the `AUXILIARY` option.

Creating a Duplicate Database on a Local or Remote Host

When you create a duplicate database, the procedure differs depending on your configuration. This section contains these topics:

- Duplicating a Database on a Remote Host with the Same Directory Structure
- Duplicating a Database on a Remote Host with a Different Directory Structure
- Creating a Duplicate Database on the Local Host

Duplicating a Database on a Remote Host with the Same Directory Structure

The simplest case is to duplicate the database to a different host and to use the same directory structure. In this case, you do not need to change the initialization parameter file or set new filenames for the duplicate datafiles.

1. Follow the steps in "Preparing the Auxiliary Instance for Duplication: Basic Steps" on page 12-9.
2. Run the DUPLICATE command, making sure to do the following:
   - If automatic channels are not configured, then allocate at least one auxiliary channel.
   - Specify the NOFILENAMECHECK parameter on the DUPLICATE command.
   - Specify the PFILE parameter if starting the auxiliary instance with a client-side parameter file. The client-side parameter file must exist on the same host as the RMAN executable used to perform the duplication.

   The following example assumes that the RMAN executable is running on the duplicate host. It duplicates the database with an automatic channel, specifies a client-side initialization parameter file, and specifies the NOFILENAMECHECK option:

   ```
   DUPLICATE TARGET DATABASE TO dupdb
     # specify client-side parameter file (on same host as RMAN executable) for
     # auxiliary instance if necessary
     PFILE = /dup/oracle/dbs/initDUPDB.ora
     NOFILENAMECHECK;
   ```

RMAN automatically allocates the configured channels, then uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. Finally, RMAN opens the database with the RESETLOGS option to create the online redo logs.

Duplicating a Database on a Remote Host with a Different Directory Structure

If you create the duplicate database on a host with a different file system, then you need to change several initialization parameters and generate new filenames for the duplicate database datafiles.

Use LOG_FILE_NAME_CONVERT or the LOGFILE clause to convert the online redo log filenames. Use DB_FILE_NAME_CONVERT, the SET NEWNAME command, or the CONFIGURE AUXNAME command for the datafile filenames.

This section contains these topics:

- Duplicating By Using Initialization Parameters
- Duplicating By Using Initialization Parameters and the LOGFILE Clause
- Duplicating By Using SET NEWNAME

- Duplicating By Using CONFIGURE AUXNAME

  See Also: Table 12–2 on page 12-6 for a table of the various datafile filename conversion options

Duplicating By Using Initialization Parameters

This procedure assumes that you set initialization parameters to rename the duplicate datafiles and log files.

1. Follow the steps in "Preparing the Auxiliary Instance for Duplication: Basic Steps" on page 12-9, making sure to use an operating system utility to copy the parameter file from its location in the target host directory structure to the same location in the duplicate host directory structure. Make sure to set:

   - All initialization parameters that end in `_DEST` or `_PATH` and specify a path name.
   - `DB_FILE_NAME_CONVERT` so that it captures *all* the target datafiles and converts them appropriately, for example, from `/oracle/oradata/` to `/dup/oracle/oradata/`.
   - `LOG_FILE_NAME_CONVERT` so that it captures *all* the online redo logs and converts them appropriately, for example, `/oracle/oradata/redo` to `/dup/oracle/oradata/redo`.

   Note: You can set multiple conversion pairs in `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT`. For example, you can specify that `DB_FILE_NAME_CONVERT` changes `/fs1` to `/dsk1` and `/fs2` to `/dsk2`.

2. Perform the following operations when running the duplication:

   - If automatic channels are not configured, then allocate at least one auxiliary channel.
   - If desired, specify the same number of redo log members and groups that are used in the target database.
   - If using a client-side parameter file to start the auxiliary instance, specify the `PFILE` parameter.

   The following example assumes that the duplicate host can access the same media manager as the primary database host. The example duplicates the database with an automatic sbt channel and uses a server-side parameter file located on the duplicate host to restart the auxiliary instance:

```
DUPLICATE
  TARGET DATABASE TO dupdb
  DEVICE TYPE sbt # restores from tape backups;
# Note that DUPLICATE DEVICE TYPE sbt works only if the sbt device is configured
# with CONFIGURE CHANNEL, CONFIGURE DEVICE TYPE, or CONFIGURE DEFAULT DEVICE.
```

RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. RMAN then shuts down, starts, and opens the database with the RESETLOGS option to create the online redo logs.

Duplicating By Using Initialization Parameters and the LOGFILE Clause

This procedure assumes that you set initialization parameters to rename the duplicate datafiles and the LOGFILE clause to specify names and sizes for the online redo logs.

1. Follow the procedure described in "Duplicating By Using Initialization Parameters" on page 12-15, but do not set the LOG_FILE_NAME_CONVERT parameter.
2. Perform the following operations when running the duplication:
   - If automatic auxiliary channels are not configured, then allocate at least one auxiliary channel.
   - Specify the names and sizes for the duplicate database redo logs in the LOGFILE clause.
   - Specify new filenames for the duplicate database datafiles.
   - If using a client-side parameter file to start the auxiliary instance, specify the PFILE parameter.

The following example duplicates the database with an automatic channel and specifies an initialization parameter file as well as the LOGFILE clause:

```
DUPLICATE TARGET DATABASE TO dupdb
  # specify client-side parameter file for auxiliary instance if necessary
  PFILE = /dup/oracle/dbs/initDUPDB.ora
  LOGFILE
    '/dup/oracle/oradata/trgt/redo01.log' SIZE 200K,
    '/dup/oracle/oradata/trgt/redo02.log' SIZE 200K,
    '/dup/oracle/oradata/trgt/redo03.log' SIZE 200K;
```

Duplicating By Using SET NEWNAME

This procedure assumes that you use the SET NEWNAME command to rename the duplicate datafiles.

1. Follow the steps in "Preparing the Auxiliary Instance for Duplication: Basic Steps" on page 12-9, making sure to use an operating system utility to copy the parameter file from its location in the target host directory structure to the same location in the duplicate host. Set all initialization parameters that end in _DEST or _PATH and specify a path name.
2. Perform the following operations when running the duplication:
   - If automatic auxiliary channels are not configured, then allocate at least one auxiliary channel.
   - If desired, specify the same number of redo log members and groups that are used in the target database.
   - Specify new filenames for the duplicate database datafiles.
   - If you use a client-side parameter file to start the auxiliary instance, then specify the PFILE parameter.

   The following example uses automatic channels and a default server-side initialization parameter file for the database duplication, and uses the LOGFILE clause to specify names and sizes for the online redo logs:

```
RUN
{
  # set new filenames for the datafiles
  SET NEWNAME FOR DATAFILE 1 TO '/dup/oracle/oradata/trgt/system01.dbf';
  SET NEWNAME FOR DATAFILE 2 TO '/dup/oracle/oradata/trgt/undotbs01.dbf';
  . . .
  # issue the duplicate command
  DUPLICATE TARGET DATABASE TO dupdb
  # create at least two online redo log groups
  LOGFILE
    GROUP1
    (
      '/dup/oracle/oradata/trgt/redo01a.log',
      '/dup/oracle/oradata/trgt/redo01b.log',
      '/dup/oracle/oradata/trgt/redo01c.log';
    ) SIZE 200K,
    GROUP2
    (
      '/dup/oracle/oradata/trgt/redo02a.log',
      '/dup/oracle/oradata/trgt/redo02b.log',
      '/dup/oracle/oradata/trgt/redo02c.log';
    ) SIZE 200K,
    GROUP3
    (
      '/dup/oracle/oradata/trgt/redo03a.log',
      '/dup/oracle/oradata/trgt/redo03b.log',
      '/dup/oracle/oradata/trgt/redo03c.log';
    ) SIZE 200K;
```

RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. RMAN shuts down, starts up, and then opens the database with the `RESETLOGS` option to create the online logs.

Duplicating By Using CONFIGURE AUXNAME

This procedure assumes that you use the `CONFIGURE AUXNAME` command to rename the duplicate datafiles.

1. Follow the steps in "Preparing the Auxiliary Instance for Duplication: Basic Steps" on page 12-9, making sure to use an operating system utility to copy the parameter file from its location in the target host directory structure to the same location in the duplicate host directory structure. Set all initialization parameters that end in `_DEST` or `_PATH` and specify a path name.
2. Perform the following operations during duplication:
   - If automatic auxiliary channels are not allocated, then allocate at least one auxiliary channel.
   - If desired, specify the same number of redo log members and groups that are used in the target database.
   - If you start the auxiliary instance with a client-side parameter file, then specify the `PFILE` parameter. The client-side parameter file must reside on the same host as the RMAN executable used to perform the duplication.

   The following example 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:

```
# run the DUPLICATE command
DUPLICATE TARGET DATABASE TO dupdb
# specify client-side parameter file for auxiliary instance if necessary
PFILE = /dup/oracle/dbs/initDUPDB.ora
.
.
.
# create at least two online redo log groups
 LOGFILE
```

```
GROUP1
(
  '/dup/oracle/oradata/trgt/redo01a.log',
  '/dup/oracle/oradata/trgt/redo01b.log',
  '/dup/oracle/oradata/trgt/redo01c.log';
) SIZE 200K,
GROUP2
(
  '/dup/oracle/oradata/trgt/redo02a.log',
  '/dup/oracle/oradata/trgt/redo02b.log',
  '/dup/oracle/oradata/trgt/redo02c.log';
) SIZE 200K,
GROUP3
(
  '/dup/oracle/oradata/trgt/redo03a.log',
  '/dup/oracle/oradata/trgt/redo03b.log',
  '/dup/oracle/oradata/trgt/redo03c.log';
) SIZE 200K;
```

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.

3. Clear the auxiliary names for the datafiles so that they are not overwritten by mistake. For example, enter the following:

```
# un-specify auxiliary names for the datafiles
CONFIGURE AUXNAME FOR DATAFILE 1 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
.
.
.
CONFIGURE AUXNAME FOR DATAFILE n CLEAR;
```

Creating a Duplicate Database on the Local Host

When creating a duplicate database on the same host as the target database, follow the same procedure as for duplicating to a remote host with a different directory structure as described in "Duplicating a Database on a Remote Host with a Different Directory Structure" on page 12-14.

Note that you can duplicate the database to the same Oracle home as the target database, but you must 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 generate an error.

Database Duplication Examples

This section contains these topics:

- Setting New Filenames Manually: Example
- Resynchronizing the Duplicate Database with the Target Database: Example
- Creating a Non-Current Duplicate Database: Example
- Duplication with a Client-Side Parameter File: Example

Setting New Filenames Manually: Example

This example assumes the following:

- You are using recovery catalog database catdb.
- The target database trgt is on host1 and contains eight datafiles.
- You want to duplicate the target to database dupdb on remote host host2.
- host1 and host2 use different file systems.
- You want to store the datafiles in host2 in the ?/oradata/dup/ subdirectory.
- You want to exclude tablespace tools from the duplicate database, but keep all of the other tablespaces.
- You have used 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 or _PATH and specify a path name.
- You have disk copies or backup sets stored on disk for all the datafiles and archived redo logs in the target database, and you have manually moved them to host2 by means of an operating system utility.
- You want two online redo logs groups, each with two members of size 200 KB.
- You have configured the default device to sbt. The media management device is accessible by host2.

- 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).

```
CONNECT TARGET;
CONNECT CATALOG rman/cat@catdb;
CONNECT AUXILIARY SYS/oracle@dupdb;

# note that a RUN command is necessary because you can only execute SET NEWNAME
# within a RUN command
RUN
{
  # the DUPLICATE command uses an automatic sbt channel
  SET NEWNAME FOR DATAFILE 1 TO '?/oradata/dup/system01.dbf';
  SET NEWNAME FOR DATAFILE 2 TO '?/oradata/dup/undotbs01.dbf';
  SET NEWNAME FOR DATAFILE 3 TO '?/oradata/dup/cwmlite01.dbf';
  SET NEWNAME FOR DATAFILE 4 TO '?/oradata/dup/drsys01';
  SET NEWNAME FOR DATAFILE 5 TO '?/oradata/dup/example01.dbf';
  SET NEWNAME FOR DATAFILE 6 TO '?/oradata/dup/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 '?/oradata/dup/users01.dbf';
  DUPLICATE TARGET DATABASE TO dupdb
    SKIP TABLESPACE tools
    LOGFILE
      GROUP 1 ('?/oradata/dup/redo01a.log',
               '?/oradata/dup/redo01b.log') SIZE 200K REUSE,
      GROUP 2 ('?/oradata/dup/redo02a.log',
               '?/oradata/dup/redo02b.log') SIZE 200K REUSE;
}
```

Resynchronizing the Duplicate Database with the Target Database: Example

This example makes the same assumptions as in "Setting New Filenames Manually: Example" on page 12-20. Additionally, it assumes that you want to update the duplicate database daily so that it stays current with the target database.

```
# start RMAN and then connect to the databases
CONNECT TARGET;
CONNECT CATALOG rman/cat@catdb;
CONNECT AUXILIARY SYS/oracle@dupdb;

# configure auxiliary names for the datafiles only once
CONFIGURE AUXNAME FOR DATAFILE 1 TO '?/oradata/dup/system01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 2 TO '?/oradata/dup/undotbs01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 3 TO '?/oradata/dup/cwmlite01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 4 TO '?/oradata/dup/drsys01';
CONFIGURE AUXNAME FOR DATAFILE 5 TO '?/oradata/dup/example01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 6 TO '?/oradata/dup/indx01.dbf';
# do not configure an auxiliary name for datafile 7, because it is in the tools
# tablespace, and you are excluding tools from the duplicate database
CONFIGURE AUXNAME FOR DATAFILE 8 TO '?/oradata/dup/users01.dbf';

# Create the duplicate database. Issue the same command daily
# to re-create the database, thereby keeping the duplicate
# in sync with the target.
DUPLICATE TARGET DATABASE TO dupdb
SKIP TABLESPACE tools
  LOGFILE
    GROUP 1 ('?/oradata/dup/redo01a.log',
             '?/oradata/dup/redo01b.log') SIZE 200K REUSE,
    GROUP 2 ('?/oradata/dup/redo02a.log',
             '?/oradata/dup/redo02b.log') SIZE 200K REUSE;
```

Creating a Non-Current Duplicate Database: Example

This duplication example assumes the following:

- The target database `trgt` and duplicate database `dupdb` are on different hosts but have exactly the same directory structure.
- You want to name the duplicate database files the same as the target files.
- You are not using a recovery catalog.
- You do not have automatic channels configured.
- You want to recover the duplicate database to one week ago in order to view the data in `prod1` as it appeared at that time.

```
CONNECT TARGET SYS/oracle@trgt
CONNECT AUXILIARY SYS/oracle@dupdb
RUN
{
  SET UNTIL TIME 'SYSDATE-7';
  ALLOCATE AUXILIARY CHANNEL dupdb1 DEVICE TYPE sbt;
  DUPLICATE TARGET DATABASE TO dupdb
  NOFILENAMECHECK;
}
```

Duplication with a Client-Side Parameter File: Example

If you use a client-side initialization parameter file to start the auxiliary instance, then it must reside on the same host as the RMAN executable used to perform the duplication. Assume the following scenario:

- The target host is `host_tar` and the duplicate host is `host_dup`

- The client-side initialization parameter file on `host_dup` is named `?/dbs/initTEST.ora`.
- The hosts `host_dup` and `host_tar` are linked by a network.

In this scenario, you can run the RMAN executable (that is, run the `DUPLICATE` command in an RMAN session) either on `host_tar` or `host_dup`.

Running RMAN from host_dup

If you run the executable on `host_dup`, you can duplicate the database as follows:

```
DUPLICATE
  TARGET DATABASE TO dupdb
  DEVICE TYPE sbt
  PFILE='?/dbs/initTEST.ora';
```

Because the initialization parameter file used by the auxiliary instance resides on the same node as the RMAN executable, you can reference the local filename of the parameter file.

Running RMAN from host_tar

In this scenario, you run RMAN on the same host as the target database rather than on the host with the duplicate database. Hence, the client-side initialization parameter file needed by the `DUPLICATE` command is not located on the same node as the RMAN executable. You must do one of the following:

- Transfer the parameter file from `host_dup` to `host_tar`.
- Use NFS to mount the `host_dup` directory containing the parameter file on the `host_tar` file system.

Copying the Parameter File from host_dup to host_tar In this scenario, you manually copy the file from one host to another as follows:

```
% cp /net/host_dup/oracle/dbs/initTEST.ora /net/host_tar/tmp
```

Then, you can start RMAN on `host_tar` and perform the duplication with the following shell script:

```
#!/usr/bin/sh
rman TARGET SYS/oracle@trgt AUXILIARY SYS/oracle@dupdb <<EOF
DUPLICATE
  TARGET DATABASE TO dupdb
  DEVICE TYPE sbt
  PFILE='/tmp/initTEST.ora';
EOF
```

Mounting the host_dup File System on host_tar In this scenario, you mount the `host_dup` file system on the `host_tar` file system by using `/tmp` as the mount point. The `/tmp/initTEST.ora` filename on `host_tar` points to the `?/dbs/initTEST.ora` file residing on `host_dup`. Then, you can run the duplication as follows:

```
DUPLICATE
  TARGET DATABASE TO dupdb
  DEVICE TYPE sbt
  PFILE='/tmp/initTEST.ora';
```

13 Creating a Standby Database with Recovery Manager

This chapter describes how to create a standby database using RMAN. This chapter contains these topics:

- Preparing a Standby Database with RMAN
- Creating a Standby Database with RMAN: Overview
- Starting RMAN and the Standby Instance
- Creating a Standby Database on a Remote Host with the Same Directory Structure
- Creating a Standby Database on a Remote Host with a Different Directory Structure
- Creating a Standby Database on the Local Host
- Creating a Standby Database with Image Copies

> **See Also:** "Backing Up Files at a Standby Database Site with RMAN" on page 9-20

Preparing a Standby Database with RMAN

The procedure for preparing a standby database with RMAN is basically the same as for preparing a duplicate database. Nevertheless, you need to amend the duplication procedures described in "Creating a Duplicate Database on a Local or Remote Host" on page 12-13 to account for the issues specific to a standby database.

The documentation for the preparation and maintenance of standby databases is located in *Oracle9i Data Guard Concepts and Administration*. Familiarize yourself with what a standby database is and how to create one *before* you attempt the RMAN creation procedures in this chapter.

This section contains these topics:

- About Standby Database Preparation Using RMAN
- Creating the Standby Control File with RMAN
- Making Image Copies of Standby Control Files
- Naming the Standby Database Datafiles When Using RMAN
- Naming the Standby Database Online Redo Logs When Using RMAN

About Standby Database Preparation Using RMAN

You can use either manual methods or the Recovery Manager `DUPLICATE` command to create a standby database from backups of your primary database. Before you perform the creation procedure, you must prepare the standby instance. You can use RMAN to do the preparation tasks described in Table 13–1.

Table 13–1 Standby Database Preparation Using RMAN

Task	**Procedure**
Make a backup of the primary database to use for creation of standby database.	Use the normal backup procedure for your primary database as documented in *Oracle9i Recovery Manager User's Guide*.
Create a backup of the primary control file that is usable as a standby control file (if you do not have one).	"Creating the Standby Control File with RMAN" on page 13-3
Choose filenames for the standby datafiles.	"Naming the Standby Database Datafiles When Using RMAN" on page 13-6

Table 13–1 Standby Database Preparation Using RMAN

Task	Procedure
Choose filenames for the standby database online redo logs.	"Naming the Standby Database Online Redo Logs When Using RMAN" on page 13-8

You cannot use RMAN to perform all necessary standby database preparation. You must manually perform these tasks:

- Set all necessary initialization parameters in the primary initialization parameter file.
- Create an initialization parameter file for the standby database and configure all necessary parameters.
- Perform any Oracle Net setup and configuration required to connect to the standby instance.
- Start the standby instance without mounting the control file.

> **See Also:** *Oracle9i Data Guard Concepts and Administration* for a complete discussion of standby database preparation, including initialization parameter settings. You must perform all necessary preparation tasks described in this document before RMAN can successfully create the standby database files and mount the standby database.

Creating the Standby Control File with RMAN

In releases prior to release 8.1.7, you were required to create the standby control file with the SQL ALTER DATABASE statement. Now, you can use RMAN to make a special backup of the primary database control file that is usable as a standby database control file.

By using RMAN, you can create a standby control file in any of the ways described in the following sections:

- Creating the Standby Control File with the BACKUP Command
- Creating the Standby Control File with the COPY Command
- Creating the Standby Control File by Cataloging a SQL-Generated Control File

Creating the Standby Control File with the BACKUP Command

You have these options for creating the standby control file with the BACKUP command:

- You can create a standby control file by connecting to the primary database and running the BACKUP CURRENT CONTROLFILE command with the FOR STANDBY option.
- You can create a standby control file by connecting to the primary database and running the BACKUP ... INCLUDE CURRENT CONTROLFILE command with the FOR STANDBY option.

To create a backup set containing only a standby control file:

1. Connect to the primary database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
   ```

2. Mount or open the primary database. For example, enter:

   ```
   STARTUP MOUNT
   ```

3. Create the standby control file with the BACKUP CURRENT CONTROLFILE FOR STANDBY command. This example uses a configured channel to create the standby control file, then archives all unarchived logs and backs up any logs that have not yet been backed up at least once:

   ```
   BACKUP CURRENT CONTROLFILE FOR STANDBY;
   SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';  # so backup is consistent and recoverable
   BACKUP ARCHIVELOG ALL NOT BACKED UP 1 TIMES;
   ```

   You cannot specify a tag for a standby control file.

4. If desired, issue a LIST command to see a listing of backup sets and pieces.

To include the standby control file within another backup:

1. Connect to the primary database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
   ```

2. Back up the primary database and include a standby control file in the primary database backup. This example backs up all the primary datafiles to disk and also creates the standby control file:

   ```
   BACKUP DATABASE
     INCLUDE CURRENT CONTROLFILE FOR STANDBY
     PLUS ARCHIVELOG;
   ```

Note that PLUS ARCHIVELOG is specified so that RMAN will archive all unarchived logs before and after running BACKUP ARCHIVELOG ALL. If you do not specify PLUS ARCHIVELOG, you should run the following command after the backup of the database:

```
SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
```

3. Issue a LIST command to see a listing of backup sets and pieces.

Creating the Standby Control File with the COPY Command

You can create a standby control file by using the COPY CURRENT CONTROLFILE command with the FOR STANDBY option.

To create a control file copy that is usable as a standby control file:

1. Connect to the primary database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
   ```

2. Copy the current primary control file. Specify the FOR STANDBY option of the COPY CURRENT CONTROLFILE command to make a copy of the current control file that is usable as a standby control file. For example, enter:

   ```
   COPY CURRENT CONTROLFILE FOR STANDBY TO '/tmp/sby_control01.ctl';
   ```

3. Issue a LIST COPY command to see a listing of image copies.

Creating the Standby Control File by Cataloging a SQL-Generated Control File

You can update the repository to include a standby control file that was generated with the ALTER DATABASE statement .

To catalog a standby control file generated with ALTER DATABASE:

1. Create a standby control file with the SQL ALTER DATABASE statement (if you have not already created one). This example creates a standby control file by using SQL*Plus:

   ```
   SQL> ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/tmp/sby_control01.ctl';
   ```

2. Connect to the primary database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
   ```

3. run the CATALOG command to add metadata about the standby control file to the recovery catalog. For example, enter:

   ```
   CATALOG CONTROLFILECOPY '/tmp/sby_control01.ctl';
   ```

   RMAN considers a control file generated with the ALTER DATABASE statement as a control file copy.

Making Image Copies of Standby Control Files

RMAN can make an image copy of a control file copy that was generated by either:

- The RMAN COPY command
- The SQL ALTER DATABASE statement

To copy an RMAN-generated control file copy or SQL-generated control file:

1. Connect to the primary database and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
   ```

2. Copy the standby control file with the COPY CONTROLFILECOPY command. For example, run the following command:

   ```
   COPY CONTROLFILECOPY '/tmp/sby_control01.ctl' TO '/backup/sby_control01.ctl';
   ```

   Note: RMAN treats SQL-generated control files and RMAN-generated standby control file copies exactly the same. It inspects the header of the control file and determines whether it is a standby or a normal control file.

3. Issue a LIST COPY command to see a listing of image copies.

Naming the Standby Database Datafiles When Using RMAN

A standby database can reside either on the same host as the primary database or on a different host. The following table illustrates the implications for renaming the standby database datafiles depending on whether the directory structures on the hosts are the same or different.

Standby Database Host	Directory Structure	Renaming
Same host as primary	Different from primary host	Necessary
Same host as primary	Same as primary host	Illegal. The standby database cannot exist in the same directories as the primary database on the same host.
Different host from primary	Same as primary host	Not necessary
Different host from primary	Different from primary host	Necessary

When the directory structures are *different* for the primary and standby hosts, you have these options for naming the standby datafiles:

- Configuring the standby database initialization parameter DB_FILE_NAME_CONVERT
- By using the CONFIGURE AUXNAME or SET NEWNAME commands in RMAN when creating the standby database

When the directory structures are the *same* for the primary and standby hosts, then you have these naming options:

- Leaving the standby filenames the same as the primary filenames (that is, not setting DB_FILE_NAME_CONVERT or issuing a CONFIGURE AUXNAME or SET NEWNAME command) and specifying the NOFILENAMECHECK option of the DUPLICATE command
- By using the DB_FILE_NAME_CONVERT parameter, or the CONFIGURE AUXNAME or SET NEWNAME commands to rename the standby datafiles

Note that when you use DB_FILE_NAME_CONVERT, the format is as follows:

```
DB_FILE_NAME_CONVERT = ('oldstring1', 'newstring1', 'oldstring2', 'newstring2', ...)
```

For example, you can specify the DB_FILE_NAME_CONVERT initialization parameter as follows:

```
DB_FILE_NAME_CONVERT = ('/dbs/t1/', '/dbs/t1/s_', '/dbs/t2/', '/dbs/t2/s_')
```

Because you can specify datafile filenames in the standby control file in multiple ways, a method for prioritizing settings is necessary. Table 13–2 specifies the hierarchy for the naming of datafiles in the standby database.

Table 13–2 Order of Precedence for Naming Datafiles in Standby Database

	Method of Standby Datafile Naming	Requirement
1	Issue `CONFIGURE AUXNAME` command.	You must be connected to a recovery catalog, and an `AUXNAME` that is not `NULL` must be stored in the catalog for the datafile.
2	Issue `SET NEWNAME` command.	You must issue this command in the `RUN` block for the creation of the standby database.
3	Datafile filename as currently specified in the standby control file. The standby filename is identical to the primary filename or is named with the `DB_FILE_NAME_CONVERT` parameter.	If the filename is different, then the `DB_FILE_NAME_CONVERT` parameter must be set in the standby initialization parameter file. If the filename is the same, then you must specify the `NOFILENAMECHECK` clause of the `DUPLICATE` command.

See Also: *Oracle9i Data Guard Concepts and Administration* for more information about how to use `DB_FILE_NAME_CONVERT` to name standby files

Naming the Standby Database Online Redo Logs When Using RMAN

RMAN does not create the online redo logs when it creates the standby database. Online redo logs are not created on the standby database by RMAN. However, as described in *Oracle9i Data Guard Concepts and Administration*, the online logs can be created by other actions that you perform on the standby database. After the online logs are created, they are maintained and archived according to the normal rules for online redo logs.

The only option when naming the online redo logs on the standby database is the filename for the logs as specified in the standby control file. If the standby online log filenames must be different from the primary online log filenames, then one option is to specify filenames for the online redo logs by setting `LOG_FILE_NAME_CONVERT` in the standby initialization parameter file.

Note these restrictions when specifying filenames for the standby online redo logs:

- You must use the `LOG_FILE_NAME_CONVERT` parameter to name the online redo logs if the primary and standby databases use different naming conventions for the logs.
- You cannot use the `SET NEWNAME` or `CONFIGURE AUXNAME` commands to rename the online redo logs.

- You cannot use the `LOGFILE` clause of the `DUPLICATE` command to specify filenames for the online redo logs.
- If you want the standby online log filenames the same as the primary online log filenames, then you must specify the `NOFILENAMECHECK` clause of the `DUPLICATE` command. Otherwise, RMAN signals an error even if the standby database is created in a different host.

> **See Also:** *Oracle9i Data Guard Concepts and Administration* for complete instructions for naming standby database online redo logs

Creating a Standby Database with RMAN: Overview

When you create a standby database, the procedure differs depending on whether the standby database is on the same host as the primary database or on a different host. The procedures in this chapter assume that you have already completed the standby setup and preparation as outlined in *Oracle9i Data Guard Concepts and Administration*. Do not attempt these procedures until you have made all necessary initialization parameter settings and network configuration.

After you have performed the steps necessary for preparing the standby instance, run the Recovery Manager `DUPLICATE ... FOR STANDBY` command to create the standby database out of backups of the primary database. Note that a standby database, unlike a duplicate database created by `DUPLICATE` *without* the `FOR STANDBY OPTION`, does not get a new DBID. Hence, you should not attempt to register the standby database in the repository for the primary database.

The steps for creating the standby database differ depending on whether you specify that RMAN should recover the standby database after creating it.

> **See Also:** Chapter 12, "Duplicating a Database with Recovery Manager" to learn how to use `DUPLICATE` to create a duplicate database that is not a standby database

RMAN Standby Creation Without Recovery

By default, RMAN does not recover the standby database after creating it. If you do not specify the `DORECOVER` option of the `DUPLICATE` command, then RMAN automates these steps of the standby creation procedure during duplication:

1. RMAN establishes connections both to the primary and standby databases, and the recovery catalog (if used).

2. RMAN queries the repository, which is either the primary control file or the recovery catalog, to identify the backups of primary database datafiles and the standby control file.
3. If you use a media manager, then RMAN contacts the media manager on the standby host to request the backup data.
4. RMAN restores the standby control file to the standby host, thereby creating the standby control file.
5. RMAN restores the primary datafile backups and copies to the standby host, thereby creating the standby database datafiles.
6. RMAN leaves the standby database mounted, but does *not* place the standby database in manual or managed recovery mode. RMAN disconnects and does not perform media recovery of the standby database. Note that you should *not* register the standby database in the recovery catalog.

RMAN Standby Creation with Recovery

If you do specify the `DORECOVER` option of the `DUPLICATE` command, then RMAN performs the same steps 1-5 in "RMAN Standby Creation Without Recovery" on page 13-9. Instead of step 6, it performs these steps:

1. After all data is restored, RMAN begins media recovery. If recovery requires archived redo logs, and if the logs have the status `AVAILABLE` in the repository, then RMAN attempts to apply them. Note that if logs marked `AVAILABLE` do not actually exist on disk, then RMAN returns an error. If a log that is not marked `AVAILABLE` is required, then RMAN attempts to restore it.
2. RMAN recovers the standby database to the specified time, system change number (SCN), or log sequence number, or to the latest archived redo log generated if none of the preceding are specified.
3. RMAN leaves the standby database mounted after media recovery is complete, but does *not* place the standby database in manual or managed recovery mode. Note that you should *not* register the standby database in the recovery catalog.

Note: After RMAN creates the standby database, you must resolve any gap sequence before placing it in manual or managed recovery mode, or opening it in read-only mode. *Oracle9i Data Guard Concepts and Administration* discusses gap sequence resolution in detail.

If you want RMAN to recover the standby database after creating it, then the standby control file must be usable for the desired recovery. Thus, these conditions must be met:

- The end recovery time of the standby database must be greater than or equal to the checkpoint SCN of the standby control file.
- An archived redo log containing the checkpoint SCN of the standby control file must be available at the standby site for recovery.

One way to ensure that these conditions are met is to issue the `ALTER SYSTEM ARCHIVE LOG CURRENT` statement after creating the standby control file. This statement archives the online logs of the primary database. Then, either back up the most recent archived log with RMAN or move the archived log to the standby site.

Whether or not you perform recovery, RMAN does *not* activate the standby database after creating it. The only way to activate a standby database is to issue the `ALTER DATABASE ACTIVATE STANDBY DATABASE` statement. After a standby database is activated and the redo logs are reset, all backups and archived logs of the old primary database are invalid for the new incarnation of the database.

Note: The procedures in this chapter assume that you are using RMAN backups to create the standby database. If you are using RMAN image copies, then refer to "Creating a Standby Database with Image Copies" on page 13-21.

See Also :

- *Oracle9i Data Guard Concepts and Administration* to learn how to manage a standby database
- *Oracle9i Recovery Manager Reference* for the list of `DUPLICATE` restrictions for creating a standby database with RMAN

Starting RMAN and the Standby Instance

No matter which standby creation scenario you choose, you must first start the standby instance and then connect RMAN to this instance. The details of this procedure vary depending on whether the standby and primary sites have a different directory structure.

To start the standby instance:

1. Use an operating system utility to copy the initialization parameter file from the target host to the standby host. Set all required parameters in the standby database initialization parameter file as described in *Oracle9i Data Guard Concepts and Administration*. For example, if creating the standby database on a separate host with a different directory structure, edit:

   - Initialization parameters that end with `_DEST` and `_PATH` and specify a path name
   - `DB_FILE_NAME_CONVERT` so that it captures *all* the target datafiles and converts them appropriately, for example, from `tbs_*` to `sbytbs_*`
   - `LOG_FILE_NAME_CONVERT` so that it captures *all* the online redo logs and converts them appropriately, for example, `log_*` to `sbylog_*`

   For example, the following are sample parameter settings in the standby database initialization parameter file:

   ```
   STANDBY_ARCHIVE_DEST = /fs3/arc_dest/
   LOG_ARCHIVE_FORMAT = log%t_%s.arc
   DB_FILE_NAME_CONVERT = ('/oracle', '/fs3/oracle', '/dbf', '/fs3/oracle')
   LOG_FILE_NAME_CONVERT = ('/oracle', '/fs3/oracle')
   ```

2. Use SQL*Plus to start the standby instance without mounting it. For example, enter the following to connect to `sbdb1` as `SYS` (who has `SYSDBA` privileges) and start the database:

   ```
   SQL> CONNECT SYS/sys_pwd@sbdb1 AS SYSDBA
   SQL> STARTUP NOMOUNT PFILE=initSBDB1.ora
   ```

3. Use SQL*Plus to mount or open the primary database if it is not already mounted or open. For example, enter the following to connect to `prod1` as `SYS` and open the database:

   ```
   SQL> CONNECT SYS/sys_pwd@prod1 AS SYSDBA
   SQL> STARTUP PFILE=initPROD1.ora
   ```

   If you use a recovery catalog, then make sure that the catalog database is open. For example, enter the following to connect to `catdb` as `SYS` and open the recovery catalog database:

   ```
   SQL> CONNECT SYS/oracle@catdb AS SYSDBA
   SQL> STARTUP PFILE=initCATDB.ora
   ```

4. The standby instance must be accessible through Oracle Net. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the standby instance. Note that you must connect to the auxiliary instance with `SYSDBA` privileges, so a password file must exist.

5. Connect to the target database, the standby instance, and (if you use one) the recovery catalog database. Note that you specify the *primary* database with the `TARGET` keyword and the *standby* instance with the `AUXILIARY` keyword.

   In the following example, connection is established without a recovery catalog by using operating system authentication:

   ```
   % rman TARGET / AUXILIARY SYS/sys_pwd@sbdb1
   ```

Creating a Standby Database on a Remote Host with the Same Directory Structure

The simplest case is to create the standby database on a different host and to use the same directory structure. In this case, you do *not* need to set the `DB_FILE_NAME_CONVERT` or `LOG_FILE_NAME_CONVERT` parameters in the standby initialization parameter file or set new filenames for the standby datafiles. The primary and standby datafiles and logs have the same filenames.

Creating the Standby Database Without Performing Recovery

To create the standby database without performing recovery, do not specify the `DORECOVER` option on the `DUPLICATE` command. By default, RMAN leaves the standby database mounted and does not recover it.

To create a standby database without performing recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.
2. Follow these steps during duplication to create but not recover the standby datafiles:

   a. If you do not have automatic channels configured, then manually allocate at least one auxiliary channel. This channel performs the work of duplication.

   b. Specify `NOFILENAMECHECK` in the `DUPLICATE` command. The `NOFILENAMECHECK` option is required when the standby and primary datafiles and logs have the same names. Otherwise, RMAN issues an error.

For example, run the following command to create the standby database:

```
DUPLICATE TARGET DATABASE FOR STANDBY
  NOFILENAMECHECK;
```

Creating the Standby Database and Performing Recovery

To create the standby database and perform recovery, specify the `DORECOVER` option on the `DUPLICATE` command.

To create a standby database and perform recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.
2. Follow these steps to restore and recover the standby datafiles:
   a. Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery.
   b. If desired, issue a `SET` command to specify the end time, SCN, or log sequence number for incomplete recovery.
   c. If automatic channels are not configured, then manually allocate at least one auxiliary channel.
   d. Specify the `NOFILENAMECHECK` parameter in the `DUPLICATE` command, and use the `DORECOVER` option.

   For example, enter the following at the RMAN prompt to use a configured channel to create the standby database:

```
# If desired, issue a LIST command to determine the SCN of the standby control file.
# The SCN to which you recover must be greater than or equal to the standby control
# file SCN.
LIST BACKUP OF CONTROLFILE;
LIST COPY OF CONTROLFILE;

RUN
{
  # If desired, issue a SET command to terminate recovery at a specified point.
  # SET UNTIL SCN 143508;
  DUPLICATE TARGET DATABASE FOR STANDBY
    NOFILENAMECHECK
    DORECOVER;
}
```

   RMAN uses all incremental backups, archived log backups, and archived logs to perform incomplete recovery. The standby database is left mounted.

Creating a Standby Database on a Remote Host with a Different Directory Structure

If you create the standby database on a host with a different directory structure, you need to specify new filenames for the standby database datafiles and online redo logs. You can do the following:

- Set `LOG_FILE_NAME_CONVERT` in the standby initialization parameter file to name the standby database online redo logs. If you do not set `LOG_FILE_NAME_CONVERT`, then you must use the `NOFILENAMECHECK` option of the `DUPLICATE` command.
- Set `DB_FILE_NAME_CONVERT` in the standby initialization parameter file to name the standby datafiles.
- Issue the `SET NEWNAME` command or the `CONFIGURE AUXNAME` command when using the RMAN `DUPLICATE` command to name the datafiles.

When creating the standby database on a host with a different directory structure, follow one of the procedures in the following sections:

- Naming Standby Database Files with DB_FILE_NAME_CONVERT
- Naming Standby Database Files with SET NEWNAME Commands
- Naming a Standby Database Files with CONFIGURE AUXNAME Commands

See Also:

- "Naming the Duplicate Datafiles" on page 12-6 to learn about the difference between `SET NEWNAME` and `CONFIGURE AUXNAME`
- *Oracle9i Data Guard Concepts and Administration* for a full discussion of standby database preparation and creation

Naming Standby Database Files with DB_FILE_NAME_CONVERT

In this procedure, you use `DB_FILE_NAME_CONVERT` to name the standby datafiles and `LOG_FILE_NAME_CONVERT` to name the standby online redo logs.

See Also: *Oracle9i Data Guard Concepts and Administration* for examples of how to use the `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` parameters to name standby database files

Creating the Standby Database Without Performing Recovery

To create the standby database without performing recovery, do not specify the DORECOVER option on the DUPLICATE command. By default, RMAN leaves the standby database mounted and does not recover it.

To use parameters to name standby files without performing recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.
2. Run the DUPLICATE command. For example, run the following:

   ```
   DUPLICATE TARGET DATABASE FOR STANDBY;
   ```

   After restoring the backups, RMAN leaves the standby database mounted.

Creating the Standby Database and Performing Recovery

After using DB_FILE_NAME_CONVERT to name the standby datafiles and LOG_FILE_NAME_CONVERT to name the standby online redo logs, specify the DORECOVER option on the DUPLICATE command to create the standby database and perform recovery. The steps in the procedure are the same as for "Creating the Standby Database and Performing Recovery" on page 13-14.

Naming Standby Database Files with SET NEWNAME Commands

In this procedure, you use SET NEWNAME commands to name the standby datafiles.

Creating the Standby Database Without Performing Recovery

To create the standby database without performing recovery, do not specify the DORECOVER option on the DUPLICATE command. By default, RMAN leaves the standby database mounted and does not recover it.

To name standby database files with the SET NEWNAME command without performing recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.
2. Run the DUPLICATE command. Perform the following operations:
   a. If automatic channels are not configured, then manually allocate at least one auxiliary channel.

b. Specify new filenames for the standby database datafiles with `SET NEWNAME` commands.

c. Issue the `DUPLICATE` command.

The following example uses a configured channel to create the standby database:

```
RUN
{
  # set new filenames for the datafiles
  SET NEWNAME FOR DATAFILE 1 TO '?/dbs/standby_data_01.f';
  SET NEWNAME FOR DATAFILE 2 TO '?/dbs/standby_data_02.f';
  .
  .
  .
  # run the DUPLICATE command
  DUPLICATE TARGET DATABASE FOR STANDBY;
}
```

Creating the Standby Database and Performing Recovery

To create the standby database and perform recovery, specify the `DORECOVER` option on the `DUPLICATE` command.

To use the SET NEWNAME command to name standby database files and perform recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.

2. Run the `DUPLICATE` command. Follow these steps:

   a. Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 13-10).

   b. If desired, issue a `SET` command to specify the end time, SCN, or log sequence number for incomplete recovery.

   c. If automatic channels are not configured, then manually allocate at least one auxiliary channel.

   d. Specify new filenames for the standby database datafiles.

   e. Issue the `DUPLICATE` command with the `DORECOVER` option.

For example, enter the following at the RMAN prompt to use a configured channel to create the standby database:

```
# If desired, issue a LIST command to determine the SCN of the standby control file.
# The SCN to which you recover must be greater than or equal to the control file SCN.

LIST BACKUP OF CONTROLFILE;
LIST COPY OF CONTROLFILE;
RUN
{
  # If desired, issue a SET command to terminate recovery at a specified point.
  # SET UNTIL TIME 'SYSDATE-7';

  # Set new filenames for the datafiles
  SET NEWNAME FOR DATAFILE 1 TO '?/dbs/standby_data_01.f';
  SET NEWNAME FOR DATAFILE 2 TO '?/dbs/standby_data_02.f';
  .
  .
  .
  DUPLICATE TARGET DATABASE FOR STANDBY
    DORECOVER;
}
```

RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

Naming a Standby Database Files with CONFIGURE AUXNAME Commands

In this procedure, you use `CONFIGURE AUXNAME` commands to name the standby datafiles.

> **See Also:** "Duplicating By Using CONFIGURE AUXNAME" on page 12-18

Creating the Standby Database Without Performing Recovery

To create the standby database without performing recovery, do not specify the `DORECOVER` option on the `DUPLICATE` command. By default, RMAN leaves the standby database mounted and does not recover it.

To use CONFIGURE AUXNAME to name standby database files without performing recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.

2. Configure the auxiliary names for the datafiles. For example, enter:

```
# set auxiliary names for the datafiles
CONFIGURE AUXNAME FOR DATAFILE 1 TO '/oracle/auxfiles/aux_1.f';
CONFIGURE AUXNAME FOR DATAFILE 2 TO '/oracle/auxfiles/aux_2.f';
.
.
.
CONFIGURE AUXNAME FOR DATAFILE n TO '/oracle/auxfiles/aux_n.f';
```

3. Run the `DUPLICATE` command. If automatic channels are not configured, manually allocate at least one auxiliary channel before issuing the `DUPLICATE` command, as in the following example:

```
RUN
{
  # allocate at least one auxiliary channel of type DISK or sbt
  ALLOCATE AUXILIARY CHANNEL standby1 DEVICE TYPE sbt;
  .
  .
  .
  # issue the DUPLICATE command
  DUPLICATE TARGET DATABASE FOR STANDBY;
}
```

4. Unspecify the auxiliary names for the datafiles so that they are not overwritten by mistake. For example, enter the following at the RMAN prompt:

```
# un-specify auxiliary names for the datafiles
CONFIGURE AUXNAME FOR DATAFILE 1 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
.
.
.
CONFIGURE AUXNAME FOR DATAFILE n CLEAR;
```

Creating the Standby Database and Performing Recovery

To create the standby database and perform recovery, specify the `DORECOVER` option on the `DUPLICATE` command.

To use CONFIGURE AUXNAME to name standby files and perform recovery:

1. Follow the steps in "Starting RMAN and the Standby Instance" on page 13-11. Make sure to set all necessary parameters in the standby initialization parameter file.

2. Set the auxiliary names for the datafiles. For example, enter the following:

```
# set auxiliary names for the datafiles
CONFIGURE AUXNAME FOR DATAFILE 1 TO '/oracle/auxfiles/aux_1.f';
CONFIGURE AUXNAME FOR DATAFILE 2 TO '/oracle/auxfiles/aux_2.f';
.
.
.
```

```
CONFIGURE AUXNAME FOR DATAFILE n TO '/oracle/auxfiles/aux_n.f';
```

3. Run the `DUPLICATE` command. Follow these steps:

   - Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 13-10).
   - If desired, issue a `SET` command to specify the end time, SCN, or log sequence number for incomplete recovery.
   - If automatic channels are not configured, then manually allocate at least one auxiliary channel.
   - Issue the `DUPLICATE TARGET DATABASE` for standby command.

   For example, enter the following at the RMAN prompt to use a configured channel to create the standby database:

   ```
   # If desired, issue a LIST command to determine the SCN of the standby control file.
   # The SCN to which you recover must be greater than or equal to the control file SCN.
   LIST BACKUP OF CONTROLFILE;
   LIST COPY OF CONTROLFILE;

   DUPLICATE TARGET DATABASE FOR STANDBY
     DORECOVER;
   ```

   RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

4. Clear the auxiliary name settings for the datafiles so that they are not overwritten by mistake. For example, enter the following at the RMAN prompt:

   ```
   # un-specify auxiliary names for the datafiles
   CONFIGURE AUXNAME FOR DATAFILE 1 CLEAR;
   CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
   .
   .
   .
   CONFIGURE AUXNAME FOR DATAFILE n CLEAR;
   ```

Creating a Standby Database on the Local Host

When creating a standby database on the same host as the primary database, follow the same procedure as for duplicating to a remote host with a different directory structure as described in "Creating a Standby Database on a Remote Host with a Different Directory Structure" on page 13-15.

Note the following restrictions when creating a standby database on the same host as the primary database:

- You can create the standby database in the same Oracle home as the target database, but you must convert the filenames with the same methods used for conversion on a separate host. That is, you must treat a standby database in the same Oracle home as if it were a database on a separate host with a different directory structure. You must *not* use the same names for standby and primary database files when the two databases are on the same machine.
- When both databases are in the same host and same Oracle home, you must set the `LOCK_NAME_SPACE` initialization parameter.

Caution: Do not use the `NOFILENAMECHECK` option when creating the standby database in the same Oracle home as the primary database. If you do, then you may overwrite the target database files or cause the `DUPLICATE` command to fail with an error.

Creating a Standby Database with Image Copies

This section contains these topics:

- Creating a Standby Database with Image Copies: Overview
- Creating the Standby Database When Copies and Datafiles Use the Same Names
- Creating the Standby Database When Copies and Datafiles Use Different Names
- Creating a Standby Database with Backups and Image Copies: Scenario

Creating a Standby Database with Image Copies: Overview

The main restriction when using RMAN image copies to create the standby datafiles is that the image copy filenames for datafiles and archived logs on the primary and standby hosts must be the same. For example, assume that datafile 1 is named `/oracle/dbs/df1.f` on the primary host. If you use the RMAN `COPY` command to copy this datafile to `/data/df1.f`, then this image copy must exist on the standby host with the same filename of `/data/df1.f`. Otherwise, RMAN cannot locate the metadata for the standby image copy in its repository.

You have two main ways of populating the standby host with the image copies:

- Transferring them manually with `ftp` or some other utility
- Mounting the standby directory structure on the primary host with a network file system (NFS)

When you use the NFS method, you can create a directory on the primary host that maps to a directory on the standby host. If you use this method, then the NFS mount point on both machines must have the same directory name. For example, you can map `/data` on the primary host to `/data` on the standby host, but you cannot map `/data` on the primary host to `/dir` on the standby host (unless you use functionality such as symbolic links in UNIX or logical drives on Windows NT).

The filename of the image copy on the standby host must be the same as the filename of the image copy on the primary host. Nevertheless, you can specify a different path name for the standby datafile by using `SET NEWNAME` commands or the `DB_FILE_NAME_CONVERT` initialization parameter.

For example, although the image copy of datafile 1 is named `/data/df1.f` on the standby host, you can specify the path name `/oracle/sb/df1.f` in the standby control file by using initialization parameters or RMAN commands. Note that you do not manually rename the physical image copy. When you run the `DUPLICATE` command, RMAN restores the image copy `/data/df1.f` and creates the standby datafile 1 as `/oracle/sb/df1.f` based on the information in the initialization parameters or RMAN commands.

Table 13–3 illustrates two scenarios for using NFS to create a standby database with one datafile.

Table 13–3 Using Image Copies to Create a Standby Database: Scenario

NFS Mount Point	Primary Datafile Filename	Image Copy Filename	Standby Datafile Filename	Procedure
`/data` (same on both hosts)	`/oracle/dbs/df1.f`	`/data/df1.f`	`/data/df1.f` (same path name as image copy)	"Creating the Standby Database When Copies and Datafiles Use the Same Names" on page 13-24
`/data` (same on both hosts)	`/oracle/dbs/df1.f`	`/data/df1.f`	`/oracle/sb/df1.f` (different path name from image copy)	"Creating the Standby Database When Copies and Datafiles Use Different Names" on page 13-25

Table 13–3 assumes that the standby directory structure is mounted on the primary host, and that the mount point is `/data` on both hosts. Because the primary host

mounts the standby host directory structure, when you create the image copy `/data/df1.f` on the primary host, you are actually creating the image copy `/data/df1.f` on the standby host.

In the first scenario, you name the standby datafiles with the same filenames as the image copies. This case is the simplest because you do not need to use RMAN at all to create the standby database. First, set the `DB_FILE_NAME_CONVERT` parameter in the standby initialization parameter file to convert the primary datafile filename `/oracle/dbs/df1.f` to the standby filename `/data/df1.f`. Then, copy the files to the standby host, and mount the standby database.

In the second scenario, you use different filenames for the standby datafiles and the image copies. To create this standby database, run the `DUPLICATE` command. The `DUPLICATE` command restores the image copy of datafile 1 and renames it according to either the `SET NEWNAME` commands or the `DB_FILE_NAME_CONVERT` initialization parameter.

Creating the Standby Database When Copies and Datafiles Use the Same Names

This procedure assumes that you are using the same filenames for the standby datafiles and the image copies of the primary datafiles.

To create a standby database when the copies and standby datafiles have the same names:

1. After connecting to the primary database and, if desired, the recovery catalog database, mount but do not open the primary database and ensure that the database was closed cleanly prior to mounting. For example, enter:

   ```
   RMAN> STARTUP MOUNT PFILE=init.ora;
   ```

2. Make sure that you set `DB_FILE_NAME_CONVERT` in the standby initialization parameter file so that standby datafile filenames are translated from the primary datafile filenames. For example:

   ```
   DB_FILE_NAME_CONVERT = ('/oracle/dbs', '/dsk2/oracle')
   ```

3. Copy all of the datafiles and the standby control file. For example, enter:

   ```
   COPY
     DATAFILE 1 TO '/dsk2/oracle/df_1.f',
     DATAFILE 2 TO '/dsk2/oracle/df_2.f',
     DATAFILE 3 TO '/dsk2/oracle/df_3.f',
     DATAFILE 4 to '/dsk2/oracle/df_4.f',
     DATAFILE 5 TO '/dsk2/oracle/df_5.f',
     DATAFILE 6 TO '/dsk2/oracle/df_6.f',
     DATAFILE 7 TO '/dsk2/oracle/df_7.f',
     DATAFILE 8 to '/dsk2/oracle/df_8.f',
     DATAFILE 9 TO '/dsk2/oracle/df_9.f',
     DATAFILE 10 TO '/dsk2/oracle/df_10.f',
     DATAFILE 11 TO '/dsk2/oracle/df_11.f',
     DATAFILE 12 to '/dsk2/oracle/df_12.f',
     CURRENT CONTROLFILE FOR STANDBY TO '/dsk2/oracle/cf.f';
   ```

4. Start the auxiliary instance and mount the standby control file. For example, start SQL*Plus and enter:

   ```
   SQL> STARTUP NOMOUNT PFILE=/dsk2/oracle/dbs/initSTANDBY1.ora
   SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
   ```

Creating the Standby Database When Copies and Datafiles Use Different Names

This procedure assumes that you use different filenames for the standby datafiles and the image copies of the primary datafiles.

Creating the Standby Database Without Performing Recovery

To create the standby database without performing recovery, you do not need to run the DUPLICATE command. By default, RMAN leaves the standby database mounted and does not recover it.

To create a standby database when the copies and standby datafiles have different names without performing recovery:

1. Connect to the primary database, standby instance, and, if desired, the recovery catalog database. For example, enter:

   ```
   % rman TARGET sys/sys_pwd@prod1 AUXILIARY sys/sys_pwd@sbdb1 CATALOG rman/cat@catdb
   ```

2. Mount but do not open the primary database and ensure that the database was closed cleanly prior to mounting. For example, enter:

   ```
   STARTUP MOUNT PFILE=initPROD1.ora
   ```

3. Either set DB_FILE_NAME_CONVERT in the standby initialization parameter file so that standby datafile filenames are translated from the primary datafile filenames, or issue SET NEWNAME commands. For example, set the DB_FILE_NAME_CONVERT parameter as follows:

   ```
   DB_FILE_NAME_CONVERT = ('/oracle/dbs', '/dsk2/oracle')
   ```

4. Use the COPY command to copy all of the datafiles and the standby control file. For example, run the following commands:

   ```
   COPY
     DATAFILE 1 TO '/dsk2/oracle/df_1.f',
     DATAFILE 2 TO '/dsk2/oracle/df_2.f',
     DATAFILE 3 TO '/dsk2/oracle/df_3.f',
     DATAFILE 4 to '/dsk2/oracle/df_4.f',
     DATAFILE 5 TO '/dsk2/oracle/df_5.f',
     DATAFILE 6 TO '/dsk2/oracle/df_6.f',
     DATAFILE 7 TO '/dsk2/oracle/df_7.f',
     DATAFILE 8 to '/dsk2/oracle/df_8.f',
     DATAFILE 9 TO '/dsk2/oracle/df_9.f',
     DATAFILE 10 TO '/dsk2/oracle/df_10.f',
     DATAFILE 11 TO '/dsk2/oracle/df_11.f',
     DATAFILE 12 to '/dsk2/oracle/df_12.f',
     CURRENT CONTROLFILE FOR STANDBY TO '/dsk2/oracle/cf.f';
   # To ensure that the control file checkpoint is archived, archive the current
   # redo log
   SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
   ```

5. Start the auxiliary instance and mount the standby control file. For example, start SQL*Plus and enter:

```
SQL> STARTUP NOMOUNT PFILE=/dsk2/oracle/dbs/initSTANDBY1.ora
SQL> ALTER DATABASE MOUNT STANDBY DATABASE;
```

Creating the Standby Database and Performing Recovery

To create the standby database and perform recovery, specify the DORECOVER option on the DUPLICATE command.

To create a standby database when the copies and standby datafiles have different names and perform recovery:

1. Connect to the primary database, standby instance, and, if desired, the recovery catalog database. For example, enter:

```
% rman TARGET sys/sys_pwd@prod1 AUXILIARY sys/sys_pwd@sbdb1 CATALOG rman/cat@catdb
```

2. Mount but do not open the primary database and ensure that the database was closed cleanly prior to mounting. For example, enter:

```
STARTUP MOUNT PFILE=initPROD1.ora
```

3. Either set DB_FILE_NAME_CONVERT in the standby initialization parameter file so that standby datafile filenames are translated from the primary datafile filenames, or issue SET NEWNAME commands. For example, set the DB_FILE_NAME_CONVERT parameter as follows:

```
DB_FILE_NAME_CONVERT = ('/oracle/dbs', '/dsk2/oracle')
```

4. Run the DUPLICATE command. Follow these steps:

   a. Ensure that the end recovery time is greater than or equal to the checkpoint SCN of the standby control file and that a log containing the checkpoint SCN is available for recovery (as described in "RMAN Standby Creation with Recovery" on page 13-10).

   b. If desired, issue a SET command to specify the end time, SCN, or log sequence number for recovery.

   c. If automatic channels are not configured, then manually allocate at least one auxiliary channel for the duplication.

   d. Copy every datafile and the standby control file.

   e. Archive the current redo logs.

   f. Issue the DUPLICATE command with the DORECOVER option.

For example, enter the following:

```
COPY
  DATAFILE 1 TO '/dsk2/oracle/df_1.f',
  DATAFILE 2 TO '/dsk2/oracle/df_2.f',
  DATAFILE 3 TO '/dsk2/oracle/df_3.f',
  DATAFILE 4 to '/dsk2/oracle/df_4.f',
  DATAFILE 5 TO '/dsk2/oracle/df_5.f',
  DATAFILE 6 TO '/dsk2/oracle/df_6.f',
  DATAFILE 7 TO '/dsk2/oracle/df_7.f',
  DATAFILE 8 to '/dsk2/oracle/df_8.f',
  DATAFILE 9 TO '/dsk2/oracle/df_9.f',
  DATAFILE 10 TO '/dsk2/oracle/df_10.f',
  DATAFILE 11 TO '/dsk2/oracle/df_11.f',
  DATAFILE 12 to '/dsk2/oracle/df_12.f',
  CURRENT CONTROLFILE FOR STANDBY TO '/dsk2/oracle/cf.f';
  SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
  DUPLICATE TARGET DATABASE FOR STANDBY
    DORECOVER;
```

RMAN uses all incremental backups, archived redo log backups, and archived redo logs to perform incomplete recovery. The standby database is left mounted.

Creating a Standby Database with Backups and Image Copies: Scenario

In this scenario, you are performing a duplication that uses both backups and image copies of the primary datafiles. The scenario illustrates how RMAN is able to use both datafile backups and datafile copies for the standby files, and also is able to use both incremental backups and archived logs to recovery the standby database.

Assume the following about the standby database environment:

- The primary database is on `host1` and the standby database is on `host2`
- Database `prod1` has thirty datafiles: datafiles 1 through 25 are on raw disk named with the pattern `/dev/rdsk`*###* (where *###* is a number starting with `001` and ending with `025`), and datafiles 26 through 30 are located in the `/primary/datafile` directory

You perform the following actions over the course of a week:

1. On Monday, you run the following incremental level 0 database backup:

   ```
   BACKUP DEVICE TYPE sbt INCREMENTAL LEVEL 0 DATABASE PLUS ARCHIVELOG;
   ```

2. On Tuesday, you copy datafiles 1 through 5 into the `/standby/datafile` directory on `host1`, then run `BACKUP ARCHIVELOG ALL`
3. On Wednesday, you copy datafiles 6 through 9 into the `/standby/datafile` directory on `host1`, then run `BACKUP ARCHIVELOG ALL`

4. On Thursday, you run the following incremental level 1 database backup:

   ```
   BACKUP DEVICE TYPE sbt INCREMENTAL LEVEL 1 DATABASE PLUS ARCHIVELOG;
   ```

5. On Friday, you copy datafiles 10 through 15 into the `/standby/datafile` directory on `host1`, then run `BACKUP ARCHIVELOG ALL`
6. On Saturday morning, you run the following RMAN commands:

   ```
   COPY CURRENT CONTROLFILE FOR STANDBY TO '/standby/datafile/cf.f';
   SQL 'ALTER SYSTEM ARCHIVELOG CURRENT';
   BACKUP DEVICE TYPE sbt ARCHIVELOG ALL;
   ```

7. On Saturday night, you `ftp` all the image copies in `/standby/datafile` on `host1` to `/standby/datafile` on `host2`, and also `ftp` all the logs on `host1` to `host2`; you also make the tape backups of `prod1` accessible to `host2`

On Sunday, you decide to create the standby database and recover it up to the point of the Saturday backup. You want all the standby datafiles to be located in the `/standby/datafile` directory on `host2`.

You must choose a method for naming the standby datafiles. You could use `DB_FILE_NAME_CONVERT` to change each pattern of the raw disk datafiles, which would require twenty-five pairs of values in the parameter (one pair for each raw disk filename that is being renamed). Instead, you decide to use `SET NEWNAME` commands for the twenty-five datafiles on raw disk, and use `DB_FILE_NAME_CONVERT` only for converting the names for the five datafiles in `/primary/datafile` to `/standby/datafile`.

The image copies are located in `/standby/datafile` on `host2`, but you only made copies of datafiles 1 through 15. This is not a problem, however, because you have incremental backups of all the datafiles. RMAN always chooses to restore image copies over backups, but if no image copies are available, then RMAN restores backups. So, you run the following script:

```
RUN
{
  # run SET NEWNAME commands for datafiles 1-25
  SET NEWNAME FOR DATAFILE 1 TO '/standy/datafile/df1.f';
  SET NEWNAME FOR DATAFILE 2 TO '/standby/datafile/df2.f';
  .
  .
  .
  SET NEWNAME FOR DATAFILE 25 TO '/standby/datafile/df25.f';
  DUPLICATE TARGET DATABASE FOR STANDBY DORECOVER;
}
```

RMAN does the following actions during the duplication:

- Uses the image copies of datafiles 1 through 15.
- Restores the backups of datafiles 16 through 30 (because no image copies are available of these datafiles).
- Uses incremental backups to recover datafiles 1 through 9 and datafiles 16 through 30, but not to recover datafiles 10 through 15 because these copies were created on Friday after the Thursday incremental level 1 backup.
- Restores and applies archived logs as needed to datafiles 1 through 30 up to the last archived log that was backed up.
- Applies archived logs on disk up to the last archived log.

14 Tuning Recovery Manager

The primary goal of RMAN tuning is to create an adequate flow of data between disk and storage device. Tuning RMAN backup and restore operations involves the following tasks discussed in this chapter:

Tuning Recovery Manager: Overview

RMAN backup and restore operations have the following distinct components:

- Reading or writing input data
- Processing data by validating blocks and copying them from the input to the output buffers

The slowest of these operations is called the **bottleneck**. RMAN tuning is the task of identifying the bottleneck (or bottlenecks) and attempting to make it more efficient by using RMAN commands, initialization parameter settings, or adjustments to physical media. The key to tuning RMAN is understanding I/O.

RMAN's backup and restore jobs use two types of I/O buffers: `DISK` and tertiary storage (usually tape). When performing a backup, RMAN reads input files using disk buffers and writes the output backup file by using either disk or tape buffers. When performing restores, RMAN reverses these roles.

Besides being divided into `DISK` and `sbt`, I/O is also divided into **synchronous** and **asynchronous**. Synchronous devices only perform one I/O task at a time. Hence, you can easily determine how much time backup jobs require. In contrast to synchronous I/O, asynchronous I/O can perform more than one task at a time.

To tune RMAN effectively, you must thoroughly understand concepts such as synchronous and asynchronous I/O, disk and tape buffers, and channel architecture. When you understand these concepts, then you can learn how to use fixed views to monitor bottlenecks, and use the techniques described in "Improving RMAN Backup Performance" on page 14-11 to solve problems.

RMAN Tuning Concepts

This section contains these topics:

- Disk Buffer Allocation
- Tape Buffer Allocation
- Synchronous and Asynchronous I/O
- Factors Affecting Backup Speed to Tape
- Channel Tuning Options
- Backup Tuning Options

Disk Buffer Allocation

RMAN I/O uses two different types of buffers: disk and tape. These buffers are typically different sizes. To understand how RMAN allocates disk buffers, you must understand how RMAN multiplexing works, as described in "Multiplexed Backup Sets" on page 5-18. Review this section before proceeding.

RMAN **multiplexing** is the number of files in a backup read simultaneously and then written to the same backup piece. The degree of multiplexing depends on the `FILESPERSET` parameter of the `BACKUP` command as well as the `MAXOPENFILES` parameter of the `CONFIGURE CHANNEL` or `ALLOCATE CHANNEL` commands.

For example, assume that you back up two datafiles with one channel. You set `FILESPERSET` to `3` and set `MAXOPENFILES` to `8`. In this case, the number of files in each backup set is 2 (the lesser of `FILESPERSET` and the files read by each channel), and so the level of multiplexing is `2` (the lesser of `MAXOPENFILES` and the number of files in each backup set).

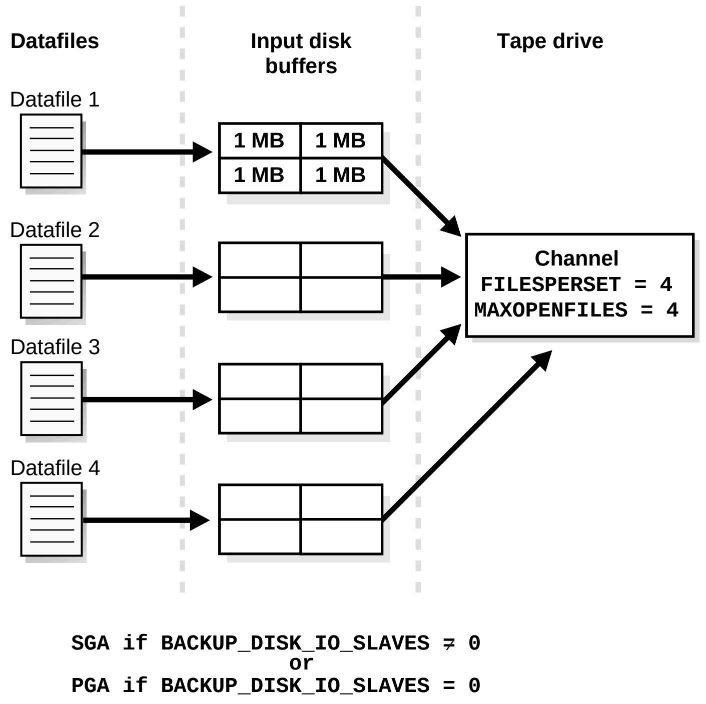
When RMAN backs up from disk, it uses the algorithm described in Table 14–1 to determine how many buffers to allocate and how large to make the buffers.

Table 14–1 Disk Buffer Allocation Algorithm

If level of multiplexing is . . .	Then . . .
Less than or equal to 4	RMAN allocates buffers of size 1 MB so that the total buffer size for all the input files is 16 MB. For example, if `FILESPERSET=1`, then RMAN allocates 16 buffers for the one file going into the set. If `FILESPERSET=4`, then RMAN allocates 4 buffers for each of the 4 files in the backup set.
Greater than 4 but less than or equal to 8	RMAN allocates disk buffers of size 512 KB so that the total buffer size for all the files is less than 16 MB.
Greater than 8	RMAN allocates a fixed 4 disk buffers of 128 KB for each file, so that the total size is 512 KB for each file.

In the example shown in Figure 14–1, "Disk Buffer Allocation", one channel is backing up four datafiles on a robust striped disk configuration. `MAXOPENFILES` is set to `4` and `FILESPERSET` is set to `4`. Hence, the level of multiplexing is 4. So, the total size of the buffers for each datafile is 4 MB.

Figure 14–1 Disk Buffer Allocation



To calculate the total size of the buffers allocated in a backup set, multiply the total bytes for each datafile by the number of datafiles being concurrently accessed by the channel, and then multiply this number by the number of channels.

Assume that you use one channel to back up four datafiles, and use the settings shown in Figure 14–1. In this case, multiply as follows to obtain the total size of the buffers allocated for the backup:

```
4 MB/datafile x 1 channel x 4 datafiles/channel = 16 MB
```

Set the `MAXOPENFILES` parameter so that the number of files read simultaneously is just enough to utilize the output device fully. This consideration is especially important when the output device is tape.

Tape Buffer Allocation

If you make a backup to an `sbt` device, then Oracle allocates four buffers for each channel for the tape writers (or reads if doing a restore). Oracle allocates these buffers *only if* the channel is an `sbt` channel. Typically, each tape buffer is 256 KB. To calculate the total size of buffers used during a backup or restore, multiply the buffer size by 4, and then multiply this product by the number of channels.

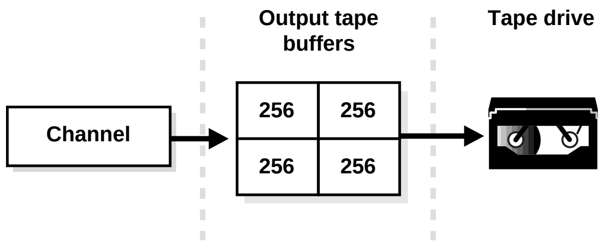
As illustrated in Figure 14–2, assume that you use one tape channel and each buffer is 256 KB. In this case, the total size of buffers used during a backup is as follows:

```
256 KB/buffer x 4 buffers/channel x 1 channel = 1024 KB
```

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 or the large pool if the `LARGE_POOL_SIZE` initialization parameter is set. If you set the parameter to `false`, then RMAN allocates the buffers from the PGA.

If you use I/O slaves, then set the `LARGE_POOL_SIZE` initialization parameter to set aside SGA memory dedicated to holding these large memory allocations. Hence, the RMAN I/O buffers do not compete with the library cache for SGA memory.

Figure 14–2 Tape Buffer Allocation



```
SGA if BACKUP_TAPE_IO_SLAVES = TRUE
                  or
PGA if BACKUP_TAPE_IO_SLAVES = FALSE
```

Synchronous and Asynchronous I/O

When RMAN reads or writes data, the I/O is either **synchronous** or **asynchronous**. When the I/O is synchronous, a server process can perform only one task at a time. When it is asynchronous, a server process can begin an I/O and then perform other work while waiting for the I/O to complete. It can also begin multiple I/O operations before waiting for the first to complete.

You can set initialization parameters that determine the type of I/O. If you set `BACKUP_TAPE_IO_SLAVES` to `true`, then the tape I/O is asynchronous. Otherwise, the I/O is synchronous. It is recommended that you always set `BACKUP_TAPE_IO_SLAVES` to `true`.

Some operating systems support native asynchronous I/O, and Oracle takes advantage of this feature if it is available. On operating systems that do not support native asynchronous I/O, Oracle can simulate it by using special I/O slave processes that are dedicated to performing I/O on behalf of another process. You can control disk I/O slaves by setting the `DBWR_IO_SLAVES` parameter to a nonzero value. Oracle allocates four backup disk I/O slaves for any nonzero value of `DBWR_IO_SLAVES`.

Figure 14–3 shows synchronous I/O in a backup to tape. The following steps occur:

1. A server process writes blocks to a tape buffer.
2. The tape process writes data to tape. *The server process must remain idle while the media manager copies data from the Oracle buffers to the media manager's internal buffers.*
3. The tape process returns a message to the server process stating that it has completed writing.
4. The server process can initiate a new task.

Figure 14–3 Synchronous I/O

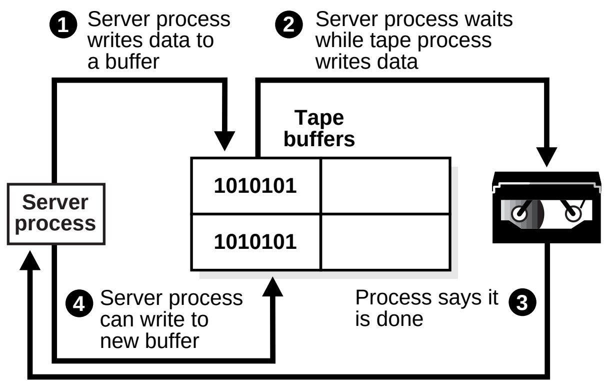
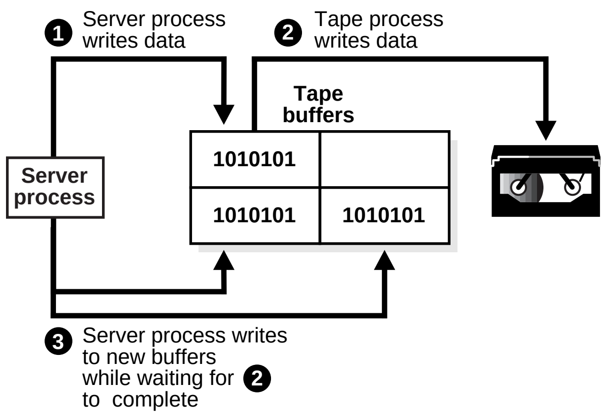

1 Server process writes data to a buffer
2 Server process waits while tape process writes data
Tape buffers
1010101
1010101
Server process
4 Server process can write to new buffer
Process says it is done 3

Figure 14–4 shows **asynchronous I/O** in a tape backup. The following steps occur:

1. A server process writes blocks to a tape buffer.
2. The tape process writes data to tape. *While the tape process is writing, other server processes are free to process more input blocks and fill more output buffers.*
3. Two spawned server processes write to tape buffers as the initial tape process writes to tape.

Figure 14–4 Asynchronous I/O


1 Server process writes data
2 Tape process writes data
Tape buffers
Server process
1010101
1010101
1010101
3 Server process writes to new buffers while waiting for 2 to complete

Factors Affecting Backup Speed to Tape

The following factors affect the speed of the backup to tape:

- Native Transfer Rate
- Tape Compression
- Tape Streaming
- Physical Tape Block Size

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.

Tape Streaming

One of the most interesting issues for backup performance is tape **streaming**. Almost all tape drives currently on the market are fixed-speed, streaming tape drives. In other words, these drives can only write data at one speed. As a result, when they run out of data to write to tape, they must slow down and stop. For example, when the drive's buffer empties, the tape is moving so quickly that it actually overshoots and must rewind past 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. The common rule is that a larger tape block size leads to a faster backup. Note that physical tape block size is not controlled by RMAN or the Oracle server, but by media management software. Larger physical tape block size leads to a faster backup. The physical tape block size is controlled by media management software.

Channel Tuning Options

You can set various channel limit parameters that apply to operations performed by the allocated server session in the `CONFIGURE CHANNEL` and `ALLOCATE CHANNEL` commands.

You can use these parameters to do the following:

- Limit the size of backup pieces.
- Prevent RMAN from consuming too much disk bandwidth.
- Determine the level of multiplexing for each channel.

You can specify the channel parameters described in Table 14–2.

Table 14–2 Tuning RMAN Channel Parameters

Parameter	Description
`MAXPIECESIZE`	Specifies the maximum size of a backup piece. Use this parameter to force RMAN to create multiple backup pieces in a backup set. RMAN creates each backup piece with a size no larger than the value specified in the parameter.

Table 14–2 Tuning RMAN Channel Parameters

Parameter	Description
RATE	Specifies the bytes / second that RMAN reads on this channel. Use this parameter to set an upper limit for bytes read so that RMAN does not consume excessive disk bandwidth and degrade online performance. For example, set RATE=1500K. If each disk drive delivers 3 MB / second, then RMAN leaves some disk bandwidth available to the online system.
MAXOPENFILES	Determines the maximum number of input files that a backup or copy can have open at a given time (default value is 8). As explained in "Disk Buffer Allocation" on page 14-3, the level of RMAN multiplexing partially determined by MAXOPENFILES. As explained in Table 14–1, the level of multiplexing in turn determines how RMAN allocates disk buffers. Multiplexing is the number of input files simultaneously read and then written into the same backup piece.

See Also:

- *Oracle9i Recovery Manager Reference* for ALLOCATE CHANNEL syntax
- *Oracle9i Recovery Manager Reference* for CONFIGURE syntax
- *Oracle9i Recovery Manager Reference* for SET syntax

Backup Tuning Options

The BACKUP command lets you set parameters that influence how RMAN selects files for input into backup sets. You can set these parameters to do the following:

- Prevent RMAN from writing a single backup set to multiple volumes.
- Prevent RMAN from reading from too many disks at once.
- Limit the number of disk drives that are read from simultaneously.

You can specify the parameters described in Table 14–3.

Table 14–3 Tuning RMAN Backup Parameters

Parameter	Description
MAXSETSIZE	Specifies the maximum size in bytes of a backup set. Use this parameter to prevent a single backup set from spanning multiple volumes.

Table 14–3 Tuning RMAN Backup Parameters

Parameter	Description
FILESPERSET	Specifies the maximum number of files to place in a backup set. The default value for FILESPERSET is the lower of these two values: 64, and the number of files / channel. If you allocate only one channel, then you can use this parameter to make RMAN create multiple backup sets. For example, if you have fifty input datafiles and two channels, you can set FILESPERSET=5 to create ten backup sets. This strategy can prevent you from splitting a backup set among multiple tapes.
DISKRATIO	Specifies the number of drives to include in the backup. Assume the datafiles are located on five disks, each disk supplies data at 10 bytes / second, and the tape drive requires 20 bytes / second to keep streaming. If you set DISKRATIO=2, then RMAN reads from two drives at a time, thereby distributing the backup load.

Note: Control the number of datafiles accessed by a channel by setting FILESPERSET on the BACKUP or CONFIGURE command.

See Also: *Oracle9i Recovery Manager Reference* for BACKUP and CONFIGURE syntax

Improving 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 get the best performance for a backup, follow the suggested steps in this section:

- Step 1: Adjust the RATE Parameter
- Step 2: If Backing Up to Tape, Set BACKUP_TAPE_IO_SLAVES
- Step 3: If You Use Synchronous Disk, Set DBWR_IO_SLAVES
- Step 4: If You Fail to Allocate Shared Memory, Set LARGE_POOL_SIZE
- Step 5: Adjust the Level of Multiplexing
- Step 6: Determine Whether Files Are Empty or Contain Few Changes
- Step 7: Use the BLKSIZE Parameter to Alter Tape Buffer Size
- Step 8: Query V$ Views to Identify Bottlenecks

Step 1: Adjust the RATE Parameter

Make sure that the RATE parameter is not set on the ALLOCATE CHANNEL or CONFIGURE CHANNEL commands, as described in Table 14–2. The RATE parameter is intended to slow down a backup so that you can run it in the background with as little effect as possible on OLTP operations.

The RATE parameter specifies units of bytes/second. Test to find a value that improves performance of your queries while still letting RMAN complete the backup in a reasonable amount of time. Note that RATE is not designed to increase backup throughput, but to decrease backup throughput so that more disk bandwidth is available for other database operations.

Step 2: If Backing Up to Tape, Set BACKUP_TAPE_IO_SLAVES

If (and only if) you are backing up to an sbt device, then set the BACKUP_TAPE_IO_SLAVES initialization parameter to true to cause the tape buffers to be allocated from the SGA. You can control the buffer size with the PARMS parameter on the ALLOCATE CHANNEL or CONFIGURE CHANNEL command.

The BACKUP_TAPE_IO_SLAVES initialization parameter simulates asynchronous tape I/O by spawning an additional process to wait for tape I/O completion, leaving the primary process free to process additional disk blocks while waiting for tape I/O to complete. If you do not set this parameter, then I/O to the tape layer is synchronous, which means no other work can occur until the tape is done writing.

The BACKUP_TAPE_IO_SLAVES parameter requires that the buffers for the respective disk or tape I/O be allocated from the shared memory (SGA), so that they can be shared between two processes. Therefore, allocate a large enough SGA size to accommodate this memory usage. If you set the BACKUP_TAPE_IO_SLAVES parameter, then also set the LARGE_POOL_SIZE parameter.

Step 3: If You Use Synchronous Disk, Set DBWR_IO_SLAVES

If (and only if) your disk does *not* support asynchronous I/O, then try setting the DBWR_IO_SLAVES initialization parameter to a nonzero value. Any nonzero value for DBWR_IO_SLAVES causes a fixed number (four) of disk I/O slaves to be used for backup and restore, which simulates asynchronous I/O. If I/O slaves are used, I/O buffers are obtained from the SGA (or the large pool, if configured).

Step 4: If You Fail to Allocate Shared Memory, Set LARGE_POOL_SIZE

Set this initialization parameter only if Oracle 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 looks something like the following:

```
ksfqxcre: failure to allocate shared memory means sync I/O will be used whenever async I/O
to file not supported natively
```

When attempting to get shared buffers for I/O slaves, Oracle does the following:

- If `LARGE_POOL_SIZE` is set, then Oracle attempts to get memory from the large pool. If this value is not large enough, then Oracle does not try to get buffers from the shared pool.
- If `LARGE_POOL_SIZE` is not set, then Oracle attempts to get memory from the shared pool.
- If Oracle cannot get enough memory, then it obtains I/O buffer memory from local process memory 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 (formerly called multi-threaded 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; Oracle does not attempt to age memory out of the large pool.

Use the `LARGE_POOL_SIZE` initialization parameter to configure the large pool. To see in which pool (shared pool or large pool) the memory for an object resides, query `V$SGASTAT.POOL`.

The Oracle9*i* formula for setting `LARGE_POOL_SIZE` is as follows:

```
LARGE_POOL_SIZE = number_of_allocated_channels * (16 MB + ( 4 * size_of_tape_buffer ) )
```

For backups to disk, the tape buffer size is obviously 0, so set `LARGE_POOL_SIZE` to 16 MB. For tape backups, the size of a single tape buffer is defined by the RMAN channel parameter `BLKSIZE`, which defaults to 256 KB. Assume a case in which you are backing up to two tape drives. If the tape buffer size is 256 KB, then set

LARGE_POOL_SIZE to 18 MB. If you increase BLKSIZE to 512 KB, then increase LARGE_POOL_SIZE to 20 MB.

> **See Also:** *Oracle9i Database Concepts* for more information about the large pool, and *Oracle9i Database Reference* for complete information about initialization parameters

Step 5: Adjust the Level of Multiplexing

As explained in "Multiplexed Backup Sets" on page 5-18, the level of multiplexing is determined by the following factors:

- The number of files going into in each backup set, which is determined by the lesser of the number of files read by each channel as compared to the FILESPERSET setting
- The lesser of the MAXOPENFILES value and the number of files going into in each backup set

You should adjust the level of multiplexing to account for the disk configuration on the server. Striped disk configurations involve hardware multiplexing, so that the level of RMAN multiplexing does not need to be as high. For example, consider the following disk configuration scenarios:

- If you stripe your datafiles across a large number of disks, then you should set the level of RMAN multiplexing to 1 because the hardware already performs multiplexing and so RMAN efficiently reads the data distributed among the multiple disks.
- If you stripe your datafiles across a small number of disks (for example, two or three), then you should set the level of multiplexing between 4 and 8 because RMAN has to perform some of the multiplexing chores.
- If you do not stripe your disks at all, then you should set the level of multiplexing to at least 8 because RMAN must perform the multiplexing itself.

For example, if each channel reads fifteen datafiles, and FILESPERSET=10 and MAXOPENFILES=8, then you can calculate the level of multiplexing as follows:

```
min( min( 15, 10 ), 8 ) = 8
```

If the datafiles are striped across two disks, then the multiplexing level is too high. In this case, you should set MAXOPENFILES to a lower value such as 6.

Step 6: Determine Whether Files Are Empty or Contain Few Changes

When performing a full backup of files that are largely empty, or when performing an incremental backup when few blocks have changed, you may not be able to supply data to the tape fast enough to keep it streaming. In either case, you can improve performance by increasing the level of multiplexing.

Note: A good way to test whether your tape backup performance is slow because of empty files is to try backing up archived logs, which contain nothing but data (in other words, they do not contain empty space).

An incremental backup is an RMAN backup in which only modified blocks are backed up. Incremental backups are not necessarily faster than full backups because Oracle still reads the entire datafile to take an incremental backup. If tape drives are not locally attached, then incremental backups can be faster. You must consider how much bandwidth exists for reading the disks compared to the bandwidth for writing to the tapes. If tape bandwidth is limited compared to disk, then incremental backups may help.

If only a few blocks have changed in an incremental backup, then you need to input many buffers from the datafile before you accumulate enough blocks to fill a buffer and write to tape. Hence, the tape drive may not stream.

If you set the level of multiplexing (as described in "Step 5: Adjust the Level of Multiplexing" on page 14-14) to a large value, then you can scan many datafiles in parallel, the output buffers for the tape drive are filled quickly, and you can write them frequently o keep the drive streaming. The `FILESPERSET` value should be less than or equal to `MAXOPENFILES`. For example, set both parameters to `8`, and raise this value if the tape drive does not stream. For an incremental backup, `50` is a good value for the level of multiplexing. For a full or incremental level 0 backup, the level of multiplexing should be a lower value such as `4` or `8`.

Step 7: Use the BLKSIZE Parameter to Alter Tape Buffer Size

If the tape is not streaming, but the problem is not due to an incremental backup or by backing up empty files, then you can try adjusting the block size of the tape buffer. You can change the size of each tape buffer using the `PARMS` parameter of the `ALLOCATE CHANNEL` or `CONFIGURE CHANNEL` command. If the `BLKSIZE` parameter for PARMS is supported on your platform, then you can set it to the desired size of each buffer. For example, configure an sbt channel as follows:

```
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS="BLKSIZE=524288";
```

A good rule of thumb is to set BLKSIZE to a value that is a little less than the tape block size of the media manager. What "a little less" means depends on the media manager. For example, if the tape block size is 512 KB and the media manager has a header of size 16 KB, then you can set BLKSIZE=49600.

Note that it is also a good idea to increase the media management physical tape block size. For example, you do not want to set the BLKSIZE parameter to 512 KB and leave the physical tape block size as 32 KB.

Step 8: Query V$ Views to Identify Bottlenecks

If none of the previous steps improves backup performance, then try to determine the exact source of the bottleneck. 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.

This section contains these topics:

- Identifying Bottlenecks with Synchronous I/O
- Identifying Bottlenecks with Asynchronous I/O

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

To determine whether your tape is streaming when the I/O is synchronous, query the EFFECTIVE_BYTES_PER_SECOND column in the V$BACKUP_SYNC_IO or V$BACKUP_ASYNC_IO view. Table 14–4 describes how to use this column.

Table 14–4 V$BACKUP_SYNC_IO View

If EFFECTIVE_BYTES_PER_SECOND is . . .	Then . . .
Less than the raw capacity of the hardware	The tape is not streaming. When the tape is not streaming the performance drop is typically huge.

Table 14–4 V$BACKUP_SYNC_IO View

If EFFECTIVE_BYTES_PER_SECOND is . . .	Then . . .
More than the raw capacity of the hardware	The tape may be streaming, depending on the compression ratio of the data.

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. The `DISCRETE_BYTES_PER_SECOND` column in the `V$BACKUP_SYNC_IO` view displays the I/O rate. Note that if you see data in `V$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 time when I/O was already ready for use and so there was no need to made an operating system call to poll for I/O completion.

The simplest way to identify the bottleneck is to query `V$BACKUP_ASYNC_IO` for the datafile that has the largest ratio for `LONG_WAITS` divided by `IO_COUNT`.

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: *Oracle9i Database Reference* for descriptions of the `V$BACKUP_SYNC_IO` and `V$BACKUP_ASYNC_IO` views

15 Recovery Manager Troubleshooting

This chapter describes how to troubleshoot Recovery Manager. This chapter contains these topics:

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. This section contains these topics:

- Identifying Types of Message Output
- Recognizing RMAN Error Message Stacks
- Identifying Error Codes
- Interpreting RMAN Error Stacks
- Identifying RMAN Return Codes

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.

Type of Output	Produced By	Location	Description
RMAN messages	RMAN	Direct this output to: ■ Standard output (typically the terminal) ■ A log file specified by `LOG` on the command line or the `SPOOL LOG` command ■ A file created by redirecting RMAN output by command line options such as UNIX > operator	Contains actions relevant to the RMAN job as well as error messages generated by RMAN, the server, and the media vendor. RMAN error messages have an `RMAN-`*`xxxxx`* prefix. Normal action descriptions do not have a prefix.
`alert_`*`SID`*`.log`	Oracle database server	The directory named in the `USER_DUMP_DEST` initialization parameter.	Contains a chronological log of errors, initialization parameter settings, and administration operations. Records values for overwritten control file records (refer to "Monitoring the Overwriting of Control File Records" on page 16-32).

Type of Output	Produced By	Location	Description
Oracle trace file	Oracle database server	The directory specified in the USER_DUMP_DEST initialization parameter.	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 Oracle fails to load the media management library.
sbtio.log	Third-party media management software	The directory specified in the USER_DUMP_DEST initialization parameter.	Contains vendor-specific information written by the media management software. Note that 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

On various occasions it may be important for you to determine whether RMAN successfully executed a command. For example, if you are trying to write a script that performs an unattended backup using RMAN, you may want to know whether the backup was a success or failure.

One way to determine whether RMAN encountered an error is to examine its return code, as described in "Identifying RMAN Return Codes" on page 15-9. A second way is to search the Recovery Manager 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. Following is sample output for a 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:** *Oracle9i Database Error Messages* for explanations of `RMAN` and `ORA` error codes

RMAN Error Message Numbers

Table 15–1 indicates the error ranges for common RMAN error messages, all of which are described in *Oracle9i Database Error Messages*.

Table 15–1 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

Media Manager Error Numbers

When errors occur through the media management API, RMAN returns an error message number prefixed as follows:

```
Additional information:
```

Table 15–2 lists media manager message numbers and their corresponding error text. In the error codes, *O/S* stands for *operating system*. The errors prefixed with an asterisk are internal and should never be seen during normal operation.

Table 15–2 Media Manager Error Message Ranges **(Page 1 of 2)**

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

Table 15–2 Media Manager Error Message Ranges **(Page 2 of 2)**

Cause	No.	Message
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
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:

- Because most of the messages in the error stack are not meaningful for troubleshooting, try to identify the one or two errors that are most important.
- Check for a line that says `Additional information` followed by an integer. This line indicates a media management error. The integer that follows refers to a code that is explained in the text of the error message.

- 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 informative.
- Look for the `RMAN-03002` or `RMAN-03009` message (`RMAN-03009` is the same as `RMAN-03002` but includes the channel ID), which immediately follows the 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 15–1 and then refer to *Oracle9i 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-01
using channel ORA_DISK_1
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 08/29/2001 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 `usesr` 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 08/29/2001 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 Oracle 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 Media Management Errors: Example

Media management errors in RMAN message output are not uncommon. Assume that you use a tape drive and receive 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 09/04/2001 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
```

Following the suggestions for reading error message stacks, you look for the `Additional information` line and notice:

```
Additional information: 7005
```

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 server. Hence, you must consult your media vendor documentation to interpret the error codes and messages.

Identifying RMAN Return Codes

One way to determine whether RMAN encountered an error is to examine its return code. RMAN returns 0 to the operating system if no errors occurred, a nonzero value otherwise. For example, if you are running UNIX with the C shell, then, when RMAN completes, the return code will be in a shell variable called `$status`.

Testing the Media Management API

On specific platforms, Oracle provides a diagnostic tool called `sbttest`. This utility performs a simple test of the media management software by acting as the Oracle database server and attempting to communicate with the media manager.

Obtaining the sbttest Utility

On UNIX, the `sbttest` utility is 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]>
```

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. The following table explains how to interpret the output.

If sbttest returns . . .	Then . . .
0	The program ran without error. In other words, the media manager is installed and can accept a data stream and return the same data when requested.
a nonzero value	The program encountered an error. 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 Oracle 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 Oracle server is not linked with the media management library, then `sbttest` can still work.
- The `sbttest` program passes all environment parameters from the shell but RMAN does not.

Terminating an RMAN Command

You have the following methods for terminating an RMAN command while it is executing:

- Press CTRL+C (or the equivalent "attention" key combination for your system) in the RMAN interface, which is the preferred method. This operation also terminates allocated channels unless they are hung in the media management code, for example, when they are waiting for a tape to be mounted.
- Kill the server session corresponding to the RMAN channel by running the SQL ALTER SYSTEM statement.
- Terminate the server session corresponding to the RMAN channel on the operating system.

Terminating the Session with ALTER SYSTEM

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 by specifying the `sid` in a SQL statement, but the commands are not the same as the operating system process `kill` commands.

You can specify the `sid` in the SQL statement `ALTER SYSTEM KILL SESSION` command. It 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 is no more effective than killing at the operating system level if the process is hung in the media manager.

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 the try killing the Oracle processes of the connections. Note that killing the Oracle process can cause problems for the media manager.

This section contains these topics:

- Components of an RMAN Session
- Process Behavior During a Hung Job
- Terminating an RMAN Session: Basic Steps

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 process** itself
- The **catalog connection** to the recovery catalog database (only if you use a recovery catalog)
- An **auxiliary connection** to an auxiliary instance (if running `DUPLICATE` or performing TSPITR, none otherwise)
- The initial connection to the target database, also called the **default channel**
- 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.
- One **target connection** to the target database corresponding to each allocated channel

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 seem 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. Target and auxiliary connections that are not hung in the media manager code are also terminated: only the target and auxiliary connections executing in the media management layer remains active. You must manually kill this process because terminating its session does not kill it. Even after termination, the media manager may keep resources busy or continue processing because it does not realize that the Oracle process is gone. This behavior depends on which media manager you use.

Terminating the catalog connection does not cause RMAN to finish 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

The best way to terminate RMAN when the connections for the allocated channels are hung in the media manager is to kill the Oracle process of the connections. 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 caveat 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 "Monitoring RMAN Interaction with the Media Manager" on page 15-22. 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, EVENT, SECONDS_IN_WAIT AS SEC_WAIT,
       STATE, 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. Kill the hung processes with an operating system utility. For example, on Solaris execute a `kill -9` command:

```
% kill -9 8642 8374
```

3. Check that the media manager also clears its processes, or else the next backup or restore may still hang 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 is unhelpful, ask the media manager technical support for the correct solution.

See Also: Your operating system specific documentation for the relevant commands

Monitoring RMAN Through V$ Views

In general, you can use `LIST`, `REPORT`, and `SHOW` to obtain most RMAN information: backups, copies, database incarnations, configuration settings, and so forth. Nevertheless, you may at times want to use `V$` views for information that these commands do not display.

Sometimes it is useful to identify what a server session performing a backup or copy operation is doing. You have access to several views that can assist in monitoring the progress of or obtaining information about RMAN jobs, as described in the following table.

View	Description
`V$PROCESS`	Identifies currently active processes.
`V$RECOVER_FILE`	Identifies which datafiles require recovery.
`V$SESSION`	Identifies currently active sessions. Use this view to determine which Oracle database server sessions correspond to which RMAN allocated channels.
`V$SESSION_LONGOPS`	Provides progress reports on RMAN backup and restore jobs.
`V$SESSION_WAIT`	Lists the events or resources for which sessions are waiting.
`V$BACKUP_SYNC_IO`	Displays rows when the I/O is synchronous to the process (or thread on some platforms) performing the backup.
`V$BACKUP_ASYNC_IO`	Displays rows when the I/O is asynchronous to the process (or thread on some platforms) performing the backup.

`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.

You can use RMAN to perform the checks discussed in the following sections:

- Correlating Server Sessions with RMAN Channels

- Monitoring RMAN Job Progress
- Monitoring RMAN Interaction with the Media Manager
- Monitoring RMAN Job Performance
- Determining Which Datafiles Require Recovery

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 NT 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%'
/
```

If you do not run the `SET COMMAND ID` command in the RMAN job, then the `CLIENT_INFO` column displays in the following format:

```
rman channel=channel_id
```

For example, the following shows sample output:

```
 SID SPID         CLIENT_INFO
---- ------------ ------------------------------
  14 8374         rman channel=ORA_SBT_TAPE_1
```

Matching Server Sessions with Channels in Multiple RMAN Sessions

If more than one RMAN session is active, 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 sbt;
     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:** *Oracle9i Recovery Manager Reference* for `SET COMMAND ID` syntax, and *Oracle9i Database Reference* for more information on `V$SESSION` and `V$PROCESS`

Monitoring RMAN Job Progress

Monitor the progress of backups, copies, and restores 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 15–3 describes column 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 15–3 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`	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 15–3 Columns of V$SESSION_LONGOPS Relevant for RMAN

Column	Description for Detail Rows
`TOTALWORK`	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, restore, or copy reports its progress compared to the total amount of work required for a job step. For example, if you perform a database restore that uses 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 that set is completely restored, RMAN begins reporting progress on the next set to restore.

To monitor job progress:

1. After connecting to the target database and, if desired, the recovery catalog database, start an RMAN job. For example, enter:

```
RESTORE DATABASE;
```

2. While the job is running, execute a script 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
/
```

If you repeat the query while the restore 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
```

3. If you run the 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 15-22 to obtain more information.

You can also run the following Perl script from the UNIX command line. It creates a temporary SQL query script and then runs the script in an infinite `while` loop. When RMAN completes the job, cancel the Perl script:

```
#!/usr/local/bin/perl5
open (TMP, ">/tmp/test.sql")
      || die "can't open /tmp/test.sql\n";
print TMP
'SET FEEDBACK OFF
SPOOL /tmp/sql.out
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;
SPOOL OFF
EXIT;
EOF';
close TMP;
while (1)
{       # replace with your sqlplus connection information
        ‘sqlplus "/ AS SYSDBA" @/tmp/test‘;
        open(SQLOUT, "</tmp/sql.out")
                || die "can't open /tmp/sql.out\n";
        @whole = <SQLOUT>;
        foreach (@whole)  {print if defined;}
        close(SQLOUT);
}
```

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:

```
sbtinit
sbtopen
sbtread
sbtwrite
sbtbackup
```

Before making a call to any of functions in the media management API, the server adds a row in `V$SESSION_WAIT`, with the `STATUS` column including the string `WAIT`. The `V$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,
       STATE, 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 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 sbtbackup         600 WAITING              rman channel=ORA_SBT_TAPE_1
```

Note: The V$SESSION_WAIT view shows only Oracle events, not media manager events.

See Also: *Oracle9i Database Reference* for descriptions of V$SESSION_WAIT

Monitoring RMAN Job Performance

Monitor backup and restore performance by querying V$BACKUP_SYNC_IO and V$BACKUP_ASYNC_IO.

See Also: *Oracle9i Database Reference* for more information on these V$ views, and "Step 8: Query V$ Views to Identify Bottlenecks" on page 14-16 to learn how to use these views to tune backup performance

Determining Which Datafiles Require Recovery

You can often use the dynamic performance view V$RECOVER_FILE to determine which files need to be recovered and why they need to be recovered. The following query shows the file numbers of datafiles that require recovery, as well as the reason for recovery (if known) and the SCN and time when recovery needs to begin:

```
COL FILE# FORMAT 999
COL ERROR FORMAT a10
SELECT * FROM V$RECOVER_FILE
/

FILE# ONLINE  ONLINE_ ERROR         CHANGE# TIME
----- ------- ------- ---------- ---------- --------------------
    4 ONLINE  ONLINE  FILE NOT            0
                      FOUND
    5 ONLINE  ONLINE  FILE NOT            0
                      FOUND
    8 OFFLINE OFFLINE OFFLINE             0
                      NORMAL
```

Note: The view 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 needed to update V$RECOVER_FILE accurately.

Query V$DATAFILE and V$TABLESPACE to obtain filenames and tablespace names for datafiles requiring recovery. For example, enter:

```
SELECT d.NAME, t.NAME
FROM V$DATAFILE d, V$TABLESPACE t
WHERE t.TS# = d.TS#
AND d.FILE# IN (14,15,21);  # use values obtained from V$RECOVER_FILE query

NAME                              NAME
--------------------------------  ---------------
/oracle/oradata/trgt/drsys01.dbf   DRSYS
/oracle/oradata/trgt/example01.dbf EXAMPLE
/oracle/oradata/trgt/users01.dbf   USERS
```

You can combine these queries in the following SQL*Plus script (sample output follows):

```
COL df# FORMAT 999
COL df_name FORMAT a35
COL tbsp_name FORMAT a10
COL status FORMAT a7
COL error FORMAT a10

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#
/

 DF# DF_NAME                             TBSP_NAME  STATUS  ERROR         CHANGE# TIME
---- ----------------------------------- ---------- ------- ---------- ---------- --------
   4 /oracle/oradata/trgt/drsys01.dbf    DRSYS      ONLINE  FILE NOT            0
                                                            FOUND
   5 /oracle/oradata/trgt/example01.dbf  EXAMPLE    ONLINE  FILE NOT            0
                                                            FOUND
   8 //oracle/oradata/trgt/users01.dbf   USERS      OFFLINE OFFLINE             0
                                                            NORMAL
```

RMAN Troubleshooting Scenarios

This section contains these topics:

- After Installation of Media Manager, RMAN Channel Allocation Fails: Scenario
- Backup Job Is Hanging: Scenario
- RMAN Fails to Start RPC Call: Scenario

- Backup Fails with Invalid RECID Error: Scenario
- Backup Fails Because of Control File Enqueue: Scenario
- RMAN Fails to Delete All Archived Logs: Scenario
- Backup Fails Because RMAN Cannot Locate an Archived Log: Scenario
- RMAN Cannot Set Target Database Character Set: Scenario
- RMAN Does Not Recognize Character Set Name: Scenario
- RMAN Denies Logon to Target Database: Scenario
- Database Duplication Fails Because of Missing Log: Scenario
- Duplication Fails with Multiple RMAN-06023 Errors: Scenario
- UNKNOWN Database Name Appears in Recovery Catalog: Scenario

After Installation of Media Manager, RMAN Channel Allocation Fails: Scenario

In this scenario, you install and test the media manager as explained in "Configuring RMAN to Make Backups to a Media Manager" on page 8-2, but you still cannot make RMAN back up to tape. For example, after allocating the `sbt` channel, you receive an error stack similar to the following:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03009: failure of allocate command on c1 channel at 08/29/2001 17:16:54
ORA-19554: error allocating device, device type: SBT_TAPE, device name:
ORA-27211: Failed to load Media Management Library
Additional information: 25
```

After Installation of Media Manager, RMAN Channel Allocation Fails: Diagnosis

The `ORA-27211` error indicates that the channel allocation is failing because Oracle is not loading the media management library. If the channel allocation fails, then Oracle generates a trace file in the `USER_DUMP_DEST` location that contains the error that caused the channel allocation to fail. The trace file should have the complete path name of the media management library loaded by Oracle as well as any other media manager errors or operating system errors. For example, the trace file on UNIX may be called something like `/oracle/rdbms/log/prod1_ora_16226.trc`, and may contain information such as the following:

```
*** 2001-08-29 17:16:54.385
SKGFQ OSD: Error in function sbtinit on line 2396
SKGFQ OSD: Look for SBT Trace messages in file /oracle/rdbms/log/sbtio.log
```

```
SBT Initialize failed for oracle.static
```

The last line of this output indicates that Oracle is loading the default static library instead of the media management library that you installed.

To test the loading of the media management library, try allocating a channel by using the `PARMS` parameter `SBT_LIBRARY` to force the loading of the media management library. For example, if your library is called `/vendor/lib/some_mm_lib.so` and is pointed to by `$ORACLE_HOME/lib/libobk.so`, then run a command such as the following, making sure to specify whatever `PARMS` settings are required by your media manager:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS='SBT_LIBRARY=?/lib/libobk.so',
          'ENV=(NSR_SERVER=tape_svr,NSR_CLIENT=oracleclnt,NSR_GROUP=oracle_tapes)';
}
```

If the channel allocation fails, then check the trace file again to see whether you can learn anything new. If the channel allocation with `SBT_LIBRARY` succeeds, but an ordinary `sbt` channel allocation fails, then Oracle is probably trying to load a library (`$ORACLE_HOME/lib/libobk.so` on UNIX, `%ORACLE_HOME%/bin/orasbt.dll` on NT) other than the one you installed. You may have more than one library in the operating system path, and the one that Oracle is loading is the wrong one.

After Installation of Media Manager, RMAN Channel Allocation Fails: Solution

If the problem is that Oracle is not loading the correct library, then make sure that the library is named correctly. For example, on UNIX you should name it `$ORACLE_HOME/rdbms/libobk.so` or create a symbolic link with this name that points to your library, and on Windows NT you should name it `%ORACLE_HOME%/bin/orasbt.dll`. It is possible to place these files in nondefault directories, but they must be included in the system path so that Oracle can locate them.

See Also: *Oracle9i Recovery Manager Reference* for descriptions of the legal `PARMS` parameters

Backup Job Is Hanging: Scenario

In this scenario, an RMAN backup job starts as normal and then pauses inexplicably:

```
Recovery Manager: Release 9.2.0.0.0 - Production
connected to target database: TRGT
connected to recovery catalog database

RMAN> BACKUP TABLESPACE SYSTEM, tools;

allocated channel: t1
channel t1: sid=16 devtype=SBT_TAPE

channel t1: starting datafile backupset
set_count=15 set_stamp=338309600
channel t1: including datafile 2 in backupset
channel t1: including datafile 1 in backupset
channel t1: including current controlfile in backupset
# Hanging here for 30 minutes now
```

Backup Job Is Hanging: Diagnosis

If a backup job is hanging, that is, not proceeding, then several scenarios are possible:

- The job abnormally terminated.
- A server-side or media management error occurred.
- RMAN is waiting for an event such as the insertion of a new cassette into the tape device.

As described in "Monitoring RMAN Interaction with the Media Manager" on page 15-22, you can query `sbt` wait events to gain more information. For example, run the following query on the target instance:

```
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,
       STATE, 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 sbtbackup         300 WAITING             rman channel=ORA_SBT_TAPE_1
```

Backup Job Is Hanging: Solution

Because the causes of a hung backup job can be varied, so are the solutions. For example, backup jobs often hang simply because the tape device has completely filled the current cassette and is waiting for a new tape to be inserted. Ideally, the query of the `sbt` wait events should indicate the problem.

If the `sbt` wait event query is unhelpful, then examine media manager process, log, and trace files for signs of abnormal termination or other errors (refer to the description of message files in "Identifying Types of Message Output" on page 15-2).

See Also: "Terminating an RMAN Session: Basic Steps" on page 15-14 to learn how to kill an RMAN session that is hanging

RMAN Fails to Start RPC Call: Scenario

In this scenario, you run a backup job and receive message output similar to the following:

```
channel c8: including datafile number 47 in backupset
RPC call appears to have failed to start on channel c9
RPC call ok on channel c9
channel c3: including datafile number 18 in backupset
```

RMAN Fails to Start RPC Call: Diagnosis

The `RPC call appears to have failed` message does not usually indicate a problem. The message indicates one of the following:

- The target database instance is slow.
- A timing problem occurred.

Timing problems occur in this way. When RMAN begins an RPC, it checks the `V$SESSION` performance view. The RPC updates the information in the view to indicate when it starts and finishes. Sometimes RMAN checks `V$SESSION` before the RPC has indicated it has started, which in turn generates the following message:

```
RPC call appears to have failed
```

If a message stating "`RPC call ok`" does not appear in the output immediately following the message stating "`RPC call appears to have failed`", then the backup job encountered a problem.

Backup Fails with Invalid RECID Error: Scenario

In this scenario, you attempt a backup and receive the following error messages:

```
RMAN-3014: Implicit resync of recovery catalog failed
RMAN-6038: Recovery catalog package detected an error
RMAN-20035: Invalid high RECID error
```

Backup Fails with Invalid RECID Error: Diagnosis

In one common scenario, you restore a backup control file created through a non-Oracle mechanism, and then open the database without the `RESETLOGS` option. If you had created the backup control file through the RMAN `BACKUP` command or the SQL `ALTER DATABASE BACKUP CONTROLFILE` statement, then Oracle would have required you to reset the online logs.

The control file and the recovery catalog are now not synchronized. The database control file is older than the recovery catalog, because at one time the recovery catalog resynchronized with the old current control file, and now the database is using a backup control file. RMAN detects that the control file currently in use is older than the control file previously used to resynchronize.

Another common scenario occurs when you attempt to copy the target database to a new machine as follows:

1. On machine 1, you shut down the database and make a copy of the control file with an operating system utility. You do not use `CATALOG` to add this control file copy to the repository.
2. You transfer the control file copy to machine 2.
3. On machine 2, you create a new initialization parameter file and new database instance.
4. You mount the control file copy on machine 2. Oracle does not recognize the control file as a backup control file: to Oracle it looks like the current control file.
5. You start RMAN and connect to the new target database and the recovery catalog on machine 2. Because the control file was not created with RMAN and

was not cataloged as a control file copy, RMAN sees the database on machine 2 as the database on machine 1.

6. You restore and recover database the new database on machine 2 and then open it. As a consequence, various records are added to the recovery catalog during the restore and recovery. For example, the highest `RECID` in the recovery catalog moves from 90 to 100.
7. On machine 1, you start RMAN and connect to the original target database and recovery catalog. The recovery catalog indicates that the highest `RECID` is 100, but the control file indicates that the highest `RECID` is 90. The control file `RECID` should always be greater than or equal to the recovery catalog `RECID`, so RMAN issues `RMAN-20035`.

Backup Fails with Invalid RECID Error: Solution 1

This solution is safest and is strongly recommended:

To reset the database with RMAN:

1. Connect to the target database with SQL*Plus. For example, enter:

```
% sqlplus '/ AS SYSDBA'
```

2. Mount the database if it is not already mounted. For example, enter:

```
ALTER DATABASE MOUNT;
```

3. Start cancel-based recovery by using the backup control file, then cancel it. The reason for canceling is that the `USING BACKUP CONTROLFILE` clause stamps the controlfile as a backup, which then permits `OPEN RESETLOGS`. For example, enter:

```
ALTER DATABASE RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE;
ALTER DATABASE RECOVER CANCEL;
```

4. Open the database with the `RESETLOGS` option. For example, enter:

```
ALTER DATABASE OPEN RESETLOGS;
```

5. Use RMAN to connect to the target database and recovery catalog. For example, enter:

```
% rman TARGET SYS/oracle@trgt CATALOG rman/cat@catdb
```

6. Reset the database. For example, enter:

```
RESET DATABASE;
```

7. Take new backups so that you can recover the database if necessary. For example, enter:

   ```
   BACKUP DATABASE PLUS ARCHIVELOG;
   ```

Backup Fails with Invalid RECID Error: Solution 2

This solution is more difficult than solution 1:

To create the control file with SQL*Plus:

1. Connect to the target database with SQL*Plus. For example, enter:

   ```
   % sqlplus 'SYS/oracle@trgt AS SYSDBA'
   ```

2. Mount the database if it is not already mounted:

   ```
   SQL> ALTER DATABASE MOUNT;
   ```

3. Back up the control file to a trace file:

   ```
   SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
   ```

4. Edit the trace file as necessary. The relevant section of the trace file looks something like the following:

   ```
   # The following commands will create a new control file and use it
   # to open the database.
   # Data used by the recovery manager will be lost. Additional logs may
   # be required for media recovery of offline data files. Use this
   # only if the current version of all online logs are available.
   STARTUP NOMOUNT
   CREATE CONTROLFILE REUSE DATABASE "TRGT" NORESETLOGS  ARCHIVELOG
   --  STANDBY DATABASE CLUSTER CONSISTENT AND UNPROTECTED
       MAXLOGFILES 32
       MAXLOGMEMBERS 2
       MAXDATAFILES 32
       MAXINSTANCES 1
       MAXLOGHISTORY 226
   LOGFILE
     GROUP 1 '/oracle/oradata/trgt/redo01.log'  SIZE 25M,
     GROUP 2 '/oracle/oradata/trgt/redo02.log'  SIZE 25M,
     GROUP 3 '/oracle/oradata/trgt/redo03.log'  SIZE 500K
   -- STANDBY LOGFILE
   DATAFILE
     '/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'
CHARACTER SET WE8DEC
;
# Take files offline to match current control file.
ALTER DATABASE DATAFILE '/oracle/oradata/trgt/tools01.dbf' OFFLINE;
ALTER DATABASE DATAFILE '/oracle/oradata/trgt/users01.dbf' OFFLINE;
# Configure RMAN configuration record 1
VARIABLE RECNO NUMBER;
EXECUTE :RECNO := SYS.DBMS_BACKUP_RESTORE.SETCONFIG('CHANNEL','DEVICE TYPE DISK
DEBUG 255');
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# All logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;
# Database can now be opened normally.
ALTER DATABASE OPEN;
# Commands to add tempfiles to temporary tablespaces.
# Online tempfiles have complete space information.
# Other tempfiles may require adjustment.
ALTER TABLESPACE TEMP ADD TEMPFILE '/oracle/oradata/trgt/temp01.dbf' REUSE;
# End of tempfile additions.
```

5. Shut down the database:

   ```
   SHUTDOWN IMMEDIATE
   ```

6. Execute the script to create the control file, recover (if necessary), archive the logs, and open the database:

   ```
   STARTUP NOMOUNT
   CREATE CONTROLFILE ...;
   EXECUTE ...;
   RECOVER DATABASE
   ALTER SYSTEM ARCHIVE LOG CURRENT;
   ALTER DATABASE OPEN ...;
   ```

Caution: If you do *not* open with the `RESETLOGS` option, then two copies of an archived redo log for a given log sequence number may exist—even though these two copies have completely different contents. For example, one log may have been created on the original host and the other on the new host. If you accidentally confuse the logs during a media recovery, then the database will be corrupted but Oracle and RMAN cannot detect the problem.

Backup Fails Because of Control File Enqueue: Scenario

In this scenario, a backup job fails because RMAN cannot make a snapshot control file. The message stack is as follows:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 08/30/2001 22:48:44
ORA-00230: operation disallowed: snapshot controlfile enqueue unavailable
```

Backup Fails Because of Control File Enqueue: Diagnosis

When RMAN needs to back up or resynchronize from the control file, it first creates a **snapshot** or consistent image of the control file. If one RMAN job is already backing up the control file while another needs to create a new snapshot control file, then you may see the following message:

```
waiting for snapshot controlfile enqueue
```

Under normal circumstances, a job that must wait for the control file enqueue waits for a brief interval and then successfully obtains the enqueue. RMAN makes up to five attempts to get the enqueue and then fails the job. The conflict is usually caused when two jobs are both backing up the control file, and the job that first starts backing up the control file waits for service from the media manager.

To determine which job is holding the conflicting enqueue:

1. After you see the first message stating "`RMAN-08512: waiting for snapshot controlfile enqueue`", start a new SQL*Plus session on the target database:

   ```
   % sqlplus 'SYS/oracle@trgt AS SYSDBA'
   ```

2. Execute the following query to determine which job is causing the wait:

   ```
   SELECT s.SID, USERNAME AS "User", PROGRAM, MODULE,
          ACTION, LOGON_TIME "Logon", l.*
   FROM V$SESSION s, V$ENQUEUE_LOCK l
   WHERE l.SID = s.SID
   AND l.TYPE = 'CF'
   AND l.ID1 = 0
   AND l.ID2 = 2;
   ```

   You should see output similar to the following (the output in this example has been truncated):

   ```
   SID User Program              Module                    Action           Logon
   --- ---- -------------------- ------------------------- ---------------- ---------
   ```

```
     9 SYS  rman@h13 (TNS V1-V3) backup full datafile: c1  0000210 STARTED  21-JUN-01
```

Backup Fails Because of Control File Enqueue: Solution

After you have determined which job is creating the enqueue, you can do one of the following:

- Wait until the job creating the enqueue completes
- Cancel the current job and restart it after the job creating the enqueue completes
- Cancel the job creating the enqueue

Commonly, enqueue situations occur when a job is writing to a tape drive, but the tape drive is waiting for a new cassette to be inserted. If you start a new job in this situation, then you will probably receive the enqueue message because the first job cannot complete until the new tape is loaded.

RMAN Fails to Delete All Archived Logs: Scenario

In this scenario, the database archives automatically to two directories: `?/oradata/trgt/arch` and `?/oradata/trgt/arch2`. You tell RMAN to perform a backup and delete the input archived redo logs afterward in the following script:

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

You then run a crosscheck to make sure the logs are gone and find the following:

```
CROSSCHECK ARCHIVELOG ALL;

validation succeeded for archived log
archivelog filename=/oracle/oradata/trgt/arch2/archive1_964.arc recid=19 stamp=368726072
```

RMAN deleted one set of logs but not the other.

RMAN Fails to Delete All Archived Logs: Diagnosis

This problem is not an error. When you specify `DELETE INPUT` without the `ALL` keyword, RMAN deletes only one copy of each input log. Even if you archive to five destinations, RMAN deletes logs from only one directory.

RMAN Fails to Delete All Archived Logs: Solution

To force RMAN to delete all existing archived redo logs, use the `DELETE ALL INPUT` clause of the `BACKUP` command. For example, enter:

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

Backup Fails Because RMAN Cannot Locate an Archived Log: Scenario

In this scenario, you schedule regular backups of the archived redo logs. The next time you make a backup, you receive this error:

```
RMAN-6089:  archive log NAME not found or out of sync with catalog
```

Backup Fails Because RMAN Cannot Locate an Archived Log: Diagnosis

This problem occurs when the archived log that RMAN is looking for cannot be accessed by RMAN, or the recovery catalog needs to be resynchronized. Often, this error occurs when you delete archived logs with an operating system command, which means that RMAN is unaware of the deletion. The `RMAN-6089` error occurs because RMAN attempts to back up a log that the repository indicates still exists.

Backup Fails Because RMAN Cannot Locate an Archived Log: Solution

Make sure that the archived logs exists in the specified directory and that the RMAN catalog is synchronized. Check the following:

1. Make sure the archived log file that is specified by the `RMAN-6089` error exists in the correct directory.
2. Check that the operating system permissions are correct for the archived log (`owner` = `oracle`, `group` = `DBA`) to make sure that RMAN can access the file.
3. If the file appears to be correct, then try synchronizing the catalog by running the following command from the RMAN prompt:

   ```
   RESYNC CATALOG;
   ```

If you know that the logs are unavailable because you deleted them by using an operating system utility, then run the following command at the RMAN prompt to update RMAN metadata:

```
CROSSCHECK ARCHIVELOG ALL;
```

It is always better to use RMAN to delete logs than to use an operating system utility. The easiest method to remove unwanted logs is to specify the `DELETE INPUT` option when backing up archived logs. For example, enter:

```
BACKUP DEVICE TYPE sbt
  ARCHIVELOG ALL
  DELETE ALL INPUT;
```

RMAN Cannot Set Target Database Character Set: Scenario

In this scenario, you are trying to connect to a release 8.0.4 target database. You receive the following error messages when you try to connect to the target database:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ====
RMAN-00571: ===========================================================
PLS-00201: identifier 'DBMS_BACKUP_RESTORE.SET_CHARSET' must be declared
ORA-06550: line 1, column 7: PL/SQL: Statement ignored
RMAN-04015: error setting target database character set to WE8ISO8859P1
```

RMAN Cannot Set Target Database Character Set: Diagnosis

Typically, this error message means that the `DBMS_BACKUP_RESTORE` package was not created during the installation of the database. Here are possible causes:

- The installation scripts contained errors.
- The PL/SQL option, which is required for RMAN, was never installed.

RMAN Cannot Set Target Database Character Set: Solution

If you did not install the PL/SQL option, then install it. If you did install the PL/SQL option, then create the required packages by connecting to SQL*Plus with `SYSDBA` privileges and running the following scripts:

```
SQL> @?/rdbms/admin/dbmsbkrs.sql
SQL> @?/rdbms/admin/prvtbkrs.plb
```

RMAN Does Not Recognize Character Set Name: Scenario

In this scenario, you are connected to the target database while it is not open and attempting to perform an RMAN operation. You receive the following error:

```
PLS-00553: character set name is not recognized
```

RMAN Does Not Recognize Character Set Name: Diagnosis

Typically, this message means that the character set in the client environment, that is, the environment in which you are running the RMAN executable, is different from the character set in the target database environment.

RMAN Does Not Recognize Character Set Name: Solution

1. Query the target database to determine the value of the `NLS_CHARACTERSET` parameter. For example, run this query:

```
SQL> SELECT VALUE FROM V$NLS_PARAMETERS WHERE PARAMETER='NLS_CHARACTERSET';
```

2. Set the character set environment variable in the client to the same value as the variable in the server. For example, you can set the NLS_LANG environment variable on a UNIX system as follows:

```
% setenv NLS_LANG american_america.we8dec
% setenv NLS_DATE_FORMAT "MON DD YYYY HH24:MI:SS"
```

RMAN Denies Logon to Target Database: Scenario

RMAN fails with the following errors when trying to connect to the target database:

```
% rman
Recovery Manager: Release 9.2.0.0.0 - Production

RMAN> CONNECT TARGET sys/change_on_install@inst1

RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
ORA-01017: invalid username/password; logon denied
```

Diagnosis of Cause

RMAN automatically requests a connection to the target database as SYSDBA. In order to connect to the target as SYSDBA, you must do one of the following:

- Be part of the operating system DBA group with respect to the target database (that is, have the ability to connect with SYSDBA privileges to the target database without a password).
- Create a password file with the orapwd command and the initialization parameter REMOTE_LOGIN_PASSWORDFILE.

If the target database does not have a password file, then the user you are logged in as must be validated with operating system authentication.

Solution

Either create a password file for the target database or add yourself to the administrator list in the operating system.

See Also: *Oracle9i Database Administrator's Guide* to learn how to create a password file

To learn how to create a password file, see

Database Duplication Fails Because of Missing Log: Scenario

In this scenario, you attempt to duplicate a database with the DUPLICATE command, but receive the following error stack:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of Duplicate Db command at 09/04/2001 12:11:29
RMAN-03015: error occurred in stored script Memory Script
RMAN-06053: unable to perform media recovery because of missing log
RMAN-06025: no backup of log thread 1 seq 16 scn 145858 found to restore
```

Database Duplication Fails Because of Missing Log: Diagnosis

The problem is that RMAN is not able to apply all the archived logs needed for complete recovery. For example, if you only backed up logs through sequence 15, but the most recent archived log is sequence 16, then DUPLICATE fails.

Database Duplication Fails Because of Missing Log: Solution

When creating the duplication script, use the SET UNTIL command to specify a log sequence number for incomplete recovery. For example, to terminate recovery after applying log sequence 15, enter:

```
RUN
{
  SET UNTIL SEQUENCE 16 THREAD 1;  # recovers up to but not including log 16
  DUPLICATE TARGET DATABASE TO 'dupdb';
}
```

See Also: "Creating a Non-Current Duplicate Database: Example" on page 12-22 for more information about performing incomplete recovery during the duplication operation

Duplication Fails with Multiple RMAN-06023 Errors: Scenario

In this scenario, you back up the database, then run the DUPLICATE command. You receive the following error stack:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of Duplicate Db command at 09/04/2001 13:55:11
RMAN-03015: error occurred in stored script Memory Script
RMAN-06026: some targets not found - aborting restore
RMAN-06023: no backup or copy of datafile 8 found to restore
RMAN-06023: no backup or copy of datafile 7 found to restore
```

```
RMAN-06023: no backup or copy of datafile 6 found to restore
RMAN-06023: no backup or copy of datafile 5 found to restore
RMAN-06023: no backup or copy of datafile 4 found to restore
RMAN-06023: no backup or copy of datafile 3 found to restore
RMAN-06023: no backup or copy of datafile 2 found to restore
RMAN-06023: no backup or copy of datafile 1 found to restore
```

Duplication Fails with Multiple RMAN-06023 Errors: Diagnosis

The `DUPLICATE` command recovers to archived redo logs, but cannot recover into online redo logs. Thus, if the restored backup cannot be made consistent without applying the online redo logs, then duplication fails with RMAN-06023 errors because RMAN is looking for backups created *before* the most recent archived log.

Duplication Fails with Multiple RMAN-06023 Errors: Solution

Issue the following statement after making a backup:

```
SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
```

This statement archives all records in the online redo logs so that RMAN can now recover the backup by applying the most recent archived redo log.

UNKNOWN Database Name Appears in Recovery Catalog: Scenario

In this scenario, you list the database incarnations registered in the recovery catalog and see a database with the name `UNKNOWN`:

```
LIST INCARNATION OF DATABASE;

RMAN-03022: compiling command: list
List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
56      57        TRGT      4052472287  YES    1            Sep 03 2001 06:45:51
1       19        UNKNOWN   4141147584  NO     1            Jan 08 2001 14:47:28
1       2         TRGT      4141147584  YES    14602        Jan 15 2001 15:32:57
```

UNKNOWN Database Name Appears in Recovery Catalog: Diagnosis

One way you get the `DB_NAME` of `UNKNOWN` is when you register a database that was once opened with the `RESETLOGS` option. The `DB_NAME` can be changed during a `RESETLOGS` operation, so RMAN does not know what the `DB_NAME` was for those old incarnations of the database because it was not registered in the recovery catalog at the time. Consequently, RMAN sets the `DB_NAME` column to `UNKNOWN` when creating the `DBINC` record.

UNKNOWN Database Name Appears in Recovery Catalog: Solution

The `UNKNOWN` name entry is expected behavior after a `RESETLOGS` operation. You should *not* attempt to remove `UNKNOWN` entries from the recovery catalog.

Part IV

Maintaining the Recovery Manager Repository

Part IV describes how to maintain and query the RMAN metadata that is stored in the target database control file and, optionally, the recovery catalog. This part contains these chapters:

16

Managing the Recovery Manager Repository

This chapter describes how to manage the RMAN repository. Depending on how you implement RMAN, you can store this data either in the recovery catalog or exclusively in the control file. This chapter contains these topics:

Creating the Recovery Catalog

This section contains these topics:

- Configuring the Recovery Catalog Database
- Creating the Recovery Catalog Owner
- Creating the Recovery Catalog

> **See Also:** "Using Basic RMAN Commands" on page 3-3 to learn how to connect to RMAN, and "Choosing an RMAN Authentication Method" on page 3-2 to learn about the advantages and disadvantages of maintaining a recovery catalog

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. You should not install the catalog in the target database: this tactic defeats the purpose of the catalog. Also, decide whether to operate the catalog database in `ARCHIVELOG` mode, which is recommended.

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 on the number of databases monitored by the catalog
- Depends on the number and size of RMAN scripts stored in the catalog
- Grows as the numbers of archived logs and backups for each database grow

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. Hence, 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 a more realistic estimate.

Allocating Disk Space for the Recovery Catalog Database

After either creating a new catalog database or finding an existing database to host the catalog, allocate disk space for the following:

- `SYSTEM` tablespace
- Temporary tablespaces
- Rollback segment 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 rollback or undo tablespaces. Table 16–1 describes typical space requirements.

Table 16–1 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
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 they are on the same disk and you lose one database, then you will probably lose the other.

Creating the Recovery Catalog 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` with password `oracle` has `SYSDBA` privileges on the recovery catalog database `catdb`.

- A tablespace called `tools` on the recovery catalog database `catdb` stores the recovery catalog. Note that to use a reserved word as a tablespace name, you must enclose it in quotes and put it in uppercase font.
- A tablespace called `temp` exists in the recovery catalog database.
- The database is configured in the same way as all normal databases, for example, `catalog.sql` and `catproc.sql` have successfully run.

To create the recovery catalog schema in the recovery catalog database:

1. Start SQL*Plus and then connect with administrator privileges to the database containing the recovery catalog. For example, enter:

   ```
   CONNECT SYS/oracle@catdb AS SYSDBA
   ```

2. Create a user and schema for the recovery catalog. For example, enter:

   ```
   CREATE USER rman IDENTIFIED BY cat
     TEMPORARY TABLESPACE temp
     DEFAULT TABLESPACE tools
     QUOTA UNLIMITED ON tools;
   ```

3. Grant the `RECOVERY_CATALOG_OWNER` role to the schema owner. This role provides the user with privileges to maintain and query the recovery catalog.

   ```
   SQL> GRANT RECOVERY_CATALOG_OWNER TO rman;
   ```

4. Grant other desired privileges to the RMAN user.

   ```
   SQL> GRANT CONNECT, RESOURCE TO rman;
   ```

Creating the Recovery Catalog

After creating the catalog owner, create the catalog itself 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. Connect to the database that will contain the catalog as the catalog owner. For example, enter the following from the operating system command line:

   ```
   % rman CATALOG rman/cat@catdb
   ```

   You can also connect from the RMAN prompt:

   ```
   % rman
   RMAN> CONNECT CATALOG rman/cat@catdb
   ```

2. Run the CREATE CATALOG command to create the catalog. If the catalog tablespace is this user's default tablespace, then you can run this command:

   ```
   CREATE CATALOG;
   ```

   Note that the creation of the catalog can take several minutes.

3. Optionally, start SQL*Plus and query the recovery catalog to see which tables were created:

   ```
   SQL> SELECT TABLE_NAME FROM USER_TABLES;
   ```

> **See Also:** *Oracle9i SQL Reference* for the SQL syntax for the GRANT and CREATE USER statements, and *Oracle9i Recovery Manager Reference* for CREATE CATALOG command syntax

Managing Target Database Records in the Recovery Catalog

This section describes how to register, unregister, and reset target database records in the recovery catalog.

This section contains these topics:

- Registering a Database in the Recovery Catalog
- Unregistering a Target Database from the Recovery Catalog
- Resetting a Database Incarnation in the Recovery Catalog

Registering a Database in the Recovery Catalog

Before using RMAN with a recovery catalog, you must register the target database in the recovery catalog. RMAN obtains all information it needs to register the target database from the database itself.

As long as each target database has a distinct DBID, you can register more than one target database in the same recovery catalog. For example, you can register target databases prod1, prod2, and prod3 in catalog schema rman1. Note that you cannot register a specific database multiple times in the same recovery catalog: registration occurs only once. However, you can register a target database in multiple recovery catalog schemas. For example, you can register target database prod1 in catalog schema rman1 and in schema rman2.

Each database registered in a given catalog must have a unique database identifier (DBID), but not necessarily a unique database name. When you copy a database with user-managed methods, the copy database has the same DBID as the original database. You can use the DBNEWID utility to change the DBID and database name of manually copied databases. However, the recommended method is to copy a database with the `DUPLICATE` command, because RMAN gives it a new DBID.

Note: If you use Oracle Enterprise Manager, then you can use also the Maintenance wizard to register the database.

To register the target database:

1. Connect to the target database and recovery catalog database. For example, issue the following to connect to the catalog database `catdb` as user `rman` (who owns the catalog schema):

   ```
   % rman TARGET / CATALOG rman/cat@catdb
   ```

2. If the target database is not mounted, then mount or open it. For example, issue:

   ```
   STARTUP MOUNT;
   ```

   The recovery catalog database must be open.

3. To use RMAN with a target database, you must first register the database. Run the following command:

   ```
   REGISTER DATABASE;
   ```

   After you run `REGISTER DATABASE`, RMAN creates rows in the repository that contain information about the target database. Then, RMAN performs a full resynchronization with the catalog in which it transfers all pertinent data about the target database from the control file and saves it in the catalog.

4. Test that the registration was successful by running `REPORT SCHEMA`. This command shows the database structure as it is stored in the repository. For example:

   ```
   RMAN> REPORT SCHEMA;

   Report of database schema
   File K-bytes    Tablespace          RB segs Datafile Name
   ---- ---------- ------------------- ------- -------------------
   1       307200 SYSTEM              ***     /oracle/oradata/trgt/system01.dbf
   2        20480 UNDOTBS             ***     /oracle/oradata/trgt/undotbs01.dbf
   3        10240 CWMLITE             ***     /oracle/oradata/trgt/cwmlite01.dbf
   4        10240 DRSYS               ***     /oracle/oradata/trgt/drsys01.dbf
   5        10240 EXAMPLE             ***     /oracle/oradata/trgt/example01.dbf
   6        10240 INDX                ***     /oracle/oradata/trgt/indx01.dbf
   ```

```
7         10240 TOOLS                ***     /oracle/oradata/trgt/tools01.dbf
8         10240 USERS                ***     /oracle/oradata/trgt/users01.dbf
```

5. If there are any existing user-created copies of datafiles or archived logs on disk that were created under Oracle release 8.0 or higher, you can add them to the recovery catalog with the CATALOG command. For example:

   ```
   CATALOG DATAFILECOPY '/tmp/users01.dbf';
   CATALOG ARCHIVELOG '/tmp/archive1_731.dbf', '/tmp/archive1_732.dbf';
   ```

   In some cases, datafile copies made of an Oracle7 database with operating systems can be cataloged. To be usable, the copies must not require any Oracle7 redo to be recovered, that is, they must be either of the following:

   - Datafile copies made when the database was shut down consistently. The database must not have been opened again before being migration.
   - Datafile copies made after a tablespace became offline normal or read-only. The tablespaces must not have been brought online or made read / write again before migration.

   Note: To determine whether log records have aged out of the control file, compare the number of logs on disk with the number of records in V$ARCHIVED_LOG.

   See Also:

   - *Oracle9i Recovery Manager Reference* for REGISTER syntax
   - *Oracle Enterprise Manager Administrator's Guide* to learn about RMAN restore and recovery
   - *Oracle9i Database Migration* for issues relating to database migration

Troubleshooting DBID Problems

Oracle uses an internal, uniquely generated number called the DBID to distinguish one database from another. Oracle generates the DBID at database creation. RMAN uses the DBID to distinguish one database from another. A problem can occur when you create a database by non-RMAN techniques (for example, user-managed backup and restore) instead of with a CREATE DATABASE statement or DUPLICATE command. In such cases, RMAN detects the duplicate database identifiers and the REGISTER DATABASE command fails.

To be able to register a copied database with the same DBID as an already registered database, use the DBNEWID utility to change the DBID of the copy database. You can also use this utility to change the database name. However, the best solution is to avoid the problem altogether by using the `DUPLICATE` command, which generates a new database identifier automatically.

See Also:

- *Oracle9i Recovery Manager Reference* for `CATALOG` syntax
- *Oracle9i Recovery Manager Reference* for `DUPLICATE` syntax
- *Oracle9i Database Utilities* to learn how to use the DBNEWID utility to change the DBID
- *Oracle9i Database Migration* for issues relating to database migration

Unregistering a Target Database from the Recovery Catalog

RMAN can unregister a database as well as register it. Make sure this procedure is what you intend, because if you make a mistake, then must reregister the database. In this case, you lose any metadata that is older than the `CONTROLFILE_RECORD_KEEP_TIME` setting in the target database control file.

To unregister a database:

1. Start RMAN and connect to the target database. Note down the DBID value that is displayed when you use RMAN to connect to the target database. For example, enter:

```
% rman TARGET / CATALOG rman/cat@catdb

connected to target database: RDBMS (DBID=1237603294)
connected to recovery catalog database
```

2. List the copies and backup sets recorded in the repository (refer to "Listing RMAN Backups, Copies, and Database Incarnations" on page 17-3). For example, enter:

```
LIST BACKUP SUMMARY;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Tag
------- -- -- - ----------- --------------- ------- ------- ---
19      B  A  A DISK        08-FEB-02       1       1       TAG20020208T155239
20      B  F  A DISK        08-FEB-02       1       1       TAG20020208T155242
```

```
21      B  A  A DISK        08-FEB-02       1       1       TAG20020208T155331
22      B  A  A DISK        08-FEB-02       1       1       TAG20020208T155604
```

3. Run DELETE statements to delete all existing physical backups (refer to "Deleting Backups and Copies" on page 18-4). For example:

   ```
   DELETE BACKUP DEVICE TYPE sbt;
   DELETE BACKUP DEVICE TYPE DISK;
   ```

   RMAN will list the backups that it intends to delete and prompt for confirmation before deleting them.

4. Use SQL*Plus to connect to the recovery catalog database as the catalog owner, then execute the following query in the recovery catalog to find the correct row of the DB table, setting DB_ID equal to the value you obtained from step 1. For example, enter:

   ```
   % sqlplus rman/cat@catdb
   SQL> SELECT DB_KEY, DB_ID FROM DB WHERE DB_ID = 1237603294;
   ```

   This query should return exactly one row.

   ```
   DB_KEY     DB_ID
   ---------- ----------
            1 1237603294
   1 row selected.
   ```

5. While still connected to the recovery catalog, enter the following, where DB_KEY and DB_ID are the corresponding columns from the row you got from the query in step 4:

   ```
   SQL> EXECUTE dbms_rcvcat.unregisterdatabase(db_key, db_id)
   ```

   For example, enter:

   ```
   SQL> EXECUTE dbms_rcvcat.unregisterdatabase(1, 1237603294)
   ```

Resetting a Database Incarnation in the Recovery Catalog

When you run either the RMAN command or the SQL statement ALTER DATABASE OPEN RESETLOGS, you create a new incarnation of the database. You can see a record of the new incarnation in the V$DATABASE_INCARNATION view of the target database.

If you run the RMAN command (not the SQL statement) ALTER DATABASE OPEN RESETLOGS, then RMAN automatically creates a new database incarnation record in the recovery catalog. RMAN implicitly and automatically issues a RESET

DATABASE command, which specifies that this new incarnation of the database is the current incarnation. RMAN associates all subsequent backups and log archiving done by the target database with the new database incarnation.

If you issue the SQL statement (not the RMAN command) ALTER DATABASE OPEN RESETLOGS, then RMAN does *not* automatically run a RESET DATABASE command. Hence, RMAN cannot access the recovery catalog because it cannot distinguish between a RESETLOGS command and an accidental restore of an old control file. To solve this problem, you must manually run the RESET DATABASE command in RMAN after executing the SQL statement ALTER DATABASE OPEN RESETLOGS. The RESET DATABASE command updates the repository to indicate that the target database has been opened with the RESETLOGS option.

In the rare situation in which you wish to restore backups of a prior incarnation of the database, use the RESET DATABASE TO INCARNATION *key* command to change the current incarnation to an older incarnation. For example, if you accidentally drop a table immediately after the most recent RESETLOGS, then you may want to recover the database to just before the time of the most recent RESETLOGS and then open it with the RESETLOGS option, thereby creating a new incarnation.

Note: If you use Oracle Enterprise Manager, then you can use also the Maintenance wizard to reset the database incarnation.

To reset the recovery catalog to an older incarnation:

1. Specify the primary key of the desired database incarnation. Obtain the incarnation key value by issuing a LIST command:

```
LIST INCARNATION;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         TRGT      1224038686  NO     1            02-JUL-01
1       582       TRGT      1224038686  YES    59727        10-JUL-01
```

2. Reset the database to the old incarnation. For example, enter:

```
RESET DATABASE TO INCARNATION 2;
```

3. 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;
}
```

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:

- *Oracle9i Recovery Manager Reference* for `RESET DATABASE` syntax, *Oracle9i Recovery Manager Reference* for `LIST` syntax
- *Oracle Enterprise Manager Administrator's Guide* to learn about the Maintenance wizard

Resynchronizing the Recovery Catalog

When RMAN performs a **resynchronization**, it compares the recovery catalog to either the current control file of the target database or a backup control file and updates it with information that is missing or changed.

This section contains the following topics:

- About Resynchronization
- Types of Records That Are Resynchronized
- When Should You Resynchronize?
- Forcing a Full Resynchronization of the Recovery Catalog
- Setting the CONTROLFILE_RECORD_KEEP_TIME Value

About Resynchronization

Resynchronizations can be full or partial. In a partial resynchronization, RMAN reads the current control file to update changed information about new backups, new archived logs, and so forth. However, RMAN does not resynchronize metadata about the database physical schema: datafiles, tablespaces, redo threads, rollback segments (only if the database is open), and online redo logs. In a full resynchronization, on the other hand, RMAN updates all changed records, including those for the database schema.

Note: Although RMAN performs partial resynchronizations when the control file is a backup, it does not perform full resynchronizations.

When resynchronizing, RMAN does the following:

1. Creates a snapshot control file.
2. Compares the recovery catalog to the snapshot.
3. Updates the catalog with information that is missing or changed.

RMAN performs partial or full resynchronizations automatically as needed when you execute certain commands, including `BACKUP` and `COPY`. For this reason, you should not need to run `RESYNC CATALOG` very often, because the `BACKUP` and `COPY` commands automatically resynchronize the catalog for you. However, you may want to run your backups in `NOCATALOG` mode most of the time, and then connect to the catalog only periodically. To ensure a full resynchronization in this case, run a `RESYNC CATALOG` command.

Note: If you use Oracle Enterprise Manager, then you can use also the Maintenance wizard to perform catalog resynchronizations.

See Also:

- *Oracle9i Recovery Manager Reference* for more information about the `RESYNC` command
- *Oracle Enterprise Manager Administrator's Guide* to learn about the Maintenance wizard

Types of Records That Are Resynchronized

Table 16–2 describes the types of records that RMAN resynchronizes.

Table 16–2 Records Updated during a Resynchronization

Records	Description
Log history	Created when a online redo log switch occurs.
Archived redo logs	Associated with archived logs that were created by archiving an online log, copying an existing archived redo log, or restoring an archived redo log backup set. RMAN tracks this information so that it knows which archived logs it should expect to find.
Backup history	Associated with backup sets, backup pieces, backup set members, and file copies. The `RESYNC CATALOG` command updates these records when a `BACKUP` or `COPY` command is executed.
Physical schema	Associated with datafiles and tablespaces. If the target database is open, then rollback segment information is also updated. Physical schema information in the recovery catalog is updated only when the target has the current control file mounted. If the target database has mounted a backup control file, a freshly created control file, or a control file that is less current than a control file that was seen previously, then physical schema information in the recovery catalog is *not* updated. Physical schema information is also not updated when you use the `RESYNC CATALOG FROM CONTROLFILECOPY` command.

When Should You Resynchronize?

RMAN automatically performs full or partial resynchronizations as needed in certain situations. Most RMAN commands such as `BACKUP`, `COPY`, `DELETE`, and so forth perform a full or partial resynchronization (depending on whether the schema metadata has changed) automatically when the target database control file is mounted and the recovery catalog database is available. Perform manual resynchronizations in the following scenarios.

Resynchronizing When 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. In this case, you do not want to make a daily backup of the target database in `CATALOG` mode because it depends on the availability of a database that is geographically distant. One solution is to connect to the catalog once a week and run the `RESYNC CATALOG` command.

Resynchronizing in ARCHIVELOG Mode When You Back Up Infrequently

Assume that you do the following:

- Run the database in `ARCHIVELOG` mode
- Back up the database infrequently (for example, 1000 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. Instead, Oracle stores information about log switches and archived redo logs in the control file. You must propagate this information periodically into the recovery catalog.

How frequently you resynchronize the recovery catalog depends on the rate at which Oracle 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 Physical Database Changes

Resynchronize the recovery catalog after making any change to the physical structure of the target database. As with redo log archive operations, the recovery catalog is *not* updated automatically when a physical schema change is made.

A physical schema change occurs after:

- Adding or dropping a tablespace
- Adding a new datafile to an existing tablespace
- Adding or dropping a rollback segment

Forcing a Full Resynchronization of the Recovery Catalog

Issue the RESYNC CATALOG command to force a full resynchronization of the recovery catalog.

To perform a full resynchronization of the recovery catalog:

1. Connect SQL*Plus to the recovery catalog database, and open the database if it is not already open. For example, enter:

   ```
   STARTUP
   ```

2. connect RMAN to the target and recovery catalog databases, and then mount the target database if it is not already mounted. For example, enter:

   ```
   STARTUP MOUNT;
   ```

3. Run the RESYNC CATALOG command at the RMAN prompt:

   ```
   RESYNC CATALOG;
   ```

See Also: *Oracle9i Recovery Manager Reference* for RESYNC CATALOG command syntax

Setting the CONTROLFILE_RECORD_KEEP_TIME Value

If you maintain a recovery catalog, then use the RMAN RESYNC CATALOG command to resynchronize the catalog before control file records are reused. In this way, RMAN never loses records because they are propagated from the control file to the recovery catalog. Note that RMAN automatically resynchronizes the catalog when making backups and copies.

To ensure that control file records are not overwritten before they are added to the recovery catalog, set the CONTROLFILE_RECORD_KEEP_TIME to a value that is slightly longer (perhaps a week) than the interval between backups or resynchronizations. For example, if you back up the database during the first and third week of every month, and if you run RESYNC CATALOG during the second and fourth week of every month, then consider setting CONTROLFILE_RECORD_KEEP_TIME to 10 or 14.

See Also: "Monitoring the Overwriting of Control File Records" on page 16-32

Managing RMAN Scripts Stored in the Recovery Catalog

This section contains these topics:

- About Recovery Catalog Stored Scripts
- Storing Scripts in the Recovery Catalog
- Executing Stored Scripts
- Replacing Stored Scripts
- Deleting Stored Scripts

See Also: "Printing Scripts Stored in the Recovery Catalog" on page 17-21 to learn how to query RMAN stored scripts

About Recovery Catalog Stored Scripts

A **stored script** is a sequence of RMAN commands stored within the recovery catalog. It provides a common repository of frequently executed RMAN commands.

For example, you can collect the RMAN commands needed to perform nightly backups into a single script called `nightly_bkup`. Storing the script in the recovery catalog instead of in an operating system text file has the advantage that it is accessible to any DBA that uses RMAN, regardless of which machine RMAN is executed on.

If you have a recovery catalog, then you can use RMAN to do the following:

- Create a script and store it in the recovery catalog
- Execute a stored script
- Replace an existing stored script
- Delete a script from the recovery catalog
- Print the contents of a stored script to the message log file or the screen
- Obtain a listing of all the stored scripts (as described in "Printing Scripts Stored in the Recovery Catalog" on page 17-21)

Note the following important aspects of stored script creation:

- You must be connected to a recovery catalog when executing any of the script commands.
- You must be connected to a target database when executing any of the script commands. Because a script is created for use with one and only one target database, you must be connected to the same target when replacing, printing, or deleting the script.

Storing Scripts in the Recovery Catalog

Use the `CREATE SCRIPT` command to create an RMAN script and store it in the recovery catalog.

To create a stored script:

1. After connecting RMAN to the target database and recovery catalog, compose the desired script using `CREATE SCRIPT`. For example, this script backs up the database and the archived redo logs:

   ```
   CREATE SCRIPT b_whole
   {
     BACKUP DATABASE PLUS ARCHIVELOG;
     DELETE OBSOLETE;
   }
   ```

2. Examine the output. If you see the `RMAN-08085` message, then the script was successfully created and stored in the recovery catalog:

   ```
   created script b_whole
   ```

   Note that this script is associated with the target database that you were connected to when you created the script.

Executing Stored Scripts

Use the `EXECUTE SCRIPT` command to run an RMAN script that you have stored in the recovery catalog.

To execute a stored script:

After connecting to the recovery catalog and target database, issue a `RUN` command to execute the desired script. For example, enter:

```
RUN { EXECUTE SCRIPT b_whole; }
```

RMAN inserts the contents of the script between the brackets of `RUN`. Note that you do not need to run `ALLOCATE CHANNEL` if you already did so *within* the script or if you have automatic channels configured.

> **See Also:** *Oracle9i Recovery Manager Reference* for `EXECUTE SCRIPT` command syntax

Replacing Stored Scripts

Use the REPLACE SCRIPT command to rewrite a catalog script. If the script does not already exist, then RMAN creates it. Note that you cannot use REPLACE SCRIPT to replace a single line of a script: you must rewrite everything.

To replace a stored script:

After connecting RMAN to the recovery catalog, issue a REPLACE SCRIPT command to replace a stored script with another. For example, this command replaces script b_whole with the following:

```
REPLACE SCRIPT b_whole
{
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

See Also: *Oracle9i Recovery Manager Reference* for REPLACE SCRIPT command syntax

Deleting Stored Scripts

Use the DELETE SCRIPT command to remove a script from the recovery catalog.

To delete a stored script:

After connecting RMAN to the recovery catalog, issue a DELETE SCRIPT command to delete the specified stored script:

```
DELETE SCRIPT 'b_whole';
```

See Also: *Oracle9i Recovery Manager Reference* for DELETE SCRIPT command syntax

Managing the Control File When You Use a Recovery Catalog

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. Hence, you must ensure that you resynchronize the recovery catalog with the control file records before these records are erased. As described in "Setting the CONTROLFILE_RECORD_KEEP_TIME Value" on page 16-15, you should perform either of the following actions at intervals less than the `CONTROL_FILE_RECORD_KEEP_TIME` setting:

- Make a backup or copy, thereby performing an implicit resynchronization of the recovery catalog
- Manually resynchronize the recovery catalog with the `RESYNC CATALOG` command

So, to ensure the currency of the information in the recovery catalog, the frequency of resynchronizations should be related to the value for the `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter.

One problem can arise if the control file becomes too large. The size of the target database's control file grows depending on the number of:

- Backups that you perform
- Archived redo logs that Oracle generates
- Days that this information is stored in the control file

As explained in "Monitoring the Overwriting of Control File Records" on page 16-32, 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 Oracle may overwrite the oldest records even if their age is less than the `CONTROL_FILE_RECORD_KEEP_TIME` setting. In this case, Oracle 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.

Note: The maximum size of the control file is port-specific. Typically, the maximum size is 20,000 Oracle blocks. Refer to your operating system-specific Oracle documentation for more information.

See Also:

- *Oracle9i Database Reference* for more information about the `CONTROL_FILE_RECORD_KEEP_TIME` parameter
- *Oracle9i User-Managed Backup and Recovery Guide* to learn how to manage control files for backup and recovery
- *Oracle9i Database Administrator's Guide* for more detailed information on all aspects of control file management

Backing Up and Recovering the Recovery Catalog

Include the recovery catalog 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. Avoid this unpleasant scenario by deciding how to back up and recover the recovery catalog.

This section contains these topics:

- Backing Up the Recovery Catalog
- Recovering the Recovery Catalog
- Re-Creating the Recovery Catalog

Backing Up the Recovery Catalog

When developing a strategy for backing up the recovery catalog, you should follow some general guidelines. This section contains these topics:

- Making Regular Backups
- Choosing the Appropriate Method for Physical Backups
- Storing the Recovery Catalog in an Appropriate Place

Making Regular Backups

Back up the recovery catalog with the same frequency that you back up the target database. For example, if you make a weekly whole database backup of the target database, then back up the recovery catalog immediately after all target database backups. The backed up catalog contains a record of the target backup preceding it, so if you need to restore the catalog you can use it to restore a target backup.

Choosing the Appropriate Method for Physical Backups

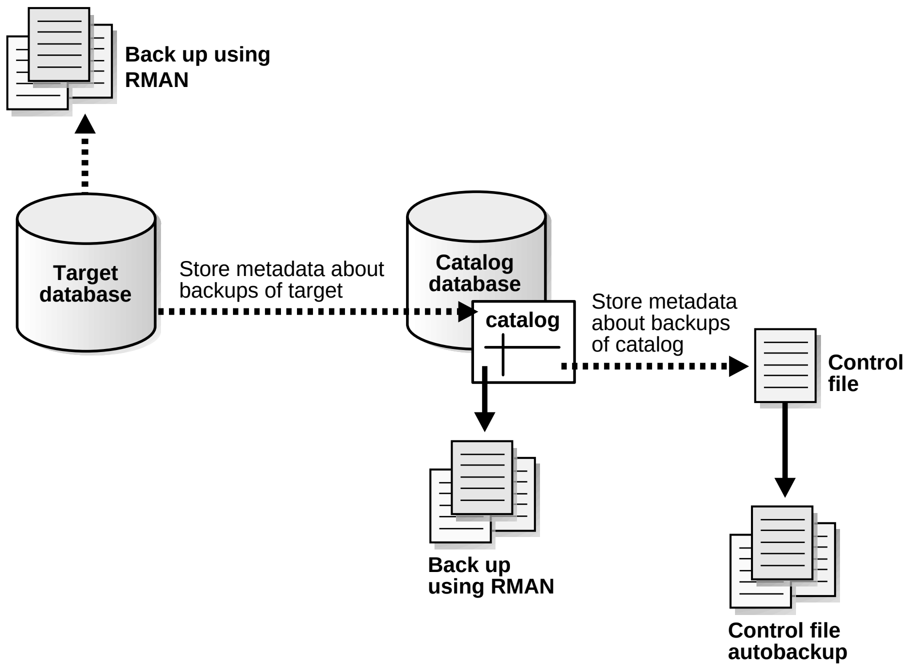
When backing up the recovery catalog database, you can use RMAN to make the backups. As illustrated in Figure 16–1, you should start RMAN with the `NOCATALOG` option so that the repository for the recovery catalog is the control file in the catalog database.

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 is needed.
- Set the retention policy to a `REDUNDANCY` value greater than `1`.
- Back up the database onto two separate media (for example, disk and tape). You can specify `BACKUP COPIES 2` when making backups.
- 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.

Figure 16–1 How to Use the Control File as the Repository for Backups of the Catalog



> **See Also:** *Oracle9i User-Managed Backup and Recovery Guide* to learn how to back up the control file with SQL statements, and "Performing Disaster Recovery" on page 10-35 for more information for recovery with a control file autobackup

Storing the Recovery Catalog in an Appropriate Place

Never store a recovery catalog containing the RMAN repository for a database in the same database as the target database or on the same disks as the target database. For example, do not store the catalog for database `prod1` in `prod1`. A recovery catalog for `prod1` is only effective if it is separated from the data that it is designed to protect.

If `prod1` suffers a total media failure, and if the recovery catalog data for `prod1` is also stored in `prod1`, then you have no catalog to aid in recovery. You will probably have to restore an autobackup of the control file for `prod1` and use it to restore and recover the database.

Separating the target and catalog databases is especially important when you back up a recovery catalog database. The following example shows what you should *not* do. For example, consider the following:

- Target database `prod1` and catalog database `catdb` are on different hosts.
- `catdb` contains the recovery catalog repository for target database `prod1`.

You decide to use a catalog to back up `catdb`, but are not sure where to create it. If you create the catalog containing the repository for `catdb` in database `catdb`, then if you lose `catdb` due to a media failure, you will have difficulty restoring `catdb` *and* will leave `prod1` without a catalog to use in a restore scenario. Hence, you should never store the recovery catalog for a target database in the target database itself: this tactic completely defeats the purpose of having a recovery catalog.

Recovering the Recovery Catalog

If you use RMAN to recover the catalog, then the procedure differs depending on where you stored the metadata for backups of the catalog. Refer to "Performing Basic RMAN Media Recovery" on page 10-5 for a table describing the recovery options.

Re-Creating the Recovery Catalog

If the recovery catalog database is lost or damaged, and 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 catalog database that has never been backed up
- A catalog database that has been backed up, but cannot be recovered because the datafile backups or archived logs are not available

You have these options for partially re-creating the contents of the missing catalog:

- Issue `CATALOG` commands to recatalog archived redo logs, backup control files, and datafile copies. Because the `CATALOG` command does not support recataloging of backup pieces or backup sets, you can re-create information about backup sets *only* by using the `RESYNC CATALOG` command.
- Use the `RESYNC CATALOG` command to extract metadata from a control file and rebuild the recovery catalog. Note that you automatically lose any metadata that was contained in old control file records that aged out of the control file.

  Depending on the state of the target control file, you can:

- Resynchronize from the current control file
- Resynchronize from a control file copy

See Also:

- *Oracle9i Recovery Manager Reference* for information about the `CATALOG` command
- *Oracle9i Recovery Manager Reference* for information about the `CROSSCHECK` command
- "Managing the Control File When You Use a Recovery Catalog" on page 16-18 to learn about how records age out of the control file

Exporting the Recovery Catalog

To move the recovery catalog from one database to another, run the Import and Export utilities. You can only import the catalog into a supported version of the Oracle database server. In general, you can import the catalog into a database of the same release or later.

This section contains the following topics:

- Considerations When Moving Catalog Data
- Exporting the Recovery Catalog
- Importing the Recovery Catalog

Considerations When Moving Catalog Data

You should only import the recover catalog into a schema that does not already contain a recovery catalog schema. In other words, the user who will own the imported recovery catalog schema should not already own a catalog schema. For example, if user `rman` owns the recovery catalog on database `catdb`, and you want to export the recovery catalog on `catdb` and import it into database `catdb2`, then `rman` should not already own a recovery catalog on `catdb2`. You should either create a new recovery catalog owner on `catdb2`, or drop the current user `rman` on `catdb2` and then re-create the user. You cannot merge a recovery catalog into an existing recovery catalog.

The basic steps for exporting a catalog from a primary database and importing the catalog into a secondary database are as follows:

1. Use the Export utility to export the catalog data from the primary database as described in "Exporting the Recovery Catalog" on page 16-25.
2. Create a user on the secondary database as described in "Creating the Recovery Catalog Owner" on page 16-3, and grant the user necessary privileges.
3. Use the Import utility to import the catalog data into the schema created in the previous step, as described in "Importing the Recovery Catalog" on page 16-25.

You should *not* run the `CREATE CATALOG` command either before or after the Import of the catalog into the secondary database. By importing the catalog data into the new schema, you effectively create the catalog in the secondary database.

Exporting the Recovery Catalog

Refer to *Oracle9i Database Utilities* for concepts and procedures relating to the Export utility.

To make a logical export of the recovery catalog from the command line:

1. Execute the Export utility at the operating system command line, making sure to do the following:

   a. Connect as the owner of the recovery catalog

   b. Specify the `OWNER` option

   c. Specify an output file

   For example, if the owner of the catalog in database `catdb` is `rman`, you can issue the following at the UNIX command line to export the catalog to file `cat.dmp`:

   ```
   % exp rman/cat@catdb FILE=cat.dmp OWNER=rman
   ```

2. Examine the output to make sure you were successful:

   ```
   Export terminated successfully without warnings.
   ```

Importing the Recovery Catalog

If you used Export to make a logical backup of the recovery catalog, then use the Import utility to import it into another database.

To make a logical import of the recovery catalog from the command line:

1. Create a new user in another database. For the recommended SQL syntax for creating a new user in a recovery catalog database, see "Creating the Recovery Catalog" on page 16-2.
2. Import the catalog data from the export file. Execute the import at the command line, making sure to do the following:

   a. Connect as the new owner of the recovery catalog.

   b. Specify the old owner with the `FROMUSER` parameter.

   c. Specify the new owner with the `TOUSER` parameter.

   d. Specify the import file.

   For example, assume the following:

   - The old owner of the catalog in database `prod1` is `rman`.
   - The user in the new recovery catalog database `catdb2` is `rman2`.
   - The file containing the export of the catalog is `cat.dmp`.

   The command is then as follows:

   ```
   % imp USERID=rman2/cat2@catdb2 FILE=cat.dmp FROMUSER=rman TOUSER=rman2
   ```

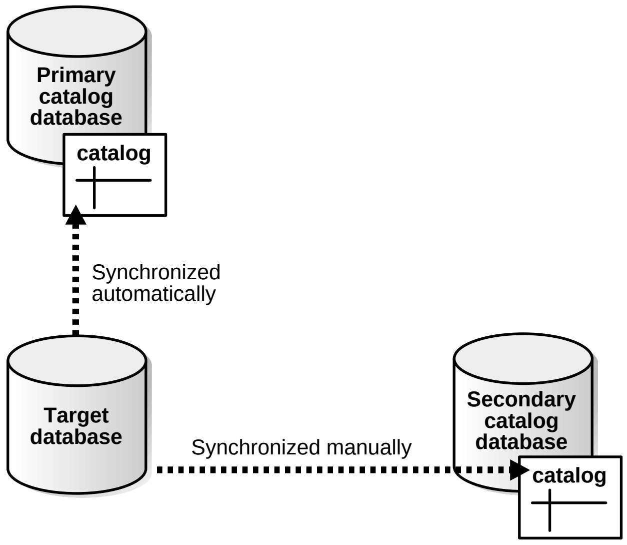
3. Use the imported catalog data for restore and recovery of your target database.

Increasing Availability of the Recovery Catalog

You may have a production system in which you want to maintain high availability for the catalog database. For example, you may have 100 target databases registered in the recovery catalog. In case the primary catalog database goes down, you can create redundancy by storing a secondary recovery catalog in a separate database, as illustrated in Figure 16–2. You must register the target database in the secondary catalog.

In this availability scenario, the main catalog is synchronized as normal during regular backups, while the secondary catalog is synchronized periodically with the `RESYNC CATALOG` command. If the primary catalog database goes down or requires routine maintenance, then you can resynchronize the secondary catalog and use it as the new primary catalog during the interim.

Figure 16–2 Registering One Database in Two Recovery Catalogs



See Also: "Creating the Recovery Catalog" on page 16-2

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.

See Also: *Oracle9i Recovery Manager Reference* for the complete set of compatibility rules governing the RMAN environment

To determine the schema version of the recovery catalog:

1. Start SQL*Plus and connect to the recovery catalog database as the catalog owner. For example:

   ```
   % sqlplus rman/cat@catdb
   ```

2. Query the RCVER table to obtain the schema version, as in the following example (sample output included):

   ```
   SELECT *
   FROM RCVER;

   VERSION
   ------------
   09.02.00
   ```

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
09.02.00
```

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 9.2.0. The current version of the catalog schema is 9.2.0.

Upgrading the Recovery Catalog

If you use a version of the recovery catalog that is older than that required by the RMAN executable, then you must upgrade it. For example, you must upgrade the catalog if you use a release 8.1 version of the RMAN executable with a release 8.0 version of the recovery 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 executable. 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.

Checking the Current Recovery Catalog Version

To determine the current release of the catalog schema, you must run a SQL query. Note that the catalog version does not necessarily change with every new release of RMAN. It changes when incompatibilities between catalogs are found. Hence, the best strategy is to let RMAN tell you when you need to upgrade.

To determine the current release of the recovery catalog:

1. Use SQL*Plus to connect to the recovery catalog database as the catalog owner. For example, enter:

   ```
   % sqlplus rman/cat@catdb
   ```

2. Query the RCVER catalog table. For example, run this query:

   ```
   SELECT * FROM RCVER;

   VERSION
   ------------
   09.02.00
   ```

Upgrading the Recovery Catalog Schema

Use the RMAN UPGRADE CATALOG command to change the catalog to a newer release.

To upgrade the recovery catalog:

1. Use RMAN to connect to the target and recovery catalog databases. For example, enter:

   ```
   % rman TARGET / CATALOG rman/cat@catdb

   connected to recovery catalog database
   PL/SQL package rcat.DBMS_RCVCAT version 08.00.04 in RCVCAT database is too old
   ```

2. Issue the UPGRADE CATALOG command:

   ```
   UPGRADE CATALOG;

   recovery catalog owner is rman
   enter UPGRADE CATALOG command again to confirm catalog upgrade
   ```

3. Enter the UPDATE CATALOG command again to confirm:

   ```
   UPGRADE CATALOG;

   recovery catalog upgraded to version 09.02.00
   DBMS_RCVMAN package upgraded to version 09.02.00
   DBMS_RCVCAT package upgraded to version 09.02.00
   ```

See Also:

- *Oracle9i Recovery Manager Reference* for `UPGRADE CATALOG` command syntax
- *Oracle9i Recovery Manager Reference* for information about recovery catalog compatibility
- *Oracle9i Database Migration* for complete compatibility and migration information

Dropping the Recovery Catalog

If you do not want to maintain a recovery catalog, then you can drop the recovery catalog schema from the tablespace. The `DROP CATALOG` command deletes all information from the recovery catalog. Hence, if you have no backups of the recovery catalog schema, then all backups of all target databases managed by this catalog become unusable.

The `DROP CATALOG` command is not appropriate for "unregistering" a single database from a catalog that has multiple target databases registered. If you try to delete the metadata for one target database by dropping the catalog, then you delete the metadata for all target databases.

To drop the recovery catalog schema:

1. Use RMAN to connect to the target and recovery catalog databases.

   ```
   % rman TARGET / CATALOG rman/cat@catdb
   ```

2. Issue the `DROP CATALOG` command twice to confirm:

   ```
   DROP CATALOG;

   recovery catalog owner is rman
   enter DROP CATALOG command again to confirm catalog removal

   DROP CATALOG;
   ```

Note: Even after you drop the recovery catalog, the control file still contains records about the backups. To purge these records, re-create the control file.

See Also: *Oracle9i Recovery Manager Reference* for DROP CATALOG command syntax, and "Unregistering a Target Database from the Recovery Catalog" on page 16-8 to learn how to unregister a database from the catalog

Managing the RMAN Repository Without a Recovery Catalog

RMAN works perfectly well without a recovery catalog. In fact, the recovery catalog receives all its information from the control file. If you choose not to use a recovery catalog, however, you must perform some additional administrative tasks. This section contains these topics:

- Understanding Catalog-Only Command Restrictions
- Monitoring the Overwriting of Control File Records
- Maintaining the Control File Repository
- Backing Up and Restoring the Control File

See Also:

- *Oracle9i Database Administrator's Guide* for a conceptual overview of the control file.
- Chapter 16, "Managing the Recovery Manager Repository" for a description of the importance of the control file for backup and recovery
- *Oracle9i Database Administrator's Guide* for more detailed information on managing the control file

Understanding Catalog-Only Command Restrictions

If you choose not to use a recovery catalog, then you can still use RMAN very effectively. RMAN obtains the information it needs from the control file of the target database. Nevertheless, some commands are not available when you use the control file as the sole repository of RMAN metadata. The following commands are only available when you use a recovery catalog:

- `CREATE CATALOG`, `UPGRADE CATALOG`, `DROP CATALOG`
- `CREATE SCRIPT`, `DELETE SCRIPT`, `REPLACE SCRIPT`, `PRINT SCRIPT`
- `LIST INCARNATION`
- `REGISTER DATABASE`

- `REPORT SCHEMA AT TIME`
- `RESET DATABASE`
- `RESYNC CATALOG`

Monitoring the Overwriting of Control File Records

When you do not use a recovery catalog, the control file is the sole source of information about RMAN backups and copies. As you make backups and copies, Oracle adds new records to the control file. These records are circularly reused, which means that Oracle overwrites older records.

The following 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.

What happens when Oracle needs to add new records to the control file, but the oldest record is less than the value specified in `CONTROL_FILE_RECORD_KEEP_TIME`? The following steps occur:

1. Oracle attempts to expand the size of the control file, which it can only do if the underlying operating system file can be expanded.
2. If it cannot expand the control file, then Oracle overwrites the oldest record—regardless of whether its age is less than the `CONTROL_FILE_RECORD_KEEP_TIME` value—and logs this action in the `alert.log`.

Hence, if you are not using a recovery catalog, then 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 database once a week, then you need to keep every backup at least a week. Set `CONTROL_FILE_RECORD_KEEP_TIME` to a value such as `10` or `14`.

Managing the Overwriting of Control File Records: Scenario

Assume the following scenario:

- You do not use a recovery catalog.
- `CONTROL_FILE_RECORD_KEEP_TIME` is set to `14`.
- All records currently in the control file are between 1 and 13 days old.

- The control file is at the maximum size permitted by the operating system.

You make a backup of the database. Because Oracle cannot expand the control file beyond the operating system file size limit, it begins overwriting records in the control file, starting with those records aged 13 days. For each record that it overwrites, it records an entry in the `alert.log` that looks something like the following:

```
krcpwnc: following controlfile record written over:
RECID #72 Recno 72 Record timestamp
07/28/00 22:15:21
Thread=1 Seq#=3460
Backup set key: stamp=372031415, count=17
Low scn: 0x0000.3af33f36
07/27/00 21:00:08
Next scn: 0x0000.3af3871b
07/27/00 23:23:54
Resetlogs scn and time
scn: 0x0000.00000001
08/05/99 10:46:44
Block count=102400 Blocksize=512
```

To guard against this type of scenario, use a recovery catalog. If you cannot use a catalog, then do the following if possible:

- Store the control file in a file system rather than raw disk so that it can expand as needed.
- Monitor the `alert.log` to make sure that Oracle is not overwriting control file records.

See Also: "Types of Records in the Control File" on page 4-15 for a conceptual overview of control file records

Maintaining the Control File Repository

RMAN provides several commands that allow you to check and delete records of backups as well as physically remove backups and copies.

See Also:

- "Managing Target Database Records in the Recovery Catalog" on page 16-5 provides a complete description of these maintenance procedures. Most of these commands work whether or not you use a recovery catalog.
- "Understanding Catalog-Only Command Restrictions" on page 16-31 for a list of the commands that require a catalog

Backing Up and Restoring the Control File

If you use the control file as the sole repository of the RMAN metadata, maintain alternate control files through multiplexing or operating system mirroring and back up the control file frequently. If you lose the control file and do not have a backup, you lose all information about RMAN backups and copies contained in the file. For this reason, you should set `CONFIGURE CONTROLFILE AUTOBACKUP` to `ON`.

So long as a control file autobackup is available, RMAN can mount and restore the database. After the control file is mounted, you can restore the remainder of the database. Note that you lose persistent configuration settings stored in the control file, but after you restore the autobackup the configuration settings are returned.

See Also:

- "Control File and Server Parameter File Autobackups" on page 5-47 to learn about the control file autobackup feature
- "Backing Up Control Files with RMAN" on page 9-8 to learn about manual and automatic control file backups
- "Performing Recovery with a Backup Control File and No Recovery Catalog" on page 10-16 to learn how to restore a database when the current control file and recovery catalog are unavailable
- *Oracle9i User-Managed Backup and Recovery Guide* to learn how to make user-managed backups of the control file

17

Querying the RMAN Repository

This chapter describes how to obtain information about RMAN from the repository. This chapter contains these topics:

About Querying the RMAN Metadata

You can obtain information from the RMAN repository in several different ways. The following table describes the basic options.

Method	Catalog Needed?	Description
LIST command	No	Use this command to list backups, copies, and database incarnations. The output displays those files operated on by the CHANGE, CROSSCHECK, and DELETE commands.
REPORT command	No	Use this command to find out which files need a backup, which backups are no longer needed, which files are in the schema, and so forth.
SHOW command	No	Use this command to display persistent RMAN configuration settings.
PRINT SCRIPT command	Yes	Use this command to display the names of the scripts stored in the recovery catalog.
Recovery catalog fixed views	Yes	Query these views to access the catalog itself. Some information, such as the names and contents of the stored scripts, can only be obtained from the catalog views.
V$ fixed views	No	Query these views to access the target database control file. RMAN obtains metadata for the recovery catalog from the control file. Some V$ views such as V$DATAFILE_HEADER, V$PROCESS, and V$SESSION contain information not found in the catalog views.

The main source of information about RMAN is the REPORT and LIST command output. Use these commands to query the RMAN repository and determine what you have backed up as well as what you need to back up. This information is extremely helpful in developing an effective backup strategy.

The LIST command displays all RMAN backups (both backup sets and proxy copies) and copies, while the REPORT command performs more complex analysis. For example, you can generate a report on which datafiles need a backup and which backup pieces are obsolete with the REPORT command. RMAN writes the output from the REPORT and LIST commands to either standard output or a log file.

The SHOW command displays persistent configuration settings. For example, if you allocate automatic channels with the CONFIGURE command, these settings are displayed in the SHOW output.

See Also:

- Chapter 16, "Managing the Recovery Manager Repository" to learn how to keep the RMAN repository current
- *Oracle9i Recovery Manager Reference* for `LIST` command syntax
- *Oracle9i Recovery Manager Reference* for `REPORT` command syntax

Listing RMAN Backups, Copies, and Database Incarnations

The `LIST` command queries the recovery catalog or control file and produces a listing of the backups, copies, archived redo logs, and database incarnations recorded there. You can specify these files when running the `CHANGE`, `CROSSCHECK`, and `DELETE` commands.

This section contains these topics:

About RMAN Lists

You can control how the output is displayed by using the `BY BACKUP` and `BY FILE` options and choosing between the `SUMMARY` and `VERBOSE` options.

The primary purpose of the `LIST` command is to determine which backups or copies are available. Note that only backups and copies that completed successfully are stored in the repository. For example, you can list:

- Backups (backup sets and proxy copies) or image copies recorded in the RMAN repository
- Backups or image copies of a specified database, tablespace, datafile, archived redo log, or control file

- Backups and image copies that have expired
- Backups and image copies restricted by options such as time, path name, device type, tag, or recoverability
- Incarnations of a specified database

Use the RMAN repository to determine what you need to back up. In particular, ensure that:

- The `STATUS` columns of the output tables list all backups and image copies as `AVAILABLE`
- All datafiles, archived redo logs, and control files that you need backed up are included in the output
- The backups and copies recorded in the repository are recent

Listing Backups by Backup

By default, RMAN lists backups by backup, which means that it serially lists each backup set or proxy copy and then identifies the files included in the backup. By default, RMAN lists backups and copies in *verbose mode*, which means that it provides extensive, multiline information.

To list backups by backup:

1. After connecting to the target database and recovery catalog (if you use one), execute `LIST BACKUP`. Specify the desired objects with the *`listObjList`* clause. For example, you can enter:

   ```
   LIST BACKUP;  # lists backup sets, backup pieces, and proxy copies
   ```

   Optionally, specify the `EXPIRED` keyword to identify those backups that were not found during a crosscheck:

   ```
   LIST EXPIRED BACKUP;
   ```

2. Examine the output (refer to *Oracle9i Recovery Manager Reference* for an explanation of the various column headings in the LIST output). Sample output follows:

```
LIST BACKUP;

List of Backup Sets
===================

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
19      201K       DISK        00:00:01     08-FEB-02
        BP Key: 30   Status: AVAILABLE   Tag: TAG20020208T155239
        Piece Name: /oracle/dbs/0jdg9v28_1_1

  List of Archived Logs in backup set 19
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    21      98086      08-FEB-02 98461      08-FEB-02
  1    22      98461      08-FEB-02 98464      08-FEB-02
  1    23      98464      08-FEB-02 98469      08-FEB-02
  1    24      98469      08-FEB-02 98472      08-FEB-02
  1    25      98472      08-FEB-02 98475      08-FEB-02

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
20      Full    197M       DISK        00:00:42     08-FEB-02
        BP Key: 31   Status: AVAILABLE   Tag: TAG20020208T155242
        Piece Name: /oracle/dbs/0kdg9v2b_1_1
  SPFILE Included: Modification time: 08-FEB-02
  List of Datafiles in backup set 20
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 98512      08-FEB-02 /oracle/oradata/trgt/system01.dbf
  2       Full 98512      08-FEB-02 /oracle/oradata/trgt/undotbs01.dbf
  3       Full 98512      08-FEB-02 /oracle/oradata/trgt/cwmlite01.dbf
  4       Full 98512      08-FEB-02 /oracle/oradata/trgt/drsys01.dbf
  5       Full 98512      08-FEB-02 /oracle/oradata/trgt/example01.dbf
  6       Full 98512      08-FEB-02 /oracle/oradata/trgt/indx01.dbf
  7       Full 98512      08-FEB-02 /oracle/oradata/trgt/tools01.dbf
  8       Full 98512      08-FEB-02 /oracle/oradata/trgt/users01.dbf

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
21      1K         DISK        00:00:02     08-FEB-02
        BP Key: 32   Status: AVAILABLE   Tag: TAG20020208T155331
        Piece Name: /oracle/dbs/0ldg9v3r_1_1

  List of Archived Logs in backup set 21
```

```
Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
---- ------- ---------- --------- ---------- ---------
1    34      98509      08-FEB-02 98529      08-FEB-02
```

Listing Backups by File

You can list copies of datafiles, control files, and archived logs. Specify the desired objects with the *`listObjList`* or *`recordSpec`* clause (refer to *Oracle9i Recovery Manager Reference*). If you do not specify an object, then RMAN displays copies of all database files and archived redo logs. By default, RMAN lists backups in verbose mode, which means that it provides extensive, multiline information.

To list backups by file:

1. After connecting to the target database and recovery catalog (if you use one), execute `LIST BACKUP` with the `BY FILE` option. Specify the desired objects and options. For example, you can enter:

   ```
   LIST BACKUP BY FILE;
   ```

   Optionally, specify the `EXPIRED` keyword to identify those backups that were not found during a crosscheck:

   ```
   LIST EXPIRED BACKUP BY FILE;
   ```

2. Examine the output (refer to *Oracle9i Recovery Manager Reference* for an explanation of the various column headings in the `LIST` output). Sample output follows:

   ```
   List of Datafile Backups
   ========================

   File Key     TY LV S Ckp SCN    Ckp Time  #Pieces #Copies Tag
   ---- ------- -  -- - ---------- --------- ------- ------- ---
   1    20      B  F  A 98512      08-FEB-02 1       1       TAG20020208T155242
   2    20      B  F  A 98512      08-FEB-02 1       1       TAG20020208T155242
   3    20      B  F  X 98512      08-FEB-02 1       1       TAG20020208T155242
   4    20      B  F  U 98512      08-FEB-02 1       1       TAG20020208T155242

   List of Archived Log Backups
   ============================

   Thrd Seq     Low SCN    Low Time  BS Key  S #Pieces #Copies Tag
   ---- ------- ---------- --------- ------- - ------- ------- ---
   1    21      98086      08-FEB-02 22      A 1       1       TAG20020208T155604
                                     19      A 1       1       TAG20020208T155239
   1    22      98461      08-FEB-02 22      A 1       1       TAG20020208T155604
                                     19      A 1       1       TAG20020208T155239
   ```

```
1    23      98464      08-FEB-02 22      A 1       1       TAG20020208T155604
                                  19      A 1       1       TAG20020208T155239

List of Controlfile Backups
===========================

CF Ckp SCN Ckp Time  BS Key  S #Pieces #Copies Tag
---------- --------- ------- - ------- ------- ---
98510      08-FEB-02 20      A 1       1       TAG20020208T155242

List of SPFILE Backups
======================

Modification Time BS Key  S #Pieces #Copies Tag
----------------- ------- - ------- ------- ---
08-FEB-02         20      A 1       1       TAG20020208T155242
```

Listing Copies

Besides listing backup sets and proxy copies, you can list image copies. Specify the desired objects with the *listObjList*, *recordSpec*, or archivelogRecordSpecifier clauses. If you do not specify an object, then LIST COPY displays all datafile copies, control file copies, and archived redo logs. Note that RMAN considers both archived redo logs and image copies of archived redo logs as copies. By default, RMAN lists backups in verbose mode which means that it provides extensive, multiline information.

To list image copies:

1. After connecting to the target database and recovery catalog (if you use one), execute LIST COPY. Specify the desired objects and options. For example, you can enter:

   ```
   LIST COPY; # lists all datafile copies, control file copies, and archived logs
   LIST ARCHIVELOG ALL; # lists all archived logs
   ```

   Optionally, specify the EXPIRED keyword to identify those copies that were not found during a crosscheck:

   ```
   LIST EXPIRED COPY;
   ```

2. Examine the output (refer to *Oracle9i Recovery Manager Reference* for an explanation of the various column headings in the LIST output). Sample output follows:

   ```
   LIST ARCHIVELOG ALL;

   List of Archived Log Copies
   ```

```
Key     Thrd Seq     S Low Time             Name
------- ---- ------- - -------------------- ----
8       1    28      A NOV 07 2001 10:50:07 /oracle/oradata/trgt/arch/archive1_28.dbf
9       1    29      A NOV 07 2001 11:54:10 /oracle/oradata/trgt/arch/archive1_29.dbf
10      1    30      A NOV 07 2001 12:00:22 /oracle/oradata/trgt/arch/archive1_30.dbf
11      1    31      A NOV 07 2001 12:01:28 /oracle/oradata/trgt/arch/archive1_31.dbf
12      1    32      A NOV 07 2001 12:44:00 /oracle/oradata/trgt/arch/archive1_32.dbf
13      1    33      A NOV 07 2001 12:59:37 /oracle/oradata/trgt/arch/archive1_33.dbf
```

Listing Backups in Summary Mode

By default the LIST output is highly detailed, but you can also specify that RMAN display the output in summarized form. Specify the desired objects with the *listObjectList* or *recordSpec* clause. If you do not specify an object, then LIST BACKUP displays all backups. By default, RMAN lists backups in verbose mode.

To list backups in summary mode:

1. After connecting to the target database and recovery catalog (if you use one), execute LIST BACKUP. Specify the desired objects and options. For example, you can enter:

   ```
   LIST BACKUP SUMMARY;
   ```

   Optionally, specify the EXPIRED keyword to identify those copies that were not found during a crosscheck:

   ```
   LIST EXPIRED BACKUP SUMMARY;
   ```

2. Examine the output (refer to *Oracle9i Recovery Manager Reference* for an explanation of the various column headings in the LIST output). Sample output follows:

   ```
   List of Backups
   ===============
   Key     TY LV S Device Type Completion Time #Pieces #Copies Tag
   ------- -- -- - ----------- --------------- ------- ------- ---
   387     B  0  A SBT_TAPE    08-FEB-01       1       2       TAG20020208T155604
   396     B  0  A SBT_TAPE    08-FEB-01       3       1       TAG20020208T155604
   423     B  0  X SBT_TAPE    08-FEB-01       1       1       TAG20020208T155604
   427     B  0  U SBT_TAPE    08-FEB-01       1       1       TAG20020208T155604
   ```

Listing Backups and Copies with Restrictions

You can specify several different conditions to narrow your LIST output.

To generate a list of copies and backups restricted by object or other conditions:

1. After connecting to the target database and recovery catalog (if you use one), execute `LIST COPY` or `LIST BACKUP` with the *`listObjList`* or *`recordSpec`* condition. For example, enter:

   ```
   # lists backups of all files in database
   LIST BACKUP OF DATABASE;

   # lists copy of specified datafile
   LIST COPY OF DATAFILE '?/oradata/trgt/system01.dbf';

   # lists specified backup set
   LIST BACKUPSET 213;

   # lists datafile copy
   LIST DATAFILECOPY '/tmp/tools01.dbf';
   ```

2. You can also restrict the search by specifying the *`maintQualifier`* or `RECOVERABLE` clause. For example, enter:

   ```
   # specify a backup by tag
   LIST BACKUP TAG 'weekly_full_db_backup';

   # specify a backup or copy by device type
   LIST COPY OF DATAFILE '?/oradata/trgt/system01.dbf' DEVICE TYPE sbt;

   # specify a backup or copy 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-2001' AND '17-DEC-2001';

   # specify logs backed up at least 2X to tape
   LIST ARCHIVELOG ALL BACKED UP 2 TIMES TO DEVICE TYPE sbt;
   ```

3. Examine the output. For example, sample output follows for a list of copies of `datafile 1`:

   ```
   LIST COPY OF DATAFILE 1;

   List of Datafile Copies
   Key     File S Completion time Ckp SCN    Ckp time        Name
   ------- ---- - --------------- ---------- --------------- ------
   3       1    A 18-JUL-00       114148     17-JUL-01       /tmp/system01.dbf
   ```

   See Also: *Oracle9i Recovery Manager Reference* for *`listObjList`* syntax, and *Oracle9i Recovery Manager Reference* for an explanation of the various columns in the `LIST` output

Listing Database Incarnations

Every time you reset the online redo logs of a target database, you create a new incarnation of the database. You can track the incarnations with the `INCARNATION` option of the `LIST` command.

To list database incarnations:

1. After connecting to the target database and (optionally) the recovery catalog, run `LIST INCARNATION`:

   ```
   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;
   ```

2. Examine the output (refer to *Oracle9i Recovery Manager Reference* for an explanation of the various column headings in the `LIST` output). Sample output follows:

   ```
   LIST INCARNATION OF DATABASE;

   List of Database Incarnations
   DB Key  Inc Key DB Name  DB ID            CUR Reset SCN  Reset Time
   ------- ------- -------- ---------------- --- ---------- ----------
   1       12      TRGT     1335481537       NO  1          NOV 12 2001 03:06:41
   1       2       TRGT     1335481537       YES 164378     NOV 12 2001 17:54:26
   ```

   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 its incarnation key.

Reporting on Backups, Copies, and Database Schema

This section contains the following topics:

- About RMAN Reports
- Reporting on Objects Needing a Backup
- Reporting on Obsolete Backups and Copies
- Reporting on Unrecoverable Backups and Copies
- Reporting on the Database Schema

About RMAN Reports

To gain more detailed information from the RMAN repository, generate a report. Use the REPORT command to answer questions such as the following:

- Which files need a backup?
- Which files have had unrecoverable operations performed on them?
- Which backups or copies are obsolete and can be deleted?
- What was the physical schema of the database at some previous time?
- Which files have not been backed up recently?

Note: For the report to be accurate, the RMAN repository must be synchronized with the control file and you must have run the CHANGE, UNCATALOG, and CROSSCHECK commands recently to update the status of all backups and copies. To learn how to maintain the RMAN repository refer to Chapter 16, "Managing the Recovery Manager Repository".

The information that you obtain from reports can be extremely important for your backup and recovery strategy. In particular, run the REPORT NEED BACKUP and REPORT UNRECOVERABLE commands regularly to ensure the following:

- The necessary backups are available to perform recovery.
- Recovery can be performed within a reasonable length of time, that is, that the mean time to recovery (MTTR) is minimized.

Reporting on Objects Needing a Backup

You can report on objects that require a backup by specifying the NEED BACKUP keyword. The REDUNDANCY parameter specifies the minimum number of backups or copies that must exist for a datafile to be considered not in need of a backup. If you do not specify the parameter, REDUNDANCY defaults to 1. The DAYS parameter indicates that recovery must begin by using logs more than *integer* days old. The INCREMENTAL parameter indicates that more than *integer* incremental backups are required for complete recovery.

Note: If you disable the retention policy, then REPORT NEED BACKUP with no other options generates an error message.

To report on objects that need a backup:

1. After connecting to the target database and recovery catalog (if you use one), run CROSSCHECK commands as needed to update the status of backups and copies. Following is a possible crosscheck session:

   ```
   # allocate maintenance channel for crosscheck if automatic channels not configured
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
   CROSSCHECK BACKUP;  # crosschecks all backups
   CROSSCHECK COPY;    # crosschecks all copies
   ```

2. If you have a retention policy configured, then you can just run REPORT NEED BACKUP without any other options to determine which files need backups (sample output follows):

   ```
   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
   3    0     /oracle/oradata/trgt/cwmlite01.dbf
   4    0     /oracle/oradata/trgt/drsys01.dbf
   7    0     /oracle/oradata/trgt/tools01.dbf
   ```

3. To override the retention policy (or if you do not have a retention policy enabled), run REPORT NEED BACKUP DAYS. Any files older than the DAYS parameter value need a new backup because their backups require the specified number of DAYS worth of archived logs for recovery. For example, run:

   ```
   REPORT NEED BACKUP DAYS = 7 DATABASE;  # needs min 7 days of logs to recover
   REPORT NEED BACKUP DAYS = 30 TABLESPACE SYSTEM;
   REPORT NEED BACKUP DAYS = 14 DATAFILE '?/oradata/trgt/tools01.dbf';
   ```

4. To determine which files need an incremental backup, specify the INCREMENTAL parameter. If complete recovery of a datafile requires more than the specified number of incremental backups, then RMAN considers it in need of a new backup. For example, enter:

   ```
   REPORT NEED BACKUP INCREMENTAL = 1 DATABASE;
   REPORT NEED BACKUP INCREMENTAL = 3 TABLESPACE SYSTEM;
   REPORT NEED BACKUP INCREMENTAL = 5 DATAFILE '?/oradata/trgt/users01.dbf';
   ```

   See Also: *Oracle9i Recovery Manager Reference* for an explanation of the various column headings in the REPORT output

Reporting on Obsolete Backups and Copies

You can report on objects that are obsolete, that is, superfluous, by specifying the OBSOLETE keyword. If you do not specify any other options, then REPORT OBSOLETE displays the backups and copies that are marked obsolete by the current retention policy. By default, the retention policy is configured to REDUNDANCY of 1.

The REPORT OBSOLETE command supports the RECOVERY WINDOW and REDUNDANCY options at the command level, which have the same meanings as the options with the same names on the CONFIGURE command.

To report on obsolete backups and copies:

1. After connecting to the target database and recovery catalog (if you use one), issue CROSSCHECK commands as needed to update the status of backups and copies. Following is a possible crosscheck session:

   ```
   # allocate maintenance channel for crosscheck if automatic channels not configured
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
   CROSSCHECK BACKUP;  # crosschecks all backups
   CROSSCHECK COPY;    # crosschecks all copies
   RELEASE CHANNEL;
   ```

2. Use the OBSOLETE option to identify which backups are obsolete because they are no longer needed for recovery. For example, enter:

   ```
   # lists backups or copies that are superfluous because they are not needed to recover
   # the database to a random point within the past week
   REPORT OBSOLETE RECOVERY WINDOW OF 7 DAYS;
   # lists backups or copies that are superfluous because more than 2 copies of the
   # files exist on tape
   REPORT OBSOLETE REDUNDANCY = 2 DEVICE TYPE sbt;
   ```

3. Use the ORPHAN option to list unusable backups and copies belonging to an incarnation that is not a direct predecessor of the current incarnation (refer to "Reports of Orphaned Backups" on page 7-5). For example, enter:

   ```
   REPORT OBSOLETE ORPHAN;
   ```

4. Optionally, delete those backups that are obsolete. You can automatically delete obsolete backups and copies by issuing the DELETE OBSOLETE command. For example, you can enter:

   ```
   # delete obsolete backups and copies displayed when you issue REPORT OBSOLETE
   DELETE OBSOLETE;
   # delete obsolete backups and copies according to a specified recovery window
   DELETE OBSOLETE RECOVERY WINDOW OF 7 DAYS;
   # delete obsolete backups and copies according to a specified redundancy
   ```

```
DELETE OBSOLETE REDUNDANCY = 2;
```

Note that RMAN prompts you for confirmation before actually deleting the files. To suppress the prompt, specify the NOPROMPT option of the DELETE command. Specify FORCE to delete the files and remove their repository records regardless of whether the files exist. RMAN ignores any I/O errors for the deleted objects.

> **See Also:** "Reports of Obsolete Backups" on page 7-4 for a conceptual overview of reports of obsolete backups, and "Backup Retention Policies" on page 5-49 for a conceptual overview of retention policies

Reporting on Unrecoverable Backups and Copies

Issue the REPORT UNRECOVERABLE command to determine which datafiles have had an unrecoverable operation performed against an object residing in the datafile after its last backup.

Assume that you perform an unrecoverable operation on the table employee by issuing an ALTER TABLE employee ... NOLOGGING statement. If the employee table is located in datafile 3, then the REPORT command can flag backups of this datafile as unrecoverable.

To report on backups and copies that are unrecoverable:

After connecting to the target database and recovery catalog (if you use one), issue REPORT UNRECOVERABLE. For example, enter:

```
REPORT UNRECOVERABLE DATABASE;             # examines all datafiles
REPORT UNRECOVERABLE TABLESPACE 'users';   # examines a specific tablespace
```

Reporting on the Database Schema

You do not have to use V$ or recovery catalog views to identify the database files. Issue REPORT SCHEMA to list the files. If you use a recovery catalog, then you also generate historical reports of the database schema at a past time. You do not need a recovery catalog, however, to report the current schema.

To report the database schema at a specified point in time:

1. After connecting to the target database and recovery catalog (if you use one), issue REPORT SCHEMA for a list of all the datafiles and tablespaces in the target database at the current time:

   ```
   REPORT SCHEMA;
   ```

If you use a recovery catalog, then you can use the *`atClause`* to specify a past time, SCN, or log sequence number:

```
REPORT SCHEMA AT TIME 'SYSDATE-14';     # schema as it existed two weeks ago
REPORT SCHEMA AT SCN 1000;              # schema as it existed at scn 1000
REPORT SCHEMA AT SEQUENCE 100 THREAD 1; # schema as it existed at log sequence 100
```

2. Examine the report. For example, here is a sample output:

```
REPORT SCHEMA AT SCN 1000;

Report of database schema
File K-bytes    Tablespace           RB segs Datafile Name
---- ---------- -------------------- ------- -------------------
1        307200 SYSTEM               YES     /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
```

This type of information is useful for incomplete recovery because you can determine the schema of the database for the time to which you want to recover.

> **See Also:** *Oracle9i Recovery Manager Reference* for `REPORT` command syntax

Showing RMAN Configuration Settings

Run the `SHOW` command to display persistent configuration settings specified with the `CONFIGURE` command. These settings are persistent in the sense of being configured for use with any RMAN session.

By using the SHOW command, you can perform the queries discussed in the following sections:

- Showing All RMAN Configuration Settings
- Showing the RMAN Retention Policy Configuration Settings
- Showing the Automatic Channel Configuration Settings
- Showing the BACKUP Command Configuration Settings
- Showing the Snapshot Control File Filename
- Showing the Default Filenames Configured for Auxiliary Channels

See Also: *Oracle9i Recovery Manager Reference* for CONFIGURE command syntax, and *Oracle9i Recovery Manager Reference* for SHOW command syntax

Showing All RMAN Configuration Settings

You can use the CONFIGURE command to specify a variety of persistent settings for the RMAN environment. The SHOW ALL command displays both the CONFIGURE commands that you have issued as well as RMAN's default configurations. Note that you can return any CONFIGURE command to its default setting by running CONFIGURE ... CLEAR.

To show all RMAN configuration settings:

After connecting to the target database and recovery catalog (if you use one), run the SHOW ALL command. For example, enter the following:

```
SHOW ALL;    # shows all CONFIGURE settings, both user-entered and default
```

Sample output for SHOW ALL follows:

```
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT';
CONFIGURE DEVICE TYPE 'SBT' PARALLELISM 1;
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE DATAFILE BACKUP COPIES FOR DISK TO 2;
CONFIGURE DATAFILE BACKUP COPIES FOR SBT TO 1; #default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR SBT TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DISK TO 1; # default
CONFIGURE MAXSETSIZE TO 3072K;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/dbs/cf_snap.f';
CONFIGURE EXCLUDE FOR TABLESPACE 'example';
```

Note that the output is displayed so that you can paste it into a script and run it as an RMAN command file; hence, you can easily change your entire configuration. You can even run the script on a different target database.

Showing the RMAN Retention Policy Configuration Settings

You can use the CONFIGURE RETENTION POLICY command to specify either the number of days in the recovery window or the level of redundancy. By default, the retention policy is set to REDUNDANCY = 1.

To show the configuration policy:

After connecting to the target database and recovery catalog (if you use one), run the SHOW RETENTION POLICY command. For example, enter:

```
SHOW RETENTION POLICY;    # shows the CONFIGURE setting for the retention policy
```

Sample output for SHOW RETENTION POLICY follows:

```
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

Showing the Automatic Channel Configuration Settings

You can use the CONFIGURE command to set the following:

- The automatic channels
- The default device type used by the automatic channels
- The default number of automatic channels allocated

Showing the Automatic Channel Settings

Issue the SHOW CHANNEL command to display the settings for all automatically allocated channels.

To show the automatic channel settings:

After connecting to the target database and recovery catalog (if you use one), issue the SHOW CHANNEL command. For example, enter:

```
SHOW CHANNEL;    # shows the CONFIGURE setting for the automatic channels
```

Sample output for SHOW CHANNEL follows:

```
RMAN configuration parameters are:
CONFIGURE CHANNEL DEVICE TYPE 'SBT' RATE 1500K;
```

Showing the Configured Device Types

Issue the SHOW DEVICE TYPE command to display the configured devices and their parallelism settings. The DISK device type is preconfigured.

To show the default device type for automatic channels:

After connecting to the target database and recovery catalog (if you use one), run the SHOW DEVICE TYPE command. For example, enter:

```
SHOW DEVICE TYPE;    # shows the CONFIGURE DEVICE TYPE ... PARALLELISM settings
```

Sample output for SHOW DEVICE TYPE follows:

```
RMAN configuration parameters are:
CONFIGURE DEVICE TYPE 'SBT' PARALLELISM 1;
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
```

Showing the Default Device Type

Issue the SHOW DEFAULT DEVICE TYPE command to display the settings for the default device type used by the automatic channels. When you issue the BACKUP command, RMAN allocates only default channels of the type set by the CONFIGURE DEFAULT DEVICE TYPE command. This default device type setting is not in effect when you use commands other than BACKUP. Note that you cannot disable the default device type: it is always either DISK (default setting) or sbt.

To show the default device type for automatic channels:

After connecting to the target database and recovery catalog (if you use one), run the SHOW DEFAULT DEVICE TYPE command. For example, enter:

```
SHOW DEFAULT DEVICE TYPE;    # shows the CONFIGURE DEFAULT DEVICE TYPE setting
```

Sample output for SHOW DEFAULT DEVICE TYPE follows:

```
RMAN configuration parameters are:
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT';
```

Showing the BACKUP Command Configuration Settings

You can use the CONFIGURE command to set the following behavior for the BACKUP command:

- Which tablespaces are excluded from whole database backups (CONFIGURE EXCLUDE)
- The number of identical copies of each backup that RMAN creates (CONFIGURE ... BACKUP COPIES)

- The default maximum backup set size for jobs using automatic channels (CONFIGURE MAXSETSIZE)
- Whether RMAN backs up files when the identical files are already backed up (CONFIGURE BACKUP OPTIMIZATION)

Showing the Tablespaces Excluded from Backups

You can use the CONFIGURE EXCLUDE command to exclude tablespaces from whole database backups.

To show the tablespaces excluded from whole database backups:

After connecting to the target database and recovery catalog (if you use one), run the SHOW EXCLUDE command. For example, enter:

```
SHOW EXCLUDE;    # shows the CONFIGURE EXCLUDE setting
```

Sample output for SHOW EXCLUDE follows:

```
RMAN configuration parameters are:
CONFIGURE EXCLUDE FOR TABLESPACE 'OLD_ACCOUNTS';
```

Showing the Number of Identical Copies of Each Backup

Use the CONFIGURE ... BACKUP COPIES command to set the number of identical copies that RMAN makes of each backup. For example, if the value is 3, RMAN produces a total of three identical copies of each backup piece in a backup set.

To show the number of identical copies of each backup:

After connecting to the target database and recovery catalog (if you use one), run the SHOW ARCHIVELOG BACKUP COPIES or SHOW DATAFILE BACKUP COPIES commands. For example, enter:

```
SHOW DATAFILE BACKUP COPIES;    # shows the CONFIGURE DATAFILE BACKUP COPIES setting
```

Sample output for SHOW DATAFILE BACKUP COPIES follows:

```
RMAN configuration parameters are:
CONFIGURE DATAFILE BACKUP COPIES FOR DISK TO 2;
CONFIGURE DATAFILE BACKUP COPIES FOR SBT TO 1; #default
```

Showing the Default Maximum Size of Backup Sets

You can run the CONFIGURE MAXSETSIZE command to set the maximum sizes for RMAN backup sets. The size of a backup set is measured in the total bytes of the included backup pieces.

To show the maximum sizes for RMAN backup sets:

After connecting to the target database and recovery catalog (if you use one), issue the `SHOW MAXSETSIZE` command. For example, enter:

```
SHOW MAXSETSIZE;      # shows the CONFIGURE MAXSETSIZE settings
```

Sample output for `SHOW MAXSETSIZE` follows:

```
RMAN configuration parameters are:
CONFIGURE MAXSETSIZE TO 3072K;
```

Showing Whether Backup Optimization Is Enabled

You can use the `CONFIGURE BACKUP OPTIMIZATION` command to enable and disable backup optimization.

To show the status of backup optimization:

After connecting to the target database and recovery catalog (if you use one), issue the `SHOW BACKUP OPTIMIZATION` command. For example, enter:

```
SHOW BACKUP OPTIMIZATION;
```

Sample output for `SHOW BACKUP OPTIMIZATION` follows:

```
RMAN configuration parameters are:
CONFIGURE BACKUP OPTIMIZATION ON;
```

Showing the Snapshot Control File Filename

You can use the `CONFIGURE SNAPSHOT CONTROLFILE` command to set the default value for the snapshot control file. Issue the `SHOW SNAPSHOT CONTROLFILE` command to display this value.

Note: In releases prior to Oracle9*i*, the `CONFIGURE SNAPSHOT CONTROLFILE` command was called `SET SNAPSHOT CONTROLFILE`.

To show the snapshot control file filename:

After connecting to the target database and recovery catalog (if you use one), run the `SHOW SNAPSHOT CONTROLFILE` command. For example, enter:

```
SHOW SNAPSHOT CONTROLFILE NAME;    # shows the CONFIGURE SNAPSHOT CONTROLFILE setting
```

Sample output for `SHOW SNAPSHOT CONTROLFILE` follows:

```
RMAN configuration parameters are:
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/dbs/cf_snap.f';
```

See Also: "Configuring the Snapshot Control File Location" on page 8-28 to learn about the snapshot control file and its function

Showing the Default Filenames Configured for Auxiliary Channels

You can use the CONFIGURE AUXNAME command to set persistent filenames for auxiliary channels. For example, you can give new filenames for duplicate or standby datafiles, or datafiles in a TSPITR operation. Issue the SHOW AUXNAME command to display these filenames.

Note: In releases prior to Oracle9*i*, the CONFIGURE AUXNAME command was called SET AUXNAME.

To show persistent settings for auxiliary filenames:

After connecting to the target database and recovery catalog (if you use one), issue the SHOW AUXNAME command. For example, enter:

```
SHOW AUXNAME;    # shows the CONFIGURE AUXNAME setting
```

Sample output for SHOW AUXNAME follows:

```
RMAN configuration parameters are:
CONFIGURE AUXNAME FOR DATAFILE '/oracle/oradata/trgt/tools01.dbf' TO '/tmp/tools01.dbf';
```

Printing Scripts Stored in the Recovery Catalog

To print the text of a specified stored script, either run the PRINT SCRIPT command or query the RC_STORED_SCRIPT_LINE catalog view. To display a list of RMAN stored scripts, query the RC_STORED_SCRIPT catalog view.

This section contains these topics:

- Displaying the Text of Stored Scripts with PRINT SCRIPT
- Displaying the Text of Stored Scripts by Querying RC_STORED_SCRIPT_LINE
- Listing Stored Scripts by Querying RC_STORED_SCRIPT

Displaying the Text of Stored Scripts with PRINT SCRIPT

Use the PRINT SCRIPT command to display the text of a stored script. If desired, you can save the output to an RMAN log file.

To print a stored script to a message log:

1. Start RMAN and connect to the recovery catalog database and target database, specifying the `LOG` argument if you want to print to a message log. For example, enter the following to specify `rman_log`:

   ```
   % rman TARGET / CATALOG rman/cat@catdb LOG = rman_log
   ```

   Note that you must connect to the target database that you connected to when you created the script.

2. Issue a `PRINT SCRIPT` command to write the script to the log:

   ```
   PRINT SCRIPT b_whole;
   ```

3. Exit RMAN and use a text editor to view the script. For example, enter:

   ```
   RMAN> EXIT
   % vi rman_log
   ```

   See Also: *Oracle9i Recovery Manager Reference* for `PRINT SCRIPT` command syntax

Displaying the Text of Stored Scripts by Querying RC_STORED_SCRIPT_LINE

The `RC_STORED_SCRIPT_LINE` view contains the text of all stored scripts for all incarnations of the target databases registered in the recovery catalog.

To list the text for a specific script:

1. Start SQL*Plus and connect to the recovery catalog database as the catalog owner. For example, enter the following:

   ```
   % sqlplus rman/cat@catdb
   ```

2. Execute the following query, replacing *`database_key`* with the numerical primary key of the target database and *`script_name`* with the name of the script:

   ```
   SELECT TEXT
   FROM RC_DATABASE_INCARNATION i, RC_STORED_SCRIPT_LINE l
   WHERE i.DB_KEY = database_key
   AND SCRIPT_NAME = script_name
   AND i.DB_KEY = s.DB_KEY
   ```

```
AND i.CURRENT_INCARNATION = 'YES'
/
```

Sample output follows:

```
TEXT
--------------------------------------------------------------------------------
{ backup database plus archivelog;}
```

Listing Stored Scripts by Querying RC_STORED_SCRIPT

The RC_STORED_SCRIPT view contains information about all stored scripts for all incarnations of the target databases registered in the catalog.

To list the scripts for the current incarnation of a target database:

1. Start SQL*Plus and connect to the recovery catalog database as the catalog owner. For example, enter the following:

   ```
   % sqlplus rman/cat@catdb
   ```

2. Execute the following query in SQL*Plus, replacing *database_key* with the numerical primary key of the target database:

   ```
   SELECT DISTINCT SCRIPT_NAME
   FROM RC_DATABASE_INCARNATION i, RC_STORED_SCRIPT s
   WHERE i.DB_KEY = database_key
   AND i.DB_KEY = s.DB_KEY
   /
   ```

   Sample output follows:

   ```
   SCRIPT_NAME
   --------------------------------------------------------------------------------
   backup_db
   backup_system
   ```

See Also: *Oracle9i Recovery Manager Reference* for information about the RC_STORED_SCRIPT view

Querying the Recovery Catalog Views

The LIST, REPORT, and SHOW commands should provide you with all the repository information that you require. Nevertheless, you can sometimes also obtain useful information from the recovery catalog views, which are views in the catalog schema prefixed with RC_.

This section contains these topics:

- About Queries to the Recovery Catalog Views
- Querying Catalog Views for the Target DB_KEY or DBID Values

> **See Also:** *Oracle9i Recovery Manager Reference* for reference information about the recovery catalog views

About Queries to the Recovery Catalog Views

The recovery catalog views are not normalized, but are optimized for RMAN usage rather than user queries. RMAN obtains backup and recovery information from the target database control file and stores it in the catalog tables.

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 10 different target databases registered in the same recovery catalog, then the catalog views show the information for all incarnations of all databases registered in the catalog. You often have to perform joins among the views to distinguish the specific incarnation of the target database that you are interested in.

Most of the catalog views have a corresponding dynamic performance view in the database server. For example, `RC_BACKUP_PIECE` corresponds to `V$BACKUP_PIECE`. The primary difference between the catalog and server views is that each catalog view contains information about all the databases registered in the catalog, whereas the server view contains information only about itself. The two types of views often use different primary keys to uniquely identify rows.

Distinguishing a Database in the Catalog Views

Most of the catalog views contain the columns `DB_KEY` and `DBINC_KEY`. Each target database 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 each target database is uniquely identified by the `DBINC_KEY` primary key. When querying information about a specific incarnation of a target database, you should use these columns to specify the database. Then, you can perform joins with most of the other catalog views to obtain the desired information.

Distinguishing a Database Object in the Catalog Views

An important difference between catalog and V$ views is that a different system of unique identifiers is used for backup and recovery objects. For example, many V$ views such as V$ARCHIVED_LOG use the RECID and STAMP columns to form a concatenated primary key. The corresponding 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.

Querying Catalog Views for the Target DB_KEY or DBID Values

The DB_KEY value which is the primary key for a target 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 the target database. The DBID, which is a unique system-defined number given to every Oracle database, is what distinguishes one target database from another target database in the RMAN metadata.

Assume that you want to obtain information about one of the target databases registered in the recovery catalog. You can easily determine the DBID from this database either by looking at the output displayed when RMAN connects to the database, or querying a V$ view as in the following:

```
SELECT DBID
FROM V$DATABASE;

DBID
---------
598368217
```

You can then obtain the DB_KEY for a target database by running the following query, where *dbid_of_target* is the DBID that you previously obtained:

```
SELECT DB_KEY
FROM RC_DATABASE
WHERE DBID = dbid_of_target;
```

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 by using 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, you can execute this script:

```
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';
```

You should use the DB_NAME column to specify a database *only if* you do not have more than one database registered in the recovery catalog with the same DB_NAME. RMAN permits you to register more than one database with the same database name, but requires that the DBID values be different. For example, you can have ten databases with the DB_NAME value of prod1, each with a different DBID. Because the DBID is the unique identifier for every database in the metadata, use this value to obtain the DB_KEY and then use DB_KEY to uniquely identify the database.

RMAN Repository Query Examples

This section contains these topics:

- Listing Objects with Restrictions: Example
- Reporting Backups and Copies Not Needed for the Recovery Window: Example
- Reporting Redundant Backups and Copies: Example
- Generating Historical Reports of Database Schema: Example
- Listing Database Incarnations: Example

Listing Objects with Restrictions: Example

Use the LIST command to query the contents of the recovery catalog or the target database control file if no recovery catalog is used. You can use several different parameters to qualify lists.

The following example lists all backups of datafiles in tablespace tbs_1 that were made after June 11, 2000:

```
LIST BACKUP OF TABLESPACE users BY FILE COMPLETED BEFORE 'JUN 11 2001 00:00:00';
```

The following example lists backups on media management devices:

```
LIST BACKUP OF DATABASE SUMMARY DEVICE TYPE sbt;
```

The following example lists all copies of datafile 2 with the tag df2__copy that are in the /copy directory:

```
LIST COPY OF DATAFILE 2 TAG df2_copy LIKE '/copy/%';
```

Reporting Backups and Copies Not Needed for the Recovery Window: Example

Use the `REPORT` command to determine which copies and backups are superfluous and so can be deleted. For example, if you only need to be able to recover the database to a point within the last two weeks, then issue this command:

```
REPORT OBSOLETE RECOVERY WINDOW OF 14 DAYS;
```

You can then delete these obsolete backups and copies by issuing this command:

```
DELETE OBSOLETE RECOVERY WINDOW OF 14 DAYS;
```

Reporting Redundant Backups and Copies: Example

The following command reports all backups and copies on disk that are obsolete because three more recent backups or copies are already available:

```
REPORT OBSOLETE REDUNDANCY 3 DEVICE TYPE DISK;
```

The following command reports all backups on tape that are obsolete because at least two backups already exist that were made no more than one week ago:

```
REPORT OBSOLETE REDUNDANCY 2 UNTIL TIME 'SYSDATE-7' DEVICE TYPE sbt;
```

Generating Historical Reports of Database Schema: Example

The following commands reports the database schema in the present, a week ago, and on September 20, 2000:

```
REPORT SCHEMA;
REPORT SCHEMA AT TIME 'SYSDATE-7';
REPORT SCHEMA AT TIME "TO_DATE('09/20/01','MM/DD/YY')";
```

The following command reports on the database schema at SCN 953:

```
REPORT SCHEMA AT SCN 953;
```

The following command reports on the database schema at log sequence number 12 of thread 2:

```
REPORT SCHEMA AT SEQUENCE 12 THREAD 2;
```

Listing Database Incarnations: Example

Every time that you perform a `RESETLOGS` operation on a database, you create a new incarnation. This example lists all database incarnation of `trgt` registered in the recovery catalog:

```
LIST INCARNATION OF DATABASE trgt;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         TRGT      1224038686  NO     1            02-JUL-01
1       582       TRGT      1224038686  YES    59727        10-JUL-01
```

You can access the same information by querying `V$DATABASE_INCARNATION` in the target database and `RC_DATABASE_INCARNATION` in the recovery catalog database.

> **See Also:** "UNKNOWN Database Name Appears in Recovery Catalog: Scenario" on page 15-39 for information about `UNKNOWN` database names in the `LIST OUTPUT`

18 Performing Maintenance with Recovery Manager

This chapter describes how to manage the RMAN repository. Depending on how you implement RMAN, you can store this data either in a recovery catalog or exclusively in the control file. This chapter contains these topics:

Crosschecking Backups and Copies

To ensure that data about backup sets and image copies 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 recorded in the recovery catalog or the control file.

This section contains these topics:

- About RMAN Crosschecks
- Crosschecking Specific Backups and Copies
- Crosschecking Backups and Copies of Database Files

See Also: "Crosschecks of RMAN Backups and Copies" on page 7-8 for a conceptual overview of crosschecks

About RMAN Crosschecks

Crosschecks can update outdated repository information about backups and copies whose repository records do not match their physical status. For example, someone removes archived logs from disk with an operating system command so that the repository indicates that the logs are on disk when in fact they have deleted.

If the backup or copy is on disk, then the CROSSCHECK command determines whether the header of the file is valid. If the backup is on tape, then the command simply checks that the backup exists.

The possible status values for backups and copies are AVAILABLE, UNAVAILABLE, and EXPIRED. View the output of the LIST command or the recovery catalog views to determine the status of backups and copies.

Note: The CROSSCHECK command does *not* delete operating system files or remove repository records. You must use the DELETE command for these operations.

See Also: "Deleting Backups and Copies" on page 18-4 to learn how to delete files and update repository records, and *Oracle9i Recovery Manager Reference* for CROSSCHECK command syntax and a description of the repository status values

Crosschecking Specific Backups and Copies

You can use the LIST command to obtain a report of the backups and copies that you have made and then use the CROSSCHECK command to check that these files still exist. You can run the DELETE EXPIRED command to delete repository records for copies and backups that fail the crosscheck.

To crosscheck specified backups or copies:

1. Identify the desired backup pieces, backup sets or proxy copies that you want to check by issuing a LIST command. For example, issue:

   ```
   LIST BACKUP;
   LIST COPY;
   ```

2. If you do not have automatic channels configured, then allocate one or more channels of type MAINTENANCE. Otherwise, skip this step. For example, issue:

   ```
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
   ```

3. Check whether the specified backups or copies exist. For example, enter:

   ```
   CROSSCHECK BACKUPSET 1338, 1339, 1340;
   CROSSCHECK BACKUPPIECE TAG = 'nightly_backup';
   CROSSCHECK CONTROLFILECOPY '/tmp/control01.ctl';
   CROSSCHECK DATAFILECOPY 113, 114, 115;
   CROSSCHECK PROXY 789;
   ```

   If the backup or copy is no longer available, then RMAN marks it as EXPIRED. If it was marked EXPIRED and is now available, then RMAN marks it AVAILABLE.

4. If you manually allocated one or more maintenance channels, then release them:

   ```
   RELEASE CHANNEL;
   ```

Crosschecking Backups and Copies of Database Files

You can use the LIST command to determine which database objects that you have backed up or copied, then use the CROSSCHECK command to check whether these backups and copies exist.

To crosscheck backups or copies of specified files:

1. If you do not have automatic channels allocated, then allocate one or more channels of type `MAINTENANCE`. Otherwise, skip this step. For example, issue:

   ```
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
   ```

2. Check for the backups of the specified database, tablespace, datafile, control file, or archived redo log. Limit the crosscheck according to the time sequence. For example, check all backups of datafile `?/oradata/trgt/system01.dbf` over the last six months, as well as all logs and server parameter files on tape:

   ```
   CROSSCHECK BACKUP OF DATAFILE "?/oradata/trgt/system01.dbf"
     COMPLETED AFTER 'SYSDATE-180';
   CROSSCHECK BACKUP OF ARCHIVELOG ALL SPFILE;
   ```

3. Release the allocated maintenance channels:

   ```
   RELEASE CHANNEL;
   ```

See Also: *Oracle9i Recovery Manager Reference* for `CROSSCHECK` command syntax

Deleting Backups and Copies

You can use RMAN to delete backups, copies, and archived logs. RMAN deletes the specified files and removes their repository records.

This section contains these topics:

- Deleting Specified Backups and Copies
- Deleting Expired Backups and Copies
- Deleting Obsolete Backups and Copies

Note: Note that backups and copies with `DELETED` status do not appear in the `LIST` command output. Query the `V$` control file views instead.

See Also:

- Chapter 7, "RMAN Concepts III: Maintenance" for a conceptual overview of the RMAN deletion commands
- *Oracle9i Recovery Manager Reference* for DELETE command syntax
- *Oracle9i Recovery Manager Reference* for descriptions of the recovery catalog views

Deleting Specified Backups and Copies

In general, use the DELETE command to remove backups and copies that you no longer want to retain. This command removes the physical files, deletes the catalog records (if you use a catalog), and updates the records in the target control file to status DELETED.

To delete backups and copies and remove their repository records:

This procedure does not require the use of a recovery catalog.

1. Run the LIST output to obtain primary keys of backups and copies.

   ```
   LIST BACKUP OF DATABASE ARCHIVELOG ALL; # lists backups of db files and logs
   LIST COPY;
   ```

2. If you do not have automatic channels configured, then allocate a maintenance channel, for example:

   ```
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
   ```

3. Run the DELETE command to eliminate the specified physical files and their repository records. You can delete any type of object in the *recordSpec* clause, for example:

   ```
   DELETE BACKUPPIECE 101;
   DELETE CONTROLFILECOPY '/tmp/control01.ctl';
   DELETE NOPROMPT ARCHIVELOG UNTIL SEQUENCE = 300;
   ```

4. You can also delete files using DELETE BACKUP, DELETE COPY, or DELETE ARCHIVELOG as in these examples:

   ```
   DELETE BACKUP OF TABLESPACE users DEVICE TYPE sbt; # deletes only tape backups
   DELETE COPY OF CONTROLFILE LIKE '/tmp/%';  # LIKE specifies name of the copy
   DELETE NOPROMPT BACKUP OF SPFILE COMPLETED BEFORE 'SYSDATE-7';
   DELETE NOPROMPT ARCHIVELOG ALL
      BACKED UP 3 TIMES TO sbt; # backs up logs only if already backed up 3X to tape
   ```

If you run RMAN interactively, then RMAN asks for confirmation before deleting any files. If you specify `NOPROMPT`, then RMAN does not ask for confirmation.

5. Release the allocated maintenance channel:

```
RELEASE CHANNEL;
```

Deleting Expired Backups and Copies

You can use the `CROSSCHECK` command to determine whether backups and copies recorded in the repository still exist on disk or tape. If RMAN cannot locate the backups and copies, then it updates their records to `EXPIRED` status. You can then use the `DELETE EXPIRED` command to remove these expired records. Note that if for some reason the expired files still exist, then the `DELETE EXPIRED` command aborts with an error message.

To delete expired repository records:

1. If you do not have automatic channels configured, then allocate one or more maintenance channels, as in the following:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
```

2. If you have not perform a crosscheck recently, then issue a `CROSSCHECK` command. For example, issue:

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

3. Delete the expired backups. For example, issue:

```
DELETE EXPIRED BACKUP;
DELETE EXPIRED COPY;
```

4. Release the allocated maintenance channels:

```
RELEASE CHANNEL;
```

Deleting Obsolete Backups and Copies

Use the `DELETE OBSOLETE` command to remove backups and copies that are obsolete, that is, eligible for deletion. You can determine what qualifies a backup or copy for obsolete status in these ways:

- By configuring a retention policy with the `CONFIGURE` command

- By using the options on the DELETE OBSOLETE command

The DELETE OBSOLETE command removes both the physical files, deletes the catalog records (if you use a catalog), and updates the records in the target control file to status DELETED.

Deleting Backups and Copies Rendered Obsolete by the Retention Policy

If you specify the DELETE OBSOLETE command with no other operands, then RMAN deletes all obsolete backups and copies defined by the retention policy.

To delete backups and copies rendered obsolete by the retention policy:

1. If you do not have automatic channels configured, then allocate a maintenance channel for each device type on which you created backups or copies. The REPORT OBSOLETE command can report on the obsolete status of backups and copies on disk as well as backups on tape. For example, enter:

   ```
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
   ```

2. Issue a DELETE OBSOLETE command without any options to delete the objects defined as obsolete by the retention policy, for example:

   ```
   DELETE OBSOLETE;
   ```

   If you run RMAN interactively, then RMAN asks for confirmation before deleting any files. If you specify NOPROMPT, then RMAN does not ask for confirmation.

3. Release the allocated maintenance channel:

   ```
   RELEASE CHANNEL;
   ```

Deleting Backups and Copies Defined as Obsolete with the DELETE Command

You can run the DELETE OBSOLETE REDUNDANCY or DELETE OBSOLETE RECOVERY WINDOW commands to delete obsolete backups and copies. The redundancy or recovery window setting on the DELETE command overrides setting on the CONFIGURE RETENTION POLICY command.

To delete backups made obsolete by the DELETE OBSOLETE command:

1. If you do not have automatic channels configured, then allocate a maintenance channel, for example:

   ```
   ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
   ```

2. Run a DELETE OBSOLETE command with the REDUNDANCY or RECOVERY WINDOW options. These options define what is obsolete during the delete job. For example, enter:

   ```
   DELETE OBSOLETE REDUNDANCY = 3;
   DELETE OBSOLETE RECOVERY WINDOW OF 7 DAYS;
   ```

3. Release the allocated maintenance channel:

   ```
   RELEASE CHANNEL;
   ```

Forcing the Deletion of Backups and Copies

It is possible for the RMAN repository to indicate that an object has one status while the actual status of the object on the media is different. For example, the RMAN repository says that a backup set is AVAILABLE when it is in fact missing from the media management catalog. If you attempt to delete the object, then you receive a warning such as the following:

```
RMAN-06207: WARNING: 1 objects could not be deleted for DISK channel(s) due
RMAN-06208:          to mismatched status.  Use CROSSCHECK command to fix status
List of Mismatched objects
==========================
  Object Type   Filename/Handle
--------------- ---------------------------------------------------
Backup Piece    0id270ud_1_1
```

You can force RMAN to delete any object and remove its repository record by specifying the FORCE keyword. RMAN ignores any I/O errors. For example:

```
DELETE FORCE NOPROMPT BACKUPSET TAG 'weekly_bkup';
```

See Also:

- "Behavior of DELETE Command When the Repository and Media Do Not Correspond" on page 7-14
- *Oracle9i Recovery Manager Reference* for DELETE command syntax

Crosschecking and Deleting on Multiple RMAN Channels

This section contains these topics:

- About Allocating Multiple RMAN Channels for Maintenance Commands
- How RMAN Crosschecks and Deletes on Multiple Channels

- Crosschecking Disk and Tape Channels with One Command: Example
- Crosschecking on Multiple Oracle Real Application Cluster Nodes: Example
- Deleting on Disk and Tape Channels with One DELETE Command: Example
- Releasing Multiple Channels: Example

About Allocating Multiple RMAN Channels for Maintenance Commands

You can configure or manually allocate multiple maintenance channels before issuing `CROSSCHECK` or `DELETE` commands. RMAN searches for each backup or copy on all channels that have the same device type as the channel used to create the backup. The multichannel feature is designed for use in these scenarios:

- To allow crosschecking or deletion of all backup pieces or proxy copies, both on disk and tape, with a single command
- To make crosschecking and deleting work correctly in an Oracle Real Application Clusters configuration in which each backup piece or proxy copy exists only on one node

How RMAN Crosschecks and Deletes on Multiple Channels

When you configure or manually allocate multiple maintenance channels and run a `CROSSCHECK` or `DELETE` command, RMAN performs the crosscheck or delete on all channels that have the appropriate device type.

For example, assume that you prefer to manually allocate channels rather than use configured channels. You have a media manager configured, but have not yet made backups to tape. You have created only one backup of a database to disk as follows:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE DISK CONNECT 'SYS/sys_pwd@node2';
  BACKUP DATABASE;
}
```

Assume that you issue the following series of commands at the RMAN prompt:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node1';
AllOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/oracle@node2';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP OF DATABASE;
```

RMAN checks the first two channels because they both have the device type of disk and finds the backup on the second channel. However, RMAN does not perform a crosscheck on the third `sbt` channel because you have not yet made backups with a media manager.

Note: The `DELETE EXPIRED` command issues warnings if any objects marked as `EXPIRED` actually exist. In rare cases, the repository can mark an object as `EXPIRED` even though the object exists. For example, a directory containing an object 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.

Crosschecking Disk and Tape Channels with One Command: Example

RMAN can perform crosschecks on more than one media with a single command. 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 command to perform a crosscheck on both `DISK` and `sbt`:

```
CROSSCHECK BACKUP OF DATABASE;
```

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
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=13c5erba_1_1 recid=33 stamp=408382829
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=14c5erce_1_1 recid=34 stamp=408382863
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-674966176-20000915-00 recid=35 stamp=408382869
```

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:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP OF DATABASE;
```

Note that you do not have to manually allocate a disk channel because RMAN uses the preconfigured disk channel.

Crosschecking on Multiple Oracle Real Application Cluster Nodes: Example

This feature is useful in an Oracle Real Application Clusters configuration in which tape backups exist on various nodes in the cluster and are only visible on the nodes where they were created.

When crosschecking on multiple nodes, it is important to allocate channels at every node where backups were created. If you omit a channel for a node, or you set the parallelism to a value less than the number of nodes, then the backups created on that node will be marked `EXPIRED` in the repository. For example, you can configure RMAN for use with Oracle Real Application Clusters nodes as follows:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE CHANNEL 1 DEVICE TYPE DISK CONNECT 'SYS/oracle@node_1';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK CONNECT 'SYS/oracle@node_2';
```

Then, crosscheck the cluster nodes with the following command:

```
CROSSCHECK BACKUP;
```

Deleting on Disk and Tape Channels with One DELETE Command: Example

You can also perform deletions on all allocated channels. In the following example, you configure an `sbt` channel:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

Then, you run a command to delete all backup sets from disk and tape:

```
DELETE BACKUPSET;
```

RMAN uses both the `sbt` channel and the preconfigured `DISK` channel when deleting (sample output follows). Note that RMAN prompts you for confirmation before deleting any files:

```
using channel ORA_SBT_TAPE_1
using channel ORA_DISK_1

List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
```

```
------- ------- --- --- ----------- ----------- ----------
388     387     1   1   AVAILABLE   SBT_TAPE    12c5erb2_1_1
397     396     1   1   UNAVAILABLE SBT_TAPE    13c5erba_1_1
424     423     1   1   AVAILABLE   SBT_TAPE    14c5erce_1_1
428     427     1   1   AVAILABLE   SBT_TAPE    c-674966176-20000915-00
433     432     1   1   AVAILABLE   DISK        /oracle/dbs/16c5esv4_1_1
437     436     1   1   AVAILABLE   DISK        /oracle/dbs/c-674966176-20000915-01

Do you really want to delete the above objects (enter YES or NO)? y
deleted backup piece
backup piece handle=/oracle/dbs/16c5esv4_1_1 recid=36 stamp=408384484
deleted backup piece
backup piece handle=/oracle/dbs/c-674966176-20000915-01 recid=37 stamp=408384496
deleted backup piece
backup piece handle=12c5erb2_1_1 recid=32 stamp=408382820
deleted backup piece
backup piece handle=13c5erba_1_1 recid=33 stamp=408382829
deleted backup piece
backup piece handle=14c5erce_1_1 recid=34 stamp=408382863
deleted backup piece
backup piece handle=c-674966176-20000915-00 recid=35 stamp=408382869
```

The following example manually allocates `DISK` and `sbt` maintenance channels and then deletes specific backup sets from both disk and tape:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
DELETE BACKUPSET 1,2,3,4,5;
```

RMAN looks for the specified backup sets on the channels and deletes any it finds. If RMAN does not find a backup on any channel, then RMAN marks the object as deleted in the control file and deletes the recovery catalog record (if you use a recovery catalog).

Releasing Multiple Channels: Example

You can release all allocated maintenance channels by running this command:

```
RELEASE CHANNEL;
```

Changing the Availability Status of a Backup or Copy Record

Run the `CHANGE ... UNAVAILABLE` command when a backup or copy cannot be found or has migrated offsite. RMAN does not use files marked `UNAVAILABLE` in `RESTORE` or `RECOVER` commands. If the file is later found or returns to the main site, then you can mark it available again by issuing `CHANGE ... AVAILABLE`.

This section contains these topics:

- Marking a Backup or Copy as Unavailable
- Marking a Backup or Copy as Available

Marking a Backup or Copy as Unavailable

Run the `CHANGE ... UNAVAILABLE` command to mark a backup or copy as unavailable.

To mark a file's status in the repository as UNAVAILABLE:

1. Issue a `LIST` command to determine the availability status of RMAN backups and copies. For example, issue:

```
LIST BACKUP;
LIST COPY;
```

2. Run a `CHANGE ... UNAVAILABLE` command to mark a backup or copy as `UNAVAILABLE` in the RMAN repository. For example, enter:

```
CHANGE DATAFILECOPY '/tmp/control01.ctl' UNAVAILABLE;
CHANGE COPY OF ARCHIVELOG SEQUENCE BETWEEN 1000 AND 1012 UNAVAILABLE;
CHANGE BACKUPSET 12 UNAVAILABLE;
CHANGE BACKUP OF CONTROLFILE UNAVAILABLE;
CHANGE BACKUP OF SPFILE TAG "TAG20020208T154556" UNAVAILABLE;
```

Marking a Backup or Copy as Available

Use the `CHANGE ... AVAILABLE` command to mark a backup or copy as available. Note that this command does not check for the existence of the files or validate the file in any way: it merely updates the repository record to `AVAILABLE`. You can run `CROSSCHECK` to validate the file.

To mark a file's status in the repository as AVAILABLE:

1. Issue a `LIST` command to see which the availability status of RMAN backups and copies. For example, issue:

```
LIST BACKUP;
LIST COPY;
```

2. If you do not have automatic channels allocated, then allocate one or more maintenance channels. Otherwise, skip this step. For example, issue:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
```

3. If a previously unavailable backup or copy is reinstated, then issue a CHANGE ... AVAILABLE command to mark it as AVAILABLE in the RMAN repository. For example, enter:

```
CHANGE DATAFILECOPY '/tmp/system01.dbf' AVAILABLE;
CHANGE BACKUPSET 12 AVAILABLE;
CHANGE BACKUP OF SPFILE TAG "TAG20020208T154556" AVAILABLE;
```

See Also: *Oracle9i Recovery Manager Reference* for CHANGE command syntax

Exempting a Backup or Copy from the Retention Policy

The BACKUP ... KEEP command can archive a backup, potentially offsite, for longer than the time dictated by the user's retention policy. The backup is still a fully valid backup, however, and can be restored just as any other RMAN backup. This type of backup is called a **long-term backup**.

Note: The KEEP FOREVER clause requires the use of a recovery catalog.

Specify KEEP ... LOGS to save archived logs for a possible incomplete recovery and KEEP ... NOLOGS not to save archived logs for a possible incomplete recovery. Note that NOLOGS is not valid with an inconsistent backup.

Use the CHANGE command to alter the KEEP status of a backup or copy. 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.

To alter the KEEP status of a backup or copy:

Alter the retention status of the backup or copy. Issue CHANGE ... KEEP to make the file exempt from the retention policy, and CHANGE ... NOKEEP to make the file conform to the retention policy. This example allows a backup set to be marked obsolete by the retention policy:

```
CHANGE BACKUPSET 231 NOKEEP;
```

This example makes a datafile copy exempt from the retention policy for 180 days (6 months):

```
CHANGE DATAFILECOPY '/tmp/system01.dbf' KEEP UNTIL 'SYSDATE+180';
```

Cataloging Archived Logs and User-Managed Copies

You can make RMAN aware of the existence of archived logs that are not recorded in the repository as well as file copies that are created through means other than RMAN. This section contains the following topics:

- About Cataloging Archived Logs and User-Managed Copies
- When Should You Catalog a User-Managed Copy?
- Cataloging a User-Managed Copy

About Cataloging Archived Logs and User-Managed Copies

The control file keeps records of all archived logs generated by the target database. If you use a recovery catalog, then RMAN propagates the archived log information from the control file to the catalog. If you have to restore a control file backup, and if you change the archiving destination or format during recovery, then the repository will not have information about archived logs needed for recovery. Hence, you must catalog these logs if you want to use them for recovery.

A different case occurs when you make user-managed copies, that is, copies made with operating system commands. In the case, the repository has no record of them. You must manually notify RMAN when you make copies with an operating system utility such as the UNIX `cp` command.

Run the RMAN `CATALOG` command when:

- Adding information about a user-managed datafile copy, archived redo log copy, or control file copy to the recovery catalog and control file
- Cataloging a 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
- Recording the existence of user-managed copies of Oracle release 8.0 and higher databases created before RMAN was used
- Recording the existence of Oracle7 backups before migrating to Oracle release 8.0 or higher

When Should You Catalog a User-Managed Copy?

Catalog user-managed copies of database files in the following situations:

- You make a copy of a datafile or an archived logs with an operating system utility.
- You want to catalog user-managed copies of Oracle7 database files that meet the criteria specified in "Cataloging User-Managed Copies of Oracle7 Database Files" on page 18-16. These datafile copies allow you to recover the database if it crashes after migration but before you have a chance to take a backup of the migrated database.

Note: You cannot catalog backup sets or pieces.

Cataloging Consistent and Inconsistent User-Managed Copies

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, or archived redo log 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 Copies of Oracle7 Database Files

RMAN cannot catalog Oracle7 files, except in the following special circumstances:

- During the migration from Oracle7 to the current release, you shut down the Oracle7 database cleanly prior to running the migration utility, and back up the datafiles.
- You make backups of tablespaces that are `OFFLINE NORMAL` or read-only at the time of the backup, and do not bring these tablespaces online or make them read/write again before migration.

Oracle accepts these Oracle7 copies because no redo from the Oracle7 database is required to recover them. RMAN can then catalog and restore these copies in the Oracle9*i* database if no other backups exist.

The following scenario generates an Oracle7 datafile backup that you can catalog with RMAN:

1. Shut down the Oracle7 database cleanly.
2. Make a user-managed copy of the datafiles with an operating system utility such as the UNIX `cp` command or Windows `COPY` command.
3. Migrate the database to the current release.

The backups made before migration are identical to user-managed copies taken after migration, and so may be cataloged with RMAN.

> **See Also:** *Oracle9i Database Migration* to learn how to migrate an Oracle database

Cataloging a User-Managed Copy

Use the `CATALOG` command to propagate information about user-managed copies to the recovery catalog.

To catalog a user-managed copy:

1. Make a copy with an operating system utility. Note that `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.

```
SQL> ALTER TALBESPACE users BEGIN BACKUP;
% cp $ORACLE_HOME/oradata/trgt/users01.dbf /tmp/users01.dbf;
SQL> ALTER TABLESPACE users END BACKUP;
```

2. After connecting to the target database and, if desired, the recovery catalog, run the `CATALOG` command. For example, enter:

```
CATALOG DATAFILECOPY '/tmp/users01.dbf';
```

> **See Also:** *Oracle9i Recovery Manager Reference* for `CATALOG` command syntax

Uncataloging RMAN Records

This section contains the following topics:

- About Uncataloging RMAN Records
- Removing Records for Files Deleted with Operating System Utilities
- Removing Catalog Records with Status DELETED

About Uncataloging RMAN Records

Run the `CHANGE ... UNCATALOG` command to perform the following actions on RMAN repository records:

- Delete a specific backup or copy record from the recovery catalog (if you use one)
- Update a backup or copy record in the target control file repository to status `DELETED`

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 or copy through a means other than RMAN. For example, if you delete archived redo logs using an operating system utility, then remove the record for this log from the repository by issuing `CHANGE ARCHIVELOG ... UNCATALOG`.

In releases prior to Oracle9*i*, RMAN sometimes updated the status of records to `DELETED` in the recovery catalog rather than removing the records altogether. In Oracle9*i*, RMAN always removes the catalog record rather than marking it as `DELETED`. Hence, catalog records should only be marked with status `DELETED` if the catalog has been upgraded or the catalog was resynchronized from a backup control file. You can remove all repository records of backups and copies with status `DELETED` using the `prgrmanc.sql` script, which is located in an operating system specific location (`$ORACLE_HOME/rdbms/admin` on UNIX).

Removing Records for Files Deleted with Operating System Utilities

To remove catalog records for files deleted with operating system utilities, use the `CHANGE ... UNCATALOG` command.

To remove the catalog record for a backup or copy deleted with an operating system utility:

1. Run a `CHANGE ... UNCATALOG` command for the backups or copies that you deleted from the operating system with operating system commands. This example deletes repository references to 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.
   ```

Removing Catalog Records with Status DELETED

Use the `prgrmanc.sql` script to remove recovery catalog records with status `DELETED`. In releases prior to Oracle9*i*, RMAN updated recovery catalog records to `DELETED` status after deleting the physical files rather than removing the records.

In Oracle9*i* and later, RMAN always removes catalog records and never updates them to status `DELETED`. However, records with status `DELETED` can appear in the recovery catalog when you upgrade a catalog created prior to Oracle9*i* to the current release. For this special case, you can run the `prgrmanc.sql` script.

To remove all copy and backup records with status DELETED:

1. Start a SQL*Plus session and connect to the recovery catalog. This example connects to the database `rcat` as user `rman`:

   ```
   % sqlplus rman/cat@catdb
   ```

2. Run the script `prgrmanc.sql` script, which is stored in an operating system specific location (`$ORACLE_HOME/rdbms/admin` on UNIX):

   ```
   SQL> @prgrmanc
   ```

   The scripts removes all records with status `DELETED` from the recovery catalog.

Index