Oracle8*i*

Recovery Manager User's Guide and Reference

Release 2 (8.1.6)

December 1999

A76990-01

ORACLE

Oracle8*i* Recovery Manager User's Guide and Reference, Release 2 (8.1.6)

A76990-01

Primary Authors: Connie Dialeris, Joyce Fee, Lance Ashdown

Contributing Authors: Don Beusee, Greg Pongracz, Francisco Sanchez, Steve Wertheimer

Graphic Designer: Valarie Moore

Contents

2 Getting Started with Recovery Manager

3 Managing the Recovery Manager Repository

4 Generating Lists and Reports with Recovery Manager

5 Making Backups and Copies with Recovery Manager

6 Restoring and Recovering with Recovery Manager

9 Recovery Manager Troubleshooting

Part II Recovery Manager Reference

10 Recovery Manager Command Syntax

11 Recovery Catalog Views

Send Us Your Comments

Oracle8*i* Recovery Manager User's Guide and Reference, Release 2 (8.1.6)

A76990-01

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

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

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

- E-mail - infodev@us.oracle.com
- FAX - (650) 506-7228 Attn: Oracle Server Documentation
- Postal service:
  Oracle Corporation
  Server Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, and telephone number below.

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

Preface

This guide includes the conceptual and task-oriented information you need to perform backup, restore, and recovery procedures using the Recovery Manager utility. It assumes that you are familiar with the basic concepts of backup and recovery described in the *Oracle8i Backup and Recovery Guide*.

The *Oracle8i Recovery Manager User's Guide and Reference* contains information that describes the features and functionality of the Oracle8*i* Standard Edition and the Oracle8*i* Enterprise Edition products. The Standard Edition and Enterprise Edition have the same basic features, but several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to use Recovery Manager to perform automated tablespace point-in-time recovery, you must have the Enterprise Edition.

> **See Also:** *Getting to Know Oracle8i* for information about the differences between Oracle8*i* Standard Edition and the Oracle8*i* Enterprise Edition and the features and options that are available.

What's New in Oracle8*i*?

This section describes new features for Recovery Manager in Oracle release 8.0 and release 8.1.

Release 8.1.6

New features in release 8.1.6 include:

- The **autolocate** option of the **set** command causes RMAN to automatically discover which nodes of an OPS cluster contain the backups that you want to restore (see "set_run_option" on page 10-142).

- The **configure compatible** command controls the compatibility of the recovery catalog packages with the RMAN executable (see "configure" on page 10-47 and "Setting Recovery Catalog Compatibility" on page 3-4).
- The **resetlogs** option of the **alter database** command allows you to open the database with the RESETLOGS option (see "alterDatabase" on page 10-16).
- The **change ... delete**, **delete expired**, and **backup ... delete input** commands can now remove catalog records rather than update them to status DELETED (see Table 3–2, "Maintenance Commands and Scripts" on page 3-18).

Release 8.1.5

New features in release 8.1.5 include the following:

- 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 (see "Proxy Copy" on page 1-22).
  - 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 (see "backup" on page 22).
  - When a channel is allocated, RMAN displays in its log a text message identifying the media management product used to take backups on that channel. When the media manager encounters an error, it returns a text error message explaining the error, which will be displayed in the RMAN log.
  - The new **send** command allows commands to be sent directly from an RMAN session to the media management software (see "send" on page 10-136).
- The commands **crosscheck backup** (see "crosscheck" on page 10-64), **change ... crosscheck** (see "change" on page 10-38), and **delete expired backup** (see "deleteExpired" on page 10-69) allow for the synchronization of the recovery catalog with the media manager's catalog. RMAN can determine whether backups and copies are actually on disk or tape and update their repository record if they are not.

- The output of the **list backup** command (see "list" on page 10-83) 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.
- A new command, **report need backup redundancy** (see "report" on page 10-110), alerts the user that a new backup is required when fewer than a user-specified number of backups of a data file exist.
- The **create catalog** command (see "createCatalog" on page 10-59) creates the recovery catalog. It replaces `catrman.sql` and associated scripts in the `dbs/admin` directory.
- 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 "upgradeCatalog" on page 10-158).
- The **drop catalog** command (see "dropCatalog" on page 10-74) removes the recovery catalog schema.
- The RMAN commands **startup** (see "startup" on page 10-152), **shutdown** (see "shutdown" on page 10-147), and **alter database ... mount**/**alter database ... open** (see "alterDatabase" on page 10-16) have the same syntax as their equivalent SQL commands.
- The **duplicate** command allows creation of a replicated database using the backups of another database (see Chapter 7, "Creating a Duplicate Database with Recovery Manager").
- When backing up on multiple nodes of a parallel server, 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.
- Recovery Manager can now create up to four concurrent copies of each backup piece (see "Duplexed Backup Sets" on page 1-44).
- 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.
- 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. RMAN now issues an error.

- You can perform TSPITR (Tablespace Point -in-Time Recovery) without a recovery catalog (see "Performing TSPITR Without a Recovery Catalog" on page 8-4).
- 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 "Monitoring RMAN Jobs" on page 9-12).

Structure

This book contains the following parts and chapters:

Part / Chapter	Contents
PART 1	**Using Recovery Manager**
Chapter 1, "Recovery Manager Concepts"	Describes the features and functionality of the Recovery Manager utility.
Chapter 2, "Getting Started with Recovery Manager"	Describes how to start using Recovery Manager, including whether you should use a recovery catalog, how you can execute commands, and how to connect to a target database.
Chapter 3, "Managing the Recovery Manager Repository"	Describes how to manage RMAN metadata using either a recovery catalog or the control file.
Chapter 4, "Generating Lists and Reports with Recovery Manager"	Describes how to generate lists of your backups and copies and reports describing which datafiles need to be backed up, which backups are obsolete, and the structure of the database schema at a specified time.
Chapter 5, "Making Backups and Copies with Recovery Manager"	Describes how to make backup and image copies of datafiles, control files, and archived redo logs using Recovery Manager.
Chapter 6, "Restoring and Recovering with Recovery Manager"	Describes how to restore backups and copies of datafiles, control files, and archived redo logs and perform media recovery on datafiles.
Chapter 7, "Creating a Duplicate Database with Recovery Manager"	Describes how to create a duplicate of your target database on either a local or remote host using backups of your target database datafiles.
Chapter 8, "Performing Point-in-Time Recovery with Recovery Manager"	Provides planning guidelines and step-by-step instructions for performing tablespace point-in-time recovery with Recovery Manager.

Part / Chapter	Contents
Chapter 9, "Recovery Manager Troubleshooting"	Provides suggestions for interpreting RMAN error messages and debugging output as well as an account of the most common RMAN problems.
PART 2	**Recovery Manager Reference**
Chapter 10, "Recovery Manager Command Syntax"	Provides syntax diagrams, parameter descriptions, and code samples for all RMAN commands.
Chapter 11, "Recovery Catalog Views"	Describes the views available with a recovery catalog.
"Glossary"	Defines terms relevant for backup and recovery.

Changes to This Book

The following aspects of this manual are new in release 8.1.6:

- In release 8.1.5, all the backup and recovery documentation was located in the *Oracle8i Backup and Recovery Guide*. In release 8.1.6, this documentation is divided into the following books:
  - *Oracle8i Backup and Recovery Guide*
  - *Oracle8i Recovery Manager User's Guide and Reference*
  - *Oracle8i Standby Database Concepts and Administration*
- Chapter 9, "Recovery Manager Troubleshooting" gives tips for debugging RMAN as well as descriptions of common problems.

Audience

This guide is for database administrators (DBAs) who administer the backup, restore, and recovery operations of an Oracle database system using the Recovery Manager utility.

Knowledge Assumed of the Reader

Readers of this guide are assumed to be familiar with:

- Relational database concepts and basic database administration as described in *Oracle8i Concepts* and the *Oracle8i Administrator's Guide*.
- Basic backup and recovery concepts and strategies as described in the *Oracle8i Backup and Recovery Guide*.

- The operating system environment under which they are running Oracle.

Conventions

This section explains the conventions used in this manual including the following:

- Text
- Recovery Manager Syntax Diagrams and Notation
- Code Examples

Text

This section explains the conventions used within the text:

UPPERCASE Characters

Uppercase text is used to call attention to tablespace names, initialization parameters, and SQL keywords.

For example, "If you create a private rollback segment, the name must be included in the ROLLBACK_SEGMENTS parameter of the `init.ora` file. You can view this information by issuing a SHOW PARAMETER statement in SQL*Plus."

Italicized Characters

Italicized words within text are book titles, new vocabulary, emphasized words, or variables in SQL or Recovery Manager syntax.

For example, "An *archived redo log* is an online redo log that has been copied offline. You *must* run your database in ARCHIVELOG mode to enable this feature. If you are using Recovery Manager, you can specify an archived redo log in a **backup** command by using the **archivelog like** *'/oracle/archive/arc_*'* sub-clause."

Bold Characters

Bold words within text are Recovery Manager keywords or operating system-specific commands.

For example, "Use the Recovery Manager **backup** command to back up your database. Alternatively, use the UNIX **cp** command to copy files."

Monospaced Characters

Filenames and directories appear in a monospaced font. Also, monospaced characters in text preceding a code example indicates a filename or keyword used in the sample code.

For example, "This command backs up the tablespace TBS_1:

```
run {
     allocate channel c1 type disk;
     backup tablespace tbs_1;
}
```

Recovery Manager Syntax Diagrams and Notation

For information about Recovery Manager syntax conventions, see "Conventions Used in this Reference" on page 10-2.

Code Examples

SQL, SQL*Plus, and Recovery Manager commands and statements appear separated from the text of paragraphs in a monospaced font. For example:

```
INSERT INTO emp (empno, ename) VALUES (1000, 'SMITH');
ALTER TABLESPACE users ADD DATAFILE 'users2.ora' SIZE 50K;
run {
     allocate channel ch1 type disk;
     backup database;
}
```

When you run RMAN from the command line, the command prompt appears as RMAN>. When you issue commands from the SQL*Plus command line, the prompt appears as SQL>. These prompts are displayed in the code examples only when they are necessary to prevent confusion.

You can execute SQL, SQL*Plus, and RMAN commands in different environments on different platforms. As much as possible, this guide attempts to provide generic documentation, that is, documentation that is not specific to any operating system or interface. Nevertheless, it is sometimes necessary for illustrative purposes to show how the syntax works at the operating system level. In these cases, this book uses examples from a UNIX command-line interface and employs the % symbol to indicate the operating system prompt. For example:

```
% rman target / rcvcat rman/rman@inst2
RMAN> startup
```

How to Use This Guide

This manual contains the following basic types of information:

- Conceptual (Chapter 1, "Recovery Manager Concepts")
- Procedural (chapters 3 - 8)
- Reference (Part II, "Recovery Manager Reference")
- Troubleshooting (Chapter 9, "Recovery Manager Troubleshooting")

To acquaint yourself with the basic features of RMAN, read Chapter 1, "Recovery Manager Concepts" and then Chapter 2, "Getting Started with Recovery Manager". Refer to the procedural chapters for information about specific tasks. Finally, refer to the reference chapters for clarification about RMAN syntax or the columns in the catalog views.

Your Comments Are Welcome

We value and appreciate your comments as an Oracle user and reader of our references. As we write, revise, and evaluate, your opinions are the most important input we receive. At the front of this reference is a reader's comment form that we encourage you to use to tell us both what you like and what you dislike about this (or other) Oracle manuals. If the form is missing, or you would like to contact us, please use the following address or fax number:

Server Technologies Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood City, CA 94065
FAX: 650-506-7228

You can also e-mail your comments to the Information Development department at the following e-mail address: infodev@us.oracle.com

Part I

Using Recovery Manager

1

Recovery Manager Concepts

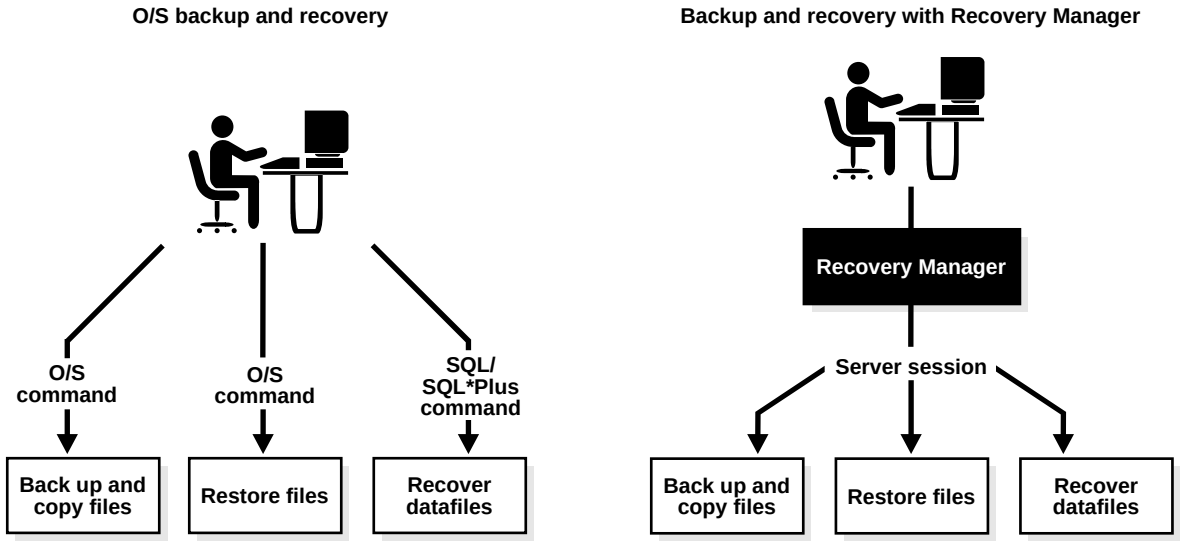
This chapter describes the basic concepts for the Oracle Recovery Manager (RMAN) utility, and includes the following topics:

Overview of Recovery Manager

Recovery Manager (RMAN) is an Oracle tool that allows you to back up, copy, restore, and recover datafiles, control files, and archived redo logs. It is included with the Oracle server and does not require separate installation. You can invoke RMAN as a command line utility from the operating system (O/S) prompt or use the GUI-based Enterprise Manager Backup Manager.

RMAN users server sessions to automate many of the backup and recovery tasks that were formerly performed manually. For example, instead of requiring you to locate appropriate backups for each datafile, copy them to the correct place using operating system commands, and choose which archived logs to apply, RMAN manages these tasks automatically.

Figure 1–1 Comparison of RMAN and O/S Backup and Recovery Procedures



When you start RMAN, the following events occur:

- The RMAN user session starts on the client.
- RMAN creates two default server sessions that connect to the target database. The *target database* is the database that you want to back up or restore.
- If you perform I/O on disk or tape, RMAN requires that you allocate one channel for each disk or device. A channel corresponds to a server session.

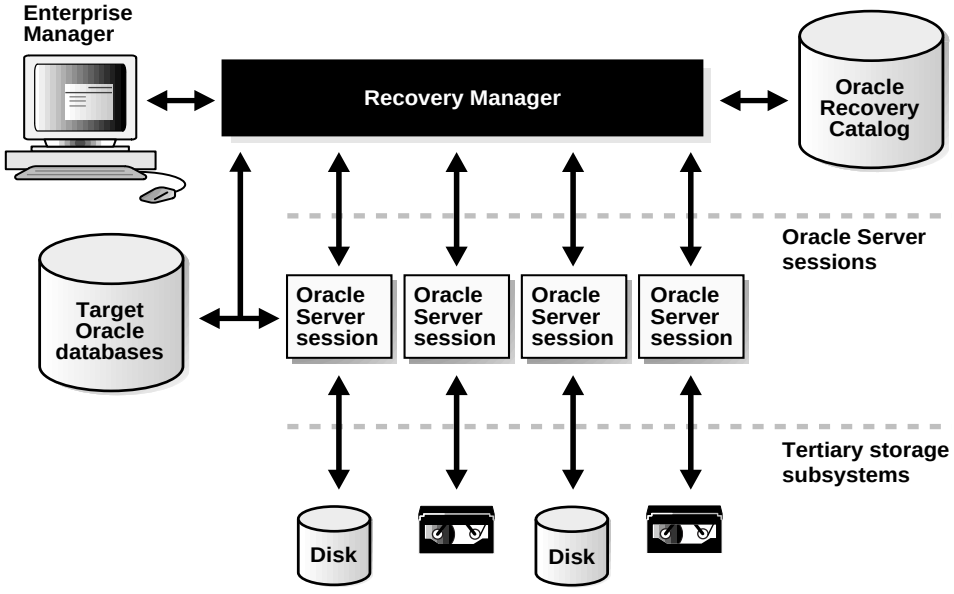
- If you connect to a recovery catalog, RMAN creates a server session on the recovery catalog database.

When you use RMAN to connect to the target database, RMAN allocates server sessions to perform the backup and recovery operations through a PL/SQL interface. RMAN physically stores its backups and copies on disk or, if you use media management software, on tape.

RMAN stores metadata about its backup and recovery operations in the recovery catalog, which is a centralized repository of information, or exclusively in the control file. Typically, the recovery catalog is stored in a separate database. If you do not use a recovery catalog, RMAN uses the control file as its repository of metadata.

Note: Recovery Manager is only compatible with Oracle databases of release 8.0 or higher. It is *not* compatible with the Oracle7 Enterprise Backup Manager (EBU).

Figure 1–2 Recovery Manager with Optional Recovery Catalog



See Also: *Oracle8i Migration* for information about RMAN compatibility and upgrade issues.

Recovery Manager Features

RMAN automates important backup and recovery operations. For example, Recovery Manager is able to:

- Use server sessions to back up and copy the database, tablespaces, datafiles, control files, and archived redo logs.
- Compress backups of datafiles so that only those data blocks that have been written to are included in a backup.
- Store frequently executed backup and recovery operations in scripts.
- Perform incremental backups, which back up only those data blocks that have changed since a previous backup.
- Recover changes to objects created with the NOLOGGING option by applying incremental backups (recovery with redo logs does not apply these changes).
- Create a duplicate of your production database for testing purposes.
- Use third-party media management software.
- Generate a printable message log of all backup and recovery operations.
- Use the recovery catalog to automate both restore and recovery operations.
- Perform automatic parallelization of backup and restore operations.
- Restore a backup using a backup control file and automatically adjust the control file to reflect the structure of the restored datafiles.
- Find datafiles that require a backup based on user-specified limits on the amount of redo that must be applied for recovery.
- Perform crosschecks to determine whether archived materials in the media management catalog are still available.
- Test whether specified backups can be restored.

> **See Also:** *Getting to Know Oracle8i* for a description of RMAN features that are new to this version of Oracle.

What Recovery Manager Is Not

RMAN is not:

- The only option for performing backup and recovery operations. You can also back up and restore files using operating system commands, and use SQL / SQL*Plus statements to recover them.

- A recovery catalog. A recovery catalog is an optional schema containing information about RMAN operations. Note also that RMAN does not store backups or copies in the recovery catalog; rather, it stores information *about* backups and copies in the recovery catalog.
- Compatible with Oracle databases *before* release 8.0.
- The Enterprise Backup Utility (EBU), which is an Oracle7 utility. RMAN is *not* compatible with EBU.
- An operating system-specific backup utility. RMAN is a generic utility that can run on a number of different operating systems.
- A scheduling utility. RMAN has no built-in scheduling functionality enabling it to perform regular backups automatically.

Recovery Manager Commands

RMAN provides commands that you can use to manage all aspects of backup and recovery operations.

Note: All RMAN commands for Oracle release 8.0 work in release 8.1.

This section contains the following topics:

- Recovery Manager PL/SQL Packages
- How Recovery Manager Compiles and Executes Commands
- Types of Recovery Manager Commands
- User Execution of Recovery Manager Commands
- User Termination of Recovery Manager Commands
- Recovery Manager Command Errors

> **See Also:** Chapter 10, "Recovery Manager Command Syntax" for a complete account of RMAN commands and their syntax.

Recovery Manager PL/SQL Packages

The RMAN executable uses PL/SQL procedures to interpret commands. The PL/SQL packages perform two main functions:

- Maintain the RMAN repository in the control file or recovery catalog.
- Communicate with Oracle and the operating system to create, restore, and recover backup sets and image copies.

The DBMS_RCVCAT and DBMS_RCVMAN packages are created by the **create catalog** command. RMAN uses DBMS_RCVCAT to maintain information in the recovery catalog and DBMS_RCVMAN to query the recovery catalog or control file.

The DBMS_BACKUP_RESTORE package is created by the `dbmsbkrs.sql` and `prvtbkrs.plb` scripts. This package is automatically installed in every Oracle database when the `catproc.sql` script is run. This package interfaces with the Oracle server and the operating system to provide the I/O services for backup and restore operations as directed by Recovery Manager.

> **See Also:** *Oracle8i Supplied PL/SQL Packages Reference* for more information about the DBMS_RCVCAT, DBMS_RCVMAN, and DBMS_BACKUP_RESTORE packages.

How Recovery Manager Compiles and Executes Commands

Recovery Manager processes commands in two phases:

- Compilation
- Execution

When you issue a Recovery Manager command such as **backup**, RMAN generates output alerting you to the various phases of command processing. Following is sample output for a **backup tablespace** command. Note the RMAN messages that begin `RMAN-03xxx`:

```
RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel ch1: starting full datafile backupset
RMAN-08502: set_count=48 set_stamp=346765191 creation_time=15-OCT-98
RMAN-08010: channel ch1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00017 name=/oracle/dbs/tbs_14.f
RMAN-08522: input datafile fno=00003 name=/oracle/dbs/tbs_11.f
RMAN-08522: input datafile fno=00004 name=/oracle/dbs/tbs_12.f
RMAN-08522: input datafile fno=00007 name=/oracle/dbs/tbs_13.f
RMAN-08013: channel ch1: piece 1 created
RMAN-08503: piece handle=/oracle/dbs/1gaamds7_1_1 comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:04
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: ch1
```

> **See Also:** Chapter 9, "Recovery Manager Troubleshooting" to learn how to interpret RMAN output.

Compilation

During the compilation phase, RMAN performs two operations:

- Name translation
- Construction of PL/SQL job steps

Name Translation Most RMAN commands require you to identify what object the command should operate on. *Name translation* is the operation whereby RMAN translates what you specified in the command into a list of one or more entities that it really operates on. An entity in this sense can be a datafile, an archived redo log, a datafile copy, a control file copy, and so forth. For example, when you enter the command **backup database,** RMAN translates the keyword **database** into a list of all the datafiles that are in the database. Similarly, when you issue the following:

```
backup archivelog from time='xxx' until time='yyy'
```

RMAN translates the *archivelogRecordSpecifier* clause (the **from ... until ...** part of the command) into a list of archived redo logs.

Construction of PL/SQL Job Steps RMAN performs its work by dynamically generating one or more PL/SQL programs and then executing them. In essence, RMAN compiles the commands you issue into PL/SQL programs. RMAN itself contains a copy of the PL/SQL engine, and executes the PL/SQL programs internally during the execution phase. These programs make remote procedure calls to the target database to perform backup, restore, and other necessary operations.

A single RMAN command can result in the generation of multiple PL/SQL programs. For example, a **backup database** command generates one PL/SQL program for each backup set that is to be created. Similarly, **restore database** generates one PL/SQL program for each backup set that needs to be restored.

Execution

During the execution phase, RMAN schedules and runs the PL/SQL programs it compiled during the compilation phase. RMAN assigns one PL/SQL program to each channel (that is, each server session) that you have allocated. The channels execute their PL/SQL program concurrently. For example, if you allocate three channels, then RMAN executes three of the PL/SQL programs at the same time.

RMAN is able to execute concurrent PL/SQL programs because the remote procedure calls (RPCs) that the programs make to the target database use the non-blocking User Program Interface (UPI) feature, thereby allowing RMAN to switch to another channel when one channel makes a non-blocking RPC call. RMAN uses an internal polling mechanism to detect when a non-blocking RPC call has completed. After it completes, RMAN resumes the PL/SQL program.

Types of Recovery Manager Commands

RMAN uses two basic types of commands: *stand-alone commands* and *job commands*. With the exception of the **change**, **crosscheck**, and **delete** commands, stand-alone commands are self-contained. In contrast, job commands must appear within the brackets of a **run** command.

After you connect to the target and optional recovery catalog, you will execute most of your RMAN commands within **run**. Following is a typical example of a **run** statement:

```
run {
     allocate channel c1 type 'sbt_tape';
     restore database;
     recover database;
}
```

> **See Also:** Chapter 10, "Recovery Manager Command Syntax" for an exhaustive description of RMAN syntax.**:**

Stand-Alone Commands

Unlike job commands, stand-alone commands do not appear as sub-commands within **run**. Following are some of the commands that can appear by themselves:

- **catalog**
- **change**
- **create catalog, drop catalog, upgrade catalog**
- **create script**, **delete script**, **replace script**
- **crosscheck**
- **delete expired backupset**
- **list**
- **report**

Some of these commands are not strictly stand-alone, however, because they must be preceded by an **allocate channel for maintenance** command.

> **See Also:** "rmanCommand" on page 10-130 to learn about stand-alone command syntax.

Job Commands

Unlike stand-alone commands, job commands must appear within the brackets of a **run** command. Following are examples of job commands:

- **allocate channel**
- **backup**
- **copy**
- **duplicate**
- **recover**
- **restore**
- **switch**

RMAN executes the job commands inside of a **run** command block sequentially. If any command within the block fails, 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:** "run" on page 10-133 to learn about job command syntax.

Command Exceptions

Most commands are either stand-alone commands or job commands. If you try to issue a job command outside of a **run** block or issue a stand-alone command inside a **run** block, RMAN issues a syntax error message. The following exceptions, however, can function as both stand-alone and job commands:

- **@**
- **@@**
- **host**

- **send**
- **shutdown**
- **startup**
- **sql**

Command-Line Arguments

RMAN supports a number of command-line arguments that you can specify when you connect to RMAN. You can specify most of these arguments only on the command line; the exceptions are **target** and **catalog**, which you can specify either on the command line or through the **connect** command after RMAN has started. The **connect** command allows you to avoid putting passwords on the command line, since this practice sometimes poses a security problem.

> **See Also:** "cmdLine" on page 10-42 to learn about command-line options.

User Execution of Recovery Manager Commands

RMAN uses a command language interpreter (CLI) that allows you to 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.

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/sys_pwd@prod1 catalog rman/rman@rcat
RMAN> run {
2>  allocate channel d1 type disk;
3>  backup database;
4>  }
```

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 should 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). RMAN parses the command file in its

entirety before compiling or executing any commands. Batch mode is most suitable for performing regularly scheduled backups via an operating system job control facility.

In this example, the RMAN sample script from "Interactive Mode" on page 1-10 has been placed into a command file called `b_whole_10.rcv`. You can run this file from the operating system command line and write the output into the log file `rman_log.f` as follows:

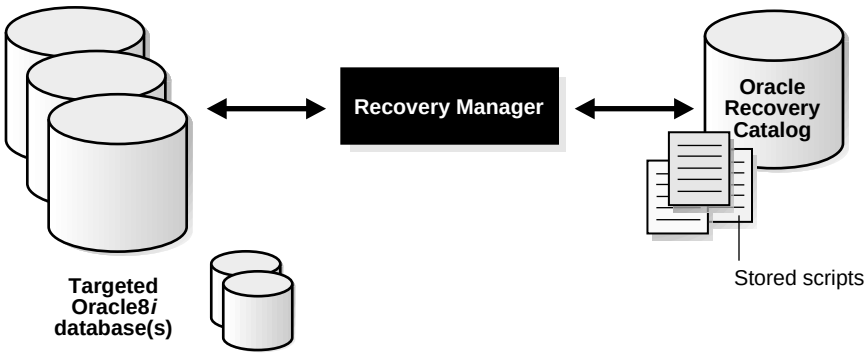
```
% rman target / catalog rman/rman@rcat @b_whole_l0.rcv log rman_log.f
```

> **See Also:** "cmdLine" on page 10-42 for more information about RMAN command line options.

Stored Scripts

A *stored script* is a set of RMAN commands that is enclosed in braces and stored in the recovery catalog. Maintaining a stored script allows you to plan, develop, and test a set of commands for backing up, restoring, or recovering the database. Stored scripts also minimize the potential for operator errors. Each stored script relates to only one database.

Figure 1–3 Recovery Manager Stored Scripts



To create a stored script, either enter your script interactively into the RMAN command-line interface, or type the RMAN commands into a command file and run the command file.

Following is an example of a stored script:

```
replace script b_whole_l0 {
     # back up whole database and archived logs
       allocate channel d1 type disk;
       allocate channel d2 type disk;
       allocate channel d3 type disk;
       backup
         incremental level 0
         tag b_whole_l0
         filesperset 6
         format '/dev/backup/prod1/df/df_t%t_s%s_p%p'
          (database);
         sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
         backup
          filesperset 20
          format '/dev/backup/prod1/al/al_t%t_s%s_p%p'
          (archivelog all
           delete input);
}
```

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: "Storing Scripts in the Recovery Catalog" on page 3-27 for more information on scripts. Also refer to the sample scripts stored in your `$ORACLE_HOME/rdbms/demo` directory.

User Termination of Recovery Manager Commands

You have two methods for terminating an RMAN job while it is executing:

- Press CTRL+C (or the equivalent "attention" key combination for your system), 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.
- Terminate the session on the operating system. For example, you can execute the UNIX `kill` command.

You can identify the session ID for an RMAN channel by looking in the RMAN log for the RMAN-08500 message, which is displayed for each allocated channel.

Recovery Manager Command Errors

On various occasions it may be important for you to determine whether RMAN successfully executed your 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.

The simplest 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, 1 otherwise. For example, if you are running UNIX and using the C shell, RMAN outputs the return code into a shell variable called $status.

The second easiest 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,
channel, comma, controlfilecopy, current, database, datafile, datafilecopy, delete,
diskratio, filesperset, format, include, (, parms, pool, ;, skip, setsize, tablespace,
tag"
RMAN-01007: at line 1 column 58 file: standard input
```

> **See Also:** Chapter 9, "Recovery Manager Troubleshooting" to learn how to interpret RMAN error messages and troubleshoot problems.

Recovery Manager Repository

The RMAN repository is the collection of metadata about your 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 3, "Managing the Recovery Manager Repository" to learn how to manage the RMAN repository, and "Understanding Catalog-Only Command Restrictions" on page 3-45 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 your target databases that RMAN uses and maintains. Conveniently, you do not need an additional database for storing the recovery catalog because you can put the recovery catalog in an existing 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 the following topics:

- Contents of the Recovery Catalog
- Resynchronization of the Recovery Catalog
- Backups of the Recovery Catalog
- Compatibility of the Recovery Catalog

Contents of the Recovery Catalog

The recovery catalog contains information about:

- 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 and SQL commands.

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 (only if the database is open), and online redo logs. In a full resynchronization, RMAN updates all changed records, including schema records.

When you issue certain commands in RMAN, the program 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.

It is a good idea to run RMAN once a day or so and issue the **resync catalog** command to ensure that the catalog stays current. 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 1-17 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 resynchronizes. This snapshot control file ensures that RMAN has a consistent view of the control file either when refreshing the recovery catalog or when querying 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 how current the recovery catalog is.

The Oracle8*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 a snapshot control file. For instructions, see "Determining the Snapshot Control File Location" on page 2-3.

> **See Also:** "Resynchronizing the Recovery Catalog" on page 3-29 to learn how to resynchronize the recovery catalog, and "resync" on page 10-127 for **resync catalog** 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 your recovery catalog.

> **See Also:** "Backing Up the Recovery Catalog" on page 3-37 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 is constituted by 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 an 8.1.6 RMAN executable with an 8.1.4 target database, and store the repository in an 8.1.5 recovery catalog database whose catalog tables were created in version 8.1.6.

The RMAN **configure compatible** command allows you to specify the compatibility level of the recovery catalog. For example, you can specify that a catalog can only function with an RMAN executable of release 8.1.5 or later. The compatibility setting determines the behavior of certain commands such as **change ... delete** and **backup ... delete input**.

See Also:

- *Oracle8i Migration* for a chart describing the compatibility of the components in the RMAN environment.
- "Setting Recovery Catalog Compatibility" on page 3-4 to learn how to set the compatibility level of the recovery catalog.
- "configure" on page 10-47 for **configure compatible** syntax.

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 another database for the sole purpose of maintaining the recovery catalog is burdensome.

Oracle does not support the following features in this operational mode:

- Stored scripts
- Restore and recovery when the control file is lost or damaged

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 *non-circular reuse records*.

Circular Reuse Records Circular reuse records contain non-critical 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 re-use records are arranged in a logical ring. When all available record slots are full, Oracle either expands the control file to make room for a new records or overwrites the oldest record. The CONTROL_FILE_RECORD_KEEP_TIME

initialization parameter 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 3-47 to learn how to manage Oracle's treatment of circular-reuse records.

Non-Circular Reuse Records Non-circular reuse records contain critical information that does not change often and cannot be overwritten. Some examples of information in circular reuse records include:

- Datafiles
- Online redo logs
- Redo threads

Recovery Without a Catalog

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

- Use a minimum of two multiplexed or mirrored control files, each on separate disks.
- Keep excellent records of which files were backed up, the date they were backed up, and the names of the backup pieces that each file was written to (see Chapter 4, "Generating Lists and Reports with Recovery Manager"). Keep all Recovery Manager backup logs.

WARNING: It is difficult to restore and recover if you lose your control files and do not use a recovery catalog. The only way to restore and recover when you have lost all control files and need to restore and recover datafiles is to call Oracle Support Services. Support will need to know:

- **The current schema of the database.**
- **Which files were backed up.**
- **When the files were backed up.**
- **The names of the backup pieces containing the files.**

See Also: "Understanding Catalog-Only Command Restrictions" on page 3-45 for a complete list of commands that are disabled unless you use a recovery catalog.

Media Management

To utilize tape storage for your 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 the purpose of backing up and recovering data.

Note: The Oracle kernel uses a Legato shared library that allows it to perform backups through the Legato Storage Manager, which is a media manager bundled with Oracle on certain operating systems. See the *Legato Storage Manager Administrator's Guide* for more information.

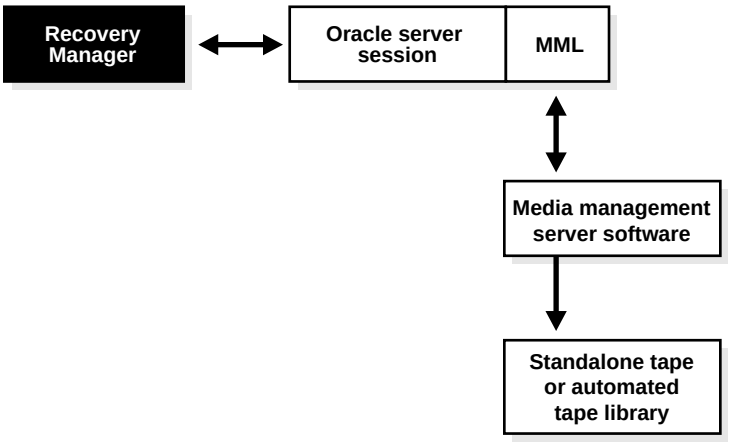
Oracle publishes a media management API that third-party vendors can use to build software that works with RMAN. To use RMAN to make backups to sequential media such as tape, integrate media management software with your Oracle software. Note that Oracle does not need to connect to the media management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

Note: Because RMAN uses the same media management API as the Oracle7 Enterprise Backup Utility (EBU), RMAN is compatible with the same media managers as EBU. Check with your vendor to see whether your version of the media software is compatible with RMAN (see "Backup Solutions Program" on page 1-23).

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.

Figure 1–4 shows the architecture for a media manager integrated with Oracle.

Figure 1–4 Architecture for MML Integrated with Oracle



The Oracle executable is the same used when a user connects to the database. The MML in the diagram above represents vendor-supplied media management software 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.

Backup and Restore Operations Using a Media Manager

The following Recovery Manager script performs a datafile backup to a tape drive controlled by a media manager:

```
run {
     # Allocating a channel of type 'sbt_tape' specifies a media management device
     allocate channel ch1 type 'sbt_tape';
     backup datafile 10;
}
```

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 requests the media manager to load a tape and 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 with robotic arm, 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:

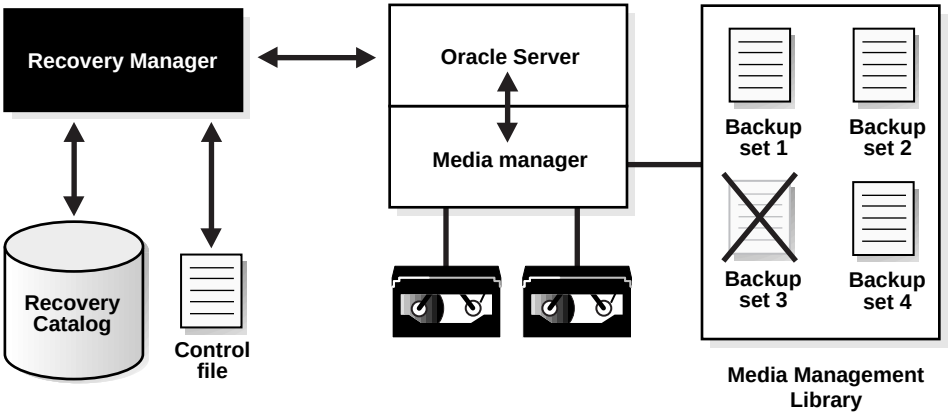
1. Oracle requests the restore of a particular file.
2. The media manager identifies the tape containing the file and reads the tape.
3. The media manager passes the information back to the Oracle server session.
4. The Oracle session writes the file to disk.

Media Manager Crosschecks

Sometimes tapes in the media management library become unavailable. RMAN can perform crosschecks to determine that backup pieces are still available; in this way, the RMAN repository stays synchronized with the media management catalog.

For example, you can issue a **crosscheck backup** command to check all backups on tape. If RMAN cannot find a particular backup, it changes its status to EXPIRED in the RMAN repository. Issue a **list** command or access the recovery catalog views to determine the status of backups and copies.

Figure 1–5 Crosschecks



See Also:

- "Crosschecking the RMAN Repository" on page 3-14 to learn how to perform crosschecks
- "crosscheck" on page 10-64 for **crosscheck** syntax
- "change" on page 10-38 for **change ... crosscheck** syntax

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 recovery to the media manager, which in turn makes all decisions regarding how and when to move the data.

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 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 a record of each proxy-copied file in the control file. RMAN uses this information to resynchronize the recovery catalog. Access the V$PROXY_DATAFILE and V$PROXY_ARCHIVEDLOG dynamic performance views to obtain the proxy copy information. Use the **change ... proxy** command to delete or change the status of a proxy backup.

Note: 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: *Oracle8i Reference* for more information about V$PROXY_DATAFILE and V$PROXY_ARCHIVEDLOG, and "backup" on page 10-22 for **backup** command syntax

Media Manager Testing

A new client program, `sbttest`, is a stand-alone 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 either the bundled Legato Storage Manager or another 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) provides a range of media management products that are compliant with Oracle's Media Management Library (MML) specification. Software that is compliant with the MML interface enables an Oracle server session to issue commands to the media manager to back up or restore a file. The media manager responds to the command by loading, labeling, or unloading the requested tape.

One MML-compliant product is the Legato Storage Manager (LSM), which is available for several common platforms. Oracle includes LSM with software that it ships for these platforms. If your version of the Oracle software includes LSM, refer to the *Legato Storage Manager Administrator's Guide* to learn about its features. If your shipment includes other BSP media management products, then refer to your platform-specific documentation for information.

Several other products may be available for your platform from media management vendors. For a current list of available products, you can link to the BSP page from the Oracle Backup and Recovery website:

`http://www.oracle.com/database/recovery`

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

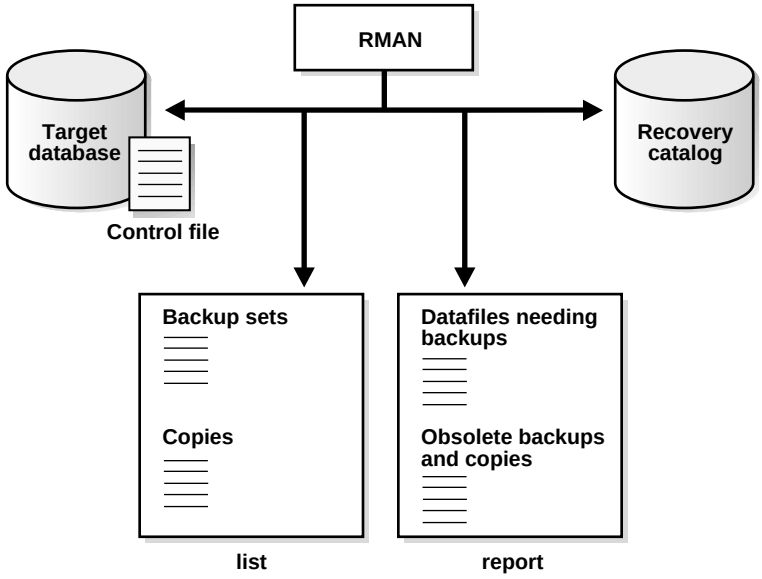
If you wish to use a specific media management product, contact the media management vendor directly to determine whether it is a member of the Oracle BSP. Note that Oracle does not certify media vendors for compatibility with RMAN, so any questions about availability, version compatibility, and functionality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

Lists and Reports

Use RMAN's **report** and **list** commands to gain information about backups and image copies. RMAN obtains the information from the RMAN repository: either the control file or the recovery catalog.

The **list** command lists the contents of the RMAN repository, whereas the **report** command performs a more detailed analysis. RMAN writes the output from these commands to the screen or to a log file.

Figure 1–6 RMAN Lists and Reports



Lists of Backups and Copies

The **list** command queries the recovery catalog or control file and produces a record of its contents. Use it to list:

- Backups of a specified list of datafiles.
- Image copies of a specified list of datafiles.
- Backups of any datafile that is a member of a specified list of tablespaces.
- Image copies of any datafile that is a member of a specified list of tablespaces.
- Backups of any archived redo logs with a specified name and/or within a specified range.
- Image copies of any archived redo log with a specified name and/or within a specified range.
- Incarnations of a specified database.

See Also: "Generating Lists" on page 4-2 to learn how to generate lists of backups and image copies, and "list" on page 10-83 for reference material on the **list** command.

Reports on Backups, Copies, and Database Schema

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 datafiles have not been backed up recently?
- Which datafiles need to be backed up because fewer than *n* number of backups or copies are available?
- Which backups and copies can be deleted?
- 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 *ophaned*, that is, unusable in a restore operation, because they belong to incarnations of the database that are not direct predecessors of the current incarnation?

Issue 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.

The **report** command lists backup sets and datafile copies that can be deleted either because they are redundant or because they are unrecoverable. A datafile is considered *unrecoverable* if an unrecoverable operation has been performed against an object residing in the datafile subsequent to the last backup.

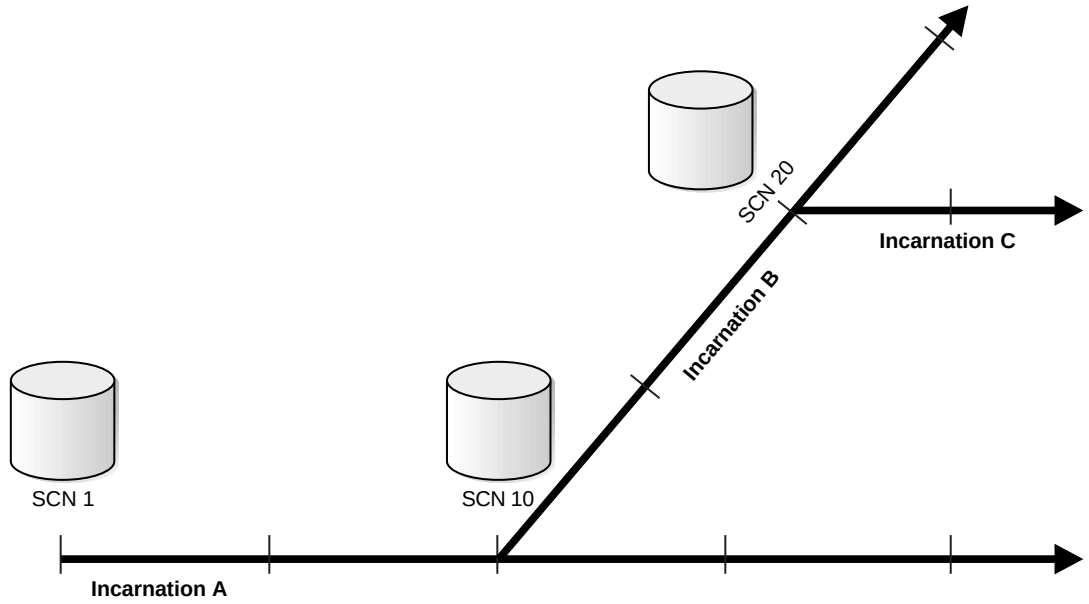
Note: A datafile that does not have a backup is not considered unrecoverable. You can recover such datafiles via the CREATE DATAFILE command provided that redo logs starting from when the file was created still exist.

Reporting on Orphaned Backups

The **report** command allows you to list *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.

For example, see the incarnation scenario depicted in Figure 1–7:

Figure 1–7 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:

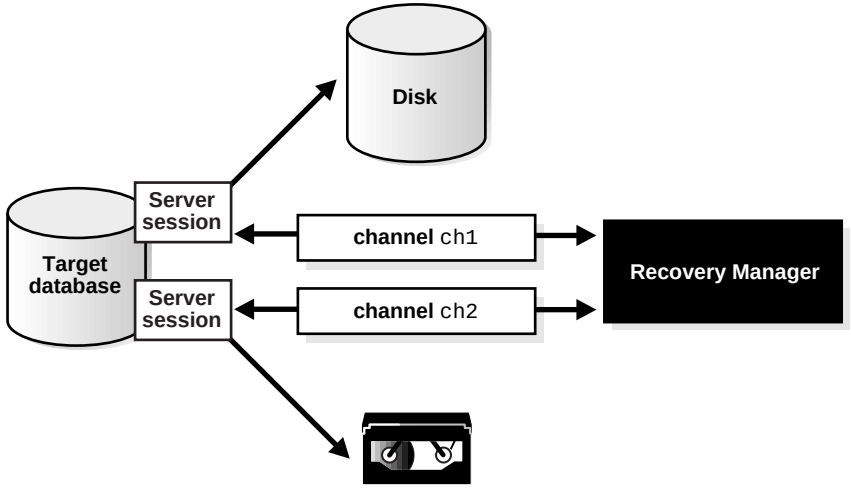
Current Incarnation	Orphaned Backups	Usable Backups (Non-Orphaned)
Incarnation A	All backups from incarnations B and C.	All backups from incarnation A.
Incarnation B	Backups from incarnation A after SCN 10. All backups from incarnation C.	Backups from incarnation A prior to SCN 10. All backups from incarnation B.
Incarnation C	All backups from incarnation A after SCN 10. All backups from incarnation B after SCN 10.	All backups from incarnation A prior to SCN 10. All backups from incarnation B prior to SCN 20. All backups from incarnation C.

See Also: "Generating Reports" on page 4-5 to learn how to generate reports, and "report" on page 10-110 for reference material on the **report** command.

Channel Allocation

You must allocate a *channel* before you execute backup and recovery commands. Each allocated channel establishes a connection from RMAN to a target or auxiliary database (either a database created with the **duplicate** command or a temporary database used in TSPITR) instance by starting a server session on the instance. This server session performs the backup and recovery operations. Only one RMAN session communicates with the allocated server sessions.

Figure 1–8 Channel Allocation



The **allocate channel** command (executed within a **run** command) and **allocate channel for maintenance** command (executed 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. Each channel usually corresponds to one output device, unless your media management library is capable of hardware multiplexing.

WARNING: Oracle does not recommend hardware multiplexing of Oracle backups.

See Also: "allocate" on page 10-10 for reference material on the **allocate channel** command, and "allocateForMaint" on page 10-14 for reference material on the **allocate channel for maintenance** command.

Channel Control Options

Use channel control commands to:

- Control the operating system resources RMAN uses when performing **backup**, **copy, restore**, and **recover** operations.

- Affect the degree of parallelism for a backup (in conjunction with the **filesperset** parameter of the **backup** command).
- Specify limits on I/O bandwidth consumption (**set limit channel ... readrate**).
- Specify limits on the size of backup pieces (**set limit channel ... kbytes**).
- Specify limits on the number of concurrently open files (**set limit channel ... maxopenfiles**).
- Send vendor-specific commands to the media manager (**send**).

On some platforms, these 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.

Whether the **allocate channel** command causes your 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.

In version 8.1, the **allocate channel** command causes RMAN to contact the media manager whenever the type specified is other than **disk**. Depending on the media management software used, the media manager allocates resources at this time or waits until a backup file is opened for reading or writing. Note that in version 8.0, 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 **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.

You must allocate a maintenance channel before issuing a **change ... delete** command, which calls the operating system to delete a file, or a **change ... crosscheck** command. 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. You can only allocate one maintenance channel at a time.

See Also: "set" on page 10-138 for reference material on the **set** command, and "allocateForMaint" on page 10-14 for reference material on the **allocate channel for maintenance** command.

Channel Parallelization

You can allocate multiple channels, 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 **allocate channel** or **allocate auxiliary channel** command 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 Parallel Server (OPS) configuration for distributing the workload across different nodes.

Factors Affecting Degrees of Parallelization

RMAN internally handles parallelization of **backup**, **copy**, and **restore** commands. You only need to specify:

- More than one **allocate channel** command.
- The objects that you want to back up, copy, or restore.

RMAN executes commands serially; that is, it completes the current command before starting the next one. Parallelism is exploited only within the context of a single command. Consequently, if you want 5 datafile copies, issue a single **copy** command specifying all 5 copies rather than 5 separate **copy** commands.

The following RMAN script uses serialization to create the file copies: 5 separate **copy** commands are used to back up the files. Only one channel is active at any one time.

```
run {
    allocate channel  c1 type disk;
    allocate channel  c2 type disk;
    allocate channel  c3 type disk;
    allocate channel  c4 type disk;
    allocate channel  c5 type disk;
    copy datafile 22 to '/dev/prod/backup1/prod_tab5_1.dbf';
    copy datafile 23 to '/dev/prod/backup1/prod_tab5_2.dbf';
    copy datafile 24 to '/dev/prod/backup1/prod_tab5_3.dbf';
```

```
    copy datafile 25 to '/dev/prod/backup1/prod_tab5_4.dbf';
    copy datafile 26 to '/dev/prod/backup1/prod_tab6_1.dbf';
}
```

The following statement uses *parallelization* on the same example; one RMAN **copy** command copies 5 files, with 5 channels available. All 5 channels are *concurrently active*—each channel copies one file.

```
run {
    allocate channel  c1 type disk;
    allocate channel  c2 type disk;
    allocate channel  c3 type disk;
    allocate channel  c4 type disk;
    allocate channel  c5 type disk;
    copy datafile 5 to '/dev/prod/backup1/prod_tab5_1.dbf',
         datafile 23 to '/dev/prod/backup1/prod_tab5_2.dbf',
         datafile 24 to '/dev/prod/backup1/prod_tab5_3.dbf',
         datafile 25 to '/dev/prod/backup1/prod_tab5_4.dbf',
         datafile 26 to '/dev/prod/backup1/prod_tab6_1.dbf';
}
```

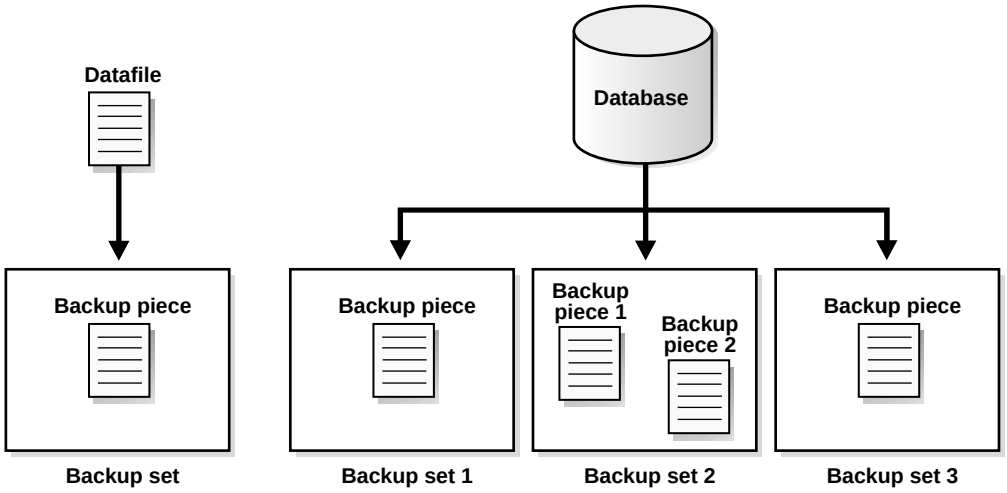
> **See Also:** *Oracle8i Parallel Server Documentation Set: Oracle8i Parallel Server Concepts; Oracle8i Parallel Server Setup and Configuration Guide; Oracle8i Parallel Server Administration, Deployment, and Performance* for information about parallelization in an OPS configuration.

Backup Sets

When you execute the **backup** command, you create one or more backup sets. A *backup set*, which is a logical construction, contains one or more physical *backup pieces*. Backup pieces are operating system files that contain the backed up datafiles, control files, or archived redo logs. You cannot split a file across different backup sets or mix archived redo logs and datafiles into a single backup set.

A backup set is a complete set of backup pieces that constitute a full or incremental backup of the objects specified in the **backup** command. Backup sets are in an RMAN-specific format; image copies, in contrast, are available for use without additional processing.

Figure 1–9 Backup Sets Contain One or More Backup Pieces



When backing up files, the target database must be mounted or open. If the database is mounted and was not shut down abnormally prior to mounting, then RMAN produces a consistent backup. The control file must be current.

If the database is in ARCHIVELOG mode, then the target database can be open or closed; you do not need to close the database cleanly (although Oracle recommends you do so that the backup is consistent). If the database is in NOARCHIVELOG mode, then you must close it cleanly prior to taking a backup.

Note: You cannot make a backup of a plugged-in tablespace until after it has been specified read-write.

This section contains the following topics:

- Storage of Backup Sets
- Backup Set Compression
- Filenames for Backup Pieces
- Number and Size of Backup Sets
- Size of Backup Pieces
- Multiplexed Backup Sets

- Duplexed Backup Sets
- Parallelization of Backups
- Backup Errors

> **See Also:** Chapter 5, "Making Backups and Copies with Recovery Manager" to learn how to make backups, and "backup" on page 10-22 for information on the **backup** command.

Storage of Backup Sets

RMAN can create backup sets that are written to disk or tertiary storage. If you specify **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 pathname.

Using a sequential output device or media management system that is available and supported by Oracle on your operating system, you can write backup sets to sequential output media such as magnetic tape. If you specify **type** *'sbt_tape'*, then you can back up to any media supported by the media management software.

Note that you cannot archive directly to tape, but RMAN does allow you to back up archived redo logs from disk to tape. If you specify the **delete input** option, RMAN deletes the file after backing it up. RMAN automatically stages the required archived logs from tape to disk during recovery.

Note: RMAN cannot delete obsolete backups automatically. To learn how to delete old backups manually, see "Deleting Backups and Copies and Updating Their Status in the RMAN Repository" on page 3-18.

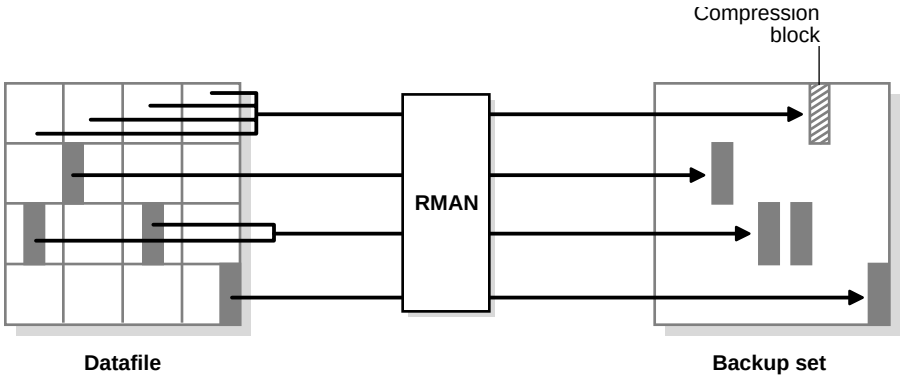
Backup Set Compression

RMAN performs *compression* on its backups, which means that the server session does not write datafile blocks that have never been used. Image copies of a datafile, however, always contain all datafile blocks.

Data blocks in the datafile are grouped into buffers. When RMAN encounters a used data block in a buffer, it writes only the used block to the backup set. When RMAN encounters four contiguous buffers of *unused* input blocks, it writes one *compression block* (of size DB_BLOCK_SIZE) to the backup set.

Use the DB_FILE_DIRECT_IO_COUNT initialization parameter to set the size of the buffer. For example, set the parameter to a value of 64K. In this case, RMAN writes one compression block for each 256K of contiguous unused input blocks in the input file.

Figure 1–10 Backup Set Compression



> **See Also:** *Oracle8i Designing and Tuning for Performance* to learn how RMAN buffers its backups.

Filenames for Backup Pieces

You can either let RMAN determine a unique name for the backup piece or use the **format** parameter to specify a name. If you do not specify a filename, RMAN uses the **%U** substitution variable to guarantee a unique name. The **backup** command provides substitution variables that allow you to generate unique filenames.

> **See Also:** "backup" on page 10-22 for reference material on the **format** parameter and the substitution variables of the **backup** command.

Number and Size of Backup Sets

Use the *backupSpec* clause to list what you want to back up as well as specify other useful options. The number and size of backup sets depends on:

- 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 **setsize** parameter, which limits the overall size in bytes of a backup set.

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

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

> **Note:** RMAN writes backup sets serially; striping a backup set across multiple output devices is not supported.

- RMAN always tries to divide the backup load so that all allocated channels have roughly the same amount of work to do.
- The maximum upper limit for the number of files per backup set is determined by the **filesperset** parameter of the **backup** command.
- The maximum upper limit for the size in bytes of a backup set is determined by the **setsize** parameter of the **backup** command.

Using the filesperset Parameter

The **filesperset** parameter limits the number of files that can go in a backup set. The default value of this parameter is calculated by RMAN as follows: RMAN compares the value 64 to the rounded-up ratio of number of files / number of channels, and sets **filesperset** to the lower value. For example, if you back up 70 files with one channel, RMAN divides 70/1, compares this value to 64, and sets **filesperset** to 64 because it is the lowest value.

The number of backup sets produced by RMAN is the rounded-up ratio of number of datafiles / **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 / 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 8 files using 3 channels. Because **filesperset** is not specified, RMAN compares 64 to 3 (8/3 rounded up) and chooses the lesser value of 3. RMAN creates 3 backup sets and groups the files into the sets so that each set is approximately the same size:

```
List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1677    90         368815605  0  368815602  98         22-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    1681    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/32avnbfi_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    10   /vobs/oracle/dbs/tbs_31.f           0  Full 75989      22-JUN-99
    19   /vobs/oracle/dbs/tbs_33.f           0  Full 75989      22-JUN-99

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1678    91         368815605  0  368815602  97         22-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    1682    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/31avnbfi_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    9    /vobs/oracle/dbs/tbs_24.f           0  Full 75988      22-JUN-99
    11   /vobs/oracle/dbs/tbs_32.f           0  Full 75988      22-JUN-99
    18   /vobs/oracle/dbs/tbs_25.f           0  Full 75988      22-JUN-99

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1679    92         368815605  0  368815601  96         22-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    1683    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/30avnbfh_1_1
```

```
    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    5    /vobs/oracle/dbs/tbs_21.f           0  Full 75990      22-JUN-99
    6    /vobs/oracle/dbs/tbs_22.f           0  Full 75990      22-JUN-99
    8    /vobs/oracle/dbs/tbs_23.f           0  Full 75990      22-JUN-99
```

Specifying filesperset: Example If you back up 8 datafiles using 3 channels and specify **filesperset** = 2, RMAN places no more than 2 datafiles in each backup set. Consequently, RMAN creates at least 8 / 2 = 4 backup sets: it may create more depending on the sizes of the datafiles as well as other factors.

```
List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1713    96         368815867  0  368815865  102        22-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ----------------------
    1718    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/36avnbnp_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    5    /vobs/oracle/dbs/tbs_21.f           0  Full 75996      22-JUN-99
    8    /vobs/oracle/dbs/tbs_23.f           0  Full 75996      22-JUN-99

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1714    97         368815867  0  368815865  103        22-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ----------------------
    1719    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/37avnbnp_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    6    /vobs/oracle/dbs/tbs_22.f           0  Full 75997      22-JUN-99
    9    /vobs/oracle/dbs/tbs_24.f           0  Full 75997      22-JUN-99

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1715    98         368815867  0  368815866  104        22-JUN-99
```

```
    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- -----------------------
    1720    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/38avnbnq_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    10   /vobs/oracle/dbs/tbs_31.f           0  Full 75998      22-JUN-99
    18   /vobs/oracle/dbs/tbs_25.f           0  Full 75998      22-JUN-99

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ----------------------
1716    99         368815870  0  368815869  105        22-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- -----------------------
    1721    1   1   AVAILABLE   22-JUN-99              /vobs/oracle/dbs/39avnbnt_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    11   /vobs/oracle/dbs/tbs_32.f           0  Full 75999      22-JUN-99
    19   /vobs/oracle/dbs/tbs_33.f           0  Full 75999      22-JUN-99
```

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 that are 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:** "backup" on page 10-22 to learn about the *backupSpec* clause, and *Oracle8i Designing and Tuning for Performance* to learn about RMAN buffer management.

Using the setsize Parameter

Each channel produces at least one backup set. To specify the maximum size of the backup set in bytes, use the **setsize** parameter in the **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 more than one backup set.

If you are writing to disk, use the **setsize** parameter of the **backup** command to restrict the total byte size of a backup set to the maximum file size supported by your media manager or operating system.

Relationship Between setsize and filesperset The following table compares the **setsize** parameter to the **filesperset** parameter:

	Meaning	You decide...	RMAN decides...
setsize	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, thus causing RMAN to create more backup sets of the specified size.	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.	How many bytes to make the backup sets to keep the parameter restriction.

Because **filesperset** has a default value, if you set **setsize** 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 / **setsize**
  - Total number of datafiles / **filesperset**
- RMAN enforces both the **filesperset** limit and the **setsize** limit. If necessary, RMAN creates more backup sets than are calculated in the above comparison.
- RMAN uses a complex algorithm of best fit so that the majority of backup sets are as close to the **setsize** limit with files as close to the **filesperset** limit as possible.

Specifying setsize: 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 10Mb, you issue the following:

```
run {
     allocate channel c1 type 'sbt_tape';
     allocate channel c2 type 'sbt_tape';
     backup database setsize = 10000;
}
```

Because you did not set **filesperset**, RMAN performs a calculation to obtain the default value. It compares 64 to 50 / 2 and sets **filesperset** = 25. RMAN then compares:

- 50000 / 10000 (blocks / **setsize**) = 5
- 50 / 25 (files / **filesperset**) = 2

Consequently, RMAN attempts to make 5 backup sets, with each backup set containing no more than 25 files and totalling no more than 10Mb in size.

Note that if you set **setsize** 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 SETSIZE: file# 1 /oracle/dbs/tbs_01.f
```

> **See Also:** "backup" on page 10-22 for information on the **setsize** parameter.

Size of Backup Pieces

Each backup set contains at least one backup piece. If you do not restrict the backup piece size, Oracle generates a backup set containing only one file. To restrict the size of the physical files in a backup set, use the **kbytes** option of the **set channel limit** command for each allocated channel.

For example, you can restrict the backup piece size for a datafile backup to 9Mb as follows:

```
run {
     allocate channel c1 type disk;
     set limit channel c1 kbytes = 9000;
     backup datafile 1;
}
```

A **list backup** command reveals that RMAN created 5 backup pieces rather than 1 backup piece to conform to the **set channel limit** size restriction:

```
List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
295     9          377549368  0  377549334  9          30-SEP-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    297     1   1   AVAILABLE   30-SEP-99              /oracle/dbs/09b81sgm_1_1
    298     2   1   AVAILABLE   30-SEP-99              /oracle/dbs/09b81sgm_2_1
```

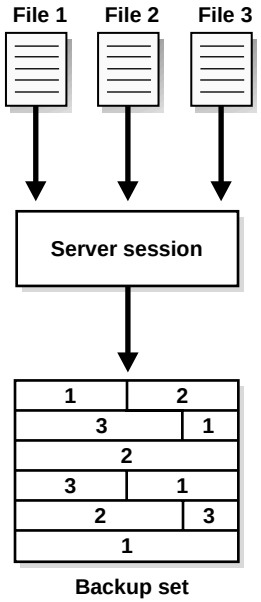
```
299     3   1   AVAILABLE   30-SEP-99             /oracle/dbs/09b81sgm_3_1
300     4   1   AVAILABLE   30-SEP-99             /oracle/dbs/09b81sgm_4_1
301     5   1   AVAILABLE   30-SEP-99             /oracle/dbs/09b81sgm_5_1

List of Datafiles Included
File Name                                LV Type Ckp SCN    Ckp Time
---- ----------------------------------- -- ---- ---------- -------------
1    /oracle/dbs/tbs_01.f            0  Full 35562      30-SEP-99
```

See Also: "set_run_option" on page 10-142 for information on the **kbytes** parameter.

Multiplexed Backup Sets

Oracle takes the datafile blocks included in the same backup set and *multiplexes* them, which means that the data blocks from all of the datafiles in the backup set are interspersed with one another. As Figure 1–11 illustrates, RMAN can back up three datafiles into a backup set that contains only one backup piece; this backup piece is constituted by the intermingled block components of the three input files.

Figure 1–11 Datafile Multiplexing



Use the **filesperset** parameter to control the number of datafiles that Oracle backs up concurrently to a backup set. Controlling concurrency is helpful when you want to keep a tape device streaming without saturating a single datafile with too many read requests, since these requests can subsequently degrade online performance. Limit the read rate by using the **readrate** option of the **set limit channel** command.

When multiplexing files, you can:

- Partition the datafiles into backup sets explicitly, 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 redo logs, but not both together. You cannot write datafiles and archived redo logs to the same

backup set because the Oracle logical block size of the objects in a multiplexed backup must be the same.

See Also:

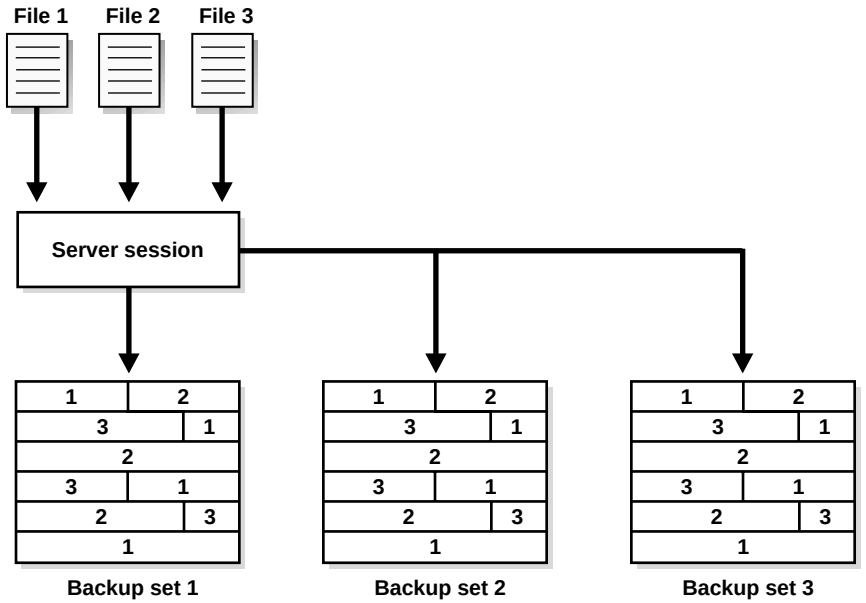
- *Oracle8i Designing and Tuning for Performance* to learn how to tune backup performance.
- "Multiplexing Datafiles in a Backup" on page 5-19 to learn how to multiplex backups.
- "backup" on page 10-22 for reference material on the **filesperset** parameter of the **backup** command.
- "set_run_option" on page 10-142 for reference material on the **set** option of the **run** command.

Duplexed Backup Sets

RMAN provides an efficient way to produce multiple copies of an archived redo log or datafile backup set. Create up to four identical copies of a backup set by issuing the **set duplex** command.

You should not confuse multiplexing, which is combining multiple input files into a single backup set, with duplexing, which is the output of two or more identical backup sets. RMAN allows you to multiplex your input files and duplex the output.

Figure 1–12 Duplexing Multiplexed Backup Sets



See Also: "Duplexing Backup Sets" on page 5-23 to learn how to duplex backups, and "set_run_option" on page 10-142 for reference material on the **set duplex** command.

Parallelization of Backups

If you want to create multiple backup sets and allocate multiple channels, then RMAN automatically parallelizes its operation and writes multiple backup sets in parallel. The allocated server sessions divide up the work of backing up the specified files.

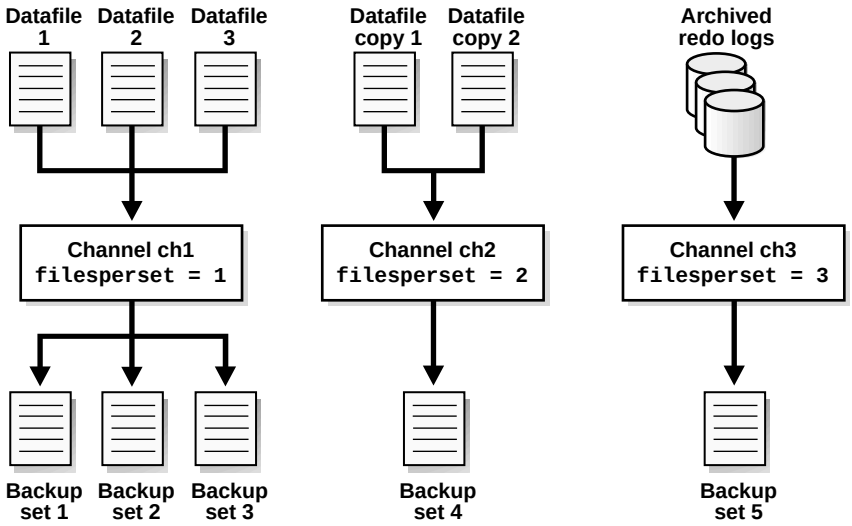
Note: You cannot stripe a single backup set across multiple channels.

RMAN automatically assigns a backup set to a device. You can specify that Oracle should write all backup sets for a *backupSpec* to a specific channel using the **channel** parameter, as in the following example:

```
run {
     allocate channel ch1 type disk;
     allocate channel ch2 type disk;
     allocate channel ch3 type disk;
     backup
      (datafile 1,2,3 filesperset = 1
       channel ch1)
      (datafilecopy '/oracle/copy/cf.f' filesperset = 2
       channel ch2)
      (archivelog from logseq 100 until logseq 102 thread 1 filesperset = 3
       channel ch3);
}
```

Figure 1–13 show an example of parallelization in which channel 1 backs up datafiles, channel 2 backs up datafile copies, and channel 3 backs up archived redo logs.

Figure 1–13 Parallelization of Backups



See Also:

- "Channel Parallelization" on page 1-31 for an overview of how allocated channels affect parallelization
- "Determining How Channels Distribute a Backup Workload" on page 5-24 to learn how to parallelize backups
- "backup" on page 10-22 for reference material on the **channel** parameter of the **backup** command

Backup Errors

RMAN is equipped to 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 abort the jobs. RMAN needs to rewrite the backup sets that it was writing at the time of the error, but it retains any backup sets that it successfully wrote prior to the abort.

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, 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 control file records through the V$BACKUP_CORRUPTION view.

Use the **set maxcorrupt** command to limit the number of previously undetected block corruptions that Oracle allows in a specified datafile or list of datafiles. If a **backup** or **copy** command detects more than this number of corruptions, then the command aborts. The default limit is zero, meaning that RMAN does not tolerate corrupt blocks.

See Also:

- "Integrity Checks" on page 1-66 for more information about fractured and corrupt blocks
- *Oracle8i Reference* for more information on V$BACKUP_CORRUPTION
- "set_run_option" on page 10-142 for reference information on **set maxcorrupt**

Backup Types

Recovery Manager allows you to control the type of backups you produce. RMAN backups can be classified in these ways:

- Full or incremental
- Open or closed
- Consistent or inconsistent

Backup Type	Definition
Full	A backup that is non-incremental, that is, it backs up all used data blocks in the datafiles. **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 an incremental level 0 backup to serve as a basis. Full backups cannot be included in an incremental strategy, although an RMAN copy made with the **level** parameter can be included.

Backup Type	Definition
Open	A backup of any part of the target database when it is open. **Note:** Do not put a tablespace in hot backup mode with the ALTER TABLESPACE BEGIN BACKUP statement. RMAN uses a different method to guarantee consistency in hot 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. **Note:** If you use a recovery catalog, the catalog database must be open.
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 prior to mounting. ■ It was shut down with the ABORT option 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, skipping only datafile blocks that have never been used. The server session 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.

Oracle allows you to create and restore full backups of the following:

- Datafiles

- Datafile copies
- Tablespaces
- Control files (current or backup)
- Database (all datafiles and current control file)

Note that backup sets containing archived redo logs are always full backups.

Incremental Backups

An incremental backup reads the entire file and then backs up only those data blocks that have changed since a previous backup. Oracle allows you to create and restore 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.

The primary reasons for making an incremental backup are:

- To save tape when using a media manager.
- 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.

If none of these criteria apply, then full backups are usually preferable because the application of the incremental backup increases recovery time while the cost savings is negligible.

This section contains the following topics:

- Multi-Level Incremental Backups
- How Incremental Backups Work
- Differential Incremental Backups
- Cumulative Incremental Backups
- Incremental Backup Strategy

Multi-Level Incremental Backups

RMAN allows you to create *multi-level incremental backups*. Each incremental level is denoted by an integer, for example, 0, 1, 2, etc. A level 0 incremental backup, which is the base for subsequent incremental backups, copies all blocks containing data.

When you generate a level *n* incremental backup in which *n* is greater than 0, you back up:

- 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*)

The benefit of performing multi-level incremental backups is that you do not back up all of the blocks all of the time. Since RMAN needs to read all of the blocks of the datafile, full backups and incremental backups taking approximately the same amount of time (assuming that the output of the backup is not a bottleneck).

Incremental backups at levels greater than 0 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.

Note: In most circumstances, cumulative backups are preferable to differential backups.

How Incremental Backups Work

Each data block in a datafile contains an SCN, which is the SCN at which the last 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, so the parent can be a level 0 backup or, depending on the incremental level, a level 1 or level 2 backup. If the SCN in the input data block is greater than the checkpoint SCN of the parent, then RMAN copies the block.

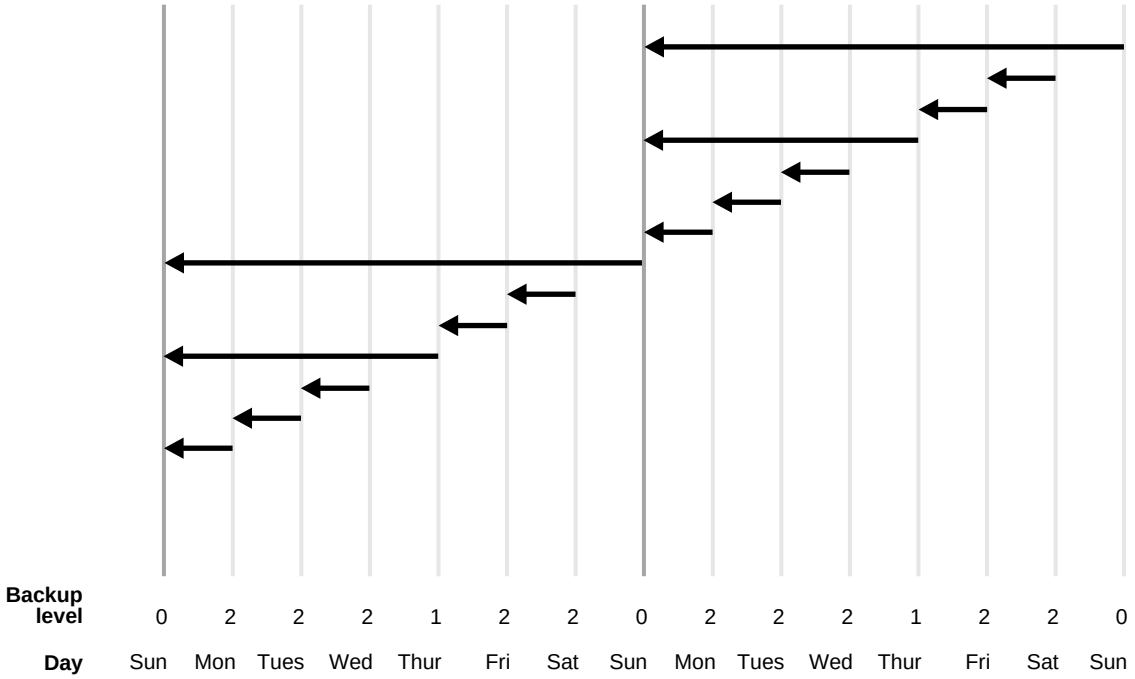
Note that one consequence of this mechanism is that RMAN applies all blocks containing changed data during recovery—even if the change is to an object created with the NOLOGGING option. Hence, making incremental backups functions as a safeguard against the loss of changes made to NOLOGGING tables.

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 since that backup. If no level 1 is available, RMAN copies all blocks changed since the base level 0 backup. Incremental backups are differential by default.

Figure 1–14 Differential Incremental Backups (Default)



In the example above:

- 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

  An 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

  An 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 only the blocks changed since the Sunday level 0 backup will be backed up.

- Friday

  An 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

  An 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.

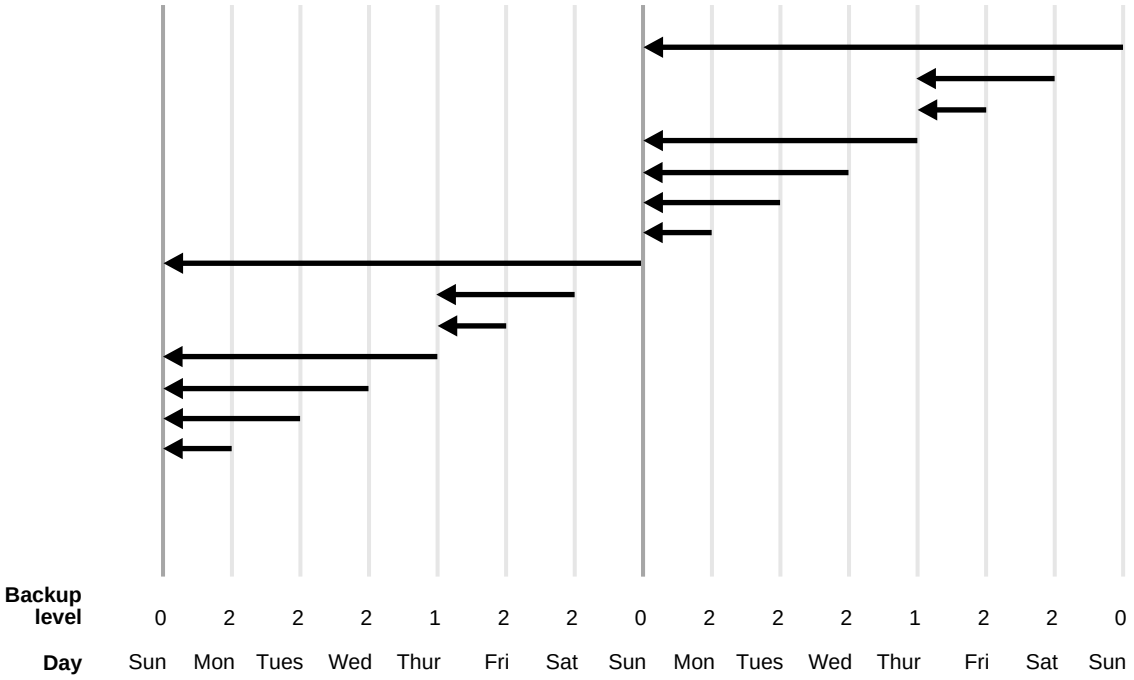
Cumulative Incremental Backups

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 1–15 Cumulative Incremental Backups



In the example above:

- 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 2-1 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 2-1 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 2-1 or less is the level 0 Sunday backup, so all the blocks changed since Sunday will be backed up. (This backup includes those which 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 1-1 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 2-1 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 2-1 or less is the level 1 Thursday backup, so all the blocks changed since Thursday will be backed up.

- The cycle is repeated.

Incremental Backup Strategy

Choose your backup scheme according to an acceptable MTTR (mean time to recover). For example, you can implement a three-level backup scheme so that a 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 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 your differential or cumulative backups to your 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.

Backup Constraints

RMAN performs backup operations only when an instance has the database mounted or open. In an Oracle parallel server environment, if the instance where the backup operation is being performed does not have the database open, then the database must not be open by any instance.

RMAN supports tablespace, datafile, archived redo log, and control file backups. It does not back up:

- Parameter files
- Password files
- Operating system files
- Online redo logs
- Transported tablespaces (before they have been specified read-write)

> **See Also:** *Oracle8i Parallel Server Documentation Set: Oracle8i Parallel Server Concepts; Oracle8i Parallel Server Setup and Configuration Guide; Oracle8i Parallel Server Administration, Deployment, and Performance* for more information about backup constraints in a parallel server environment.

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. Use 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 registering 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.

RMAN Image Copies

Use the RMAN **copy** command to create an image copy. If the original file needs to be replaced, and if the image copy is of a datafile, then you do not need to restore the copy. Instead, Oracle provides a **switch** command to point the control file at the copy and update the recovery catalog 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 media recovery to make the copy current.

RMAN can catalog an image copy and read the metadata. 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.

O/S Image Copies

Oracle supports image copies created by mechanisms other than RMAN, also known as *O/S copies*. For example, a copy of a datafile that you make with the UNIX **cp** command is an O/S copy. You must catalog such O/S copies with RMAN before using them with the **restore** or **switch** commands.

You can create an O/S copy when the database is open or closed. If the database is open and the datafile is not offline normal, then you must place the tablespace in *hot backup mode*, that is, issue the SQL statement ALTER TABLESPACE BEGIN BACKUP before creating the copy.

WARNING: If you do not put a tablespace in hot backup mode before making an online backup, Oracle can generate fractured blocks. See "Detection of Logical Block Corruption" on page 1-67.

Some sites store their datafiles on mirrored disk volumes, which permits the creation of image copies by breaking the mirrors. After you have broken the mirror, you can notify RMAN of the existence of a new O/S 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, if the mirror is resilvered (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:** "How Do You Catalog an Operating System Backup?" on page 3-36 to learn how to catalog copies, and "change" on page 10-38 for reference material on the **change** command.

Tags for Backups and Image Copies

You can assign a user-specified character string called a *tag* to backup sets and image copies (either RMAN-created copies or O/S-created copies). A tag is a 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.

Tags do not need to be unique: multiple backup sets or image copies can have the same tag. When a tag is not unique, then with respect to a given datafile, the tag refers to the most current suitable file. By default, Recovery Manager 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. Thus, 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, RMAN considers *only* backup sets with a matching tag when choosing which particular backup set or image copy to use.

See Also: "switch" on page 10-154 for reference information on the **switch** command, and "restore" on page 10-120 for reference information on the **restore** command.

Restoring Files

Use the RMAN **restore** command to restore datafiles, control files, and archived redo logs from backup sets or image copies on disk. Because a backup set is in an Oracle proprietary format, you cannot simply import it; you must use the RMAN **restore** command to extract it. In contrast, Oracle can use image copies created using RMAN without additional processing.

You can restore:

- Datafiles
- Control files (current or copies)
- Archived redo logs

Note: You do not normally restore archived redo logs because RMAN performs this operation automatically as needed during recovery. You can improve the recovery performance, however, by pre-restoring backups of archived redo logs that you will need during the recovery.

See Also: "Restoring Datafiles, Control Files, and Archived Redo Logs" on page 6-2 to learn how to restore backup sets and copies, and "restore" on page 10-120 for reference material on the **restore** command.

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 using the **set newname** command.

When RMAN performs a restore, the **restore** command creates datafile copies and records them in the repository. If you do not specify a **set newname** command for the restored files, then RMAN restores the backups to the default location and immediately updates the datafile copies created during the restore to status DELETED.

If you do specify **set newname** commands for the restored files, then you can execute a **switch** command so that RMAN considers the restored files as the current database files. If you fail to execute a **switch** command to point the control file to the datafile copies, then the datafile copy records remain in the repository and can confuse RMAN in the future. RMAN considers the un-switched files as valid datafile copies and hence candidates for future restore operations.

File Selection in Restore Operations

RMAN uses the recovery catalog—or target database control file if no recovery catalog is available—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 the *untilClause*.

All specifications of the **restore** command must be satisfied before RMAN restores a backup set or file copy. The **restore** command also considers the device types of the allocated channels when performing automatic selection.

If no available backup or copy in the recovery catalog satisfies all the specified criteria, then RMAN returns an error during the compilation phase of the restore job. If the file cannot be restored because no backup sets or datafile copies reside on media compatible with the device types allocated in the job, then create a new job specifying channels for devices that are compatible with the existing backup sets or datafile copies.

Restore Constraints

Note the following constraints on the **restore** command:

- You must perform restore operations on a started instance; however, the database does not need to be mounted. This functionality allows you to perform restore operations when the control file is lost.
- A tablespace or datafile that is to be restored must be offline, or the database must be closed.

- A restore operation either overwrites the existing datafiles or directs its output to a new file through the **set newname** command. RMAN considers the restored files as datafile copies. If you restore to the default location, records for the restored files appear in the repository with status DELETED to distinguish them from usable image copies.
- You cannot restore a plugged-in tablespace until after it has been specified READ WRITE.
- You cannot use RMAN to restore image copies created on one host to a different host. You must transfer the files manually and use the **catalog** command to catalog them before restoring.

Media Recovery

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 incremental backups automatically. You can only recover current datafiles.

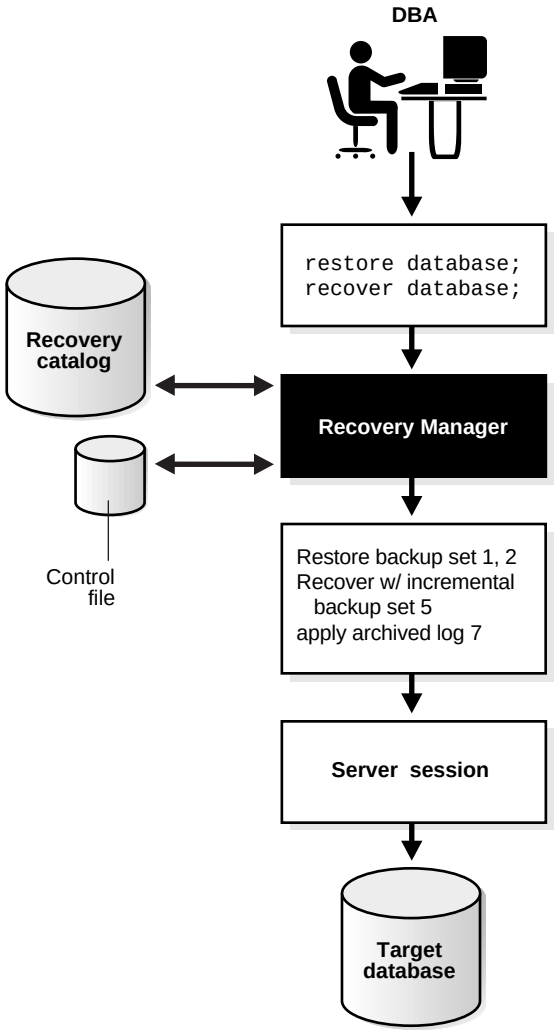
If possible, make the recovery catalog available to perform the media recovery. If it is not available, then RMAN uses information from the target database control file. Note that if control file recovery is required, then you must make the recovery catalog available. RMAN cannot operate when neither the recovery catalog nor the target database control file is available.

The generic steps for media recovery using RMAN are:

- If you want to perform incomplete recovery, use the **set until** command to specify the time, SCN, or log sequence number at which recovery terminates.
- Restore the necessary files using the **restore** command.
- Recover the datafiles using the **recover** command.

> **See Also:** Chapter 6, "Restoring and Recovering with Recovery Manager" for detailed restore and recovery procedures.

Figure 1–16 Performing RMAN Media Recovery



See Also: "Recovering Datafiles" on page 6-18 to learn how to recover datafiles, and "recover" on page 10-96 for reference material on the **recover** command.

Application of Incremental Backups and Redo Records

If RMAN has a choice between applying an incremental backup or applying redo to the restored datafiles, then it always chooses to use the incremental backup. If over-lapping levels of incremental backup are available, then RMAN automatically chooses the one covering the longest period of time.

If RMAN cannot find an incremental backup, it looks for an archived redo log. Whenever ARC*n* archives a redo log, Oracle immediately records it in the control file. Recovery Manager propagates this information into the recovery catalog during resynchronization, classifying archived redo logs as image copies. Use the **list** command to display them.

During recovery, RMAN looks for the appropriate archived redo logs in the default locations specified in the parameter file. If it cannot find them anywhere on disk, it looks in backup sets and restores archived redo logs as needed to perform the media recovery.

By default, RMAN restores the archived redo logs to the current log archive destination specified in the initialization parameter file. Use the **set archivelog destination** command to specify a different restore location.

> **See Also:** "set_run_option" on page 10-142 for **set archivelog destination** syntax.

Incomplete Recovery

RMAN can perform either complete or incomplete recovery. Using the **set until** command, you can specify a time, SCN, or log sequence number as a limit for incomplete recovery. Typically, you use this command before issuing the **restore** and **recover** commands. After performing incomplete recovery, always open the database with the RESETLOGS option and then immediately back up the database.

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 situations:

- 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 8, "Performing Point-in-Time Recovery with Recovery Manager" to learn how to perform TSPITR using RMAN.

Database Duplication

Use the RMAN **duplicate** command to create a test database on which to practice your backup and recovery procedures. The command takes disk backups of your primary database's files and uses them to create a new database. A test database is especially useful if your production database must be up and running 24 hours per day, 7 days a week.

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 redo 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.

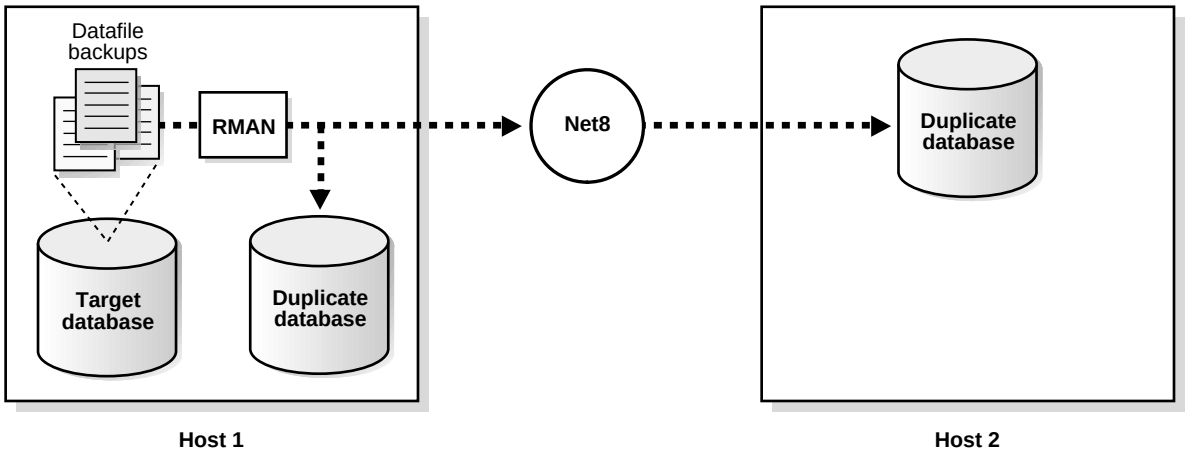
Note also the following features of RMAN duplication. You can:

- Skip read-only tablespaces with the **skip readonly** clause. Read-only tablespaces are included by default. If you omit them, 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 re-use the target datafile filenames for the duplicate datafiles.
- Create your duplicate database by using the **set until** option to recover it to a non-current 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 1–17 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 file system. For example, in some cases you can keep the same directory structure and filenames in your duplicate database, while in other cases you must reset the filenames using **set newname** commands, the DB_FILE_NAME_CONVERT initialization parameter, or both.

> **See Also:** Chapter 7, "Creating a Duplicate Database with Recovery Manager" to learn how to make a duplicate database, and "duplicate" on page 10-76 for **duplicate** command syntax.

Integrity Checks

Oracle prohibits any attempts to perform operations that result in unusable backup files or corrupt restored datafiles. Oracle performs integrity checks to:

- Ensure that restore operations do not corrupt the database by applying backups from a previous incarnation of the database.
- Ensure that incremental backups are applied in the correct order.
- Prohibit accessing datafiles that are in the process of being restored or recovered.
- Allow only one restore operation per datafile at a time.
- Prohibit backups of unrecovered backup files.
- Control information stored in backups to ensure that corrupt backup files are detected.

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 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`.

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 RMAN 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.

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 corruption.

Detection of Logical Block Corruption

RMAN can test 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 logs the block in the `alert.log` and server session trace file.

Provided the sum of physical and logical corruptions detected for a file remain 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, then the command terminates without populating the views.

Note: For **copy** and **backup** commands the **maxcorrupt** setting represents the total number of physical and logical corruptions permitted on a file.

Detection of Fractured Blocks During Open Backups

When performing open backups *without* using Recovery Manager, you must put tablespaces in *hot backup mode* in case the operating system reads a block for a backup that is currently being written by DBW*n*, and is thus inconsistent. Thus, the block is a *fractured block*.

When performing a backup using RMAN, an Oracle server session reads the datafiles, not an operating system utility. The Oracle server session reads whole Oracle blocks and checks to see whether the block is fractured by comparing control information stored in the header and footer of each block. If the session detects a fractured block, then it re-reads the block. For this reason, do not put tablespaces into hot backup mode when using Recovery Manager to back up or copy database files.

See Also: *Oracle8i Backup and Recovery Guide* for information about hot backup mode.

2

Getting Started with Recovery Manager

This chapter describes how to get started using RMAN. It includes the following topics:

- Setting Up Recovery Manager
- Deciding Whether to Use a Recovery Catalog
- Connecting to RMAN
- Using Basic RMAN Commands
- Configuring a Media Manager
- Using Sample Scripts and Scenarios

Setting Up Recovery Manager

Before using RMAN, familiarize yourself with the following topics:

- Using Password Files
- Setting NLS Environment Variables
- Determining the Snapshot Control File Location
- Using RMAN with a Multi-Threaded Server

Using Password Files

Typically, you need to use a password file when connecting to the target database over a non-secure Net8 connection, especially when you:

- Run RMAN remotely on a different machine from the target database.
- Use RMAN with a net service name in the database connect string.
- Run the database in OPS mode and wish to back up this database from more than one node in the cluster concurrently while using only one RMAN session.

Note: Recovery Manager does not back up initialization or password files. When developing your backup and recovery strategy, plan how you will protect these files from media failure (see *Oracle8i Backup and Recovery Guide*).

See Also: *Oracle8i Parallel Server Documentation Set: Oracle8i Parallel Server Concepts; Oracle8i Parallel Server Setup and Configuration Guide; Oracle8i Parallel Server Administration, Deployment, and Performance* for an example of a backup distributed over two nodes in an OPS cluster.

Setting NLS Environment Variables

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 Dates in RMAN Commands

When specifying dates in RMAN commands, the date string can be either:

- A literal string whose format matches the NLS_DATE_FORMAT setting.
- A SQL expression of type DATE, for example, 'SYSDATE-10' or "TO_DATE('01/30/1997', 'MM/DD/YYYY')". Note that the second example includes its own date format mask and so is independent of the current NLS_DATE_FORMAT setting.

Following are examples of typical date settings in RMAN commands:

```
backup archivelog from time 'SYSDATE-31' until time 'SYSDATE-14';
restore database until time "TO_DATE('12/20/98','MM/DD/YY')";
```

Specifying the Database Character Set

If you are going to use RMAN to connect to a non-mounted database and then mount the database later while RMAN is still connected, set the NLS_LANG 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: *Oracle8i Reference* for more information on the NLS_LANG and NLS_DATE_FORMAT parameters, and *Oracle8i National Language Support Guide*.

Determining 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*. The default name for the snapshot control file is port-specific. Use the **set snapshot controlfile name** command to change the name of the snapshot control file; subsequent snapshot control files that RMAN creates use the name specified in the command.

For example, start RMAN and then enter:

```
set snapshot controlfile name to '/oracle/dba/prod/snap_prod.ctl';
```

You can also set the snapshot control file name to a raw device. This operation is important for OPS databases in which more than one instance in the cluster use RMAN because server sessions on each node must be able to create a snapshot control file with the same name and location. For example, enter:

```
set 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:

```
RMAN-08512: 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 starts backing up the control file first waits for service from the media manager.

See Also:

- "Resynchronization of the Recovery Catalog" on page 1-15 for an overview of RMAN resynchronization using the snapshot control file
- "set" on page 10-138 for **set** command syntax
- "Backup Fails Because of Control File Enqueue" on page 9-29 for a scenario involving a backup that fails because of an enqueue

Using RMAN with a Multi-Threaded Server

RMAN cannot connect to the target database through a multi-threaded server (MTS) dispatcher: it requires a dedicated server process. Nevertheless, you can connect specified sessions to dedicated servers, even when your database is configured for MTS.

To ensure that RMAN does not connect to a dispatcher when the target database is configured to use the MTS architecture, the net service name used by RMAN must include `(SERVER=DEDICATED)` in the `CONNECT_DATA` attribute of the connect string.

To use RMAN with an MTS database:

Net8 configuration varies greatly from system to system. The following procedure illustrates only one method.

This scenario assumes that the following net service name in the `tnsnames.ora` file connects to the target database using the MTS architecture, where `inst1` is a value of the SERVICE_NAMES initialization parameter:

```
inst1_mts =
  (description=
    (address=(protocol=tcp)(host=inst1_host)(port1521))
    (connect_data=(service_name=inst1)(server=shared))
  )
```

1. Create a net service name in the `tnsnames.ora` file that connects to the non-shared SID. For example, enter:

   ```
   inst1_ded =
     (description=
       (address=(protocol=tcp)(host=inst1_host)(port1521))
       (connect_data=(service_name=inst1)(server=dedicated))
     )
   ```

2. Connect using SQL*Plus using both the MTS and dedicated 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 an MTS session, you can issue:

   ```
   SQL> connect sys/oracle@inst1_mts
   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) using the dedicated service name. For example, enter:

```
% rman target sys/oracle@inst1_ded catalog rman/rman@rcat
```

> **See Also:** Your operating system-specific Oracle documentation and your *Net8 Administrator's Guide* for a complete description of Net8 connect string syntax.

Deciding Whether to Use a Recovery Catalog

Perhaps the most important decision you make when getting started with RMAN is whether to use a recovery catalog as the RMAN repository. 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, see "Creating the Recovery Catalog" on page 3-2.

> **See Also:** "Recovery Manager Repository" on page 1-13 for an overview of the function of the RMAN repository.

Consequences 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. For this reason, Oracle recommends that you use a recovery catalog with RMAN whenever possible.

When you use a recovery catalog, RMAN requires that you maintain a recovery catalog schema as well as any associated space used by that 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 Recovery Manager scripts stored in the catalog.
- Grows as the numbers of archived logs and backups for each database grow.

If you use a recovery catalog, decide which database you will use to install the recovery catalog schema, and also how you will back up this database. If you use RMAN to back up several databases, you may wish to create a separate recovery catalog database and create the RMAN user in that database. Also, decide whether to operate this database in ARCHIVELOG mode, which is recommended.

If you store the recovery catalog in a separate database, you need a small amount of disk space for the following:

- System tablespace
- Temp tablespace
- Rollback segment tablespaces
- Online redo log files

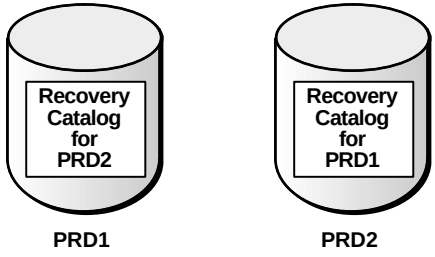
An additional benefit of maintaining a separate recovery catalog is that it is only unavailable at your discretion. Most of the space used in this database is devoted to supporting tablespaces, for example, the system, temp, and rollback tablespaces.

Table 2–1 Typical Recovery Catalog Space Requirements for 1 Year

Type of Space	Space Requirement
System	50 megabytes
Temp	5 megabytes
Rollback	5 megabytes
Recovery catalog	10 megabytes
Online redo logs	1 megabyte each (3 groups, each with 2 members)

If you have more than one database to back up, then you can create more than one recovery catalog and have each database serve as the other's recovery catalog. For example, assume you maintain two production databases, one called PRD1 and a second called PRD2. You can install the recovery catalog for PRD1 in the PRD2 database, and the recovery catalog for the PRD2 database in PRD1.

Figure 2–1 Using Production Databases as Recovery Catalog Databases



By allowing the production databases to serve as each other's recovery catalog, you avoid the extra space requirements and memory overhead of maintaining a

separate recovery catalog database. This solution is not practical, however, if the recovery catalog databases for both reside in tablespaces on the same physical disk.

Note: You *must* install the recovery catalog schema in a different database from the target database. If you do not, the benefits of using a recovery catalog are lost if you lose the database and need to restore.

WARNING: Ensure that the recovery catalog and target databases do *not* reside on the same disk. If they are on the same disks and you lose database, then you will probably lose the other.

See Also: Chapter 3, "Managing the Recovery Manager Repository" to learn how to manage the recovery catalog.

Consequences of Using the Control File as the RMAN Repository

When you use a control file as the RMAN repository, RMAN still functions very effectively. If you choose not to use a recovery catalog, follow the guidelines in "Managing the RMAN Repository Without a Recovery Catalog" on page 3-45. Specifically, make sure you understand which commands require a catalog, and develop a strategy for backing up the repository.

Connecting to RMAN

To use RMAN, you must first connect to it. This connection is necessary to:

- Authenticate you as a valid user.
- Identify the target database, which is the database that you are backing up or restoring.
- Identify the database containing the recovery catalog (if you use a recovery catalog).
- Identify an auxiliary database (if you use an auxiliary database).

Whenever you start RMAN, you must connect to a target database. Note that RMAN connects you to the target database with the SYSDBA privilege: if you do not have this privilege, the connection fails.

You have several options for how you connect. For example, you can start RMAN:

- With or without a recovery catalog.
- By connecting to databases at the operating system command line or the RMAN command line.
- In batch mode (using a command file that contains a series of RMAN commands) or interactive mode.
- With a log file that records RMAN output when you run in batch mode.
- By appending to or overwriting the log file.

This section includes the following sample Recovery Manager connection situations:

- Connecting to RMAN Without a Recovery Catalog
- Connecting to RMAN with a Recovery Catalog
- Connecting to an Auxiliary Database
- Hiding Passwords When Connecting to RMAN
- Disconnecting from RMAN

> **See Also:** "cmdLine" on page 10-42 for an exhaustive list of command-line options.

Connecting to RMAN Without a Recovery Catalog

In these examples, assume that:

sys	User with SYSDBA privileges
target_pwd	The password for connecting as SYSDBA specified in the target database's `orapwd` file
target_str	The net service name for the target database

Connecting to RMAN Using Operating System Authentication

If the target database does not have a password file, then the user must be validated using operating system authentication. You can use operating system authentication only if you connect locally, that is, RMAN and the target database reside on the same machine. You cannot connect to the target database using operating system authentication in conjunction with a net service name.

1. At the UNIX command line, enter:

```
% ORACLE_SID=PROD1; export ORACLE_SID
```

2. Issue the following statement:

```
% rman nocatalog
```

3. At the RMAN prompt enter:

```
RMAN> connect target /
```

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.

Connecting to RMAN Using Password Files

If the target database uses password files, 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 using a net service name.

Connecting from the Operating System Command Line To connect from the operating system command line, enter the following, where *sys_pwd* is the password for SYS and *target_str* is the net service name for the target database:

```
% rman target sys/target_pwd@target_str nocatalog
```

Connecting from the RMAN Prompt Alternatively, start RMAN and connect to your target database from the RMAN prompt:

```
% rman nocatalog
RMAN> connect target sys/target_pwd@target_str
```

See Also:

- "cmdLine" on page 10-42 for information about command line options
- "connect" on page 10-51 for information about the **connect** command
- *Oracle8i Administrator's Guide* to learn about password files

Connecting to RMAN with a Recovery Catalog

In these examples, assume that you maintain a recovery catalog and:

sys	User with SYSDBA privileges

rman	Owner of the recovery catalog having RECOVERY_CATALOG_OWNER privilege
target_pwd	The password for connecting as SYSDBA specified in the target database's `orapwd` file
target_str	The net service name for the target database
cat_pwd	The password for user RMAN specified in the recovery catalog's `orapwd` file
cat_str	The net service name for the recovery catalog database

Connecting to RMAN Using Operating System Authentication

If the target database does not have a password file, then the user must be validated using operating system authentication. You can use operating system authentication only if you connect locally, that is, RMAN and the target database reside on the same machine. You cannot connect to the target database using operating system authentication in conjunction with a net service name.

1. If RMAN is running on the same machine as your target database, set the `ORACLE_SID` to the target database. For example, at the UNIX prompt type:

   ```
   % ORACLE_SID=PROD1; export ORACLE_SID
   ```

2. Issue the following statement to connect to the recovery catalog as user RMAN:

   ```
   % rman catalog rman/cat_pwd@cat_str
   ```

3. Once RMAN has started, issue a **connect target** command (which assumes that you have SYSDBA privileges):

   ```
   RMAN> connect target
   ```

Connecting to RMAN Using Password Files

If the target and recovery catalog databases use 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 through a net service name.

Connecting from the Operating System Command Line To connect to RMAN from the operating system command line, enter the following:

```
% rman target sys/target_pwd@target_str catalog rman/cat_pwd@cat_str
```

Connecting from the RMAN Prompt Alternatively, you can start RMAN and connect to the target database from the RMAN prompt:

```
% rman
RMAN> connect target sys/target_pwd@target_str
RMAN> connect catalog rman/cat_pwd@cat_str
```

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:

sys	User with SYSDBA privileges
rman	Owner of the recovery catalog having RECOVERY_CATALOG_OWNER privilege
target_pwd	The password for connecting as SYSDBA specified in the target database's `orapwd` file
target_str	The net service name for the target database
cat_pwd	The password for user RMAN specified in the recovery catalog's `orapwd` file
cat_str	The net service name for the recovery catalog database
aux_pwd	The password for connecting as SYSDBA specified in the auxiliary database's `orapwd` file.
aux_str	The net service name for the auxiliary database.

If the auxiliary 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 through a net service name.

Connecting from the Operating System Command Line To connect to an auxiliary instance from the operating system command line, enter the following:

```
% rman auxiliary sys/aux_pwd@aux_str
```

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

```
% rman target sys/target_pwd@target_str catalog rman/cat_pwd@cat_str \
> auxiliary sys/aux_pwd@aux_str
```

Connecting 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_pwd@aux_str
```

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

```
% rman
RMAN> connect target sys/target_pwd@target_str
RMAN> connect catalog rman/cat_pwd@cat_str
RMAN> connect auxiliary sys/aux_pwd@aux_str
```

> **See Also:** Chapter 8, "Performing Point-in-Time Recovery with Recovery Manager" to learn how to perform RMAN TSPITR, and "duplicate" on page 10-76 for **duplicate** command syntax.

Hiding Passwords When Connecting to RMAN

If you want to connect to RMAN from the operating system command line and hide authentication information, you can write a connect script and then create execute-only privileges on the file.

For example, if you are running RMAN in an UNIX environment, you can place the following connection information in a text file called `connect_rman.sh`:

```
rman target sys/target_pwd@target_str catalog rman/cat_pwd@cat_str
```

Then, change the permissions on the connect script so that everyone can execute the script but only the desired users have read access:

```
% chmod 711 connect_rman.sh
```

To connect to RMAN, users can execute the script from the operating system command line:

```
% connect_rman.sh
```

Disconnecting from RMAN

To disconnect from RMAN, type **exit** at the RMAN prompt:

```
RMAN> exit
```

Using Basic RMAN Commands

Once you have learned how to connect to RMAN, you can immediately begin performing backup, recovery, and maintenance operations. Use the examples in this section to sample a few basic commands.

These examples assume the following:

- You are running in ARCHIVELOG mode.
- RMAN is running on the same machine as your target database.
- You are connecting from the command line using operating system authentication.
- You are not running an OPS configuration.

You will learn how to perform the following tasks:

- Connecting to RMAN
- Mounting the Database
- Reporting the Current Schema
- Copying a Datafile
- Backing Up a Tablespace
- Listing Backups and Copies
- Validating a Restore

Connecting to RMAN

Your 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 without a recovery catalog.

At the command line, enter the following:

```
% rman target / nocatalog
```

If your database is already mounted or open, you will see output similar to the following:

```
Recovery Manager: Release 8.1.6.0.0

RMAN-06005: connected to target database: RMAN (DBID=1237603294)
RMAN-06009: using target database controlfile instead of recovery catalog
```

The DBID displayed is the database identifier for your database.

If your database is not started, RMAN displays the following message when it connects:

```
RMAN-06193: connected to target database (not started)
```

> **See Also:** "cmdLine" on page 10-42 for command line connection options.

Mounting the Database

Because you are not using a recovery catalog, RMAN must obtain the information it needs from the target database control file. Consequently, the database must be mounted or open. For these examples, we will mount the database but not open it.

If the database is not started, issue the **startup** command, specifying a parameter file if the file is in a non-default location. This example uses the parameter file `initPROD1.ora`:

```
RMAN> startup mount pfile=/oracle/dbs/temp/initPROD1.ora

RMAN-06196: Oracle instance started
RMAN-06199: database mounted

Total System Global Area      19799144 bytes

Fixed Size                       64616 bytes
Variable Size                 11001856 bytes
Database Buffers               8192000 bytes
Redo Buffers                    540672 bytes

RMAN>
```

RMAN displays the size of the SGA, including the size of the database and redo buffers, and returns you to the RMAN prompt.

If the database is open, issue the following to close it cleanly and then mount it:

```
RMAN> shutdown immediate

RMAN-06405: database closed
RMAN-06404: database dismounted
RMAN-06402: Oracle instance shut down

RMAN> startup mount pfile = initPROD1.ora  # specify a parameter file if necessary
```

> **See Also:** "startup" on page 10-152 for **startup** syntax.

Reporting the Current Schema

In this example, ask RMAN to report which datafiles your target database contains. Use the **report schema** command as follows:

```
RMAN> report schema;
```

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

```
RMAN-03022: compiling command: report
Report of database schema
File K-bytes    Tablespace           RB segs Name
---- ---------- -------------------- ------- ------------------
1         47104 SYSTEM               ***     /oracle/dbs/tbs_01.f
2           978 SYSTEM               ***     /oracle/dbs/tbs_02.f
3           978 TBS_1                ***     /oracle/dbs/tbs_11.f
4           978 TBS_1                ***     /oracle/dbs/tbs_12.f
5           978 TBS_2                ***     /oracle/dbs/tbs_21.f
6           978 TBS_2                ***     /oracle/dbs/tbs_22.f
7           500 TBS_1                ***     /oracle/dbs/tbs_13.f
8           500 TBS_2                ***     /oracle/dbs/tbs_23.f
9           500 TBS_2                ***     /oracle/dbs/tbs_24.f
10         5120 SYSTEM               ***     /oracle/dbs/tbs_03.f
11         2048 TBS_1                ***     /oracle/dbs/tbs_14.f
12         2048 TBS_2                ***     /oracle/dbs/tbs_25.f
```

See Also: Chapter 4, "Generating Lists and Reports with Recovery Manager" to learn how to make lists and reports, and "report" on page 10-110 for **report** syntax.

Copying a Datafile

In this example, copy `datafile 1` to a new location. Of course, if you do not want to copy this particular datafile, copy any datafile you choose. This example allocates the disk channel `c1` and creates a datafile copy named `df1.bak`.

Use the **copy** command as follows:

```
RMAN> run {
2> allocate channel c1 type disk;
3> copy datafile 1 to 'df1.bak';
4> }
```

You see output similar to the following:

```
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: c1
```

```
RMAN-08500: channel c1: sid=12 devtype=DISK

RMAN-03022: compiling command: copy
RMAN-03023: executing command: copy
RMAN-08000: channel c1: copied datafile 1
RMAN-08501: output filename=/oracle/dbs/df1.bak recid=3 stamp=352381826
RMAN-08031: released channel: c1
```

The `RMAN-08000` message informs you that the copy was successful. Note that RMAN displays the full filename of the output copy in message `RMAN-08501`.

See Also:

- "copy" on page 10-55 for **copy** syntax
- "allocate" on page 10-10 for **allocate** syntax
- "Making Image Copies" on page 5-13 to learn how to make image copies

Backing Up a Tablespace

In this example, back up your SYSTEM tablespace to disk. Of course, you can choose to back up a different object.

This example backs up the tablespace to its default backup location, which is port-specific: on UNIX systems the location is `$ORACLE_HOME/dbs`. Because you do not specify the **format** parameter, RMAN automatically assigns the backup a unique filename.

Use the **backup** command as follows:

```
RMAN> run {
2> allocate channel c1 type disk;
3> backup tablespace system;
4> }
```

You see output similar to the following:

```
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: c1
RMAN-08500: channel c1: sid=12 devtype=DISK

RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel c1: starting full datafile backupset
RMAN-08502: set_count=1 set_stamp=352382211 creation_time=18-DEC-98
RMAN-08010: channel c1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00001 name=/vobs/oracle/dbs/tbs_01.f
```

```
RMAN-08011: including current controlfile in backupset
RMAN-08522: input datafile fno=00016 name=/oracle/dbs/tbs_03.f
RMAN-08522: input datafile fno=00002 name=/oracle/dbs/tbs_02.f
RMAN-08013: channel c1: piece 1 created
RMAN-08503: piece handle=/oracle/dbs/1hag1r83_1_1 comment=NONE
RMAN-08525: backup set complete, elapsed time: 00:00:27
RMAN-08031: released channel: c1
```

The RMAN-08525 message informs you that RMAN created the backup set successfully. Note that RMAN displays the full filename for the backup piece in message RMAN-08503.

> **See Also:** "Making Backups" on page 5-2 to learn how to make image copies, and "backup" on page 10-22 for **backup** syntax.

Listing Backups and Copies

In this example, list your backup sets and image copies. Issue the **list** command as follows:

```
RMAN> list backup;
```

You see output similar to the following:

```
List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
3       3          352382231  0  352382211  49         18-DEC-98

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    2       1   1   AVAILABLE   18-DEC-98              /oracle/dbs/1hag1r83_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    1    /oracle/dbs/tbs_01.f             0  Full 114149     18-DEC-98
    2    /oracle/dbs/tbs_02.f             0  Full 114149     18-DEC-98
    16   /oracle/dbs/tbs_03.f             0  Full 114149     18-DEC-98
```

RMAN tells you which backup sets and pieces it created as well as which datafiles it included in those sets.

Now list your image copies as follows:

```
RMAN> list copy;
```

You see output similar to the following:

```
List of Datafile Copies
Key     File S Completion time Ckp SCN    Ckp time        Name
------- ---- - -------------- ---------- --------------- ------
2       1    A 18-DEC-98       114148     18-DEC-98       /oracle/dbs/df1.bak
```

> **See Also:** "list" on page 10-83 for an explanation of the column headings in the **list** output, and "Generating Lists" on page 4-2 to learn how to make lists and reports.

Validating a Restore

Finally, check that you are able to restore your backup in case of a media failure. Use the output from the **list backup** command to determine the primary key for the backup set:

```
List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
3       3          352382231  0  352382211  49         18-DEC-98
```

In this example, the primary key is 3. Use the primary key value in your backup set in the **validate backupset** command as follows:

```
RMAN> run {
2> allocate channel c1 type disk;
3> validate backupset 3;
4> }
```

You should see output similar to the following:

```
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: c1
RMAN-08500: channel c1: sid=12 devtype=DISK

RMAN-03022: compiling command: validate
RMAN-03023: executing command: validate
RMAN-08096: channel c1: starting validation of datafile backupset
RMAN-08502: set_count=49 set_stamp=352382211 creation_time=18-DEC-98
RMAN-08023: channel c1: restored backup piece 1
RMAN-08511: piece handle=/oracle/dbs/1hag1r83_1_1 params=NULL
RMAN-08098: channel c1: validation complete
RMAN-08031: released channel: c1
```

If there are no error messages, then RMAN confirms that it is able to restore the backup set. When there is an error, RMAN always displays an error banner and provides messages indicating the nature of the error.

For example, if you try to allocate a channel on the target database when not connected to it you see:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure during compilation of command
RMAN-03013: command type: allocate
RMAN-06171: not connected to target database
```

> **See Also:** "Restoring Datafiles, Control Files, and Archived Redo Logs" on page 6-2 to learn how to restore backups and copies, and "validate" on page 10-160 for **validate** syntax.

Configuring a Media Manager

To back up to and restore from sequential media such as tape you must integrate a media manager with Oracle. This section includes the following topics:

- Linking with a Media Manager
- Generating Unique Filenames
- Limiting File Size
- Sending Device-Specific Strings to the Media Manager

> **See Also:** "Media Management" on page 1-19 for an overview of media management software.

Linking with a Media Manager

To integrate Oracle with a media manager, you must:

- Install and configure media manager software and hardware.
- Obtain your vendor's media management library (MML) interface software, which you then link with the Oracle Server. This integration enables Oracle server sessions to call the media manager.

Note: For instructions on how to achieve this integration on your platform, see your operating system-specific Oracle documentation and the documentation supplied by your media manager.

See Also: "After Linking to the Media Manager on UNIX, RMAN Fails to Back Up to Tape" on page 9-19 and "After Installing the Media Manager on NT, RMAN Fails to Back Up to Tape" on page 9-22 for troubleshooting scenarios involving media manager link problems.

Generating Unique Filenames

Use the substitution variables provided by RMAN to generate unique backup piece names when writing backups to a media manager (see "backup" on page 10-22 for the complete list of variables). A backup piece name is determined by the format string specified either in the **backup** command or in the **allocate channel** command.

If you do not specify the **format** parameter, RMAN automatically generates a unique filename using the **%U** substitution variable. The media manager considers the backup piece name as the filename backed up, so this name must be unique in the media manager catalog.

Note: Some media managers only support a 14-character backup piece name. See your media management documentation to determine the limit for your media manager.

Limiting File Size

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 out by RMAN do not exceed these limits. To limit backup piece file sizes, use the parameter **kbytes** in a **set limit channel** command. See the scripts in your $ORACLE_HOME/rdbms/demo directory for an example.

> **See Also:** "set_run_option" on page 10-142 to learn about various parameters affecting the **run** command.

Sending Device-Specific Strings to the Media Manager

Use the **send** command to send a vendor-specific quoted string to the media management software. See your media management documentation to determine which commands it supports.You can use:

- The **send** command with no other operands to send the string to all allocated channels.
- The **send device type** command to send the string to all channels of the specified device type.
- The **send channel** command to send the string to channels specified in the command.

> **See Also:** "send" on page 10-136 for **send** command syntax.

Troubleshooting the Media Manager

To aid in troubleshooting media management, Oracle offers a client program, sbttest. This program, which is linked to RMAN to perform backups to tape, provides a stand-alone test of the media management software. Use it when Oracle is unable to create or restore backups using either the bundled Legato Storage Manager or another vendor's media management product. Only use the sbttest program at the direction of Oracle support.

> **See Also:** Chapter 9, "Recovery Manager Troubleshooting" for more information about troubleshooting RMAN.

Using Sample Scripts and Scenarios

The `$ORACLE_HOME/rdbms/demo` sub-directory (the location may differ depending on your operating system) contains a number of sample RMAN scripts. These files are executable Recovery Manager command files that are fully documented so that you can understand the features used. Edit them to customize them for your site.

The first file provides a number of scripts that back up, restore, and recover a database. These scripts are typical of how some administrators back up their databases; use them as a starting point for developing backup, restore, and recovery

scripts. The remaining files contain either a backup, recovery, or duplication scenario.

Run command files from either the operating system command line or the RMAN prompt. If you use a recovery catalog, you can also create scripts using the **create script** command and execute them within a **run** command.

See Also:

- "rmanCommand" on page 10-130 to learn how to run command files from the RMAN prompt
- "cmdLine" on page 10-42 to learn how to run command files from the command line
- "Storing Scripts in the Recovery Catalog" on page 3-27 to learn how to create and execute stored scripts

3

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. The chapter includes the following topics:

- Creating the Recovery Catalog
- Setting Recovery Catalog Compatibility
- Maintaining the RMAN Repository
- Storing Scripts in the Recovery Catalog
- Backing Up and Recovering the Recovery Catalog
- Upgrading the Recovery Catalog
- Dropping the Recovery Catalog
- Managing the RMAN Repository Without a Recovery Catalog

Creating the Recovery Catalog

To use a recovery catalog, you need to set up the schema. Oracle suggests you put the recovery catalog schema in its own tablespace; however, you can put it in the SYSTEM tablespace if necessary. Note that SYS cannot be the owner of the recovery catalog.

Install the recovery catalog schema in a different database from the target database you will be backing up. If you do not, the benefits of using a recovery catalog are lost if you lose the database and need to restore.

WARNING: 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.

Assume the following for the examples below:

- User SYS with password CHANGE_ON_INSTALL has SYSDBA privileges on the recovery catalog database RCAT.
- There is a tablespace called CATTBS on the recovery catalog database RCAT that stores the recovery catalog. Note that if you want to use a reserved word as a tablespace name, you must enclose it in quotes and put it in uppercase font (see "Reserved Words" on page 10-4 for a list of keywords).
- There is a tablespace called TEMP 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.

> **See Also:** "Connecting to RMAN" on page 2-8 to learn how to connect to RMAN, and "Deciding Whether to Use a Recovery Catalog" on page 2-6 to learn about the advantages and disadvantages of maintaining a recovery catalog.

To set up the recovery catalog schema:

1. Start SQL*Plus and then connect with administrator privileges to the database containing the recovery catalog. For example, enter:

   ```
   SQL> CONNECT sys/change_on_install@rcat
   ```

2. Create a log file that you can use to check for errors. For example, enter:

   ```
   SQL> SPOOL create_rman.log
   ```

3. Create a user and schema for the recovery catalog. For example, enter:

```
SQL> CREATE USER rman IDENTIFIED BY rman
2> TEMPORARY TABLESPACE temp
3> DEFAULT TABLESPACE cattbs
4> QUOTA UNLIMITED ON cattbs;
```

4. 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;
```

5. Grant other desired privileges to the RMAN user.

```
SQL> GRANT connect, resource TO rman;
```

6. Host out to the operating system to check the `create_rman.log` file for any errors before continuing. For example, a UNIX user can issue:

```
SQL> host
% vi create_rman.log
```

To create the recovery catalog:

1. Connect to the database that will contain the catalog as the catalog owner. For example, from the operating system command line enter:

```
% rman catalog rman/rman@rcat
```

   You can also connect from the RMAN prompt:

```
% rman
RMAN> connect catalog rman/rman@rcat
```

2. Issue the **create catalog** command to create the catalog, specifying the CATTBS tablespace:

```
RMAN> create catalog tablespace cattbs;
```

   Note that the creation of the catalog can take several minutes.

3. Host out to the operating system to check the `create_rman.log` file for any errors before continuing. For example, a UNIX user can issue:

```
RMAN> host;
% vi create_rman.log
```

4. Optionally, query the recovery catalog to see which tables were created:

```
SQL> SELECT table_name FROM user_tables;
```

> **See Also:** *Oracle8i SQL Reference* for the SQL syntax for the GRANT and CREATE USER statements, and "createCatalog" on page 10-59 for **create catalog** command syntax.

Setting Recovery Catalog Compatibility

For RMAN to function effectively, the RMAN executable and recovery catalog versions must be compatible. Compatibility between RMAN and the recovery catalog is determined by two factors:

- The interface versions supported by a recovery catalog PL/SQL package release
- The **compatible** parameter of the **configure** command

The **compatible** parameter specifies the minimum acceptable release of the RMAN executable that can function with the catalog. For example, if the recovery catalog compatibility is set to 8.1.4, then only an RMAN executable of release 8.1.4 or later can connect to the catalog.

You cannot set the **compatible** parameter to a version that is not supported by the currently-installed DBMS_RCVMAN or DBMS_RCVCAT packages. If you try to set compatibility to a version that is not supported by one of the packages, then you receive an error message that indicates the lowest and highest value you can set, as in the following example:

```
RMAN> configure compatible = 8.0.1;

RMAN-03022: compiling command: configure
RMAN-03026: error recovery releasing channel resources
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure during compilation of command
RMAN-03013: command type: configure
RMAN-06455: illegal compatible value 8.0.1:  must be between 08.00.04 and 08.01.06
```

The compatibility level of the catalog affects the way that repository records are updated and deleted, as explained in the following table:

Table 3–1 Effect of Compatibility on Repository Record Removal

If compatibility is	Then change ... delete and backup ... delete input	And change ... uncatalog and prgrmanc.sql
8.1.6 or higher	Delete backup sets, image copies, and archived logs and remove their records from the recovery catalog	Can remove archived log records from the repository without creating problems when you attempt to restore archived logs whose records have been removed
8.1.5 or lower	■ Delete backup sets and image copies and remove the records from the catalog ■ Delete archived logs but change catalog records to status DELETED (and not remove them)	Can remove archived log records from the repository while sometimes creating problems when you attempt to restore logs whose records have been removed

The following table illustrates possible catalog creation and upgrade scenarios for a release 8.1.6 RMAN executable:

If you use an 8.1.6 RMAN executable to	Then the recovery catalog	And catalog compatibility is automatically set to
Execute the **create catalog** command	Is created as a release 8.1.6 recovery catalog	8.1.6. **Note:** You cannot use the 8.1.6 catalog with a pre-8.1.6 release of the RMAN executable.
Execute the **upgrade catalog** command	Is upgraded from a pre-8.1.6 release to a release 8.1.6 catalog	8.0.4 . **Note:** The 8.1.6 catalog is backwards compatible with older releases of the RMAN executable.

You can manually increase (but not decrease) the compatibility level of the catalog. For example, if you upgrade the catalog so that compatibility is automatically set to 8.0.4, then you can manually increase compatibility to 8.1.5.

The following table illustrates scenarios involving manual increase of compatibility levels in a release 8.1.6 recovery catalog:

If you set catalog compatibility to	Then you cannot use an RMAN executable of release
8.1.6	8.1.5 or lower
8.1.5	8.1.4 or lower
8.0.6	8.0.5 or lower

To view the compatibility level of the recovery catalog:

1. Start a SQL*Plus session and connect to the recovery catalog database as the recovery catalog owner. For example, enter:

   ```
   SQL> connect rman/rman@rcat
   ```

2. Issue the following query:

   ```
   SQL> SELECT value FROM config WHERE name='compatible';

   VALUE
   ------------------------------------------------------------------------------
   080106
   ```

To create an 8.1.6 recovery catalog that is usable with a pre-8.1.6 RMAN executable:

This procedure assumes that you have the use of both an 8.1.6 and pre-8.1.6 RMAN executable. The method employed it to use a pre-8.1.6 RMAN executable to create the recovery catalog, and a release 8.1.6 RMAN executable to upgrade it. The compatibility level is automatically set to 8.0.4, which has the implications described in Table 3–1.

1. Use a pre-8.1.6 RMAN executable to connect to the catalog database. For example, enter:

   ```
   % rman catalog rman/rman@rcat

   Recovery Manager: Release 8.1.5.0.0

   RMAN-06008: connected to recovery catalog database
   ```

2. Create the recovery catalog and then exit. For example, enter:

   ```
   RMAN> create catalog;
   RMAN> exit
   ```

> **Note:** If you use a release 8.0 RMAN executable to create the catalog, you must use the `catrman.sql` script to create the recovery catalog.

3. Use the 8.1.6 RMAN executable to connect to the newly created recovery catalog:

```
% rman catalog rman/rman@rcat

Recovery Manager: Release 8.1.6.0.0

RMAN-06008: connected to recovery catalog database
```

4. Upgrade the recovery catalog to release 8.1.6:

```
RMAN> upgrade catalog;
RMAN> exit
```

The compatibility level of the recovery catalog is now automatically set to 8.0.4.

To increase the compatibility level of the recovery catalog:

This procedure assumes that you have an 8.1.6 recovery catalog that can function with a pre-8.1.6 RMAN executable. This procedure increases the compatibility level of the recovery catalog:

1. To determine the current compatibility setting of the recovery catalog, use SQL*Plus connect to the recovery catalog database as the recovery catalog owner, and then issue the following SELECT statement:

```
SQL> connect rman/rman@rcat
SQL> SELECT value FROM config WHERE name='compatible';
VALUE
--------------------------------------------------------------------------------
080004
```

2. Use RMAN to connect to the target database and the recovery catalog:

```
% rman target / catalog rman/rman@rcat

Recovery Manager: Release 8.1.6.0.0

RMAN-06005: connected to target database: RMAN (DBID=1237603294)
RMAN-06008: connected to recovery catalog database
```

3. Raise the compatibility of the catalog to the desired level. For example, enter:

```
RMAN> configure compatible = 8.1.5;
```

4. View the compatibility information to ensure that the level is correct:

```
SQL> connect rman/rman@rcat
SQL> SELECT value FROM config WHERE name='compatible';
VALUE
--------------------------------------------------------------------------------
080105
```

The recovery catalog will not work with an RMAN executable from a release below the value specified with the **configure compatible** command.

Maintaining the RMAN Repository

This section describes how to manage the RMAN information repository. It assumes that you are using a recovery catalog. If you use a control file as the exclusive repository for RMAN metadata, then most RMAN maintenance commands continue to work.

> **See Also:** "Understanding Catalog-Only Command Restrictions" on page 3-45 for a list of commands that are unavailable when you use the control file as the sole repository.

This section includes the following topics:

- Registering a Database with the Recovery Catalog
- Unregistering a Database from the Recovery Catalog
- Resetting the Recovery Catalog
- Changing the Availability of a Backup or File Copy
- Crosschecking the RMAN Repository
- Deleting Backups and Copies and Updating Their Status in the RMAN Repository
- Validating the Restore of Backups and Copies
- Resynchronizing the Recovery Catalog
- Managing Records in the Control File
- Cataloging Operating System Backups

Registering a Database with the Recovery Catalog

Before using RMAN with a target database, register the target database in the recovery catalog. If you do not, RMAN cannot use the recovery catalog to store information about the target database. RMAN obtains all information it needs to register the target database from the database itself. You can register more than one target database in the same recovery catalog, but you can register a database only once in the same catalog.

To register the target database:

1. Connect to the target database and recovery catalog. For example, issue the following to connect to the catalog database RCAT as RMAN:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. If the database is not mounted, then mount or open it. For example, issue:

   ```
   startup mount;
   ```

3. If you are creating a new database or migrating an existing Oracle7 database, or if RMAN is installed for use with an existing version 8.0 or higher database, issue the following command:

   ```
   register database;
   ```

4. If there are any existing user-created backups on disk that were created under version 8.0 or higher, add them to the recovery catalog using the **catalog** command. For example, catalog `'/os_backup/df1.b'` by issuing the following command:

   ```
   catalog datafilecopy '/os_backup/df1.b';
   ```

   For an Oracle7 backup to be usable for recovery in database of a later version, it must have been part of a tablespace that was offline normal or read-only when the database was migrated. See "Cataloging Operating System Backups" on page 3-34.

   > **Note:** Backups made with the Enterprise Backup Manager (EBU) cannot be used or cataloged by RMAN.

5. RMAN automatically obtains information about the original archived redo logs from the target database control file. If you have made additional operating system backups of your archived logs, or if log records have aged out of the

target control file, catalog them. For example, you can catalog logs as follows (where `log1` etc. refers to the fully specified filename for an archived log):

```
catalog archivelog 'log1', 'log2', 'log3', ... 'logN';
```

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: "register" on page 10-101 for **register** command syntax, and *Oracle8i Migration* for issues relating to database migration.

Troubleshooting Database Identifier Problems

Oracle uses an internal, uniquely generated number called the *db identifier* to distinguish one database from another. Oracle generates this number when you create the database.

Typically, each database has a unique identifier; however, an exception occurs with databases that you create by copying files from an existing database instead of using a CREATE DATABASE statement or **duplicate** command. In such cases, RMAN detects the duplicate database identifiers and the **register database** command fails. Avoid this problem by using the **duplicate** command, which copies the database from backups and generates a new database identifier.

If a failure occurs because of duplicate database identifiers, you can create a second recovery catalog in another user's schema by re-executing the **create catalog** command using a different Oracle userid. Then, you can register the database with a duplicate database identifier into the newly created recovery catalog in the new schema.

Note: If you are using RMAN with different target databases that have the same database name and identifier, be extremely careful to always specify the correct recovery catalog schema when invoking Recovery Manager.

See Also:

- "catalog" on page 10-35 for **catalog** command syntax
- "duplicate" on page 10-76 for **duplicate** command syntax
- *Oracle8i Migration* or issues relating to database migration

Unregistering a Database from the Recovery Catalog

RMAN allows you to unregister a database as well as register it. Make sure this procedure is what you intend, because afterwards RMAN can longer recover all the backups for the unregistered database.

To unregister a database:

1. Start RMAN and connect to your target database. Note down the `DBID` value that is displayed when you use RMAN to connect to your target database. For example, enter:

```
% rman target sys/change_on_install@prod1 nocatalog

RMAN-06005: connected to target database: RMAN (DBID=1231209694)
```

2. List the copies and backup sets recorded in the control file (see "Generating Lists" on page 4-2). For example, enter:

```
RMAN> list backup of database;
RMAN-03022: compiling command: list

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
989     1          368895909  0  368895908  1          23-JUN-99

    List of Backup Pieces
    Key     Pc# Cp# Status      Completion Time        Piece Name
    ------- --- --- ----------- ---------------------- ------------------------
    990     1   1   AVAILABLE   23-JUN-99              /vobs/oracle/dbs/01avppt4_1_1

    List of Datafiles Included
    File Name                                LV Type Ckp SCN    Ckp Time
    ---- ----------------------------------- -- ---- ---------- -------------
    2    /vobs/oracle/dbs/tbs_02.f           0  Full 34968      23-JUN-99
```

3. Issue **change ... delete** statements to delete all backups from the operating system (see "Deleting Backups and Copies and Updating Their Status in the RMAN Repository" on page 3-18). For example, enter:

```
allocate channel for maintenance type disk;
change backupset 989 delete;

RMAN-03022: compiling command: change
RMAN-08073: deleted backup piece
RMAN-08517: backup piece handle=/vobs/oracle/dbs/01avppt4_1_1 recid=1 stamp=368895908
RMAN-03023: executing command: partial resync
```

```
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
```

4. Execute another list command to confirm that RMAN removed all backups:

```
list backup;
RMAN-03022: compiling command: list
```

5. Use SQL*Plus to connect to your recovery catalog database and 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:

```
SQL> SELECT db_key, db_id FROM db WHERE db_id = 1231209694;
```

This query should return exactly one row.

```
DB_KEY     DB_ID
---------- ----------
         1 1237603294
1 row selected.
```

6. 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 5:

```
SQL> EXECUTE dbms_rcvcat.unregisterdatabase(db_key, db_id)
```

For example, enter:

```
SQL> EXECUTE dbms_rcvcat.unregisterdatabase(1 , 1237603294)
```

Note: The DBMS_RCVCAT.UNREGISTERDATABASE package works on Oracle release 8.0 and higher.

Resetting the Recovery Catalog

Before you can use RMAN again with a target database that you have opened with the RESETLOGS option, notify RMAN that you have reset the database incarnation. The **reset database** command directs RMAN to create a new database incarnation record in the recovery catalog. This new incarnation record indicates 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 ALTER DATABASE OPEN RESETLOGS statement but do not reset the database, then RMAN cannot access the recovery catalog because it cannot distinguish between a RESETLOGS command and an accidental restore of an old

control file. By resetting the database, you inform RMAN that the database has been opened with the RESETLOGS option.

In the rare situation in which you wish to undo the effects of opening with the RESETLOGS option by restoring 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.

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         PROD1     1224038686  NO     1            02-JUL-98
1       582       PROD1     1224038686  YES    59727        10-JUL-98
```

2. Reset the database to the old incarnation. For example, enter:

```
reset database to incarnation 2;
```

3. After resetting the database, issue **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:

```
run {
     allocate channel ch1 type disk;
     restore database;
     recover database;
     alter database open resetlogs;
}
```

See Also: "reset" on page 10-118 for **reset database** command syntax, "list" on page 10-83 for **list** command syntax.

Changing the Availability of a Backup or File Copy

The **unavailable** option provides for cases when a backup or copy cannot be found or has migrated offsite. A file that is marked UNAVAILABLE is not used in a **restore** or **recover** command. If the file is later found or returns to the main site, then you can mark it available again by using the **available** operand. Note that you do not need to allocate a channel of type **maintenance** for this operation.

Note: You must use a recovery catalog when executing **change ... unavailable** and **change ... available**.

To mark a backup piece or copy as available or unavailable:

1. Issue a **change ... unavailable** command to mark a backup or copy as UNAVAILABLE. For example, enter:

```
change datafilecopy '/oracle/backup/cf_c.f' unavailable;
change backupset 12 unavailable;
```

2. If a previously unavailable file is reinstated, issue **change ... available** to change its status back to AVAILABLE.

```
change datafilecopy '/oracle/backup/cf_c.f' available;
change backupset 12 available;
```

See Also: "change" on page 10-38 for **change** command syntax.

Crosschecking the RMAN Repository

Because backups and copies can disappear from disk or tape or become corrupted, the RMAN metadata repository can contain outdated information. 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*.

Use either the **change ... crosscheck** or **crosscheck backup** command to check the specified files. Note that these commands do *not* delete operating system files or remove repository records; you must use separate commands for those operations.

The following table explains the difference between the crosscheck commands:

Command	Catalog Needed?	Purpose
change ... crosscheck	Only for **backupset** and **backuppiece** options	To determine whether the backups or copies exist. If RMAN cannot find backup pieces, it marks them as EXPIRED. It marks all other types of absent files—image copies and archived redo logs—as DELETED. If the files are on disk, RMAN queries the file headers. For other device types, RMAN queries the media manager to see if the file exists in the media catalog.

Command	Catalog Needed?	Purpose
crosscheck backup	No	To determine whether backups stored on disk or tape exist. Backups are either backup sets or media-managed proxy copies. This command checks only backup sets marked AVAILABLE or EXPIRED, either by examining the backup pieces for **type disk** or by querying the media manager for **type** *'sbt_tape'*. It only processes backups created on the specified channel. RMAN does not delete backup pieces that it cannot find, but marks them as EXPIRED.

See Also: "Deleting Backups and Copies and Updating Their Status in the RMAN Repository" on page 3-18 to learn how to delete files and update repository records, and "crosscheck" on page 10-64 for **crosscheck** command syntax.

Crosschecking Backups

Use the crosscheck feature to check the status of a backup on disk or tape. If the backup is on disk, the **change backupset ... crosscheck** and **crosscheck backup** commands determine whether the header of the backup piece is valid; if on tape, the commands simply check that the backups exist.

If a backup piece is unreadable or absent, then RMAN marks the backup piece EXPIRED in the output of the **list** command and the recovery catalog views. If it was marked EXPIRED but is now available, RMAN marks the backup piece as AVAILABLE in the output of the **list** command and the recovery catalog views.

Use **change backupset ... crosscheck** when you want to provide a list of backup sets or pieces to check; use **crosscheck backup** when you wish to restrict the crosscheck to a specified device type, object type, or date range and let RMAN generate the list of backup sets or pieces.

Note: The **change** command operates only on files that are recorded in the recovery catalog or the control file. The same is true for other commands except **catalog** and **resync from controlfilecopy**.

To provide a list of backups for a crosscheck:

1. Allocate a channel of type **maintenance**:

   ```
   allocate channel for maintenance type 'sbt_tape';
   ```

2. Identify the desired backup piece, backup set, or proxy copy that you want to check by issuing a **list** command:

   ```
   list backup;
   ```

3. Check whether the specified backup sets exist. This example checks whether backup sets with the primary keys of 1338, 1339, and 1340 still exist:

   ```
   RMAN> change backupset 1338, 1339, 1340 crosscheck;

   RMAN-03022: compiling command: change
   RMAN-08074: crosschecked backup piece: found to be 'EXPIRED'
   RMAN-08517: backup piece handle=/oracle/dbs/2eafnuj3_1_1 recid=77 stamp=352057957
   RMAN-08074: crosschecked backup piece: found to be 'AVAILABLE'
   RMAN-08517: backup piece handle=/oracle/dbs/2dafnuj2_1_1 recid=78 stamp=352057957
   RMAN-08074: crosschecked backup piece: found to be 'AVAILABLE'
   RMAN-08517: backup piece handle=/oracle/dbs/2fafnuj3_1_1 recid=79 stamp=352057960
   ```

   If a backup set is no longer available, RMAN marks it as EXPIRED. If it was marked EXPIRED and is now available, RMAN marks it AVAILABLE.

4. Release the allocated maintenance channel:

   ```
   release channel;
   ```

To let RMAN generate the list of backups for a crosscheck:

1. Allocate a channel of type **maintenance**:

   ```
   allocate channel for maintenance type 'sbt_tape';
   ```

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 `tbs_8.f` over the last six months:

   ```
   crosscheck backup of datafile "/oracle/dbs/tbs_8.f" completed after 'SYSDATE-180';
   ```

   If a backup set is no longer available, RMAN marks it as EXPIRED. If it was marked EXPIRED and is now available, RMAN marks it AVAILABLE.

3. Release the allocated maintenance channel:

   ```
   release channel;
   ```

Crosschecking Image Copies

Use the **change ... crosscheck** command to determine whether image copies of datafiles, control files, or archived redo logs on disk or tape are valid. If RMAN is unable to find the specified image copy or archived redo log, it updates its status to DELETED.

> **See Also:** Chapter 11, "Recovery Catalog Views" for information about recovery catalog views that you can query for copies and logs with DELETED status.

RMAN considers archived redo logs as image copies. If for some reason one or more archived redo logs becomes unavailable, issue a **change archivelog all crosscheck** command so that RMAN updates the STATUS column of the absent logs to DELETED. You do not need to use a recovery catalog to execute this command.

To crosscheck image copies:

1. Connect to RMAN either with or without a recovery catalog. For example, enter one of the following:

```
% rman target / nocatalog
% rman target / catalog rman/rman@rcat
```

2. Identify the image copy that you want to check by issuing a **list** command. This example lists all recorded image copies and archived redo logs:

```
list copy of database archivelog all;

List of Datafile Copies
Key     File S Completion time Ckp SCN    Ckp time    Name
------- ---- - --------------- ---------- ----------  ------
1262    1    A 18-AUG-98       219859     14-AUG-98   /oracle/dbs/copy/tbs_01.f

List of Archived Log Copies
Key     Thrd Seq     S Completion time Name
------- ---- ------- - --------------- -----------------------------------
789     1    1       A 14-JUL-98       /oracle/work/arc_dest/arcr_1_1.arc
790     1    2       A 11-AUG-98       /oracle/work/arc_dest/arcr_1_2.arc
791     1    3       A 12-AUG-98       /oracle/work/arc_dest/arcr_1_3.arc
```

3. Use **change ... crosscheck** to check whether the specified copy exists. If not, RMAN updates its status to DELETED. This example checks whether the datafile copy with primary key 1262 exists:

```
change datafilecopy 1262 crosscheck;
```

```
RMAN-03022: compiling command: change
RMAN-06154: validation succeeded for datafile copy
RMAN-08513: datafile copy filename=/oracle/dbs/copy/tbs_01.f recid=1 stamp=351194732
```

If RMAN is unable to validate the copy, you will see:

```
RMAN-06153: validation failed for datafile copy
```

To update the RMAN repository after archived redo logs have been removed:

1. Connect to RMAN with or without a catalog. For example, enter either:

   ```
   % rman target / nocatalog
   % rman target / catalog rman/rman@rcat
   ```

2. Issue a **change ... crosscheck** command to update the repository records for the absent archived redo logs. This example crosschecks all archived redo logs:

   ```
   change archivelog all crosscheck;
   ```

   See Also:

   - "crosscheck" on page 10-64 for **crosscheck** command syntax
   - "change" on page 10-38 for **change** command syntax
   - "list" on page 10-83 for **list** command syntax

Deleting Backups and Copies and Updating Their Status in the RMAN Repository

You can use RMAN to delete backups, copies, and archived logs and update their status in the repository to DELETED status or remove the records entirely.

Note: Note that backup and copies with DELETED status do not appear in the **list** command output: query the recovery catalog views instead.

Table 3–2 describes the functionality of the maintenance commands and scripts:

Table 3–2 Maintenance Commands and Scripts **(Page 1 of 2)**

Command or Script	Need Cat?	Purpose
change ... uncatalog	Yes	To remove the record of a specified backup or copy from the recovery catalog. This command does *not* delete physical backups or copies: it only removes their records.

Table 3–2 Maintenance Commands and Scripts **(Page 2 of 2)**

Command or Script	Need Cat?	Purpose
`prgrmanc.sql`	Yes	To remove all records of backups or copies with status DELETED from the catalog at one time. The script does not delete physical backups or copies: it only removes their records.
delete expired backup	Yes	To remove backup set repository records with status EXPIRED. RMAN also physically deletes any expired backups if they still exist. Typically, you issue this command after performing a crosscheck. A crosscheck marks inaccessible backups as EXPIRED.
change ... delete	No	To delete physical backups and image copies and remove their records from the repository. If you use a recovery catalog, and the catalog compatibility level is below 8.1.6, this command deletes archived logs and updates their repository records to status DELETED. If the compatibility level is 8.1.6 or higher, this command deletes archived redo logs and removes their repository records. Unlike the **delete expired** command, **change ... delete** operates on *any* backup or copy—not just those marked EXPIRED. **See Also**: "Setting Recovery Catalog Compatibility" on page 3-4.
backup archivelog ... delete input	No	If the catalog compatibility level is below 8.1.6, this command deletes archived logs and marks their repository records to status DELETED. If the recovery catalog compatibility is 8.1.6 or higher, this command deletes archived redo logs and removes their repository records. **See Also**: "Setting Recovery Catalog Compatibility" on page 3-4.
`$ORACLE_HOME/rdbms/demo/rman1.sh`	No	To delete obsolete physical backup sets, image copies, or archived redo logs and either remove their records or update their repository records to status DELETED. **See Also**: **change ... delete** for a description of the affect of the script on the repository. **Note:** This script works only on UNIX systems.

See Also:

- "change" on page 10-38 for **change** command syntax
- "deleteExpired" on page 10-69 for **delete** command syntax
- Chapter 11, "Recovery Catalog Views" for descriptions of the recovery catalog views

To update the catalog record for a backup or copy deleted with an operating system utility:

1. Allocate a channel of type **maintenance**:

   ```
   allocate channel for maintenance type disk;
   ```

2. Issue a **change ... uncatalog** command for the backups or copies that you deleted from the operating system using operating system commands. This example removes references to copies of the control file and `datafile 1` from the repository:

   ```
   change controlfilecopy '/oracle/backup/cf_c.f' uncatalog;
   change datafilecopy '/oracle/backup/df_1_c.f' uncatalog;
   ```

3. Release the allocated channel:

   ```
   release channel;
   ```

To physically delete backups and image copies and remove their repository records:

This procedure does not require the use of a recovery catalog.

The behavior of **change ... delete** differs depending on what you are deleting. If you are deleting backups and image copies (but *not* archived logs), then **change ... delete** deletes the physical files and removes the records for these files from the repository.

If you are deleting archived redo logs and you use a catalog, then the outcome of the following procedure depends on the compatibility setting of the recovery catalog (see "Setting Recovery Catalog Compatibility" on page 3-4), as explained in the following table:

If compatibility is set to	Then
8.1.6 or higher	RMAN deletes the specified backups, copies, or archived logs and completely removes their records from the repository.
8.1.5 or lower	RMAN deletes the specified backups, copies, or archived logs and updates their repository records to status DELETED.

1. Check for the expired backup sets and copies. Use the **list** output to obtain their primary keys.

   ```
   list backup of database archivelog all; # lists backups of database files and logs
   list copy;
   ```

2. Allocate a channel of type **delete**:

   ```
   allocate channel for delete type disk;
   ```

3. Issue a **change ... delete** command to eliminate the specified physical files and the repository records. If you are deleting archived logs, then depending on the compatibility setting of the recovery catalog, this command either changes the recovery catalog record to status DELETED *or* removes the record altogether.

   This example deletes the backup piece with key 101, the control file copy with key 63, and all archived logs through log sequence 300 from disk:

   ```
   change backuppiece 101 delete;
   change controlfilecopy 63 delete;
   change archivelog until logseq = 300 delete;  # effect on catalog records for these
                                                 # logs depends on compatibility setting
   ```

4. Release the allocated maintenance channel:

   ```
   release channel;
   ```

To physically delete obsolete backups and copies and remove their repository records:

This procedure does not require the use of a recovery catalog. The script below works only on UNIX systems.

1. Change into the `$ORACLE_HOME/rdbms/demo` directory and edit the following shell script as needed:

   ```
   % vi rman1.sh
   ```

2. Execute the script:

   ```
   % rman1.sh
   ```

3. If desired, check `deleted.log` to see the command output.

To remove EXPIRED backup records in the catalog (and delete any existing expired backup pieces):

This procedure requires the use of a recovery catalog.

1. Optionally, perform a crosscheck operation (see "Crosschecking the RMAN Repository" on page 3-14) to mark all inaccessible or absent backups as EXPIRED. The **delete expired backup** command only operates on expired backups.
2. Allocate a channel of type **delete**:

   ```
   allocate channel for delete type disk;
   ```

3. Issue a **delete expired backup** command to check for backups marked EXPIRED and remove their catalog records. This example updates all backups registered in the recovery catalog that are expired:

   ```
   delete expired backup;
   ```

   If the expired pieces exist, RMAN removes them from the operating system.
4. Release the allocated channel:

   ```
   release channel;
   ```

To remove image copy and archived redo log records from the recovery catalog (without physically deleting the files):

This procedure requires the use of a recovery catalog.

1. Check for the expired datafile copies, control file copies, or archived redo logs. Use the **list** output to obtain their primary keys. For example, enter:

   ```
   list copy;
   ```

2. Issue a **change ... uncatalog** command to remove references to the specified copies from the recovery catalog. Note that this command does not remove records of backup sets and backup pieces.

   This example removes records of copies of the control file and of the datafile copy with the primary key 4833:

   ```
   change controlfilecopy '/oracle/backup/cf_c.f' uncatalog;
   change datafilecopy 4833 uncatalog;
   ```

   Note that **change ... uncatalog** does not remove files from the operating system; it only removes recovery catalog records.
3. 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:

   ```
   SQL> SELECT cdf_key, status FROM rc_datafile_copy WHERE cdf_key = 4833;
   ```

```
CDF_KEY    S
---------- -
0 rows selected.
```

To remove all copy and backup records from the recovery catalog:

You can remove all recovery catalog records of deleted backups and copies at once using the `$ORACLE_HOME/rdbms/prgrmanc.sql` script. Only perform this operation if you do not want a historical record of what you have backed up.

See Also: "change" on page 10-38 for **change** command syntax.

1. Allocate a channel of type **maintenance**:

   ```
   allocate channel for maintenance type 'sbt_tape';
   ```

2. Issue **change ... delete** commands to update the desired records to DELETED status and remove the file from the operating system or media manager. Issue **list** commands or query the recovery catalog views to obtain primary keys for archived redo logs, backup sets, control file copies, or datafile copies.

   ```
   change backupset 100, 101, 102, 103 delete;
   ```

3. Release the allocated maintenance channel:

   ```
   release channel;
   ```

4. Start a SQL*Plus session and connect to the recovery catalog. This example connects to the database RCAT as user RMAN:

   ```
   % sqlplus rman/rman@rcat
   ```

5. Run the script `prgrmanc.sql` script, which is stored in the `$ORACLE_HOME/rdbms/admin` directory:

   ```
   SQL> @prgrmanc.sql
   ```

   RMAN removes all records with status DELETED from the recovery catalog.

To remove incarnation records from the recovery catalog:

1. Allocate a channel of type **maintenance**:

   ```
   allocate channel for maintenance type 'sbt_tape';
   ```

2. Issue **change ... delete** commands to remove unwanted backup pieces, archived redo logs, and image copies. Issue **list** commands or query the recovery catalog

views to obtain primary keys for archived redo logs, backup sets, control file copies, or datafile copies.

```
change backupset 100, 101, 102, 103 delete;
```

3. Release the allocated maintenance channel:

```
release channel;
```

4. Start a SQL*Plus session and connect to the recovery catalog. This example connects to database RCAT as user RMAN:

```
% sqlplus rman/rman@rcat
```

5. Obtain the DBINC_KEY values for the incarnations whose records you want to delete by querying the RC_DATABASE_INCARNATION recovery catalog view:

```
SQL> SELECT * FROM rc_database_incarnation;
```

6. Execute the following DML statement, where `key_value` is the value of DBINC_KEY:

```
SQL> DELETE FROM dbinc WHERE dbinc_key=key_value;
```

RMAN will remove the specified incarnation records from the recovery catalog.

> **See Also:** "RC_DATABASE_INCARNATION" on page 11-15 for more information about the RC_DATABASE_INCARNATION recovery catalog view.

Validating the Restore of Backups and Copies

A restore *validation* executes a restore test run without actually restoring the files. Test the restore of 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 should use:

- **restore ... validate** when you want RMAN to choose which backups or copies should be tested.
- **validate backupset** when you want to specify which backup sets should be tested.

> **See Also:** "restore" on page 10-120 for **restore** command syntax, and "validate" on page 10-160 for **validate** command syntax.

To let RMAN choose which backup sets or copies to validate:

1. If you want to validate the whole database, close it. If you want to validate individual files, then either close the database or take the individual datafiles offline. This example checks the status of the database, aborts the instance, and then mounts it:

```
SQL> SELECT status FROM v$instance;

STATUS
-------
OPEN
1 row selected.

SQL> STARTUP FORCE MOUNT;
```

2. Start RMAN and connect to the target database and optional recovery catalog database:

```
% rman target / catalog rman/rman@rcat
```

3. If you do not want to validate the whole database, identify the backup sets and copies that you want to validate by issuing **list** commands, noting primary keys:

```
list backupset;
list copy;
```

4. Validate the restore of the backup sets and copies. This example validates the restore of the backup control file, SYSTEM tablespace, and all archived redo logs from disk:

```
run {
     allocate channel ch1 type disk;
     allocate channel ch2 type tape;
     restore controlfile validate;
     restore tablespace 'system' validate;
     restore archivelog all validate;
}
```

5. Check the output. If you see an error message stack and then the following, 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-03002: failure during compilation of command
```

```
RMAN-03013: command type: restore
RMAN-03007: retryable error occurred during execution of command: IRESTORE
RMAN-07004: unhandled exception during command execution on channel c1
RMAN-10035: exception raised in RPC: 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
RMAN-10031: ORA-19624 occurred during call to DBMS_BACKUP_RESTORE.RESTOREBACKUPPIECE
```

If you do not see an error stack, then RMAN successfully validated the files.

To specify which backup sets to validate:

1. Start RMAN and connect to the target database and optional recovery catalog database:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. Identify the backup sets that you want to validate by issuing **list** commands:

   ```
   list backup of database archivelog all;
   ```

3. Validate the restore of the backup sets. This example validates the restore of the backup control file, SYSTEM tablespace, and all archived redo logs from disk:

   ```
   run {
        allocate channel ch1 type disk;
        validate backupset 1121;
   }
   ```

4. Check the output. If you see the `RMAN-08024` message then RMAN successfully validated the restore of the specified backup set.

   ```
   RMAN-03022: compiling command: allocate
   RMAN-03023: executing command: allocate
   RMAN-08030: allocated channel: ch1
   RMAN-08500: channel ch1: sid=10 devtype=DISK

   RMAN-03022: compiling command: validate
   RMAN-03023: executing command: validate
   RMAN-08016: channel ch1: starting datafile backupset restore
   RMAN-08502: set_count=47 set_stamp=346169465 creation_time=08-OCT-98
   RMAN-08023: channel ch1: restored backup piece 1
   RMAN-08511: piece handle=/vobs/oracle/dbs/1faa483p_1_1 params=NULL
   RMAN-08024: channel ch1: restore complete
   RMAN-08031: released channel: ch1
   ```

Storing Scripts in the Recovery Catalog

A *stored script* is a sequence of RMAN commands stored within the recovery catalog. It provides a common repository of frequently executed collections of 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 using RMAN, regardless of which machine RMAN is executed on.

RMAN allows you to:

- Create a script and store it in the recovery catalog.
- Execute a stored script.
- Replace a stored script.
- Delete a script from the recovery catalog.
- Print a stored script to the message log file or the screen.
- Obtain a listing of all your stored scripts.

To create a stored script:

1. Start RMAN and connect to the recovery catalog database. For example, enter:

```
% rman catalog rman/rman@rcat
```

2. Write the desired script. For example, this script backs up the database and the archived redo logs:

```
create script b_whole{
     allocate channel ch1 type disk;
     allocate channel ch2 type disk;
     backup database;
     sql 'ALTER SYSTEM ARCHIVE LOG ALL';
     backup archivelog all;
}
```

3. Examine the output. If you see message `RMAN-08085`, then the script was successfully created and stored in the recovery catalog:

```
RMAN-03022: compiling command: create script
RMAN-03023: executing command: create script
RMAN-08085: created script b_whole
```

To execute a stored script:

1. Start RMAN and connect to the recovery catalog database and target database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. Issue a **run** command to execute the script. RMAN inserts the contents of the script between the brackets of **run**. Note that you do not need to allocate channels if you already did so *within* the script:

   ```
   run { execute script b_whole; }
   ```

   See Also: "run" on page 10-133 for **execute script** command syntax.

To replace a stored script:

1. Start RMAN and connect to the recovery catalog database and target database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. 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 {
        allocate channel ch1 type 'sbt_tape';
        backup database;
   }
   ```

   See Also: "replaceScript" on page 10-105 for **replace script** command syntax.

To delete a stored script:

1. Start RMAN and connect to the recovery catalog database. For example, enter:

   ```
   % rman catalog rman/rman@rcat
   ```

2. Issue a **delete script** command to delete the specified stored script:

   ```
   delete script 'b_whole';
   ```

   See Also: "deleteScript" on page 10-71 for **delete script** command syntax.

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/rman@rcat log rman_log
   ```

2. Issue a **print script** command to write the script to the log:

   ```
   print script b_whole;
   ```

3. Host out to the operating system and use a text editor to view the script. For example, enter:

   ```
   RMAN> host;
   % vi rman_log
   ```

> **See Also:** "printScript" on page 10-94 for **print script** command syntax.

To obtain a listing of all stored scripts:

1. Start SQL*Plus and connect to the recovery catalog database. For example, to connect to database RCAT enter:

   ```
   % sqlplus rman/rman@rcat
   ```

2. Issue a SELECT statement on the RC_STORED_SCRIPT view:

   ```
   SQL> SELECT script_name FROM rc_stored_script;
   SCRIPT_NAME
   --------------------------------------------------------------------------------
   backupdb
   binc
   binc1
   3 rows selected.
   ```

> **See Also:** "RC_STORED_SCRIPT" on page 11-25 for information about the RC_STORED_SCRIPT view.

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.

Resynchronizations can be *full* or *partial*. In a partial resynchronization, RMAN reads the current control file to update changed information, but 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, 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 (see "resync" on page 10-127 for more information). To ensure a full resynchronization, issue a **resync catalog** command.

This section contains the following topics:

What Is Resynchronized?

Table 3–3 describes the types of records that RMAN resynchronizes.

Table 3–3 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.

Table 3–3 Records Updated during a Resynchronization

Records	Description
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. Execution of the following commands performs 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 when the command is executed:

- **backup**
- **copy**
- **crosscheck**
- **delete expired backupset**
- **duplicate**
- **list**
- **recover**
- **report**
- **restore**
- **switch**

For example, if you execute a **backup** command you see partial resynchronization messages after the backup completes:

```
RMAN-08525: backup set complete, elapsed time: 00:00:02
```

```
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-08031: released channel: c1
```

Perform manual resynchronizations in the following scenarios.

Resynchronizing when the Recovery Catalog is Unavailable If the recovery catalog is unavailable when you issue **backup** or **copy** commands, then open the catalog database later and resynchronize it manually using the **resync catalog** command.

Resynchronizing when You Run in ARCHIVELOG Mode If you are running in ARCHIVELOG mode, 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. Thus, you can perform this operation frequently—for example, hourly—without incurring undue costs.

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 when you:

- Add or drop a tablespace.
- Add a new datafile to an existing tablespace.
- Add or drop a rollback segment.

When resynchronizing from the current control file, RMAN automatically detects the records in the control file that have been updated and resynchronizes only those records. If the target database is open, then RMAN also updates information about rollback segments in the recovery catalog (this information is used for TSPITR).

When resynchronizing from a backup control file, RMAN *does not* verify that the backup pieces or file copies actually exist. Thus, you may need to use the **change ...**

crosscheck and **crosscheck** commands to remove records for files that no longer exist.

How Do You Resynchronize?

Issue the **resync catalog** command to force a full resynchronization of the recovery catalog.

To perform a full resynchronization of the recovery catalog:

1. Open the recovery catalog database (if it is not already open). For example, enter:

   ```
   SQL> STARTUP pfile=initRCAT.ora
   ```

2. Use RMAN to connect to the target and recovery catalog databases. For example, enter:

   ```
   % rman target sys/change_on_install@prod1 catalog rman/rman@rcat
   ```

3. Mount the target database if it is not already mounted:

   ```
   startup mount;
   ```

4. Issue the **resync catalog** command:

   ```
   resync catalog;

   RMAN-03022: compiling command: resync
   RMAN-03023: executing command: resync
   RMAN-08002: starting full resync of recovery catalog
   RMAN-08004: full resync complete
   ```

   See Also: "resync" on page 10-127 for **resync catalog** command syntax.

Managing Records in the Control File

The recovery catalog is dependent on the target database control file for information. Consequently, the currency of the information in the control file determines the effectiveness of the recovery catalog.

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.

You can use the CONTROL_FILE_RECORD_KEEP_TIME parameter to specify the minimum number of days that Oracle keeps this information in the control file. Entries older than the number of days specified are candidates for overwrites by newer information. The larger the CONTROL_FILE_RECORD_KEEP_TIME setting, the larger the control file.

At a minimum, resynchronize your recovery catalog at intervals less than the CONTROL_FILE_RECORD_KEEP_TIME setting, because after the number of days specified in this parameter, Oracle overwrites the information in the control file with the most recently created information. If you have not resynchronized the recovery catalog, and Oracle has overwritten the information, then this information cannot be propagated to the recovery catalog.

Note: The maximum size of the control file is port-specific. See your operating system-specific Oracle documentation.

See Also:

- *Oracle8i Reference* for more information about the CONTROL_FILE_RECORD_KEEP_TIME parameter
- *Oracle8i Backup and Recovery Guide* for more information about managing control files for backup and recovery
- *Oracle8i Administrator's Guide* for more detailed information on all aspects of control file management

Cataloging Operating System Backups

You can make RMAN aware of the existence of file copies that are created through means other than RMAN. This section contains the following topics:

What Is Cataloging?

If you do not make backups or image copies by using RMAN commands, then the recovery catalog has no record of them. You must manually notify RMAN when you make backups with an operating system utility. Use the RMAN **catalog** command to:

- Add information about an operating system datafile copy, archived redo log copy, or control file copy to the recovery catalog and control file.
- Catalog a datafile copy as a level 0 backup, thus enabling you to perform an incremental backup later using the datafile copy as the base of an incremental backup strategy.
- Record the existence of backups of version 8 databases created before RMAN was installed.
- Record the existence of Oracle7 backups of read-only or offline normal files made before migrating to Oracle8*i*.

When Should You Catalog an Operating System Backup?

Catalog backups whenever:

- You make a copy through an operating system utility.
- You want to catalog Oracle7 operating system backups that were made offline normal or read-only prior to migration.

> **Note:** You cannot re-catalog backup sets or pieces.

Cataloging Hot and Cold Operating System Backups Whenever you make a cold operating system backup, for example, by using the UNIX cp command to copy a datafile, make sure to catalog it. Oracle8*i* supports the ALTER TABLESPACE ... BEGIN/END BACKUP command, which allows open database operating system backups. Although RMAN does not create such backups, you can add them to the recovery catalog so that RMAN is aware of them.

For a backup to be cataloged, it must be:

- Accessible on disk.
- A complete image copy of a single file.
- Either a consistent or inconsistent whole database, tablespace, datafile, control file, or archived redo log backup. RMAN treats all such backups as datafile copies.

For example, if you store datafiles on mirrored disk drives, you can create an operating system copy by breaking the mirror. In this scenario, use the **catalog** command to notify RMAN of the existence of the operating system copy after

breaking the mirror. Before reforming the mirror, issue a **change ... uncatalog** command to notify RMAN that the file copy is being deleted.

Cataloging Oracle7 Operating System Backups RMAN cannot catalog Oracle7 files, except in the following special circumstances. During the migration from Oracle7 to Oracle8*i*, you shut down your Oracle7 database cleanly prior to running the migration utility. At this time, you can take operating system backups of your Oracle7 datafiles and catalog the backups with RMAN: Oracle accepts them because no redo from the old database is required to recover them. RMAN can then restore those backups in your Oracle8*i* database if no other backups exist.

Following is a scenario that generates an Oracle7 backup that you can catalog with RMAN:

1. Take a datafile offline clean or read-only.
2. Close the database. You must not have made the files read-write again before closing the database.
3. Make an operating system backup of the offline or read-only files.
4. Migrate the database to Oracle8*i*.

The pre-migration backups are identical to the backups that would be taken prior to migration, and may also be cataloged with RMAN.

> **See Also:** *Oracle8i Migration* to learn how to migrate an Oracle database.

How Do You Catalog an Operating System Backup?

Use the **catalog** command to propagate information about operating system backups to the recovery catalog.

To catalog an operating system backup:

1. Make a backup via an operating system utility. This example backs up a datafile.

   ```
   % cp $ORACLE_HOME/dbs/sales.f $ORACLE_HOME/backup/sales.bak';
   ```

2. Use RMAN to connect to the target and recovery catalog databases. For example, enter:

   ```
   % rman target change_on_install@prod1 catalog rman/rman@rcat
   ```

3. Issue the **catalog** command. For example, enter:

```
catalog datafilecopy '$ORACLE_HOME/backup/sales.bak';

RMAN-03022: compiling command: catalog
RMAN-03023: executing command: catalog
RMAN-08050: cataloged datafile copy
RMAN-08513: datafile copy filename=/oracle/backup/sales.bak recid=121 stamp=342972501
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
```

See Also: "catalog" on page 10-35 for **catalog** command syntax.

Backing Up and Recovering the Recovery Catalog

Include the recovery catalog in your backup and recovery strategy. If you do not back up your recover catalog and a disk crash occurs, you may lose some or all of your data. Avoid this unpleasant scenario by deciding how to back up and recover the recovery catalog.

This section contains the following 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, follow these guidelines:

- Make Regular Backups
- Use the Appropriate Method for Physical Backups
- Store the Recovery Catalog in an Appropriate Place
- Make Logical Backups

Make Regular Backups

Back up the recovery catalog with the same frequency that you back up your target database. For example, if you make a weekly whole database backup of your target database, then back up your recovery catalog immediately after each target database backup. The backed up catalog will have a record of the target backup

preceding it, so if you are forced to restore the catalog you will also be able to use it to restore the target database backup.

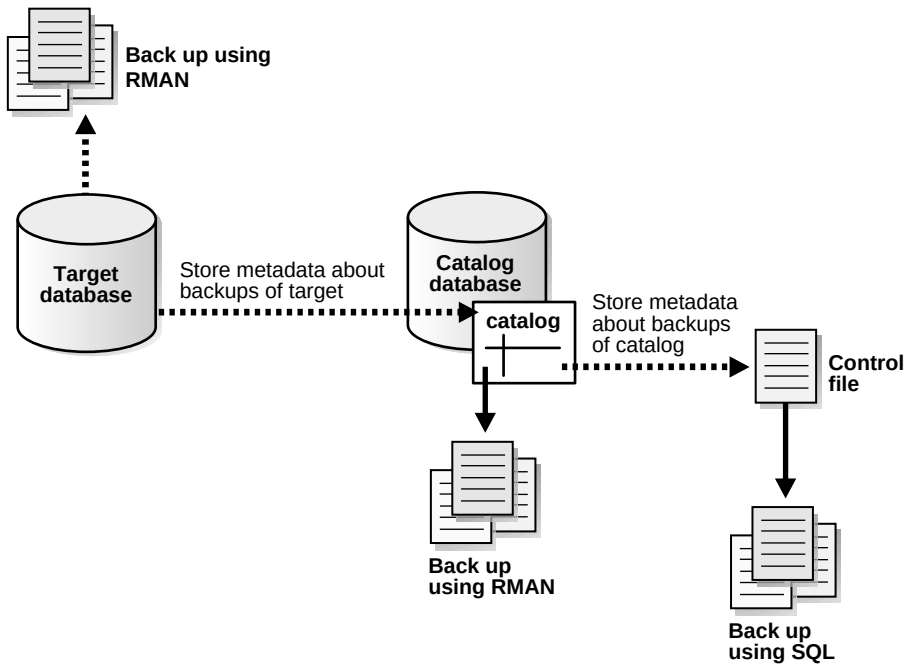
Use the Appropriate Method for Physical Backups

When backing up the recovery catalog, you have a choice of making operating system or RMAN backups. The advantage of making operating system backups is that you can restore them using operating system methods without relying on RMAN or a recovery catalog. For operating system backup procedures, see *Oracle8i Backup and Recovery Guide*.

If you use RMAN to back up the recovery catalog database, consider how you will restore the recovery catalog in case of failure. For example, you can:

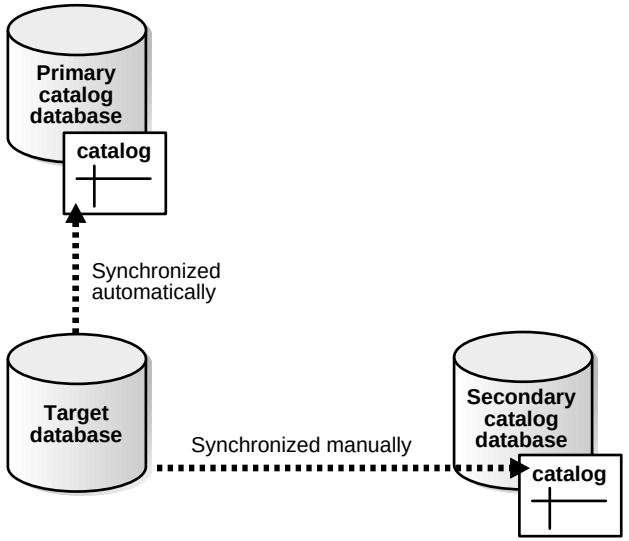
- Back up the recovery catalog database using RMAN, but start RMAN with the **nocatalog** option so that the backup repository for the recovery catalog is the control file in the catalog database. Make frequent backups of this control file using SQL. Remember to set the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter to a value that is high enough to store an adequate amount of historical backup data for the catalog.

Figure 3–1 Using the Control File as the Repository for Backups of the Catalog



- Create a second recovery catalog in a separate database. The main catalog is kept synchronized as normal, while the secondary catalog is synchronized manually by periodically issuing the **resync catalog** command. To learn how to create a catalog, see "Creating the Recovery Catalog" on page 3-2.

Figure 3–2 Registering One Database in Two Catalogs



See Also: *Oracle8i Backup and Recovery Guide* to learn how to back up the control file using SQL statements.

Store the Recovery Catalog in an Appropriate Place

Never store a recovery catalog containing the RMAN repository for a database in the same database as your 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 the recovery catalog data for PROD1 is also stored in PROD1, then you have no catalog to aid in recovery.

This rule is especially important when you want to back up a recovery catalog database. Take a case in which database RCAT contains the recovery catalog repository for target database PROD1. You decide to use a recovery catalog to back up RCAT, but are not sure where to store this catalog. If you store the catalog containing the repository for RCAT in database RCAT itself, then if you lose RCAT because of a media failure you will have difficulty restoring RCAT *and* will leave PROD1 unprotected.

Make Logical Backups

Use the Export utility to make convenient backups of your recovery catalog data. An export of the catalog allows the most flexibility when the recovery catalog must be restored, because it can be restored to any existing Oracle8*i* database.

To make a logical export of the recovery catalog from the command line:

1. Execute the export operation at the 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 PROD1 is RMAN, you can issue the following at the UNIX command line to export the catalog to file `cat.dmp`:

   ```
   % exp rman/rman@prod1 file=cat.dmp owner=rman
   ```

2. Examine the output to make sure you were successful:

   ```
   Export terminated successfully without warnings.
   ```

> **See Also:** *Oracle8i Utilities* to learn how to use the Export utility, and *Oracle8i Backup and Recovery Guide* to learn how to make operating system backups.

Recovering the Recovery Catalog

The method you use to recover the catalog depends on the method you use to back it up. You have these options:

- Recovering the Catalog Using Operating System Methods
- Recovering the Catalog Using RMAN
- Importing a Logical Backup of the Catalog

Recovering the Catalog Using Operating System Methods

If you backed up the recovery catalog using operating system commands, then restore the backup of the catalog using operating system commands and issue SQL*Plus commands to recover it. The method you use depends on whether you are recovering the entire recovery catalog database or only the tablespace in which

the recovery catalog is stored. For procedures, see *Oracle8i Backup and Recovery Guide*.

Recovering the Catalog Using RMAN

If you use RMAN to recover the catalog, then see Chapter 6, "Restoring and Recovering with Recovery Manager" for the relevant procedures.

Importing a Logical Backup of the Catalog

If you used Export to make a logical backup of the recovery catalog, then use Import to recover it. If a media failure damages your recovery catalog database, do the following:

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 3-2.
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 using the FROMUSER parameter.
   c. Specify the new owner using 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 NEW_CAT is RCAT.
   - The file containing the export of the catalog is `cat.dmp`.

   ```
   % imp userid=rcat/rcat_pwd@new_cat file=cat.dmp fromuser=rman touser=rcat
   ```
3. Use the imported catalog data for restore and recovery of your target database.

See Also: *Oracle8i Utilities* to learn how to use the Import utility.

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 mechanisms is not possible, then you must re-create the catalog.

You have two options for partially re-creating the contents of the old catalog:

- Issue **catalog** commands to re-catalog archived redo logs, backup control files, and datafile copies.
- Use the **resync catalog from controlfilecopy** command to extract information from a backup control file and rebuild the recovery catalog from it.

You can re-create information about backup sets *only* by using the **resync catalog from controlfilecopy** command, because the **catalog** command does not support re-cataloging of backup pieces or backup sets. RMAN does not verify that the files being re-cataloged still exist, so the resynchronization may add records for files that no longer exist. Remove such records by issuing **change ... crosscheck** or **crosscheck backup** commands.

> **See Also:** "change" on page 10-38 for information about the **change** command, and "crosscheck" on page 10-64 for information about the **crosscheck** command.

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 an 8.1 version of RMAN with an 8.0 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.

To upgrade the recovery catalog:

1. Use RMAN to connect to the target and recovery catalog databases. For example, enter:

```
% rman target / catalog rman/rman@rcat

RMAN-06008: connected to recovery catalog database
RMAN-06186: PL/SQL package rcat.DBMS_RCVCAT version 08.00.04 in RCVCAT
            database is too old
```

2. Issue the **upgrade catalog** command:

```
RMAN> upgrade catalog
```

```
RMAN-06435: recovery catalog owner is rman
RMAN-06442: enter UPGRADE CATALOG command again to confirm catalog upgrade
```

3. Enter the **update catalog** command again to confirm:

```
RMAN> upgrade catalog

RMAN-06408: recovery catalog upgraded to version 08.01.05
RMAN-06452: DBMS_RCVMAN package upgraded to version 08.01.05
RMAN-06452: DBMS_RCVCAT package upgraded to version 08.01.03
```

See Also:

- "upgradeCatalog" on page 10-158 for **upgrade catalog** command syntax
- "configure" on page 10-47 for information about recovery catalog compatibility
- *Oracle8i 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 registers multiple target databases. If you try to delete the information for one target database by dropping the catalog, you delete the information 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/rman@rcat
```

2. Issue the **drop catalog** command twice to confirm:

```
RMAN> drop catalog

RMAN-06435: recovery catalog owner is rman
RMAN-06436: enter DROP CATALOG command again to confirm catalog removal
RMAN> drop catalog
```

Note: Even after you drop the recovery catalog, the control file still contains records about your backups. To purge those records, re-create the control file.

See Also: "dropCatalog" on page 10-74 for **drop catalog** command syntax, and "Unregistering a Database from the Recovery Catalog" on page 3-11 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 its information from the control file. If you choose not to use a recovery catalog, however, you must perform the following tasks:

- Understanding Catalog-Only Command Restrictions
- Monitoring the Overwriting of Control File Records
- Maintaining the Control File Repository
- Backing Up the Control File

See Also:

- *Oracle8i Concepts* for a conceptual overview of the control file.
- Chapter 3, "Managing the Recovery Manager Repository" for a description of the importance of the control file for backup and recovery
- *Oracle8i 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, 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:

- **change ... available**, **change ... unavailable**, **change ... uncatalog**, **change backupset ... crosscheck**, **change backuppiece ... crosscheck**

- **configure compatible**
- **create catalog**, **upgrade catalog**, **drop catalog**
- **create script**, **delete script**, **replace script**, **print script**
- **crosscheck backup**
- **delete expired backup**
- **list incarnation**
- **register database**
- **report schema at time**
- **reset database**
- **restore** (when no control file is available)
- **resync catalog**
- **set auxname**

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

- Use a minimum of three multiplexed or mirrored control files, with each control file on a separate disk.
- Keep excellent records of which files were backed up, the date they were backed up, and also the names of the backup pieces each file was written to. Use the **list** command to obtain information about backups and copies. Keep all RMAN backup logs.

WARNING: The only way to restore and recover when you have lost all control files and do not use a recovery catalog is to call Oracle Support Services. Support will need to know the following:

- **Current schema of the database**
- **Which files were backed up**
- **What time the files were backed up**
- **Names of the backup pieces containing the files**

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.

> **See Also:** "Types of Records in the Control File" on page 1-17 for a conceptual overview of control file 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. Any information in an overwritten record is lost.

> **Note:** If you maintain a recovery catalog, use the RMAN **resync catalog** command to resynchronize the catalog before records are reused. In this way, RMAN never loses records because they are recorded in the recovery catalog.

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`.

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 your 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
07/28/99 22:15:21
Thread=1 Seq#=3460
Backup set key: stamp=372031415, count=17
Low scn: 0x0000.3af33f36
07/27/99 21:00:08
Next scn: 0x0000.3af3871b
07/27/99 23:23:54
Resetlogs scn and time
scn: 0x0000.00000001
08/05/98 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.

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. "Maintaining the RMAN Repository" on page 3-8 provides a complete description of these maintenance procedures. Most of these commands work whether or not you use a recovery catalog.

For a list of the commands that require a catalog, see "Connecting to RMAN" on page 2-8. You do not lose much in the way of maintenance functionality when you do not use a recovery catalog, because you can:

- Use the **restore ... validate** or **validate backupset** commands to test whether backups and copies are available and can be restored. See "Validating the Restore of Backups and Copies" on page 3-24 for a description of this procedure.

- Use **change ... delete** to mark records as DELETED in the control file and physically delete unwanted or corrupted backups and copies from the operating system. See "Deleting Backups and Copies and Updating Their Status in the RMAN Repository" on page 3-18 for a description of this procedure.

Backing Up 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 your control file and do not have a backup, you lose all information about RMAN backups and copies contained in the file.

To learn how to multiplex and mirror the control file and to make operating system backups of the control file, see *Oracle8i Backup and Recovery Guide*.

4

Generating Lists and Reports with Recovery Manager

This chapter describes how to use Recovery Manager to make useful lists and reports of your backups and image copies, and includes the following topics:

- Using Lists and Reports in Your Backup and Recovery Strategy
- Generating Lists
- Generating Reports
- List and Report Scenarios

Using Lists and Reports in Your Backup and Recovery Strategy

Use the **report** and **list** commands to determine what you have backed up or copied as well as what you need to back up or copy. The information, which is available to you whether or not you use a recovery catalog, is extremely helpful in developing an effective backup strategy. Refer to Chapter 3, "Managing the Recovery Manager Repository" to learn how to keep the RMAN repository current.

The **list** command displays all RMAN backups (both backup sets and proxy copies) and image copies, while the **report** command performs more complex analysis. For example, generate a report on which datafiles need a backup and which backup pieces are obsolete. Oracle writes the output from the **report** and **list** commands to either the screen or a log file.

> **See Also:** "list" on page 10-83 for **list** command syntax, and "report" on page 10-110for **report** command syntax.

Generating Lists

The **list** command queries the recovery catalog or control file and produces a listing of its contents. The primary purpose of the **list** command is to determine which backups or copies are available. For example, list:

- All 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 restricted by time, pathname, device name, tag, or recoverability.
- Incarnations of a specified database or of all databases known to the recovery catalog.

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 want backed up are included in the output.
- The backups and copies are recent.

To generate a list of image copies and backups:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log '/oracle/log/mlog.f'
   ```

2. Execute **list copy** and **list backup** commands. If you do not specify the **of** *listObjList* clause, **list** defaults to **of database**:

   ```
   list copy of database archivelog all;  # lists datafiles and archived redo logs
   list backup;  # lists backup sets, backup pieces, and proxy copies
   ```

3. Examine the output. See "list" on page 10-83 for an explanation of the various column headings in the **list** output. Following is sample output:

   ```
   list copy of database archivelog all;

   List of Datafile Copies
   Key     File S Completion time Ckp SCN    Ckp time    Name
   ------- ---- - -------------- ---------- ----------  ------
   1262    1    A 18-AUG-98       219859     14-AUG-98   /oracle/dbs/copy/tbs_01.f

   List of Archived Log Copies
   Key     Thrd Seq     S Completion time Name
   ------- ---- ------- - -------------- -----------------------------------
   789     1    1       A 14-JUL-98       /oracle/work/arc_dest/arcr_1_1.arc
   790     1    2       A 11-AUG-98       /oracle/work/arc_dest/arcr_1_2.arc
   791     1    3       A 12-AUG-98       /oracle/work/arc_dest/arcr_1_3.arc

   list backup;

   List of Backup Sets
   Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
   ------- ---------- ---------- -- ---------- ---------- ---------------------
   1174    12         341344528  0  341344502  16         14-AUG-98

       List of Backup Pieces
       Key     Pc# Cp# Status     Completion Time    Piece Name
       ------- --- --- ---------- ------------------ ----------------------------
       1176    1   1   AVAILABLE  14-AUG-98          /oracle/dbs/0ga5h07m_1_1

       Controlfile Included
       Ckp SCN    Ckp time
       ---------- ---------------
   ```

```
219857     14-AUG-98

List of Datafiles Included
File Name                                  LV Type Ckp SCN    Ckp Time
---- ------------------------------------- -- ---- ---------- -------------
1    /oracle/dbs/tbs_01.f             0  Full 199943     14-AUG-98
2    /oracle/dbs/tbs_02.f             0  Full 199943     14-AUG-98
```

To generate a list of copies and backups restricted by object or other conditions:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. To restrict by object, use **list copy** or **list backup** with the **of** *listObjList* condition. For example, enter:

   ```
   list backup of database;     # lists backups of all files in database
   list copy of datafile '/oracle/dbs/tbs_1.f'; # lists copy of specified datafile
   list backup of tablespace SYSTEM; # lists all backups of SYSTEM tablespace
   list copy of archivelog all;  # lists all archived redo logs and copies of logs
   list backup of controlfile; # lists all control file backups
   ```

   You can also restrict your search by specifying a combination of tag, device type, filename pattern, or time options. For example, enter:

   ```
   list backup tag 'weekly_full_db_backup';    # by tag
   list copy of datafile '/oracle/dbs/tbs_1.f' type 'sbt_tape';   # by type
   list backup like '/oracle/backup/tbs_4%';    # by filename pattern
   list backup of archivelog until time 'SYSDATE-30';   # by time
   list copy of datafile 2 completed between '10-DEC-1999' and '17-DEC-1999'; # by time
   ```

3. Examine the output. For example, following is sample output for a list of copies of `datafile 1`:

   ```
   RMAN> list copy of datafile 1;

   RMAN-03022: compiling command: list

   List of Datafile Copies
   Key     File S Completion time Ckp SCN    Ckp time        Name
   ------- ---- - -------------- ---------- --------------- ------
   3       1    A 18-DEC-98       114148     18-DEC-98       /oracle/dbs/df1.bak
   ```

   See Also: "listObjList" on page 10-92 for *listObjList* syntax, and "List Output" on page 10-86 for an explanation of the various columns in the **list** output.

Generating 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 not been backed up recently?
- Which files are listed as unrecoverable?
- Which backups or copies are obsolete and can be deleted?
- What was the physical schema of the database at some previous time?
- Which backups and copies are on disk and which are on tape?

> **Note:** For the report to be accurate, the RMAN repository must be current and you must have used the **change**, **uncatalog**, and **crosscheck** commands appropriately to update the status of all backups and copies is correct. To learn how to maintain the RMAN repository see "Maintaining the RMAN Repository" on page 3-8.

The information that you glean from reports can be extremely important for your backup and recovery strategy. In particular, use the **report need backup** and **report unrecoverable** commands regularly to ensure that:

- 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.

To report on objects that need a backup:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   To write the output to a log file, specify a file at startup:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. If necessary, issue **crosscheck** commands to update the status of backups and **change ... crosscheck** commands to update the status of image copies (if you

want to specify image copies by primary key, issue a **list** command to obtain the keys).

Following is a possible crosscheck session:

```
# must allocate maintenance channel for crosscheck
allocate channel for maintenance type disk;
crosscheck backup;  # crosschecks all backups
change datafile copy 100,101,102,103,104,105,106,107 crosscheck; # specified by key
change archivelog copy 50,51,52,53,54 crosscheck;  # specified by key
release channel;
```

3. Use the **need backup** option to identify which datafiles need a new backup, restricting the report by a threshold number of days or incremental backups. RMAN considers any backups older than the **days** parameter value as needing a new backup because backups require **days** worth of archived redo logs for recovery.

   For example, enter:

   ```
   report need backup days = 7 database;  # needs at least 7 days of logs to recover
   report need backup days = 30 tablespace system;
   report need backup days = 14 datafile '/oracle/dbs/tbs_5.f';
   ```

   You can also 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 '/oracle/dbs/tbs_5.f';
   ```

4. Examine the report and back up those datafiles requiring a new backup.

   See Also: "Report Output" on page 10-114 for an explanation of the various column headings in the **report** output.

To report on backups that are obsolete:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a message log file, specify the file at startup:

   ```
   % rman target / catalog rman/rman@rcat log '/oracle/log/mlog.f'
   ```

2. If necessary, issue **crosscheck** commands to update the status of backups and **change ... crosscheck** commands to update the status of image copies (if you want to specify copies by primary key, issue a **list** command to obtain them):

```
allocate channel for maintenance type disk;
crosscheck backup;
change datafile copy 100,101,102,103,104,105,106,107 crosscheck;
change archivelog copy 50,51,52,53,54 crosscheck;
release channel;
```

3. Use the **obsolete** option to identify which backups are obsolete because they are no longer needed for recovery. The **redundancy** parameter specifies the minimum level of redundancy considered necessary for a backup or copy to be obsolete. If you do not specify the parameter, **redundancy** defaults to 1.

   A datafile copy is obsolete if at least *integer* more recent backups of this file exist; a datafile backup set is obsolete if at least *integer* more recent backups or image copies of each file contained in the backup set exist. For example, enter:

```
# lists backups or copies that have at least 2 more recent backups or copies
report obsolete redundancy = 2;
```

   Use the *untilClause* to use make the redundancy check for backups sets or copies that are more recent, but not later than the specified time, SCN, or log sequence number:

```
# obsolete if there are at least 2 copies/backups that are no more than 2 weeks old
report obsolete redundancy = 2 until time 'SYSDATE-14';
report obsolete until scn 1000;
report obsolete redundancy = 3 until logseq = 121 thread = 1;
```

4. Use the **orphan** option to list which backups and copies are unusable because they belong to a incarnation that is not a direct predecessor of the current incarnation:

```
report obsolete orphan;
```

   For an explanation of orphaned backups, see "Reporting on Orphaned Backups" on page 1-27.

5. Examine the report and delete those backups that are obsolete.

```
RMAN> report obsolete;

RMAN-03022: compiling command: report
Report of obsolete backups and copies
Type                 Recid  Stamp     Filename
-------------------- ------ --------- --------------------------
```

```
Backup Set           4      345390311
Backup Piece         4      345390310 /oracle/dbs/04a9cf76_1_1

RMAN> allocate channel for delete type disk;

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: delete
RMAN-08500: channel delete: sid=11 devtype=DISK

RMAN> change backuppiece '/oracle/dbs/04a9cf76_1_1' delete;

RMAN-03022: compiling command: change
RMAN-03023: executing command: change
RMAN-08073: deleted backup piece
RMAN-08517: backup piece handle=/oracle/dbs/04a9cf76_1_1 recid=4 stamp=345390310
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
```

To report on backups that are unrecoverable:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a message log file, specify the file at startup:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. Use the **unrecoverable** option of the **report** command to determine which datafiles have had an unrecoverable operation performed against an object residing in the datafile since its last backup.

   ```
   report unrecoverable database; # examines all datafiles
   ```

To report the database schema at a specified point in time:

You must use a recovery catalog for reporting on database schema at a past time, SCN, or log sequence number.

1. Start RMAN and connect to the target database and the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a message log file, specify the file at startup:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. Issue **report schema** for a list of all the datafiles and tablespaces in the target database at the current time:

```
report schema;
```

Use the *untilClause* to specify a past time, SCN, or log sequence number:

```
report schema at time 'SYSDATE-14';
report schema at scn 1000;
report schema at logseq 100;
```

3. Examine the report. For example, here is sample output:

```
RMAN> report schema at scn 1000;
RMAN-03022: compiling command: report

Report of database schema
File K-bytes    Tablespace           RB segs Name
---- ---------- -------------------- ------- ------------------
1         35840 SYSTEM               YES     /oracle/dbs/tbs_01.f
2           978 SYSTEM               YES     /oracle/dbs/tbs_02.f
3           978 TBS_1                NO      /oracle/dbs/tbs_11.f
4           978 TBS_1                NO      /oracle/dbs/tbs_12.f
5           978 TBS_2                NO      /oracle/dbs/tbs_21.f
6           978 TBS_2                NO      /oracle/dbs/tbs_22.f
```

This type of information is useful for incomplete recovery because you can determine the schema of the database for the time that you want to recover to.

> **See Also:** "report" on page 10-110 for **report** command syntax, and "list" on page 10-83 for **list** command syntax.

List and Report Scenarios

Following are some examples of list and report generation:

- Makings Lists of Backups and Copies
- Using Lists to Determine Obsolete Backups and Copies
- Reporting Datafiles Needing Backups
- Reporting Unrecoverable Datafiles
- Reporting Obsolete Backups and Copies
- Manually Deleting Obsolete Backups and Copies

Makings Lists of Backups and Copies

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 your lists.

The following example lists all backups of datafiles in tablespace TBS_1 that were made after November 1, 1999:

```
list backup of tablespace tbs_1 completed before 'Nov 1 1999 00:00:00';
```

The following example lists all backup sets or proxy copies on media management devices:

```
list backup of database device type 'sbt_tape';
```

The following example lists all copies of `datafile 2` using the tag `weekly_df2_ _copy` that are in the `copy` sub-directory:

```
list copy of datafile 2 tag weekly_df2_copy like '/copy/%';
```

Using Lists to Determine Obsolete Backups and Copies

Use the **list** command to determine which copies and backups can be deleted. For example, if you created a full backup of the database on November 2, and you know you will not need to recover the database to an earlier date, then the backups and image copies listed in the following report can be deleted:

```
list backup of database completed before 'Nov 1 1999 00:00:00';
list copy completed before 'Nov 1 1999 00:00:00';
```

Reporting Datafiles Needing Backups

The following command reports all datafiles in the database that require the application of three or more incremental backups to be recovered to their current state:

```
report need backup incremental 3 database;
```

The following command reports all datafiles from tablespace SYSTEM that have not had a full or incremental backup in five or more days:

```
report need backup days 5 tablespace system;
```

The following command reports which of datafiles 1 - 5 need backups because they do not have two or more backups or copies stored on tape:

```
report need backup redundancy 2 datafile 1,2,3,4,5 device type 'sbt_tape';
```

Reporting Unrecoverable Datafiles

The following example reports on all datafiles on tape that need a new backup because they contain unlogged changes that were made after the last full or incremental backup.

```
report unrecoverable database device type 'sbt_tape';
```

Reporting Obsolete Backups and Copies

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_tape';
```

The following command reports which datafiles are obsolete because they belong to a database incarnation that is not a direct predecessor of the current incarnation:

```
report obsolete orphan;
```

Manually Deleting Obsolete Backups and Copies

In this scenario, assume that you want to delete the following:

- Datafile copies for which there are at least two more recent copies.
- Datafile backups for which there are at least two more recent backups or image copies of each datafile contained in the backup set.

1. Generate a report with redundancy set to 2:

```
report obsolete redundancy 2;
```

```
RMAN-03022: compiling command: report
Report of obsolete backups and copies
Type                 Recid  Stamp     Filename
-------------------- ------ --------- -------------------------
Datafile Copy        23     345392880 /oracle/dbs/tbs_01.copy
Datafile Copy        22     345392456 /oracle/dbs/tbs_01_copy.f
Backup Set           31     345552065
Backup Piece         31     345552061 /oracle/dbs/0va9hd5o_1_1
Backup Set           23     345399397
Backup Piece         23     345399391 /oracle/dbs/0ma9co2p_1_1
Backup Set           20     345397468
Backup Piece         20     345397464 /oracle/dbs/0ka9cm6l_1_1
```

2. Issue **change ... delete** commands for the copies and backups to delete them and remove their repository records. Use the filenames or issue a **list** command to obtain the primary keys:

```
allocate channel for delete type disk;
change backuppiece '/oracle/dbs/0va9hd5o_1_1', '/oracle/dbs/0ma9co2p_1_1',
   '/oracle/dbs/0ka9cm6l_1_1' delete;
change datafilecopy '/oracle/dbs/tbs_01.copy', '/oracle/dbs/tbs_01_copy.f'
   delete;
release channel;
```

Deleting Obsolete Backups and Copies Using a UNIX Shell Script

Oracle provides the `$ORACLE_HOME/rdbms/demo/rman1.sh` UNIX script to automate deletion of obsolete backups and copies. The script uses `sed` and `grep` commands to process the output of the RMAN **report obsolete** command and constructs an RMAN command file containing the necessary **change ... delete** commands. This script does not require the use of a recovery catalog.

You can edit the **report obsolete** commands in the script as desired: the script uses the default value for **report obsolete** for both disk and tape devices. The output of the commands are stored in `deleted.log`.

1. Change into the `$ORACLE_HOME/rdbms/demo` directory and run the following shell script:

```
% rman1.sh
```

2. If desired, check `deleted.log` to see the command output.
3. If you use a recovery catalog, you can query the catalog to check that the records were updated correctly. For example, run the SQL*Plus script described in "Reporting Deleted Backups and Copies" on page 4-13 against the recovery catalog database to issue a report of all deleted backups and copies.

Generating Historical Reports of Database Schema

The following commands reports the database schema in the present, a week ago, two weeks ago, and a month ago:

```
report schema;
report schema at time 'SYSDATE-7';
report schema at time "TO_DATE('12/20/98','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 logseq 12 thread 2;
```

Listing Database Incarnations

Every time that you perform a RESETLOGS operation on a database, you create a new incarnation. This example lists all database incarnation of PROD1 registered in the recovery catalog:

```
list incarnation of database prod1;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         PROD1     1224038686  NO     1            02-JUL-98
1       582       PROD1     1224038686  YES    59727        10-JUL-98
```

See Also: "UNKNOWN Database Name Appears in Recovery Catalog" on page 9-35 for information about UNKNOWN database names in the **list output**.

Reporting Deleted Backups and Copies

The RMAN **report** command does not show deleted backups and copies. If compatibility is set to 8.1.6 or higher, then **change ... delete** commands automatically purge records from the RMAN repository. If compatibility is not set to 8.1.6, and records have not been purged using `prgrmanc.sql`, then you can perform a query against the V$ or recovery catalog tables to list these records.

If you use a recovery catalog, you can execute the following SQL script to display deleted backups and copies:

Note: If you do not use a recovery catalog, you can substitute the corresponding V$ view for the recovery catalog view and run the script against the target database. For example, substitute V$DATAFILE_COPY for RC_DATAFILE_COPY.

```
col bp_key format 999999
col bs_key format 999999
col backup_type format a4 heading "TYPE"
col piece# format 999999
col copy# format 99
col status format a6
col name format a40
col tag format a20
col thread# format 9999 heading "THRD#"
col sequence# format 9999 heading "SEQ#"
col handle format a30

ttitle 'Deleted Backup Pieces'
SELECT bp_key, bs_key, handle, backup_type, piece#, copy#
FROM rman.rc_backup_piece
WHERE status = 'D';

ttitle 'Deleted Datafile Copies'
SELECT cdf_key, name, tag
FROM rman.rc_datafile_copy
WHERE status = 'D';

ttitle 'Deleted Archived Redo Logs'
SELECT al_key, thread#, sequence#, name
FROM rman.rc_archived_log
WHERE status = 'D';

ttitle 'Deleted Control File Copies'
SELECT ccf_key, name, tag
FROM rman.rc_controlfile_copy
WHERE status = 'D';
ttitle off
```

The following shows sample output for the script:

```
Fri Jul 16                                                          page    1
                          Deleted Backup Pieces

 BP_KEY  BS_KEY HANDLE                         TYPE  PIECE# COPY#
------- ------- ------------------------------ ---- ------- -----
   1037    1035 /oracle/dbs/02b1h3ah_1_1  D          1     1
   1044    1042 /oracle/dbs/03b1h3b3_1_1  D          1     1
```

```
   1051    1049 /oracle/dbs/04b1h3bf_1_1  D          1     1
   1058    1056 /oracle/dbs/05b1h3bl_1_1  D          1     1


Fri Jul 16                                                                page    1
                            Deleted Datafile Copies

   CDF_KEY NAME                                    TAG
---------- ---------------------------------------- --------------------
      1069 /oracle/work/df1.f
      1073 /oracle/work/df2.f

Fri Jul 16                                                                page    1
                          Deleted Archived Redo Logs

    AL_KEY THRD#  SEQ# NAME
---------- ----- ----- ----------------------------------------
       972     1   947 /oracle/work/arc_dest/arcr_1_947.arc
       973     1   948 /oracle/work/arc_dest/arcr_1_948.arc
       974     1   949 /oracle/work/arc_dest/arcr_1_949.arc

Fri Jul 16                                                                page    1
                         Deleted Control File Copies

   CCF_KEY NAME                                    TAG
---------- ---------------------------------------- --------------------
      1066 /oracle/work/cf1.f
```

5

Making Backups and Copies with Recovery Manager

This chapter describes how to use Recovery Manager to manage backup and copy operations, and includes the following topics:

- Making Backups
- Making Image Copies
- Backup and Copy Scenarios

> **See Also:** "Monitoring RMAN Jobs" on page 9-12 to learn how to monitor backup and copy operations.

Making Backups

Perform backups of any of the following objects using the RMAN **backup** command:

- Database (all datafiles and current control file)
- Tablespace
- Datafile (current or image copy)
- Archived redo log
- Control file (current or image copy)

Although RMAN backs up and restores only datafiles, archived redo log files, and control files, the database depends on other files for production operation. You must define a backup and restore strategy for files that RMAN does not support. Examples include initialization files, password files, network configuration files, external procedure files, and external LOB files (BFILEs).

RMAN backs up the 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.

Note: Backups using the disk API are not supported (see "After Linking to the Media Manager on UNIX, RMAN Fails to Back Up to Tape" on page 9-19). To back up to disk, allocate a channel of **type disk**.

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; open backups are always inconsistent. Consistent backups can be restored without recovery. An inconsistent backup will require some media recovery when it is restored, but is otherwise just as valid as a consistent backup.

RMAN backup are further divided into *full* and *incremental* backups. Full backups are non-incremental, that is, every used block is backed up.

This section contains the following procedures:

- Making Consistent and Inconsistent Backups
- Making Whole Database Backups

- Backing Up Tablespaces and Datafiles
- Backing Up Control Files
- Backing Up Archived Redo Logs
- Making Incremental Backups
- Making Split Mirror Backups

See Also:

- "Backup Sets" on page 1-32 for an overview of RMAN backups
- "backup" on page 22 for a complete description of **backup** command syntax
- "Backup Types" on page 1-48 for a discussion of the various RMAN backup types

Making Consistent and Inconsistent Backups

The procedures in this chapter allow you to make backups when the database is open or closed. Closed backups are either consistent or inconsistent; open backups are always inconsistent. Note that Oracle does not permit inconsistent backups in NOARCHIVELOG mode.

To make a consistent backup, the database:

- Must be mounted, but not open.
- Must *not* have crashed or aborted the last time it was open.

If these conditions are not met, the backup will be inconsistent.

Making Whole Database Backups

If you can afford to close your database, Oracle recommends taking closed, consistent backups of your whole database. If you cannot shut down your database, then your only option is to make an open backup.

To make a whole database backup:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   To write the output to a log file, specify the file at startup. For example, enter:

```
% rman target / catalog rman/rman@rcat log /oracle/log/mlog.f
```

2. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. If the database is open, or is mounted but *not* closed cleanly when last opened, the backup will be inconsistent.

3. Allocate one or more channels of type **disk** or type *'sbt_tape'*. This example backs up all the datafiles as well as the control file. 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):

   ```
   run {
        allocate channel ch1 type disk;
        backup database;
        sql 'ALTER SYSTEM ARCHIVE LOG CURRENT'; # archives current redo log as well as
                                                # all unarchived logs
   }
   ```

   Optionally, use the **format** parameter to specify a filename for the backup piece. For example, enter:

   ```
   run {
        allocate channel ch1 type 'sbt_tape';
        backup database
        format '/oracle/backup/%U';  # %U generates a unique filename
   }
   ```

   Optionally, use the **tag** parameter to specify a tag for the backup. For example, enter:

   ```
   run {
        allocate channel ch1 type 'sbt_tape';
        backup database
        tag = 'weekly_backup';   # gives the backup a tag identifier
   }
   ```

4. Issue a **list** command to see a listing of your backup sets and pieces.

Backing Up Tablespaces and Datafiles

Back up tablespaces and datafiles 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. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a message log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. If the database is open, or is mounted but *not* closed cleanly when last opened, the backup will be inconsistent.

3. Allocate one or more channels before issuing the **backup** command. This example backs up three tablespaces, using the **filesperset** parameter to specify that no more than three datafiles should go in each backup set, and also backs up the control file:

   ```
   run {
        allocate channel ch1 type disk;
        allocate channel ch2 type disk;
        allocate channel ch3 type disk;
        backup filesperset = 3
          tablespace inventory, sales
          include current controlfile;
   }
   ```

4. Issue a **list** command to see a listing of your backup sets and pieces.

To back up a datafile:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a message log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log '/oracle/log/mlog.f'
   ```

2. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. If the database is open, or is

mounted but *not* closed cleanly when last opened, the backup will be inconsistent.

3. Allocate one or more channels before issuing the **backup** command. This example backs up datafiles 1-6 as well as an image copy of a datafile:

```
run {
     allocate channel ch1 type disk;
     backup
       (datafile 1,2,3,4,5,6
       filesperset 3)
       datafilecopy '/oracle/copy/tbs_1_c.f';
}
```

4. Issue a **list** command to see a listing of your backup sets and pieces.

To back up a datafile copy:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

   If you want to write the output to a message log file, specify the file at startup. For example, enter:

```
% rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
```

2. Mount or open the database.
3. Allocate one or more channels before issuing the **backup** command. This example backs up datafile copy `df1.copy` to tape:

```
run {
     allocate channel ch1 type 'sbt_tape';
     backup datafilecopy '/oracle/copy/df1.copy';
}
```

4. Issue a **list** command to see a listing of your backup sets and pieces.

Backing Up Control Files

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.

Whole database backups automatically include the current control file, but the current control file does not contain a record of the whole database backup. To

obtain a control file backup with a record of the whole database backup, make a backup of the control file after executing the whole database backup.

Include a backup of the control file within any backup by specifying the **include current controlfile** option.

To back up the current control file:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. Mount or open the database.
3. Allocate a channel before issuing the **backup** command. This example backs up the current control file to tape and uses a tag:

   ```
   run {
        allocate channel ch1 type 'sbt_tape';
        backup current controlfile
        tag = mondayPMbackup;
   }
   ```

4. Optionally, issue a **list** command to see a listing of your backup sets and pieces.

To back up a control file copy:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. Mount or open the database.
3. Allocate a channel before issuing the **backup** command. This example backs up the control file copy `'/oracle/copy/cf.f'`:

   ```
   run {
   ```

```
    allocate channel ch1 type disk;
    backup controlfilecopy '/oracle/copy/cf.f';
}
```

4. Optionally, issue a **list** command to see a listing of your backup sets and pieces.

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 FOO and includes the current control file in the backup:

```
run {
     allocate channel c1 type disk;
     backup tablespace foo
       include current controlfile;
}
```

Backing Up Archived Redo Logs

The archived redo logs are the key to successful recovery. Back them up regularly. To back up archived logs, allocate one or more channels and issue **backup archivelog** with the desired filtering options:

```
run {
     allocate channel t1 type 'sbt_tape';
     backup archivelog all
       delete input;
}
```

> **See Also:** "Backing Up in an OPS Environment" on page 5-9 to learn about special considerations when backing up in OPS mode.

Note 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,

If you specify the **delete input** option, then RMAN only deletes the specific copy of the archived redo log that it backs up. Note that you can specify the **delete input** option in the **backup** command, which deletes the archived redo logs after you have backed them up. For example, issue:

```
run {
     allocate channel t1 type 'sbt_tape';
     backup archivelog all
       delete input;
}
```

Thus, you can back up archived logs to tape and clear your disk space of old logs in one step.

> **See Also:** "Backing Up and Deleting Multiple Copies of an Archived Redo Log" on page 5-21 for a scenario involving a backup of logs located in multiple destinations.

To back up archived redo logs:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. Mount or open the database.

3. Allocate a channel before issuing the **backup** command. This example backs up all of the archived redo log files to tape and deletes the input logs from disk:

   ```
   run {
        allocate channel ch1 type 'sbt_tape';
        backup archivelog all         # Backs up all archived redo logs.
          delete input;               # Optionally, delete the input logs
   }
   ```

   You can also specify a range of archived redo logs by time, SCN, or log sequence number. This example backs up all archived logs created more than 7 and less than 30 days ago:

   ```
   run {
        allocate channel ch1 type disk;
        backup archivelog
          from time 'SYSDATE-30' until time 'SYSDATE-7';
   }
   ```

4. Issue a **list** command to see a listing of your backup sets and pieces.

Backing Up in an OPS Environment

Backing up archived redo logs in an OPS environment poses special problems. To illustrate, assume the following:

- Node 1 archives to its local file system
- Node 2 archives to its local file system
- Node 3 archives to its local file system

If you allocate channels and then issue **backup archivelog all**, the default channel attempts to validate all the logs on only one node, causing the job to fail. Solve this problem by specifying which channel should back up which logs.

To back up archived redo logs in an OPS environment:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   If you want to write the output to a log file, specify the file at startup. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat log "/oracle/log/mlog.f'
   ```

2. Mount or open the database.
3. Perform the following tasks within the **run** command:
   - Allocate a channel for each node of the OPS cluster.
   - Force each channel to back up only those logs to which it has access by using the **like** *pathname* parameter.

   This example connects to three different nodes and informs each channel that it should only back up those logs contained in the directory to which it has access (where *node1_arc_dest* refers to a directory containing the logs on node 1, *node2_arc_dest* refers to a directory containing the logs on node 2, etc.):

   ```
   run {
        allocate channel node1 type 'sbt_tape' connect '@node1';
        allocate channel node2 type 'sbt_tape' connect '@node2';
        allocate channel node3 type 'sbt_tape' connect '@node3';

        backup (archivelog like '/node1_arc_dest%' channel node 1)
               (archivelog like '/node2_arc_dest%' channel node 2)
               (archivelog like '/node3_arc_dest%' channel node 3);
   }
   ```

See Also: "backup" on page 10-22 for **backup** syntax and a description of the **like** parameter, and "Backing Up in a Parallel Server Environment" on page 5-26 for a scenario involving backing up archived logs in an OPS environment.

Making Incremental Backups

You can make consistent or inconsistent incremental backups of the database or individual tablespaces or datafiles. This procedure makes an incremental backup of a database that was shut down cleanly.

To make a consistent, incremental backup:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. If the database is open, or is mounted but *not* closed cleanly when last opened, the backup will be inconsistent.

3. Allocate a channel before issuing the **backup** command. This example makes a level 0 backup of the database:

   ```
   run {
        allocate channel ch1 type disk;
        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:

   ```
   run {
        allocate channel ch1 type disk;
        backup
          incremental level = 1
          tablespace system
          datafile '/oracle/dbs/sales.f';
   }
   ```

This example makes a cumulative level 2 backup of the tablespace TBS_2. It will only back up those data blocks changed since the most recent level 1 or level 0 backup:

```
run {
     allocate channel ch1 type disk;
     backup
       incremental level = 2 cumulative  # specify cumulative option
       tablespace tbs_1;
}
```

4. Optionally, issue a **list** command to see a listing of your backup sets and pieces.

Making Split Mirror Backups

Many sites keep an on-disk backup of the database in case a failure occurs on the primary database or an incorrect user action such as a drop table requires incomplete recovery. An on-disk datafile backup simplifies the restore step of recovery, making recovery much quicker and more reliable.

Note: Never make backups, split mirror or otherwise, of online redo logs. Also, it is best to use the **backup controlfile** command rather than a split mirror to make control file backups.

A common way of creating an on-disk datafile backup 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 once a day to use as a backup.

RMAN does not automate the splitting of the mirrors, but can make use of the 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.

To make a split mirror backup of a tablespace:

1. Place the tablespaces that you want to back up into hot backup mode through ALTER TABLESPACE ... BEGIN BACKUP. For example, to place tablespace TBS_3 in hot backup mode enter:

   ```
   ALTER TABLESPACE tbs_3 BEGIN BACKUP;
   ```

2. Split the mirrors for the underlying datafiles contained in these tablespaces. See *Oracle8i Backup and Recovery Guide* for more information about splitting mirrors.

3. Take the tablespaces out of hot backup mode:

```
ALTER TABLESPACE tbs_3 END BACKUP;
```

4. Catalog the mirror copies as datafile copies using the **catalog** command. For example, enter:

```
catalog datafilecopy '/disk2/dbs/tbs_3.f';  # catalog split mirror
```

5. Back up these datafile copies using the rman **backup** command. For example, enter:

```
run {
     allocate channel c1 type disk;
     backup datafilecopy '/disk2/dbs/tbs_3.f';
}
```

6. When you are ready to resilver the split mirror, first use the **change ... delete** command to uncatalog the datafile copies you cataloged in step 4. For example, enter:

```
allocate channel for delete type disk;
change datafilecopy '/disk2/dbs/tbs_3.f' delete;
```

7. Resilver the split mirror for the affected datafiles.

Making Image Copies

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)

To make consistent copies of all database files:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. If the database is open, or is mounted but *not* closed cleanly when last opened, the backup will be inconsistent.
3. Generate a report of the current database schema:

```
RMAN> report schema;

RMAN-03022: compiling command: report
Report of database schema
File K-bytes    Tablespace           RB segs Name
---- ---------- -------------------- ------- ------------------
1         35840 SYSTEM               YES     /oracle/dbs/tbs_01.f
2           978 SYSTEM               YES     /oracle/dbs/tbs_02.f
3           978 TBS_1                NO      /oracle/dbs/tbs_11.f
4           978 TBS_1                NO      /oracle/dbs/tbs_12.f
```

4. Copy all of the datafiles and include the current control file. For example, enter:

```
run {
     allocate channel ch1 type disk;
     copy
       datafile 1 to '/oracle/copy/df_1.f',
       datafile 2 to '/oracle/copy/df_2.f',
       datafile 3 to '/oracle/copy/df_3.f',
       datafile 4 to '/oracle/copy/df_4.f',
       current controlfile to '/oracle/copy/cf.f';
}
```

5. Issue a **list copy** command to see a listing of your copies.

To copy datafiles, archived redo logs, and control files:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. For a consistent backup, mount but do not open the database and ensure that the database was closed cleanly prior to mounting. If the database is open, or is mounted but *not* closed cleanly when last opened, the backup will be inconsistent.
3. Copy the desired datafiles, archived redo logs, and control files. For example, enter:

```
run {
     allocate channel ch1 type disk;
     # allocate multiple channels for parallelization. For parallelization, issue one
     # copy command rather than multiple copy commands.
     allocate channel ch2 type disk;
     allocate channel ch3 type disk;
     copy
       # copy datafiles and datafile copies
       datafile '/oracle/dbs/tbs_8.f' to '/oracle/copy/df_8.f',
       datafilecopy '/oracle/copy/df_2.cp' to '/oracle/dontouch/df_2.f',
       datafilecopy tag = 'weekly_df1_copy' to '/oracle/copy/df_1.f',

       # copy archived redo logs
       archivelog '/oracle/arc_dest/arcr_1_1.arc' to '/oracle/copy/arcr_1_1.arc',
       archivelog '/oracle/arc_dest/arcr_1_2.arc' to '/oracle/copy/arcr_1_2.arc',

       # copy a control file copy
       controlfilecopy '/oracle/copy/cf.f' to '/oracle/dontouch/cf.f';
}
```

4. Issue a **list copy** command to see a listing of your copies.

Backup and Copy Scenarios

Following are some useful scenarios for performing backups and copies:

- Reporting Datafiles Needing Backups
- Skipping Files when Backing Up a Database
- Spreading a Backup Across Multiple Disk Drives
- Backing Up a Large Database to Multiple Filesystems
- Specifying the Size of Backup Sets
- Specifying the Size of Backup Pieces
- Multiplexing Datafiles in a Backup

Reporting Datafiles Needing Backups

The following command reports all datafiles in the database that would require the application of 6 or more incremental backups to be recovered to their current state:

```
report need backup incremental 6 database;
```

The following command reports all datafiles from tablespace SYSTEM that have not had a backup (full or incremental) in five or more days:

```
report need backup days 5 tablespace system;
```

Skipping Files when Backing Up a Database

The following example, which assumes that the database is running in ARCHIVELOG mode, shows a common way to back up the database (skipping tablespaces that are offline):

```
run {
     allocate channel dev1 type 'sbt_tape';
     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.

Spreading a Backup Across Multiple Disk Drives

Typically, you do not need to specify a format when backing up to tape because the default **%U** conversion variable generates a unique filename for all tape backups. When backing up to disk, however, you can specify a format if you want to spread your backup across several disk drives for improved performance. In this case, allocate one **disk** channel per disk drive and specify the format string on the **allocate channel** command. Specify the format so that the filenames are on different disks.

For example, issue:

```
run {
     allocate channel disk1 type disk format '/disk1/%d_backups/%U';
     allocate channel disk2 type disk format '/disk2/%d_backups/%U';
     allocate channel disk3 type disk format '/disk3/%d_backups/%U';
     backup database;
}
```

Backing Up a Large Database to Multiple Filesystems

In this scenario, you have a 40Gb database that you want to back up to disk. Because RMAN does not back up to raw disk, you must spread the backup across filesystems. You decide to back up to 4 filesystems and make each backup set roughly the same size: 10Gb. You want to make each backup piece no more than 2Gb so that each backup set contains 5 backup pieces.

You decide to use the **format** parameter of the **allocate channel** command so that each channel will write to a different filesystem. You use conversion variables to guarantee unique names for the backup pieces. For example, the following RMAN script spreads the backup across four filesystems (`/fs1`, `/fs2`, `/fs3`, `/fs4`), creating 4 backup sets in these directories and grouping the datafiles so that each backup set is about the same size.

```
run {
     allocate channel fs1 type disk format='/fs1/%u.%p';
     allocate channel fs2 type disk format='/fs2/%u.%p';
     allocate channel fs3 type disk format='/fs3/%u.%p';
     allocate channel fs4 type disk format='/fs4/%u.%p';

     set limit channel fs1 kbytes=2000000;  #limit file size to 2Gb
     set limit channel fs2 kbytes=2000000;
     set limit channel fs3 kbytes=2000000;
```

```
     set limit channel fs4 kbytes=2000000;

     backup database;
}
```

Specifying the Size of Backup Sets

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 **setsize** parameters to override this calculation and specify how many files should go in each backup set.

Using filesperset

When you specify the **filesperset** parameter, RMAN compares the **filesperset** value to the automatically calculated value (number of files / allocated channels) and takes the lowest of the two values, thereby ensuring that all channels are used. For example, if you are backing up 12 datafiles with 3 channels, and set **filesperset** = 2, RMAN puts 2 datafiles into each backup set rather than 4.

If the number of files specified or implied by the combined *backupSpec* clauses is greater than **filesperset**, for example, 8 total files need backing up when **filesperset** = 4, then RMAN creates multiple backup sets to maintain the correct ratio of files per backup set.

If you do not specify **filesperset**, RMAN compares the calculated value (number of files / allocated channels) to the default value of 64 and takes the lowest 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 **setsize** 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 and **filesperset** = 1, then two channels are necessarily idle.

This example parallelizes a backup, specifying that no more than 3 datafiles go in any one backup set, and no more than 1 archived redo log in any one backup set:

```
run {
     allocate channel ch1 type disk;
     allocate channel ch2 type disk;
     allocate channel ch3 type disk;
     allocate channel ch4 type disk;
     backup
       datafile 1,2,3,4,5,9,10,11,12,15
```

```
        filesperset = 3
      archivelog all
        filesperset = 1;
}
```

Using setsize

The **setsize** parameter specifies a maximum size for a backup set in units of 1K (1024 bytes). Thus, to limit a backup set to 2Mb, specify **setsize** = 2000. RMAN attempts to limit all backup sets to this size, which is useful in media manager configurations when you want each backup set to be no larger than one tape.

Configure your backup sets so that they fit on one tape volume rather than span multiple tape volumes. Otherwise, if one tape of a multi-volume backup set fails, then you lose the whole backup set.

The **setsize** parameter is easier to use than **filesperset** when you make backups of archived redo logs. This example backs up archived redo logs to tape, setting the size at 2Mb to ensure that each backup set fits on one tape volume:

```
run {
     allocate channel ch1 type 'sbt_tape';
     allocate channel ch2 type 'sbt_tape';
     backup setsize = 2000
       archivelog all;
}
```

Specifying the Size of Backup Pieces

Backup piece size is an issue in those situations where it exceeds the maximum file size of the file system or media management device. Use the kbytes parameter of the set channel limit command to limit the size of backup pieces.

For example, if your media manager only support files sized 8Mb or less, you can issue the following command when making tape backups of your database:

```
run {
     allocate channel c1 type 'sbt_tape';
     set limit channel kbytes = 8000;
     backup database;
}
```

Multiplexing Datafiles in a Backup

Assume you want to back up a database called FOO. The following conditions obtain:

- You have three tape drives available for the backup.
- The database contains 26 datafiles.
- You want to multiplex datafiles into the backup, placing 4 datafiles into each backup set.

You set **filesperset** equal to 4 because it is sufficient to keep the tape drive streaming. In this example, you are not concerned about how datafiles are grouped into backup sets.

Issue the following commands:

```
create script foo_full {
   allocate channel t1 type 'SBT_TAPE';
   allocate channel t2 type 'SBT_TAPE';
   allocate channel t3 type 'SBT_TAPE';
   backup filesperset 4
   database format 'FOO.FULL.%n.%s.%p';
}
```

This script backs up the whole database, including all datafiles and the control file. Because there are 27 files to be backed up (26 datafiles and a control file) and a maximum of 4 files per backup set, Oracle creates seven backup sets. 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 backup piece number:

```
FOO.FULL.db_name.set_num.piece_num
# for example, a file may have the following name:
FOO.FULL.prod1.3.1
```

Assuming no backup sets have been recorded in the recovery catalog prior to this job, then *set_num* will range from 1 through 7 and *piece_num* will be 1 or more.

Note that in the SBT API version 2.0, media vendors can specify the maximum size of a backup piece, causing RMAN to comply with this restriction automatically.

Backing Up Archived Redo Logs

You can also back up archived redo logs to tape. You can specify the range of archived redo logs by time, SCN, or log sequence number.

> **Note:** If specifying by time range, set the NLS_LANG and NLS_DATE_FORMAT environment variables before invoking Recovery Manager. See "Setting NLS Environment Variables" on page 2-2.

Specifying an archived log range does not guarantee that RMAN backs up all redo in the range. For example, the last archived log may end before the end of the range, or a log in the range may be missing. RMAN backs up the logs it finds and does not issue a warning. Online logs cannot be backed up; you must first archive them.

This example backs up the logs archived between 8:57 p.m. and 9:06 p.m. on November 18, 1998.

```
NLS_LANG=american
NLS_DATE_FORMAT='Mon DD YYYY HH24:MI:SS'
run {
     allocate channel dev1 type 'sbt_tape';
     backup
     archivelog all
       from time  'Nov 13 1998 20:57:13'
       until time 'Nov 13 1998 21:06:05';
}
```

The following example backs up all archived logs from sequence# 288 to sequence# 301 and deletes the archived redo logs after the backup is complete. If the backup fails the logs are not deleted.

```
run {
     allocate channel dev1 type 'sbt_tape';
     backup
       archivelog from logseq 288 until logseq 301 thread 1
       delete input;
}
```

The following command backs up all archived logs generated during the last 24 hours. The example archives the current redo log first to ensure that all redo generated up to the present gets backed up.

```
run {
     allocate channel dev1 type 'sbt_tape';
     sql "ALTER SYSTEM ARCHIVE LOG CURRENT";
     backup archivelog from time 'SYSDATE-1';
}
```

> **See Also:** Your operating system-specific documentation for more information about your environment variables.

Backing Up and Deleting Multiple Copies of an Archived Redo Log

In this scenario, you set your initialization parameters so that you automatically archive your redo logs to two locations: `/oracle/arch/dest_1/*` and `/oracle/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 your archived redo logs and then delete the originals.

Because the **backup archivelog all** command backs up exactly one copy of each distinct log sequence number, RMAN does not put two copies of the same log sequence number into the same backup set. Furthermore, if you specify the **delete input** option, RMAN only deletes the specific copy of the archived redo log that it backs up.

The easiest solution in this case is to back up both copies of each archived redo log and then delete both copies. Use the **like** parameter of the *archivelogRecordSpecifier* to indicate which destination to use. The **like** parameter allows you to match file filenames in both of your archive destinations. For example, execute the following:

```
run {
     allocate channel t1 type 'sbt_tape';
     allocate channel t2 type 'sbt_tape';
     backup
       filesperset=20
       format='al_%d/%t/%s/%p'
       (archivelog like '/oracle/arch/dest1/%' channel t1 delete input)
       (archivelog like '/oracle/arch/dest2/%' channel t2 delete input);
}
```

This example backs up the archived redo logs in the primary destination on one set of tapes and the logs from the second destination on another set of tapes. This scenario assumes that you have two tape drives available. Note that some tape subsystems will combine the two RMAN channels into a single data stream and write them to a single tape drive. You may need to configure your media management vendor to prevent this scenario from occurring.

Performing Differential Incremental Backups

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.

```
run {
     allocate channel dev1 type 'sbt_tape';
     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, RMAN copies all blocks changed since the base level 0 backup.

```
run {
     allocate channel dev1 type 'sbt_tape';
     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 will fail because Recovery Manager does not find a parent backup for the new datafiles.

```
run {
     allocate channel dev1 type 'sbt_tape';
     backup incremental level 0
       tablespace new_tbs;
}
```

Note that you can perform incremental backups in NOARCHIVELOG mode.

Performing Cumulative Incremental Backups

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.

```
run {
     allocate channel dev1 type 'sbt_tape';
     backup incremental level 2 cumulative  # blocks changed since level 0 or level 1
       database;
}
```

Duplexing Backup Sets

Prudence suggests making multiple copies of your backups to protect against disaster, media damage, or human error. Oracle allows you to make up to four backup sets simultaneously, each an exact duplicate of the others.

The **set duplex** command, which affects only the **backup** command, specifies the number of copies of each backup piece that RMAN should create. The **set duplex** command affects all channels allocated after issuing the command and is in effect until explicitly disabled (**OFF**) or changed during the session.

> **Note:** The **set duplex** command will generate an error if there are previously allocated channels.

For example, you can enter:

```
run {
     set duplex=3;
     allocate channel ch1 type 'sbt_tape';
     backup datafile 1;
}
```

This command will create three identical backups of `datafile 1`. Each backup piece will have a unique name since the **%U** default format guarantees it.

Note that you must set the BACKUP_TAPE_IO_SLAVES initialization parameter to TRUE in order to perform duplexed backups; otherwise, an error will be signaled. RMAN will configure as many slaves as needed for the number of backup copies you request.

> **See Also:** *Oracle8i Reference* for more information on the BACKUP_TAPE_IO_SLAVES initialization parameter.

Determining How Channels Distribute a Backup Workload

If you want to create multiple backup sets and allocate multiple channels, then RMAN automatically parallelizes its operation and writes multiple backup sets in parallel. The allocated server sessions divide up 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 specify that Oracle should write all backup sets for a *backupSpec* to a specific channel using the **channel** parameter.

For example, this example parallelizes the backup operation by specifying which channels RMAN should allocate for which operations:

```
run {
     allocate channel ch1 type 'SBT_TAPE';
```

```
     allocate channel ch2 type disk;
     allocate channel ch3 type disk;
     backup
       # channel ch1 backs up datafiles to tape
       (datafile 1,2,3,4
       channel ch1)
       # channel ch2 backs up control file copy to disk
       (controlfilecopy '/oracle/copy/cf.f'
       channel ch2)
       # channel ch3 backs up archived redo logs to disk
       (archivelog from time 'SYSDATE-14'
       channel ch3);
}
```

Backing Up in NOARCHIVELOG Mode

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 performing a duplexed 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 crashed or was not shutdown cleanly prior to
# starting this script. This will perform a 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;

# Here, we know that the database is cleanly closed and is now ready for a cold
# backup. RMAN requires that the database be started and mounted to perform a backup,
# so do that now.

startup mount;
run {
        # duplex the backup
        set duplex = 2;

     #  allocate channel t1 type 'SBT_TAPE';
     #  allocate channel t2 type 'SBT_TAPE';
        allocate channel t1 type disk;
```

```
        allocate channel t2 type disk;

        set limit channel t1 kbytes 2097150;
        set limit channel t2 kbytes 2097150;

        backup
          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 a Parallel Server Environment

The following script distributes datafile and archived redo log backups across two nodes in a parallel server environment:

```
run {
     allocate channel node_1 type disk connect 'sys/sys_pwd@node_1';
     allocate channel node_2 type disk connect 'sys/sys_pwd@node_2';
        backup filesperset 1
          (tablespace system, rbs, data1, data2
           channel node_1)
          (tablespace temp, reccat, data3, data4
           channel node_2);
        backup filesperset 20
          (archivelog
           until time 'SYSDATE'
           like "/node1/arc/%"
           delete input
           channel node_1);
          (archivelog
           until time 'SYSDATE'
           like "/node2/arc/%"
           delete input
           channel node_2);
}
```

See Also: *Oracle8i Parallel Server Documentation Set: Oracle8i Parallel Server Concepts; Oracle8i Parallel Server Setup and Configuration Guide; Oracle8i Parallel Server Administration, Deployment, and Performance* for more information about OPS backups.

Cataloging Operating System Copies

You can use operating system utilities to make datafile copies and then catalog them in the recovery catalog. Note that you can only catalog copies made to disk. Because the format of backup pieces is proprietary, operating system utilities cannot write backups that Recovery Manager can read.

You must make the datafile copies using operating system methods. If the database is open and the datafile is online, first issue ALTER TABLESPACE ... BEGIN BACKUP. The resulting image copy can be cataloged:

```
catalog datafilecopy '?/dbs/tbs_33.f';
```

Maintaining Backups and Copies

How long you must keep backups and copies depends on factors such as:

- How frequently you take backups.
- How far in the past point-in-time recovery is needed.

For example, if you back up all datafiles daily, do not require point-in-time recovery, and need only one backup per datafile, then you can delete previous backups as soon as the new one completes.

```
# delete a specific datafile copy
change datafilecopy '?/dbs/tbs_35.f' delete;

# delete archived redo logs older than 31 days
change archivelog until time 'SYSDATE-31' delete';
```

You must allocate a channel before deleting a backup piece. The specified backup piece must have been created on the same type of device. Note that the **allocate channel for maintenance** command is *not* issued inside of a **run** command.

```
# delete a backup piece
allocate channel for maintenance type 'sbt_tape';
change backuppiece 'testdb_87fa39e0' delete;
release channel;
```

Handling Errors During Backups and Copies

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, the block is marked corrupt in the backup.

The **set maxcorrupt for datafile** command determines how many corrupt blocks in a datafile that **backup** or **copy** will tolerate. If any datafile has more corrupt blocks than specified by the **maxcorrupt** parameter, the command aborts. Note that 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 various parameters to prevent termination:

If you specify...	Then RMAN skips...
The **skip inaccessible** option	Inaccessible datafiles. Note that a datafile is only considered inaccessible if it cannot be read. Some offline datafiles can still be read because they still exist on disk. Others have been deleted or moved and so cannot be read, making them inaccessible.
The **skip offline** option	Offline datafiles.
The **skip readonly** option	Datafiles with read-only status.

```
run {
     allocate channel dev1 type 'sbt_tape';
     set maxcorrupt for datafile 1,2,3 to 5;
     backup database
       skip inaccessible
       skip readonly
       skip offline;
}
```

6

Restoring and Recovering with Recovery Manager

This chapter describes how to use Recovery Manager to perform restore and recovery operations, and includes the following topics:

- Restoring Datafiles, Control Files, and Archived Redo Logs
- Recovering Datafiles
- Restore and Recovery Scenarios

> **See Also:** "Monitoring RMAN Jobs" on page 9-12 to learn how to monitor restore and recovery operations.

Restoring Datafiles, Control Files, and Archived Redo Logs

Use the RMAN **restore** command to restore datafiles, control files, or archived redo logs from backup sets or image copies. RMAN restores backups from disk or tape, but image copies only from disk.

When restoring files, you should:

- Allocate at least one channel before restoring backups or copies. Specify multiple channels to parallelize the restore operation.
- Allocate a channel of type **disk** if you use the **from copy** option.
- Allocate the appropriate **disk** or '*sbt_tape*' channel when restoring files. If the appropriate type of device is not allocated, then you may not be able to find a candidate backup set or copy to restore, and the **restore** command fails.

Restore files to either:

- The default location, which overwrites the files with the same name.
- A new location specified by the **set newname** command. If you restore datafiles to a new location, then RMAN considers them datafile copies and records them in the repository. Unless you issue a **switch** command to point the control file to these copies, they are considered valid copies for use in future restore operations.

This section contains the following topics:

> **See Also:** "restore" on page 10-120 for **restore** syntax, and "set_run_option" on page 10-142 for **set newname** syntax.

Restoring a Database

When restoring a target database, you can:

- Restore the database to its default location because of a media failure.
- Move the database to a new host because of a media failure.

- Test restore and recovery procedures by creating a database based on backups of your target database.

To restore the database to its default location, issue the **restore database** command. To move your target database to a new host, rename the datafiles as needed using **set newname**. To create a test database using backups of your target database, use the **duplicate** command (see Chapter 7, "Creating a Duplicate Database with Recovery Manager" for complete instructions).

This chapter contains the following topics:

Restoring the Database to its Default Location

If you do not specify **set newname** commands for the datafiles during a restore job, the database must be closed or the datafiles must be offline. Otherwise, you see output similar to the following, which results from an attempt to restore `datafile 3` while the file is online:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure during compilation of command
RMAN-03013: command type: restore
RMAN-03006: non-retryable error occurred during execution of command: IRESTORE
RMAN-07004: unhandled exception during command execution on channel ch1
RMAN-10035: exception raised in RPC: ORA-19573: cannot obtain exclusive enqueue
for datafile 3
ORA-19600: input file is datafile-copy 102 (/vobs/oracle/dbs/df.3)
ORA-19601: output file is datafile 3 (/vobs/oracle/dbs/tbs_11.f)
RMAN-10031: ORA-19573 occurred during call to DBMS_BACKUP_RESTORE.COPYDATAFILECOPY
```

The database must be closed when you restore the whole database. If the target database is mounted, then its control file is updated with any applicable datafile copy and archived log records to describe the restored files.

To restore the database to its default location:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. If the database is open, shut it down and then mount it:

```
shutdown immediate;
startup mount;
```

3. After allocating channels, restore the database:

```
run {
     allocate channel ch1 type disk;
     allocate channel ch2 type disk;
     allocate channel ch3 type disk;
     restore database;
}
```

Moving the Target Database to a New Host with the Same File System

A media failure may force you to move a database by restoring a backup from one host to another. You can perform this procedure so long as you have a valid backup and a recovery catalog or control file.

Because your restored database will not have the online redo logs of your production database, you will need to perform incomplete recovery up to the lowest SCN of the most recently archived redo log in each thread and then open the database with the RESETLOGS option.

This scenario assumes that:

- You are restoring backups from HOST_A onto HOST_B.
- You are restoring from tape backups.
- The HOST_A database has the same DB_NAME as the HOST_B database.
- The HOST_B filenames and directory paths are the same as in HOST_A.

The restore procedure differs depending on whether the target database uses a recovery catalog.

Note: You cannot use RMAN to restore image copies created on one host to a different host. Nevertheless, you can transfer the files using an operating system utility. If the files are located in the same location in the new host, then you do not need to recatalog them. If you transfer them to new location, then use the **catalog** command to update the RMAN repository with the new locations and use the **change ... uncatalog** command to uncatalog the old locations.

See Also: "Restoring Datafile Copies to a New Host" on page 6-34.

To restore the database from HOST_A to HOST_B with a recovery catalog:

1. Copy the initialization parameter file for HOST_A to HOST_B using an operating system utility.
2. Connect to the HOST_B target instance and HOST_A recovery catalog. For example, enter:

```
% rman target sys/change_on_install@host_b catalog rman/rman@rcat
```

3. Start the instance without mounting it:

```
startup nomount;
```

4. Restore and mount the control file. Execute a **run** command with the following sub-commands:
   a. Allocate at least one channel.
   b. Restore the control file.
   c. Mount the control file.

```
run {
     allocate channel ch1 type disk;
     restore controlfile;
     alter database mount;
}
```

5. Because there may be multiple threads of redo, use change-based recovery. Obtain the SCN for recovery termination by finding the lowest SCN among the most recent archived redo 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#);
```

6. Execute a **run** command with the following sub-commands:
   a. Set the SCN for recovery termination using the value obtained from the previous step.
   b. Allocate at least one channel.

c. Restore the database.

d. Recover the database.

e. Open the database with the RESETLOGS option.

```
run {
     set until scn = 500;  # use appropriate SCN for incomplete recovery
     allocate channel ch1 type 'sbt_tape';
     restore database;
     recover database;
     alter database open resetlogs;
}
```

To restore from HOST_A to HOST_B without a recovery catalog:

1. Copy the initialization parameter file for HOST_A to HOST_B using an operating system utility.
2. Use an operating system utility to make an image copy of the HOST_A control file and transfer it to HOST_B using an operating system utility.
3. Connect to the HOST_B target instance with the **nocatalog** option. For example, enter:

   ```
   % rman target sys/change_on_install@host_b nocatalog
   ```

4. Mount the database:

   ```
   startup mount;
   ```

5. Because there may be multiple threads of redo, use change-based recovery. Obtain the SCN for recovery termination by finding the lowest SCN among the most recent archived redo 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#);
   ```

6. Execute a **run** command with the following sub-commands:

   a. Set the SCN for recovery termination using the value obtained from the previous step.

   b. Allocate at least one channel.

   c. Restore the database.

d. Recover the database.

e. Open the database with the RESETLOGS option.

```
run {
     set until scn 500;  # use appropriate SCN for incomplete recovery
     allocate channel ch1 type 'sbt_tape';
     alter database mount;
     restore database;
     recover database;
     alter database open resetlogs;
}
```

Moving the Target Database to a New Host with a Different File System

The procedure for moving the database to a machine with a different file system is basically the same as described in "Moving the Target Database to a New Host with the Same File System" on page 6-4; the difference is that you need to rename each datafile using **set newname**.

For example, assume that:

- The database on HOST_A has ten datafiles.
- You are restoring from tape backups.
- You restore some datafiles to `/disk1` and others to `/disk2` on HOST_B.

To restore to HOST_B with a recovery catalog:

1. Follow the procedure in "Moving the Target Database to a New Host with the Same File System" on page 6-4 (with a recovery catalog), stopping before you execute the **run** command. Make sure to reset all *_DEST and *_PATH parameters in the initialization parameter file that specify a pathname.
2. Execute this **run** command instead:
   a. Set the end SCN obtained from the SQL*Plus query.
   b. Allocate at least one channel.
   c. Specify a new filename for each datafile.
   d. Mount the database.
   e. Restore the database.
   f. Switch the datafiles.
   g. Recover the database.

h. Open the database with the RESETLOGS option.

```
run {
     set until scn 500;  # use appropriate SCN for incomplete recovery
     allocate channel ch1 type disk;
     set newname for datafile 1 to '/disk1/%U'; # rename each datafile manually
     set newname for datafile 2 to '/disk1/%U';
     set newname for datafile 3 to '/disk1/%U';
     set newname for datafile 4 to '/disk1/%U';
     set newname for datafile 5 to '/disk1/%U';
     set newname for datafile 6 to '/disk2/%U';
     set newname for datafile 7 to '/disk2/%U';
     set newname for datafile 8 to '/disk2/%U';
     set newname for datafile 9 to '/disk2/%U';
     set newname for datafile 10 to '/disk2/%U';
     alter database mount;
     restore database;
     switch datafile all;  # points the control file to the renamed datafiles
     recover database;
     alter database open resetlogs;
}
```

To restore to HOST_B without a recovery catalog:

1. Follow the procedure in "Moving the Target Database to a New Host with the Same File System" on page 6-4 (without a recovery catalog), stopping before you execute the **run** command in step 6. Make sure to reset all *_DEST and *_PATH parameters in the initialization parameter file that specify a pathname.
2. Execute a **run** command with the following sub-commands:
   a. Set the end SCN obtained from the SQL*Plus query.
   b. Allocate at least one channel.
   c. Specify a new filename for each datafile.
   d. Restore the database.
   e. Switch the datafiles.
   f. Recover the database.
   g. Open the database with the RESETLOGS option.

```
run {
     set until scn 500;  # use appropriate SCN for incomplete recovery
     allocate channel ch1 type disk;
     set newname for datafile 1 to '/disk1/%U'; # rename each datafile manually
     set newname for datafile 2 to '/disk1/%U';
     set newname for datafile 3 to '/disk1/%U';
```

```
        set newname for datafile 4 to '/disk1/%U';
        set newname for datafile 5 to '/disk1/%U';
        set newname for datafile 6 to '/disk2/%U';
        set newname for datafile 7 to '/disk2/%U';
        set newname for datafile 8 to '/disk2/%U';
        set newname for datafile 9 to '/disk2/%U';
        set newname for datafile 10 to '/disk2/%U';
        restore database;
        switch datafile all;  # point control file to renamed datafiles
        recover database;
        alter database open resetlogs;
    }
```

Testing the Restore of a Database to a New Host

To create a duplicate database for testing while maintaining your original database, use the **duplicate** command instead of the **restore** command (see Chapter 7, "Creating a Duplicate Database with Recovery Manager"). RMAN automatically creates a unique database identifier for the duplicate database.

To test the restore of a database to a new host using the **restore** command, follow the procedures described in "Moving the Target Database to a New Host with the Same File System" on page 6-4 or "Moving the Target Database to a New Host with a Different File System" on page 6-7. To prevent the generation of unnecessary records in the recovery catalog, do one of the following:

- Run RMAN with the **nocatalog** option 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, export the recovery catalog and import it into a different schema or database and use the copied recovery catalog for the test restore or duplicated database.

The following table describes the impact on the RMAN repository of the various restore scenarios:

Command	Catalog?	Affect on Repository
duplicate	yes	Generates a new db_id for the duplicate database, which you must manually register in the catalog. After registration, RMAN is aware of two distinct databases: the target and the duplicate.
duplicate	no	Generates a new db_id for the duplicate database. The repository for the target database is not affected

Command	Catalog?	Affect on Repository
restore	yes	If you issue **switch** commands, RMAN considers the restored database as the target database, and the recovery catalog becomes corrupted. If you do not issue **switch** commands, RMAN considers the restored datafiles as image copies that are candidates for future restore operations.
restore	no	If you issue **switch** commands, RMAN considers the restored database as the target database. If you do not issue **switch** commands, the restore operation has no effect on the repository.

Restoring Tablespaces and Datafiles

If a datafile is lost or corrupted but the disk is accessible, then you can restore the datafile to its previous location. Take the tablespace offline and issue a restore tablespace command. If the old location is inaccessible, then take the tablespace offline and restore the associated datafiles to a new location.

If you cannot restore datafiles to the default location, then use the **set newname** command before restoring. In this case, Oracle considers the restored datafiles as datafile copies; perform a **switch** to make them the current datafiles. Oracle creates the filename or overwrites it if it already exists.

The RMAN **switch** command is equivalent to the ALTER DATABASE RENAME DATAFILE statement. Note that a switch effectively causes the location of the current datafile to change. Also note that switching consumes the copy, that is, deletes the corresponding records in the recovery catalog and the control file.

If you do not specify the target of the switch, then the filename specified in a prior **set newname** for this file number is used as the switch target. If you specify **switch datafile all**, then all datafiles for which a **set newname** has been issued in this job are switched to their new name.

If you issue **set newname** commands to restore datafiles to a new location with the intention of performing a recovery afterwards, perform a switch after restoring but before recovering to make the restored datafiles the current datafiles.

See Also: "switch" on page 10-154 for **switch** command syntax.

To restore a tablespace to its default location:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. After allocating channels, do the following:
   - Take the tablespace that you want to recover offline.
   - Restore the tablespace.

   For example, to restore tablespace USER_DATA to disk you might issue:

```
run {
     sql 'ALTER TABLESPACE user_data OFFLINE TEMPORARY';
     allocate channel ch1 type disk;
     restore tablespace user_data;
}
```

3. You will need to perform media recovery on the restored tablespace. See "Recovering an Inaccessible Datafile in an Open Database" on page 6-38 for the required procedure.

To restore a tablespace to a new location:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. After allocating channels, do the following:
   - Take the tablespace offline.
   - Specify an accessible location to which you can restore the damaged datafile for the offline tablespace.
   - Restore the datafile to the new location.
   - Switch the restored datafile so that the control file considers it the current datafile.

   To restore the datafiles for tablespace TBS_1 to a new location on disk, you might enter:

```
run {
     allocate channel ch1 type disk;
     sql 'ALTER TABLESPACE user_data OFFLINE TEMPORARY';
     # restore the datafile to a new location
     set newname for datafile '/disk1/oracle/tbs_1.f' to '/disk2/oracle/tbs_1.f';
     restore tablespace tbs_1;
     # make the control file recognize the restored file as current
     switch datafile all;
}
```

3. You will need to perform media recovery on the restored tablespace. See "Recovering an Inaccessible Datafile in an Open Database" on page 6-38 for the required procedure.

Restoring Control Files

If a media failure damages your control file and you do not have multiplexed copies, you must restore a backup. Issue **restore controlfile** to restore the control file to the first CONTROL_FILES location specified in the parameter file. RMAN automatically replicates the control file to all CONTROL_FILES locations specified in the parameter file.

Specify a destination name with **restore controlfile to** '*filename*' when restoring a control file to a non-default location. If the filename already exists, then Oracle overwrites the file. When you restore the control file to a new location, use the **replicate controlfile from** '*filename*' command to copy it the CONTROL_FILES destinations: RMAN does not replicate the control file automatically.

Using the **replicate controlfile** command is equivalent to using multiple **copy controlfile** commands. After your specify the input control file by name, RMAN replicates the file to the locations specified in the CONTROL_FILES initialization parameter of the target database.

To restore the control file to its default location using a recovery catalog:

1. Start RMAN and connect to the target and recovery catalog databases. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. Start the instance without mounting the database:

   ```
   startup nomount;
   ```

3. Do the following:

   a. If for some reason you need to restore a control file created before a certain date, issue a **set until** command for that date. Otherwise, go to the next step.

   b. Allocate one or more channels.

   c. Restore the control file.

   d. Mount the database.

```
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';
     allocate channel ch1 type 'sbt_tape';
     restore controlfile;
     alter database mount;
}
```

RMAN automatically replicates to the control file to the locations specified by the CONTROL_FILES initialization parameter.

4. If you need to perform media recovery on the datafiles after restoring the control file, see "Performing Complete Recovery" on page 6-20 or "Performing Incomplete Recovery" on page 6-25.

To restore the control file to a new location without a recovery catalog:

Note that the control file that contains information about a given backup is not the control file that is backed up along with the backup. For example, if you issue **backup database**, then the backup control file in this whole database backup does not contain the record of the whole database backup. The next control file backup will contain information about the whole database backup.

1. Start RMAN and connect to the target database. For example, enter:

```
% rman target / nocatalog
```

2. Mount the database:

```
startup mount;
```

3. Do the following:

   a. If you need to restore a control file created before a certain date, issue a **set until** command for that date. Otherwise, go to the next step.

   b. Allocate one or more channels.

   c. Restore the backup control file to a temporary location to prevent accidental overwriting of the current control file.

   d. Shut down the database.

   e. Replicate the control file from the restored location to all locations specified in the CONTROL_FILES parameter of the parameter file.

   f. Mount the database.

```
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';
     allocate channel ch1 type 'sbt_tape';
     # restore control file to new location
     restore controlfile to '/oracle/dbs/cf1.ctl';
     shutdown immediate;
     # replicate the control file manually to locations in parameter file
     replicate controlfile from '/oracle/dbs/cf1.ctl';
     startup mount;
}
```

4. If you need to perform media recovery on the datafiles after restoring the control file, see "Performing Complete Recovery" on page 6-20 or "Performing Incomplete Recovery" on page 6-25.

> **See Also:** "replicate" on page 10-108 for **replicate controlfile** command syntax.

Restoring Archived Redo Logs

RMAN restores archived redo logs with names constructed using the LOG_ARCHIVE_FORMAT parameter and either the LOG_ARCHIVE_DEST or 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.

Override the destination parameter with the **set archivelog destination** command. By issuing this command, you can manually stage many 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 parameter file.

For example, if you specify a different destination from the one in the initialization parameter file and restore backups, subsequent restore and recovery operations detect this new location and do not look for the files in the initialization parameter parameter destination.

If desired, you can also specify multiple restore destinations for archived redo logs, although you cannot specify these destinations simultaneously. For example, you can issue:

```
run {
     allocate channel ch1 type disk;
     # Set a new location for logs 1 through 10.
     set archivelog destination to '/disk1/oracle/temp_restore';
```

```
     restore archivelog from logseq 1 until logseq 10;
     # Set a new location for logs 11 through 20.
     set archivelog destination to '/disk1/oracle/arch';
     restore archivelog from logseq 11 until logseq 20;
     # Set a new location for logs 21 through 30.
     set archivelog destination to '/disk2/oracle/temp_restore';
     restore archivelog from logseq 21 until logseq 30;
     . . .
     recover database;
}
```

Note that if you restore archived redo logs to multiple locations, you only need to issue a single **recover** command. RMAN finds the restored archived logs automatically and applies them to the datafiles.

To restore archived redo logs:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   Optionally, specify a message log file at connect time:

   ```
   % rman target / catalog rman/rman@rcat log = rman_log
   ```

2. If the database is open, shut it down and then mount it:

   ```
   shutdown immediate;
   startup mount;
   ```

3. Perform the following operations within your **run** command:

   a. If desired, specify the new location for the restored archived redo logs using **set archivelog destination**. Otherwise, go to next step.

   b. Allocate channels.

   c. Restore the archived redo logs.

   For example, this job restores all backup archived redo logs:

   ```
   run {
        # Optionally, set a new location for the restored logs.
        set archivelog destination to '/oracle/temp_restore';
        allocate channel ch1 type disk;
        restore archivelog all;
   }
   ```

> **See Also:** "set_run_option" on page 10-142 for **set archivelog destination** command syntax.

Restoring in Preparation for Incomplete Recovery

Use the **set until** command to specify the termination point for recovery. This command affects any subsequent **restore**, **switch**, and **recover** commands that are in the same **run** command.

To restore the database in preparation for incomplete recovery:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   Optionally, specify a message log file at connect time:

   ```
   % rman target / catalog rman/rman@rcat log = rman_log
   ```

2. If the database is open, shut it down and then mount it:

   ```
   shutdown immediate;
   startup mount;
   ```

3. Perform the following operations within your **run** command:

   a. Determine whether you want to recover to a specified time, SCN, or log sequence number and issue the appropriate **set until** command.

   b. Allocate channels.

   c. Restore the database.

   For example, this job restores the database in anticipation of an incomplete recovery until December 15, 1998 at 9 a.m.

   ```
   run {
        set until time 'Dec 15 1998 09:00:00';
        allocate channel ch1 type 'sbt_tape';
        restore database;
   }
   ```

Restoring in an OPS Configuration

In some customer configurations, tape backups can only be restored from the node that created the backups. Consequently, if node A makes a backup to tape in an OPS configuration, node A—and not node B or node C—must perform the restore. Issue

the **set autolocate** command to force RMAN to discover which nodes of an OPS cluster should attempt to restore which backups. If you do *not* issue **set autolocate** when restoring in conjunction with certain media management products or when restoring from a file system, the restore can fail because RMAN attempts to restore a backup from a node where it does not reside.

Issue the **set autolocate on** command only if:

- The command precedes **restore** or **recover** commands.
- Channels are allocated on different nodes of an OPS cluster.
- The media management servers do not offer cluster-wide service.
- It is necessary for an OPS restore (the **autolocate** option incurs system overhead).

To restore a database in an OPS configuration:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   Optionally, specify a message log file at connect time:

   ```
   % rman target / catalog rman/rman@rcat log = rman_log
   ```

2. If the database is open, shut it down and then mount it:

   ```
   shutdown immediate;
   startup mount;
   ```

3. Perform the following operations within your **run** command:

   a. Allocate channels as usual for each node.

   b. Issue **set autolocate on**.

   c. Restore the database.

   ```
   run {
        allocate channel node_1 type disk connect 'sys/sys_pwd@node_1';
        allocate channel node_2 type disk connect 'sys/sys_pwd@node_2';
        allocate channel node_3 type disk connect 'sys/sys_pwd@node_3';
        set autolocate on;
        restore database;
   }
   ```

See Also: "set_run_option" on page 10-142 for **set autolocate** command syntax.

Recovering Datafiles

Media recovery is the application of redo logs or incremental backups to a restored file in order to update it to the current or non-current time. You can only recover or apply incremental backups to current datafiles, not datafile copies.

Perform media recovery when:

- Media failure has damaged datafiles and you need to recover them to the current time.
- You need to recover the entire database to a previous time.
- Media failure has damaged the control file.
- You have executed the CREATE CONTROLFILE command.

RMAN restores backup sets of archived redo logs as needed to perform the media recovery. By default, RMAN restores the archived redo logs to the current log archive destination specified in the initialization parameter file. Use the **set archivelog destination** command to specify a different location.

If RMAN has a choice between applying an incremental backup or applying redo, then it always chooses the incremental backup. If overlapping levels of incremental backup are available, then RMAN automatically chooses the one covering the longest period of time.

If possible, make the recovery catalog available to perform the media recovery. If it is not available, RMAN uses information from the target database control file.

Note: If control file recovery is required, then you must make the recovery catalog available. RMAN cannot operate when neither the recovery catalog nor the target database control file are available.

This section contains the following topics:

- Preparing for Media Recovery
- Performing Complete Recovery
- Performing Incomplete Recovery

See Also: "Incremental Backups" on page 1-50 for an overview of incremental backups.

Preparing for Media Recovery

When and how to recover depends on the state of the database and the location of its datafiles.

To determine whether media recovery is necessary:

1. Start SQL*Plus and connect to your target database. For example, issue the following to connect to PROD1:

   ```
   % sqlplus sys/change_on_install@prod1;
   ```

2. Determine the status of the database by executing the following SQL query at the command line:

   ```
   SELECT parallel, status FROM v$instance;

   PAR STATUS
   --- -------
   NO  OPEN
   ```

   If the STATUS column reads OPEN, then the database is open, but it is still possible that you need to restore or recover some tablespaces and their datafiles.

3. Execute the following query to check the datafile headers and respond according to the table below:

   ```
   SELECT file#, status, error, recover, tablespace_name, name
   FROM v$datafile_header
   WHERE error IS NOT NULL
   OR recover = 'YES';
   ```

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.

ERROR column	RECOVER column	Solution
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 unreadable data blocks but its header block is intact.

Performing Complete Recovery

When performing complete recovery, recover either the whole database or a subset of the database. For example, you can perform a complete recovery of a majority of your tablespaces, and then recover the remaining tablespaces later. It makes no difference if the datafiles are read-write or offline normal.

The method you use for complete recovery depends on whether the database is open or closed.

If the database is...	Then...
Closed	Do one of the following: ■ Recover the whole database in one operation. ■ Recover individual tablespaces in separate operations.
Open	Do one of the following: ■ Close it and recover. ■ Take individual tablespaces offline and recover them.

The **skip** clause is useful for avoiding recovery of tablespaces containing only temporary data or for postponing recovery of some tablespaces. The **skip** clause takes the datafiles in the specified tablespaces offline before starting media recovery and keeps them offline until after media recovery completes.

Issue at least one **allocate channel** command before you issue the **recover** command unless you do not need to restore archived redo log or incremental backup sets. Allocate the appropriate type of device for the backup sets that you want to restore.

If the appropriate type of storage device is not available, then the **recover** command will fail.

Recovering the Database

The procedure for performing complete recovery on the database differs depending on whether the control file is available.

To recover the database when the control file is intact:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. If the database is open, shut it down, then mount it:

```
shutdown immediate;
startup mount;
```

3. After allocating channels, restore the database and recover it. This example skips the read-only TEMP tablespace:

```
run {
     allocate channel ch1 type disk;
     restore database;
     recover database
       skip tablespace temp;
}
```

4. Examine the output to see if recovery was successful. After RMAN restores the necessary datafiles, look for `RMAN-08055` in the output:

```
RMAN-08024: channel ch1: restore complete
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: recover

RMAN-03022: compiling command: recover(1)

RMAN-03022: compiling command: recover(2)

RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
RMAN-08054: starting media recovery
RMAN-08515: archivelog filename=/oracle/arc_dest/arcr_1_40.arc thread=1 sequence=40
RMAN-08515: archivelog filename=/oracle/arc_dest/arcr_1_41.arc thread=1 sequence=41
```

```
RMAN-08055: media recovery complete

RMAN-03022: compiling command: recover(4)
RMAN-08031: released channel: ch1
```

To recover the database using a backup control file:

When you perform a restore operation using a backup control file and you use a recovery catalog, RMAN automatically adjusts the control file to reflect the structure of the restored backup.

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. If you use a recovery catalog, RMAN updates the control file. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. Start the instance without mounting the database:

   ```
   startup nomount;
   ```

3. After allocating one or more channels, do the following:

   a. Use the **restore controlfile** command to restore the control file to all locations specified in the CONTROL_FILES initialization parameter.

   b. Mount the database.

   c. Restore and recover the database.

   d. Open the database with the RESETLOGS option.

   ```
   run {
        allocate channel ch1 type 'sbt_tape';
        restore controlfile;
        alter database mount;
        restore database;
        recover database;
        alter database open resetlogs;
   }
   ```

4. Reset the database:

   ```
   reset database;
   ```

5. Immediately back up the database. Because the database is a new incarnation, the pre-RESETLOGS backups are not usable. For example, enter:

   ```
   run {
        allocate channel ch1 type 'sbt_tape';
   ```

```
     backup database;
}
```

Recovering Tablespaces

The procedure for recovery tablespaces depends on whether the database is open or closed and whether the default tablespace location is accessible.

To recover an accessible tablespace when the database is closed:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. After allocating channels, restore the tablespace and recover it. This example recovers tablespace TBS_3:

   ```
   run {
        allocate channel ch1 type disk;
        restore tablespace tbs_3;
        recover tablespace tbs_3;
   }
   ```

3. Examine the output to see if recovery was successful.

To recover an inaccessible tablespace when the database is closed:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. After allocating channels, do the following:

   a. Rename the damaged datafile, specifying an accessible location.

   b. Restore the backup datafile to the new location.

   c. Switch the restored datafile so that the control file considers it the current datafile.

   d. Recover the tablespace.

   ```
   run {
        allocate channel ch1 type disk;
        set newname for datafile '/disk1/oracle/tbs_1.f' to '/disk2/oracle/tbs_1.f';
        restore tablespace tbs_1;
        switch datafile all;
   ```

```
     recover tablespace tbs_1;
}
```

To recover an accessible tablespace while the database is open:

If a datafile is lost or corrupted but the disk is accessible, restore the datafile to its default location.

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. Do the following:

   a. Take the tablespace that you want to recover offline.

   b. Allocate channels.

   c. Optionally, set a restore destination for archived redo logs necessary for recovery. Because RMAN is restoring the logs to this location, it knows where to find them.

   d. Restore and then recover the tablespace.

   e. Bring the tablespace online.

   ```
   run {
        sql 'ALTER TABLESPACE user_data OFFLINE TEMPORARY';
        allocate channel ch1 type disk;
        set archivelog destination to '/oracle/temp/arcl_restore';
        restore tablespace user_data;
        recover tablespace user_data;
        sql 'ALTER TABLESPACE user_data ONLINE';
   }
   ```

To recover an inaccessible tablespace while the database is open:

If a tablespace or datafile is inaccessible because of media failure, restore the datafile to a new location or switch to an existing datafile copy.

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

2. After allocating channels, do the following:

   a. Take the tablespace that you want to recover offline.

   b. Rename the damaged datafile, specifying an accessible location.

c. Restore the backup datafile to the new location.

d. Switch the restored datafile so that the control file considers it the current datafile.

e. Recover the tablespace.

f. Bring the tablespace online.

```
run {
     sql 'ALTER TABLESPACE user_data OFFLINE IMMEDIATE';
     allocate channel ch1 type disk;
     set newname for datafile '/disk1/oracle/tbs_1.f' to '/disk2/oracle/tbs_1.f';
     restore tablespace tbs_1;
     switch datafile all;
     recover tablespace tbs_1;
     sql 'ALTER TABLESPACE tbs_1 ONLINE';
}
```

Performing Incomplete Recovery

RMAN allows you to perform recovery of the whole database to a specified non-current time, SCN, or log sequence number. This type of recovery is called *incomplete recovery*; if it is recovery of the whole database, it is sometimes called *database point-in-time recovery (DBPITR)*.

Incomplete recovery differs in several ways from complete recovery. The most important difference is that incomplete recovery requires you to open the database with the RESETLOGS option. Using this option gives the online redo logs a new timestamp and SCN, thereby eliminating the possibility of corrupting your datafiles by the application of obsolete archived redo logs.

Because you must open RESETLOGS after performing incomplete recovery, 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, which 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 timestamps more recent than the set time. Hence, it is usually best to specify the **set until** command *before* the **restore** or **switch** command.

> **See Also:** "untilClause" on page 10-156 for **set until** command syntax.

Performing Incomplete Recovery with a Recovery Catalog

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 (see "Setting NLS Environment Variables" on page 2-2). For example, enter:

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

To recover the database until a specified time:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

   ```
   % rman target / catalog rman/rman@rcat
   ```

   Optionally, specify a log file at connect time:

   ```
   % rman target / catalog rman/rman@rcat log = rman_log
   ```

2. If the database is open, shut it down and then mount it:

   ```
   shutdown immediate;
   startup mount;
   ```

3. Determine which time you want to recover to. For example, if you discover at 9:15 a.m. 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.

4. Perform the following operations within your **run** command:

   a. Set the end recovery time using the date format specified in your NLS_LANG and NLS_DATE_FORMAT environment variables.

   b. Allocate channels.

   c. Restore the database.

   d. Recover the database.

   e. Open the database with the RESETLOGS option.

   For example, this job performs an incomplete recovery until Nov 15 at 9 a.m.

   ```
   run {
   ```

```
     set until time 'Nov 15 1998 09:00:00';
     allocate channel ch1 type 'sbt_tape';
     restore database;
     recover database;
     alter database open resetlogs;
}
```

5. Reset the database:

```
reset database;
```

6. Immediately back up the database. Because the database is a new incarnation, the pre-RESETLOGS backups are not usable. For example, enter:

```
run {
     allocate channel ch1 type 'sbt_tape';
     backup database;
}
```

To recover the database until a specified SCN:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

   Optionally, specify a message log file at connect time:

```
% rman target / catalog rman/rman@rcat log = rman_log
```

2. If the database is open, shut it down and then mount it:

```
shutdown immediate;
startup mount;
```

3. Determine the SCN to which you want to recover. For example, if you made a backup of tablespace TBS_1 and then shortly afterwards a user accidentally overwrote a datafile in TBS_3, then you can issue a **list** command to determine the SCN for the TBS_1 backup and then restore yesterday's whole database backup and recover to that SCN.
4. Perform the following operations within your run command:
   a. Set the end recovery SCN.
   b. Allocate channels.
   c. Restore the database.
   d. Recover the database.

e. Open the database with the RESETLOGS option.

For example, this job performs an incomplete recovery until SCN 1000.

```
run {
     set until scn 1000;
     allocate channel ch1 type 'sbt_tape';
     restore database;
     recover database;
     alter database open resetlogs;
}
```

5. Reset the database:

```
reset database;
```

6. Immediately back up the database. Because the database is a new incarnation, the pre-RESETLOGS backups are not usable. For example, enter:

```
run {
     allocate channel ch1 type 'sbt_tape';
     backup database;
}
```

To recover the database until a specified log sequence number:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

Optionally, specify a message log file at connect time:

```
% rman target / catalog rman/rman@rcat log = rman_log
```

2. If the database is open, shut it down and then mount it:

```
shutdown immediate;
startup mount;
```

3. Determine the log sequence number to which you want to recover. For example, query V$LOG_HISTORY to view the redo logs that you have archived.

```
RECID      STAMP      THREAD#    SEQUENCE#  FIRST_CHAN FIRST_TIM NEXT_CHANG
---------- ---------- ---------- ---------- ---------- --------- ----------
         1  344890611          1          1      20037 24-SEP-98      20043
         2  344890615          1          2      20043 24-SEP-98      20045
         3  344890618          1          3      20045 24-SEP-98      20046
         4  344890621          1          4      20046 24-SEP-98      20048
         5  344890624          1          5      20048 24-SEP-98      20049
         6  344890627          1          6      20049 24-SEP-98      20050
```

```
7  344890630          1          7      20050 24-SEP-98      20051
8  344890632          1          8      20051 24-SEP-98      20052
8 rows selected.
```

4. Perform the following operations within your **run** command:

   a. Set the log sequence number for recovery termination.

   b. Allocate channels.

   c. Restore the database.

   d. Recover the database.

   e. Open the database with the RESETLOGS option.

   For example, this job performs an incomplete recovery until log sequence number 6 on thread 1:

```
run {
     set until logseq 6 thread 1;
     allocate channel ch1 type 'sbt_tape';
     restore database;
     recover database;
     alter database open resetlogs;
}
```

5. Reset the database:

```
reset database;
```

6. Immediately back up the database. Because the database is a new incarnation, the pre-RESETLOGS backups are not usable. For example, enter:

```
run {
     allocate channel ch1 type 'sbt_tape';
     backup database;
}
```

Performing Incomplete Recovery Without a Recovery Catalog

Performing DBPITR without a recovery catalog requires that you adhere to the following precautionary measures:

- Back Up the Control File Separately
- Make New Backups After Adding Tablespaces or Datafiles
- Save RMAN Output for All Backups
- Specify the Maximum Age of Control File Records

Back Up the Control File Separately Make a backup of the control file after your RMAN database backups because you need a backup control file that contains information about the database backup that you just made. Even if your database backup included backing up the control file, as it does if you back up `datafile 1` or specify **include current controlfile**, the backup control file contained in the backup set is not self-referential. Consider this command:

```
backup database;
```

This command produces a backup set that contains a backup of the control file. Nevertheless, this backup control file does *not* contain a record for the backup set in which it is itself contained. Consequently, if you restore this backup control file and then mount it, you will not be able to restore files out of the backup set because the control file has no record of them.

To back up the control file separately, issue commands within your **run** command as in the following example:

```
backup database;
backup current controlfile tag = 'database backup';
```

These commands create two backup sets, each of which contains a backup control file. The control file backup created by the second command is the useful one, that is, it will contain all the records related to the database backup. Creating a tag for the backup control file is useful if you need to specify it later for a restore. Do not change the tag when you take a subsequent control file backup.

Make New Backups After Adding Tablespaces or Datafiles Immediately following the addition of a new tablespace or datafile to the database, make a new backup. If you are running in NOARCHIVELOG mode, you must back up the entire database. If you are running in ARCHIVELOG mode, then you can back up just the tablespace or datafiles that you added. Follow the backups with a backup of the control file using the **backup current controlfile** command as described in "Back Up the Control File Separately" on page 6-30.

Save RMAN Output for All Backups This operation is insurance in case you need to manually restore a control file from a backup set without using RMAN (as described in "Restoring the Control File from a Backup Set Without Using RMAN" on page 6-37).

To make RMAN write output to a file, either redirect STDOUT at the operating system level or use the RMAN **log** command line option.

> **See Also:** "cmdLine" on page 10-42 for command line options.

Specify the Maximum Age of Control File Records To ensure that the control file contains backup records that you need for performing DBPITR, set the following initialization parameter to a non-zero value (where *integer* is some number of days):

```
CONTROL_FILE_RECORD_KEEP_TIME = integer
```

This value should be equal to or greater than the maximum number of days that you need to go back during point-in-time recovery. For example, if you need to recover to a point two weeks before the present, then set the parameter to 14 or higher.

> **See Also:** "Monitoring the Overwriting of Control File Records" on page 3-47 and "Managing Records in the Control File" on page 3-33 to learn how to manage CONTROL_FILE_RECORD_KEEP_TIME.

To perform DBPITR without a recovery catalog:

1. Start RMAN and connect to the target database, specifying the **nocatalog** option:

   ```
   % rman target / nocatalog
   ```

2. If the database is open, shut it down and then mount it:

   ```
   startup force mount;
   ```

3. Restore a backup control file to a temporary location. Restore one of the backup control files that you created as directed in "Back Up the Control File Separately" on page 6-30.

   > **Note:** If you created a separate control file backup as suggested, RMAN usually restores this backup automatically: RMAN only chooses the wrong backup control file if you specify a time that was in the interval between the **backup database** and the **backup current controlfile**.
   >
   > To ensure that you restore the correct control file, use the **tag** option on the **backup current controlfile** command, and then specify this tag on the restore to force RMAN to pick the control file you want.

   For example, execute the following script to restore the control file to a temporary location:

```
run {
     set until time 'Jun 18 1998 16:32:36';
     allocate channel ch1 type disk;
     # restore a backup controlfile to a temporary location.
     restore controlfile to '/tmp/cf.tmp' from tag = 'database backup';
}
```

Note: This example assumes the NLS_DATE_FORMAT environment variable has been set to 'MON DD YYYY HH24:MI:SS'. You can set it to any format you like, but you must specify the date to RMAN in that format. For the NLS_DATE_FORMAT to take effect, you must also explicitly set the NLS_LANG environment variable to whatever locality, language, and character set that you are using. For more information, see "Setting NLS Environment Variables" on page 2-2

---

4. If you did not specify the control file restore using a tag, then verify that the control file that RMAN restored is the correct one. If you saved the RMAN output as directed in "Back Up the Control File Separately" on page 6-30, then you can look in the output file. For example, look for `RMAN-08021`:

   ```
   RMAN-08021: channel c1: restoring controlfile
   RMAN-08505: output filename=/oracle/dbs/cf1.f
   RMAN-08023: channel c1: restored backup piece 1
   RMAN-08511: piece handle=/oracle/dbs/0ab81tct_1_1 tag=post_wholedb params=NULL
   RMAN-08024: channel c1: restore complete
   ```

5. Connect to the target database using SQL*Plus and make a copy of the current control file. This operation is a safety measure in case the current control file is needed again for some reason. For example, enter:

   ```
   SQL> alter database backup controlfile to '/tmp/original_cf';
   ```

6. Shut down the database and copy the control file that you restored to a temporary location to the location specified for the control file in the initialization parameter file.

   For example, assume that the CONTROL_FILES parameter is set as follows:

   ```
   CONTROL_FILES = (?/dbs/cf1.f, ?/dbs/cf2.f)
   ```

   Then, shut down the database and use operating system commands to copy the control file that you restored to the temporary location to the initialization parameter locations. For example, enter:

```
SQL> SHUTDOWN ABORT

% cp /tmp/cf.tmp $ORACLE_HOME/dbs/cf1.f
% cp /tmp/cf.tmp $ORACLE_HOME/dbs/cf2.f
```

7. Mount the database. For example, enter:

```
SQL> STARTUP MOUNT
```

8. Execute the following operations within a **run** command:

   a. Set an end time, SCN, or log sequence number (if you run in ARCHIVELOG mode) for recovery.

   b. Allocate one or more channels.

   c. Restore and recover the database. If the database is running in NOARCHIVELOG mode, specify the **noredo** option on the **recover** command. If the database runs in ARCHIVELOG mode, then omit the **noredo** option.

   d. Open the database with the RESETLOGS option

```
run {
     set until time 'Jun 18 1998 16:32:36';
     allocate channel ch1 type disk;
     restore database;
     recover database noredo;
     alter database open resetlogs;
}
```

9. Reset the database:

```
reset database;
```

10. Immediately back up the database. For example, enter:

```
run {
     allocate channel ch1 type disk;
     backup database;
}
```

    If you are running in ARCHIVELOG mode, then stop here. If you are running in NOARCHIVELOG mode, then proceed to the next step.

11. If you are running in NOARCHIVELOG mode, then you should ensure that you can perform this restore and recovery again if necessary. To do so, back up the control file that you saved in step 5 as follows:

```
run {
```

```
    allocate channel ch1 type disk;  # or type 'sbt_tape'
    backup backup controlfile '/tmp/original_cf' format ...;
}
```

Alternatively, you can copy the backup control file that you made in step 5 to a permanent location and then make RMAN aware of it by using the **catalog** command. First, copy the control file to a permanent location, giving it a meaningful filename:

```
% cp /tmp/original_cf $ORACLE_HOME/dbs/backup_cf_JUN-20-1999
```

Then, use the **catalog** command make RMAN aware of this control file:

```
catalog backup controlfile '/oracle_home/dbs/backup_cf_JUN-20-1999';
```

Restore and Recovery Scenarios

Following are useful scenarios for performing restore and recovery operations:

- Restoring Datafile Copies to a New Host
- Restoring When Multiple Databases Share the Same Name
- Restoring the Control File from a Backup Set Without Using RMAN
- Recovering an Inaccessible Datafile in an Open Database
- Recovering an Inaccessible Datafile Using Backups from Disk and Tape
- Performing Recovery After a Total Media Failure
- Recovering a Pre-RESETLOGS Backup
- Recovering a Database in NOARCHIVELOG Mode
- Recovering a Lost Datafile Without a Backup

Restoring Datafile Copies to a New Host

To move the database to a new host using 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 your target database and recovery catalog, issue a **list** command to see a listing of your datafile copies and their associated primary keys:

   ```
   list copy;
   ```

2. Copy the datafile copies to the new host using an operating system utility. For example, a UNIX user could enter:

```
% cp -r /oracle/copies /net/new_host/oracle/dbs
```

3. Uncatalog the datafile copies on the old host. For example, enter:

```
change datafile copy 1,2,3,4,5,6,7,9,10 uncatalog;
```

4. Catalog the transferred datafile copies, using their new filenames. For example, enter:

```
catalog datafilecopy '/oracle/dbs/tbs_1.f', '/oracle/dbs/tbs_2.f',
'/oracle/dbs/tbs_3.f', '/oracle/dbs/tbs_4.f', '/oracle/dbs/tbs_5.f',
'/oracle/dbs/tbs_6.f', '/oracle/dbs/tbs_7.f', '/oracle/dbs/tbs_8.f',
'/oracle/dbs/tbs_9.f', '/oracle/dbs/tbs_10.f';
```

5. Perform the restore and recovery operation described in "Moving the Target Database to a New Host with the Same File System" on page 6-4 or "Moving the Target Database to a New Host with a Different File System" on page 6-7. Specify a channel of type **disk** rather than type '*sbt_tape*'.

Restoring When Multiple Databases Share the Same Name

The database identifier is a 32-bit number that is computed when the database is created. If you want to restore a database that shares a name with another database, you must distinguish it. Use the RMAN **set dbid** command to specify a database according to its database identifier.

Obtaining the DBID of a Database You Want to Restore

If you have saved your RMAN output, refer to this information to determine the database identifier, since RMAN automatically provides it whenever you connect to the database:

```
% rman target /

Recovery Manager: Release 8.1.5.0.0

RMAN-06005: connected to target database: RMAN (DBID=1231209694)
```

If you have not saved your RMAN output and need the DBID value of a database for a restore operation, obtain it via 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 non-unique 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 queries below. 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 Using the DBID

Only use the **set dbid** command to restore the control file when all of these conditions are met:

- The control file has been lost and must be restored from a backup.
- You are using a recovery catalog.
- Multiple databases registered in the recovery catalog share a database name.

If these conditions are not met, you receive the `RMAN-20005: target database name is ambiguous` message when you attempt to restore the control file. RMAN will correctly identify the control file to restore, so you do not need to use the **set dbid** command.

RMAN accepts **set dbid** only if you have not yet connected to the target database, that is, **set dbid** must precede the **connect target** command. If the target database is mounted, then RMAN verifies that the user-specified DBID matches the DBID from the database; it not, RMAN signals an error. If the target database is not mounted, RMAN uses the user-specified DBID to restore the control file. After restoring the control file, you can mount the database to restore the rest of the database.

To set the database id enter the following, where *target_dbid* is an integer value:

```
set dbid = target_dbid;
```

To restore the control file to its default location enter:

```
run {
     allocate channel dev1 type disk;
```

```
    restore controlfile;
    alter database mount;
}
```

Restoring the Control File from a Backup Set Without Using RMAN

You must use a non-standard procedure to restore a control file from an RMAN backup set in the following situations:

- You are using a pre-8.0.5 version of RMAN to restore a database when more than one database with the same name is registered in the recovery catalog (see "Restoring When Multiple Databases Share the Same Name" on page 6-35 for a discussion of this problem).
- You are not using a recovery catalog, and your only control file backup is in an RMAN backup set.

If you have no other backup of the control file except in a RMAN backup set, and you need the control file to perform a restore operation, use the following PL/SQL program to extract the control file from the backup set. Run this program from SQL*Plus while connected as SYSDBA to the target database:

```
DECLARE
  devtype varchar2(256);
  done    boolean;
BEGIN
  devtype := dbms_backup_restore.deviceallocate('devtype', params=>'');
  # Replace 'devtype' with the device type you used when creating the backup: NULL or
  # sbt_tape.  If you used an sbt_tape device and specified a 'parms' option on the RMAN
  # allocate channel command, then put that parms data in the 'params' operand here.

  dbms_backup_restore.restoresetdatafile;

  dbms_backup_restore.restorecontrolfileto('/tmp/foo.cf');
  # This path specifies the location for the restored control file. If there are multiple
  # control files specified in the init.ora file, copy the control file to all specified
  # locations before mounting the database.

  dbms_backup_restore.restorebackuppiece('handle',done=>done);
  # Replace 'handle' with the your backup piece handle. This example assumes that the
  # backup set contains only one backup piece. If there is more than one backup piece in
  # the backup set (which only happens if the RMAN command set limit kbytes is used), then
  # repeat the restorebackuppiece statement for each backup piece in the backup set.

END;
/
```

After you have successfully restored the control file, you can mount the database and perform restore and recovery operations.

Recovering an Inaccessible Datafile in an Open Database

In this scenario, the database is open but you cannot access a datafile. You execute the following SQL query to determine its status:

```
SELECT * FROM v$recover_file;

     FILE# ONLINE  ERROR          TIME
---------- ------- -------------- ----------
        19 ONLINE  FILE NOT FOUND
```

You then decide to start RMAN and connect to the target and recovery catalog databases:

```
% rman target / catalog rman/rman@rcat
```

You issue a **report** command to determine the datafile's tablespace and filename:

```
RMAN> report schema;

RMAN-03022: compiling command: report
Report of database schema
File K-bytes    Tablespace           RB segs Name
---- ---------- -------------------- ------- ------------------
1         47104 SYSTEM               YES     /oracle/dbs/tbs_01.f
2           978 SYSTEM               YES     /oracle/dbs/tbs_02.f
3           978 TBS_1                NO      /oracle/dbs/tbs_11.f
4           978 TBS_1                NO      /oracle/dbs/tbs_12.f
5           978 TBS_2                NO      /oracle/dbs/tbs_21.f
6           978 TBS_2                NO      /oracle/dbs/tbs_22.df
7           500 TBS_1                NO      /oracle/dbs/tbs_13.f
8           500 TBS_2                NO      /oracle/dbs/tbs_23.f
9           500 TBS_2                NO      /oracle/dbs/tbs_24.f
10          500 TBS_3                NO      /oracle/dbs/tbs_31.f
11          500 TBS_3                NO      /oracle/dbs/tbs_32.f
12          500 TBS_4                NO      /oracle/dbs/tbs_41.f
13          500 TBS_4                NO      /oracle/dbs/tbs_42.f
14          500 TBS_5                YES     /oracle/dbs/tbs_51.f
15          500 TBS_5                YES     /oracle/dbs/tbs_52.f
16         5120 SYSTEM               YES     /oracle/dbs/tbs_03.f
17         2048 TBS_1                NO      /oracle/dbs/tbs_14.f
18         2048 TBS_2                NO      /oracle/dbs/tbs_25.f
19         2048 TBS_3                NO      /oracle/dbs/tbs_33.f
20         2048 TBS_4                NO      /oracle/dbs/tbs_43.f
21         2048 TBS_5                YES     /oracle/dbs/tbs_53.f
```

Because you need to take the datafile online immediately before you investigate the media failure, you decide to restore the datafile to a new location and switch to a copy of that datafile:

```
run {
     sql 'ALTER TABLESPACE tbs_3 OFFLINE IMMEDIATE';
     allocate channel ch1 type disk;
     set newname for datafile '/oracle/dbs/tbs_33.f' to '/oracle/temp/tbs_33.f';
     restore tablespace tbs_3;
     switch datafile all;
     recover tablespace tbs_3;
     sql 'ALTER TABLESPACE tbs_3 ONLINE';
}
```

Recovering an Inaccessible Datafile Using Backups from Disk and Tape

If you cannot access a datafile due to a disk failure, you should probably restore it to a new location or switch to an existing datafile copy. The following example restores and recover tablespace TBS_1, which contains four datafiles. Because some copies of these files are on disk and some backups on tape, the example allocates one disk channel and one tape channel to allow **restore** to restore from both media:

```
run {
     allocate channel dev1 type disk;
     allocate channel dev2 type 'sbt_tape';
     sql "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE";
     set newname for datafile '/disk7/oracle/tbs11.f'
       to '/disk9/oracle/tbs11.f';
     set newname for datafile '/disk7/oracle/tbs12.f'
       to '/disk9/oracle/tbs12.f';
     set newname for datafile '/disk7/oracle/tbs13.f'
       to '/disk9/oracle/tbs13.f';
     set newname for datafile '/disk7/oracle/tbs14.f'
       to '/disk9/oracle/tbs14.f';
     restore tablespace tbs_1;
     switch datafile all;     # makes the renamed datafile the current datafile
     recover tablespace tbs_1;
     sql "ALTER TABLESPACE tbs_1 ONLINE";
}
```

Performing Recovery After a Total Media Failure

The following scenario assumes:

- You have lost the whole database, all control files, the online redo logs, and recovery catalog.

- You are restoring the database to its original location after fixing the media problem.
- There are four tape drives.
- You are using a recovery catalog.

Before restoring the database, you must:

- Restore your initialization parameter file, password file (if you use one), and your recovery catalog from your most recent backup using operating system commands or utilities.
- Catalog any archived redo logs, datafile copies, or backup sets that are on disk, but are not registered in the recovery catalog. The archived redo logs up to the log sequence number being restored to must be cataloged in the recovery catalog, or Recovery Manager will not know where to find them. If you resynchronize the recovery catalog frequently, and have an up-to-date copy from which you have restored, you should not have many archived redo logs that need cataloging.

The following scenario restores and recovers the database to the most recently available archived log, which is log 124 in thread 1. The example:

- Starts the instance without mounting the database and restricts connections to DBA-only users.
- Restores the control file to the locations specified by the initialization parameter parameter CONTROL_FILES.
- Mounts the control file.
- Catalogs any archived redo logs not in the recovery catalog.
- Restores the database files to their original locations. If volume names have changed, use the **set newname** command before the restore and perform a switch after the restore to update the control file with the new locations for the datafiles.
- Recovers the datafiles by either using a combination of incremental backups and redo, or just redo. RMAN stops recovery when it reaches the log sequence number specified.
- Opens the database in RESETLOGS mode. Only complete this last step if you are certain that no other archived redo logs can be applied.

Note: Oracle recommends that you back up your database after the RESETLOGS operating (not shown in the example).

```
% rman target sys/sys_pwd@prod1 catalog rman/rman@rcat
startup nomount dba;

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';
     #  set newname for datafile 4 to '/dev/vgd_1_0/rlvt5_500M_4';
     #  etc...

     #  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 Recovery Manager restores
     #  the database to the same structure that the database had at the specified time.
     set until logseq 124 thread 1;

     allocate channel t1 type 'SBT_TAPE';
     allocate channel t2 type 'SBT_TAPE';
     allocate channel t3 type 'SBT_TAPE';
     allocate channel t4 type 'SBT_TAPE';

     restore controlfile;
     alter database mount;

     #  Catalog any archivelogs that are not in the recovery catalog:
     #  catalog archivelog '/oracle/db_files/prod1/arch/arch_1_123.rdo';
     #  catalog archivelog '/oracle/db_files/prod1/arch/arch_1_124.rdo';
     #  etc...

     restore database;

     #  Update the control file by telling it the new location of the datafiles, but
     #  only if you used set newname commands.
     #  switch datafile all;
     recover database;

     #  Complete this last step only if no more archived logs need to be applied.
     alter database open resetlogs;
}
```

Recovering a Pre-RESETLOGS Backup

Assume the following situation:

- You run RMAN with a recovery catalog.
- You made a backup of PROD1 on July 2, 1999.
- You performed incomplete recovery on this database and opened it with the RESETLOGS option on July 10, 1999. A new database incarnation was created.

On July 25, you discover that you need crucial data that was dropped from the database at 8:00 a.m. on July 8, 1999. You decide to reset PROD1 to the prior incarnation, restore the July 2 backup, and then recover to 7:55 a.m. on July 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 using the 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 prod1;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         PROD1     1224038686  NO     1            02-JUL-99
1       582       PROD1     1224038686  YES    59727        10-JUL-99
```

2. You reset the incarnation using the primary key that you just obtained:

```
# reset database to old incarnation
reset database to incarnation 2;
```

3. You 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.
   - Allocate one or more channels.
   - Abort the instance and then re-start it.
   - Restore the control file and mount it.

- Restore and recover the database.
- Reset the online redo logs.

```
run {
     set until time 'Jul 8 1999 07:55:00'; # set time to just before data was lost
     allocate channel dev1 type disk;
     shutdown abort;
     startup nomount;
     restore controlfile;
     alter database mount; # mount database after restoring control file
     restore database;
     recover database;
     alter database open resetlogs; # this command automatically resets the database
                                    # so that this incarnation is the new incarnation
}
```

Recovering a Database in NOARCHIVELOG Mode

You can recover a database running in NOARCHIVELOG mode using incremental backups. Assume the following scenario:

- You run database PROD1 in NOARCHIVELOG mode.
- You use a recovery catalog.
- You make a level 0 backup of database PROD1 to tape on Monday.
- You make a level 1 differential incremental backups to tape at 7:00 a.m. on Wednesday and Friday.
- The database suffers a media failure on Sunday, causing you to lose half of the datafiles.

In this case, you are forced to perform an incomplete media recovery until Friday, since that is the date of your most recent incremental backup. Note that RMAN always looks for incremental backups before looking for archived logs during recovery.

RMAN can perform the desired incomplete media recovery automatically if you specify the **noredo** option in the **recover** command. If you do not specify **noredo**, RMAN searches for archived redo logs after applying the Friday incremental backup, and issues an error message when it does not find them.

After connecting to PROD1 and the catalog database, recover the database using the following command:

```
run {
     allocate channel dev1 type 'sbt_tape';
```

```
     restore database;
     recover database noredo;
     alter database open resetlogs;
}
```

Recovering a Lost Datafile Without a Backup

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 `rmantarg_t1.dbf`.
3. You populate the newly created datafile with data.
4. You archive all the active online redo logs.
5. Someone accidentally deletes `rmantarg_t1.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 lost data by creating a new datafile with the exact same filename as the lost datafile, then issuing the RMAN **recover** command to apply the redo for this file.

For example, run the following job:

```
run {
     allocate channel c1 type disk;
     # take the missing datafile offline
     sql "alter database datafile
              ''/eg2k001/u02/ORACLE/rmantarg/rmantarg_t1.dbf" offline";

     # create a new datafile with the same name as the missing datafile
     sql "alter database create datafile
              ''/eg2k001/u02/ORACLE/rmantarg/rmantarg_t1.dbf''";

     # recover the newly created datafile
     recover datafile '/eg2k001/u02/ORACLE/rmantarg/rmantarg_t1.dbf';

     # bring the recovered datafile back online
     sql "alter database datafile
              ''/eg2k001/u02/ORACLE/rmantarg/rmantarg_t1.dbf'' online";
}
```

7

Creating a Duplicate Database with Recovery Manager

This chapter describes how to use Recovery Manager to create a duplicate database for testing purposes, and includes the following topics:

- Creating a Duplicate Database: Overview
- Creating a Duplicate Database on a Local or Remote Host
- Duplication Scenarios

Creating a Duplicate Database: Overview

The RMAN **duplicate** command allows you to use your target database backups to create a test database while still retaining your original database. The command takes image copies or backup sets of your target database's files and generates a new database. A duplicate database is especially useful if your production database must be up and running 24 hours per day, 7 days a week.

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 redo logs 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.

When duplicating a database you can:

- Skip read-only tablespaces with the **skip readonly** clause. Read-only tablespaces are included by default. If you omit them, you can add them later.
- Create your duplicate database in a new host. If the directory structure is the same on the new host, you can use the **nofilenamecheck** option and reuse the target datafile filenames for the duplicate datafiles.
- Use the **set until** option when creating the duplicate database to recover it to a non-current 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 backups and archived redo logs.
- Use the **duplicate** command 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, the database identifier of the copied database remains the same so you cannot register it in the same recovery catalog.

> **See Also:** "duplicate" on page 10-76 for **duplicate** command syntax.

Obeying Restrictions

RMAN duplication has the following restrictions. You *cannot*:

- Duplicate offline tablespaces, although you can add these tablespaces manually after duplication.
- Duplicate a database when some backups of the target database do not exist. RMAN attempts to duplicate the following:
  - All datafiles in online tablespaces, whether or not the datafiles are online.
  - All tablespaces taken offline with the IMMEDIATE option.

  If no valid backups exist of any tablespace or datafile, the command fails.
- Use **nofilenamecheck** when the names of the duplicate datafiles are the same as the target datafiles and the databases are in the same host. If you do specify **nofilenamecheck**, RMAN may overwrite the datafiles or signal the following error:

```
RMAN-10035: exception raised in RPC: ORA-19504: failed to create file
            "/oracle/dbs/tbs_01.f"
ORA-27086: skgfglk: unable to lock file - already in use
SVR4 Error: 11: Resource temporarily unavailable
Additional information: 8
RMAN-10031: ORA-19624 occurred during call to
DBMS_BACKUP_RESTORE.RESTOREBACKUPPIECE
```

  Note that if you want the filenames to be the same, and the databases are in *different* hosts, then you must use the **nofilenamecheck** option.
- Duplicate when the target database is not mounted.
- Use image copies located on one host to create a duplicate database on a new host. You must either use tape backups or create the image copies in the same file location in the new host.
- Recover the duplicate database to the current point in time. RMAN recovers the duplicate database up to or before the most recent available archived redo log. If the duplication is to another host, then you must make the archived redo logs available in the expected location in the new host.

Generating Files for the Duplicate Database

When duplicating a database, perform the following operations:

- Creating the Control Files

Creating the Control Files

The **duplicate** command creates the control files by using the names listed in the initialization parameter file of the duplicate database. When choosing names for the duplicate database control files, make sure that you do not overwrite the initialization parameter settings for the production files at the target database.

Creating the Online Redo Logs

You have these options for creating the names of the duplicate online redo logs, which are listed in the order of precedence:

Table 7–1 Order of Precedence for Redo Log Filename Creation

Order	Method	Result
1	Specify the **logfile** clause of **duplicate** command.	Creates redo logs as specified.
2	Set LOG_FILE_NAME_CONVERT initialization parameter.	Transforms target filenames, for example, from `log_*` to `duplog_*`. **Note:** 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 above.	Reuses the target filenames. You must specify the **nofilenamecheck** option when using this method.

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, RMAN uses the **logfile** clause. If you specify all options, then RMAN uses the **logfile** clause and ignores the others.

> **WARNING: If the target and duplicate databases are in the same host, do not use the name of an online redo log currently in use by the target database. Also, do not use the name of a redo log currently in use by the target database if the duplicate database is in a different host and the nofilenamecheck keyword is not used.**

Renaming the Datafiles

If you want to have different filenames in your duplicate datafile, then you must use parameters or commands to specify them. You have these options for renaming datafiles, listed in the order of precedence:

Table 7–2 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 want to rename files.
2	Issue **set auxname** command.	Creates new datafile filenames. This setting stays in effect until disabled with a **set auxname ... to null** command.
3	Set DB_FILE_NAME_CONVERT initialization parameter.	Transforms target filenames, for example, from `tbs_*` to `dupdbs_*`. You can use this parameter for those files not renamed with either **set newname** and **set auxname**.
4	Do none of the above.	Reuses the target filenames. You must specify the **nofilenamecheck** option when using this method.

The order of precedence determines how RMAN names the datafiles. For example, if you specify all the commands and the initialization parameter, RMAN uses **set newname**. If you specify the **set auxname** command and DB_FILE_NAME_CONVERT, RMAN uses **set 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 these tablespaces. You see the following values in the specified views or tables:

Table/View	Column	Value
V$DATAFILE	STATUS	OFFLINE
V$DATAFILE	ENABLED	READ ONLY
V$DATAFILE	NAME	MISSINGxxx
SYS.DBA_DATA_FILES	STATUS	AVAILABLE
SYS.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.

You see the following values in the specified views or tables:

Table/View	Column	Value
V$DATAFILE	STATUS	OFFLINE
V$DATAFILE	ENABLED	DISABLED
V$DATAFILE	NAME	MISSINGxxx
SYS.DBA_DATA_FILES	STATUS	AVAILABLE
SYS.DBA_TABLESPACES	STATUS	OFFLINE

Note that when you take a tablespace offline with the IMMEDIATE option, RMAN duplicates rather than skips the tablespace. As with online tablespaces, RMAN requires a valid backup for duplication.

Preventing Filename Checking It is possible for a **set 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. 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 issue:

```
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
     allocate ...
     duplicate target database to newdb;
}
```

Even though you issued **set newname** commands for all your 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 without using either **set newname** or **set auxname** if all the datafiles will be converted by this parameter, that is, all of the datafiles have the same suffix or prefix.

Preparing the Auxiliary Instance for Duplication

Follow these guidelines before performing RMAN duplication:

- Create an Oracle Password File for the Auxiliary Instance
- Create a Parameter File for the Auxiliary Instance
- Start the Auxiliary Instance
- Ensure Net8 Connectivity to the Auxiliary Instance
- Mount or Open the Target Database
- Start the RMAN Command Line Interface
- Make Sure You Have the Necessary Backups and Archived Redo Logs
- Allocate Auxiliary Channels

Create an Oracle Password File for the Auxiliary Instance

For information about creating and maintaining Oracle password files, see *Oracle8i Administrator's Guide*.

Create a Parameter File for the Auxiliary Instance

Create an initialization parameter file for the auxiliary instance and set the following required parameters:

Parameter	Specify:
DB_NAME	The same name that you use in the **duplicate** command.
CONTROL_FILES	See "Creating the Control Files" on page 7-4.

Optionally, set the following parameters:

Parameter	Specify:
DB_FILE_NAME_CONVERT	See "Renaming the Datafiles" on page 7-5.
LOG_FILE_NAME_CONVERT	See "Creating the Online Redo Logs" on page 7-4.

Set other parameters, including the parameters that allow you to connect as SYSDBA through Net8, 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.

Following are examples of the initialization parameter settings for the duplicate database:

```
DB_NAME=newdb
CONTROL_FILES=(/oracle/dup_prod/cf/cf1.f,/oracle/dup_prod/cf/cf2.log)
DB_FILE_NAME_CONVERT=(/oracle/prod/db,/oracle/dup_prod/db)
LOG_FILE_NAME_CONVERT=("/oracle/prod/log","/oracle/dup_prod/log")
```

> **See Also:** *Net8 Administrator's Guide* for more information about Net8.

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 parameter file if necessary). In this example, *aux_pwd* is the password for the user with SYSDBA authority and *aux_str* is the net service name for the auxiliary instance:

```
SQL> connect sys/aux_pwd@aux_str
SQL> startup nomount pfile='/oracle/aux/dbs/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.

Ensure Net8 Connectivity to the Auxiliary Instance

The auxiliary instance must be accessible via Net8. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

Mount or Open the Target Database

Before beginning RMAN duplication, mount or open the target database—specifying a parameter file if necessary—if it is not already mounted or open. For example, enter:

```
SQL> startup pfile='/oracle/dbs/initPROD1.ora';
```

Start the RMAN Command Line Interface

Use one of the following methods to start the RMAN command line interface:

- Connect at the Operating System Command Line
- Connect at the RMAN Prompt

Connect at the Operating System Command Line You must connect to the auxiliary instance with SYSDBA privileges, so you must use a password file. To connect to the auxiliary instance, target instance, and recovery catalog, supply the following information when starting up Recovery Manager:

```
% rman target sys/target_pwd@target_str catalog rman/cat_pwd@cat_str auxiliary  \
> sys/aux_pwd@aux_str
```

Where:

target_pwd	The password for connecting as SYSDBA specified in the target database's `orapwd` file
target_str	The net service name for the target database
cat_pwd	The password for user RMAN specified in the recovery catalog's `orapwd` file
cat_str	The net service name for the recovery catalog database
aux_pwd	The password for connecting as SYSDBA specified in the auxiliary database's `orapwd` file.
aux_str	The net service name for the auxiliary database.

Connect at the RMAN Prompt You can start the RMAN command line interface without a connection to the auxiliary instance, and then use the **connect auxiliary** command at the RMAN prompt to make the auxiliary connection:

```
% rman
RMAN> connect auxiliary sys/aux_pwd@aux_str
RMAN> connect target sys/target_pwd@target_str
RMAN> connect catalog rman/cat_pwd@cat_str
```

Make Sure You Have the Necessary Backups and Archived Redo Logs

Make sure you have backups all the datafiles in your target database. If you do not have backups of everything, 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.
- The archived redo logs are accessible on the node where the **duplicate** command is operating.

Allocate Auxiliary Channels

Before issuing the **duplicate** command, allocate at least one auxiliary channel within the same **run** command. The channel type (**disk** or '*sbt_tape*') 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 allocate a channel of type 'sbt_tape' issue:
     allocate auxiliary channel ch1 type 'sbt_tape';

     # to allocate three auxiliary channels for disk issue (specifying whatever channel
     # id that you want):
     allocate auxiliary channel aux1 type disk;
     allocate auxiliary channel aux2 type disk;
     allocate auxiliary channel aux3 type disk;
     . . .
}
```

Creating a Duplicate Database on a Local or Remote Host

When you create your duplicate database, you have the following options:

- 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 your 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.

To create a duplicate database on a different host with the same file system:

1. Use an operating system utility to copy your parameter file from its location in the target host directory structure to the same location in the duplicate host directory structure. For example, you might issue:

   ```
   cp /net/host1/oracle/dbs/initPROD1.ora /net/host2/oracle/dbs/initDUPDB.ora
   ```

2. Use SQL*Plus to start the duplicate instance without mounting it. For example, enter:

   ```
   SQL> startup nomount pfile=initDUPDB.ora;
   ```

3. Use SQL*Plus to mount or open the target database if it is not already mounted or open. For example, enter:

   ```
   SQL> startup pfile=initPROD1.ora;
   ```

4. The auxiliary instance must be accessible via Net8. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

5. Use RMAN to connect to the target database, the duplicate database, and (if you use one) the recovery catalog database. In this example, connection is established without a recovery catalog using operating system authentication:

   ```
   % rman target / auxiliary sys/sys_pwd@dupdb
   ```

   In this example, user SCOTT has SYSDBA privileges and a net service name is used for the target:

   ```
   % rman auxiliary scott/tiger@dupdb target sys/sys_pwd@prod
   ```

   In this example, connection is established to three databases, all using net service names:

   ```
   % rman catalog rman/rman@rcat target sys/sys_pwd@prod1 auxiliary scott/tiger@dupdb
   ```

6. Perform the following operations:

   - Allocate at least one auxiliary channel.

- Specify the **nofilenamecheck** parameter.

For example, enter the following:

```
run {
     allocate auxiliary channel ch1 type 'sbt_tape';
     duplicate target database to dupdb
       nofilenamecheck;
}
```

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.

Duplicating a Database on a Remote Host with a Different Directory Structure

If you create your duplicate database on a host with a different file system, you need to change several initialization parameter file 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 **set auxname** command for the datafile filenames.

> **See Also:** Table 7–2 on page 7-5 for a table of the various datafile filename conversion options.

To duplicate a database with DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT:

1. Use an operating system utility to copy your 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 *_DEST and *_PATH initialization parameters that specify a pathname.
   - DB_FILE_NAME_CONVERT so that it captures *all* the target datafiles and converts them appropriately, for example, from `tbs_*` to `duptbs_*`.
   - LOG_FILE_NAME_CONVERT so that it captures *all* the online redo logs and converts them appropriately, for example, `log_*` to `duplog_*`.
2. Use SQL*Plus to start the duplicate instance without mounting it, for example:

   ```
   SQL> startup nomount pfile=initDUPDB.ora;
   ```

3. Use SQL*Plus to mount or open the target database if it is not already mounted or open. For example, enter:

```
SQL> startup pfile=initPROD1.ora;
```

4. The auxiliary instance must be accessible via Net8. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

5. Use RMAN to connect to the target database, the duplicate database, and (if you use one) the recovery catalog database. In this example, connection is established without a recovery catalog using operating system authentication:

```
% rman target / auxiliary sys/sys_pwd@dupdb
```

   In this example, user SCOTT has SYSDBA privileges and a net service name is used for the target:

```
% rman auxiliary scott/tiger@dupdb target sys/sys_pwd@prod
```

   In this example, RMAN connects to three databases, all using net service names:

```
% rman catalog rman/rman@rcat target sys/sys_pwd@prod1 auxiliary scott/tiger@dupdb
```

6. Issue the **duplicate** command. For example, enter the following:

```
run {
     allocate auxiliary channel ch1 type 'sbt_tape';
     duplicate target database to dupdb;
}
```

   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.

To duplicate a database with DB_FILE_NAME_CONVERT and the logfile clause:

Follow the same procedure for creating a duplicate database using the parameter LOG_FILE_NAME_CONVERT, but make the following substitutions:

- In step 1, do not set the LOG_FILE_NAME_CONVERT parameter.
- In step 6, change the **run** command as follows, specifying the names for the duplicate database redo log members:

```
run {
     allocate auxiliary channel ch1 type 'sbt_tape';
```

```
    duplicate target database to dupdb
      logfile
        '/oracle/dbs/log1.f' size 200K,
        '/oracle/dbs/log2.f' size 200K;
}
```

To duplicate a database using the set newname command:

1. Use an operating system utility to copy your parameter file from its location in the target host directory structure to the same location in the duplicate host directory structure. Set all *_DEST and *_PATH initialization parameters that specify a pathname.
2. Use SQL*Plus to start the duplicate instance without mounting it:

   ```
   SQL> startup nomount pfile=initDUPDB.ora;
   ```

3. Use SQL*Plus to mount or open the target database if it is not already mounted or open:

   ```
   SQL> startup pfile=initPROD1.ora;
   ```

4. The auxiliary instance must be accessible via Net8. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.
5. Use RMAN to connect to the target database, the duplicate database, and (if you use one) the recovery catalog database. In this example, connection is established without a recovery catalog using operating system authentication:

   ```
   % rman target / auxiliary sys/sys_pwd@dupdb
   ```

   In this example, user SCOTT has SYSDBA privileges and a net service name is used for the target:

   ```
   % rman auxiliary scott/tiger@dupdb target sys/sys_pwd@prod
   ```

   In this example, connection is established to three databases, all using net service names:

   ```
   % rman catalog rman/rman@rcat target sys/sys_pwd@prod1 auxiliary scott/tiger@dupdb
   ```

6. Perform the following operations:
   - Allocate at least one auxiliary channel.
   - Specify the same number of redo log members and groups that are used in your target database.

- Specify new filenames for the duplicate database datafiles.

For example, enter the following:

```
run {
     # allocate at least one auxiliary channel of type disk or tape
     allocate auxiliary channel dupdb1 type 'sbt_tape';
     . . .
     # set new filenames for the datafiles
     set newname for datafile 1 TO '$ORACLE_HOME/dbs/dupdb_data_01.f';
     set newname for datafile 2 TO '$ORACLE_HOME/dbs/dupdb_data_02.f';
     . . .
     # issue the duplicate command
     duplicate target database to dupdb
     # create at least two online redo log groups
     logfile
       group 1 ('$ORACLE_HOME/dbs/dupdb_log_1_1.f',
                '$ORACLE_HOME/dbs/dupdb_log_1_2.f') size 200K,
       group 2 ('$ORACLE_HOME/dbs/dupdb_log_2_1.f',
                '$ORACLE_HOME/dbs/dupdb_log_2_2.f') 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.

To duplicate a database using the set auxname command:

1. Use an operating system utility to copy your parameter file from its location in the target host directory structure to the same location in the duplicate host directory structure. Set all *_DEST and *_PATH initialization parameters that specify a pathname.
2. Use SQL*Plus to start the duplicate instance without mounting it:

   ```
   SQL> startup nomount pfile=initDUPDB.ora;
   ```

3. Use SQL*Plus to mount or open the target database if it is not already mounted or open:

   ```
   SQL> startup pfile=initPROD1.ora;
   ```

4. The auxiliary instance must be accessible via Net8. Before proceeding, use SQL*Plus to ensure that you can establish a connection to the auxiliary instance. Note that you must connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

5. Use RMAN to connect to the target database, the duplicate database, and (if you use one) the recovery catalog database. In this example, connection is established without a recovery catalog using operating system authentication:

```
% rman target / auxiliary sys/sys_pwd@dupdb
```

In this example, user SCOTT has SYSDBA privileges and a net service name is used for the target:

```
% rman auxiliary scott/tiger@dupdb target sys/sys_pwd@prod
```

In this example, connection is established to three databases, all using net service names:

```
% rman catalog rman/rman@rcat target sys/sys_pwd@prod1 auxiliary scott/tiger@dupdb
```

6. Set the auxiliary names for your datafiles. For example, enter the following:

```
# set auxiliary names for the datafiles
set auxname for datafile 1 to '/oracle/auxfiles/aux_1.f';
set auxname for datafile 2 to '/oracle/auxfiles/aux_2.f';
...
set auxname for datafile n to '/oracle/auxfiles/aux_n.f';
```

7. Perform the following operations:

   - Allocate at least one auxiliary channel.
   - Specify the same number of redo log members and groups that are used in your target database.

```
run {
     # allocate at least one auxiliary channel of type disk or tape
     allocate auxiliary channel dupdb1 type 'sbt_tape';
     . . .
     # issue the duplicate command
     duplicate target database to dupdb
     . . .
     # create at least two online redo log groups
     logfile
       group 1 ('$ORACLE_HOME/dbs/dupdb_log_1_1.f',
                '$ORACLE_HOME/dbs/dupdb_log_1_2.f') size 200K,
       group 2 ('$ORACLE_HOME/dbs/dupdb_log_2_1.f',
                '$ORACLE_HOME/dbs/dupdb_log_2_2.f') 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.

8. Unspecify the auxiliary names for your datafiles so that they are not overwritten by mistake. For example, enter the following:

```
# un-specify auxiliary names for the datafiles
set auxname for datafile 1 to null;
set auxname for datafile 2 to null;
...
set auxname for datafile n to null;
```

Creating a Duplicate Database on the Local Host

When creating a duplicate database on the same host as your target database, follow the same procedure as for duplicating to a remote host with a different directory structure ("Duplicating a Database on a Remote Host with a Different Directory Structure" on page 7-12).

Note that you can duplicate your database to the same $ORACLE_HOME as your target, but you must convert the filenames using the same methods used for conversion on a separate host.

WARNING: Do not use the nofilenamecheck option when duplicating to the same $ORACLE_HOME as your primary database. If you do, you may overwrite your target database files or cause the duplicate command to fail with an error.

Duplication Scenarios

Following are some useful scenarios for creating a duplicate database:

- Setting New Filenames Manually
- Resynchronizing the Duplicate Database with the Target Database
- Creating a Non-Current Duplicate Database

Setting New Filenames Manually

This example assumes the following:

- You are using recovery catalog database RCAT.
- Your target database is on HOST1 and contains nine datafiles.

- You want to duplicate your target database to database DUPDB on remote host HOST2.
- HOST1 and HOST2 use different file systems.
- You want to store all the datafiles in HOST2 in the `/oracle/dbs` sub-directory and use the `tbs_*` prefix for each datafile.
- You have used an operating system utility to copy your parameter file from HOST1 to an appropriate location in HOST2.
- You have reset all *_DEST and *_PATH initialization parameters that specify a pathname.
- You have disk copies or backup sets stored on disk for all the datafiles and archived redo logs in the target database, and have manually moved them to HOST2 using an operating system utility.
- You want to have two online redo logs groups, each with two members of size 200K.

```
connect target;
connect catalog rman/rman@rcat;
connect auxiliary sys/change_on_install@dupdb;
run {
     allocate auxiliary channel dupdb1 type disk;
     allocate auxiliary channel dupdb2 type disk;
     allocate auxiliary channel dupdb3 type disk;
     allocate auxiliary channel dupdb4 type disk;
     set newname for datafile 1 TO '$ORACLE_HOME/dbs/tbs_01.f';
     set newname for datafile 2 TO '$ORACLE_HOME/dbs/tbs_02.f';
     set newname for datafile 3 TO '$ORACLE_HOME/dbs/tbs_03.f';
     set newname for datafile 4 TO '$ORACLE_HOME/dbs/tbs_04.f';
     set newname for datafile 5 TO '$ORACLE_HOME/dbs/tbs_05.f';
     set newname for datafile 6 TO '$ORACLE_HOME/dbs/tbs_06.f';
     set newname for datafile 7 TO '$ORACLE_HOME/dbs/tbs_07.f';
     set newname for datafile 8 TO '$ORACLE_HOME/dbs/tbs_08.f';
     set newname for datafile 9 TO '$ORACLE_HOME/dbs/tbs_09.f';
     duplicate target database to dupdb logfile
       group 1 ('$ORACLE_HOME/dbs/log_1_1.f',
                '$ORACLE_HOME/dbs/log_1_2.f') size 200K reuse,
       group 2 ('$ORACLE_HOME/dbs/log_2_1.f',
                '$ORACLE_HOME/dbs/log_2_2.f') size 200K reuse;
}
```

Resynchronizing the Duplicate Database with the Target Database

This example makes the same assumptions as in "Setting New Filenames Manually" on page 7-17. Additionally, it assumes that you want to update your 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/rman@rcat
connect auxiliary sys/change_on_install@dupdb

# set auxiliary names for the datafiles only once
set auxname for datafile 1 TO '$ORACLE_HOME/dbs/tbs_01.f';
set auxname for datafile 2 TO '$ORACLE_HOME/dbs/tbs_02.f';
set auxname for datafile 3 TO '$ORACLE_HOME/dbs/tbs_03.f';
set auxname for datafile 4 TO '$ORACLE_HOME/dbs/tbs_04.f';
set auxname for datafile 5 TO '$ORACLE_HOME/dbs/tbs_05.f';
set auxname for datafile 6 TO '$ORACLE_HOME/dbs/tbs_06.f';
set auxname for datafile 7 TO '$ORACLE_HOME/dbs/tbs_07.f';
set auxname for datafile 8 TO '$ORACLE_HOME/dbs/tbs_08.f';
set auxname for datafile 9 TO '$ORACLE_HOME/dbs/tbs_09.f';

# Create the duplicate database. Issue the same command daily
# to re-create the database, thereby keeping the duplicate
# in sync with the target.
run {
     # allocate auxiliary channels
     allocate auxiliary channel dupdb1 type disk;
     allocate auxiliary channel dupdb2 type disk;
     allocate auxiliary channel dupdb3 type disk;
     allocate auxiliary channel dupdb4 type disk;
     duplicate target database to dupdb
       logfile
         group 1 ('$ORACLE_HOME/dbs/log_1_1.f',
                  '$ORACLE_HOME/dbs/log_1_2.f') size 200K reuse,
         group 2 ('$ORACLE_HOME/dbs/log_2_1.f',
                  '$ORACLE_HOME/dbs/log_2_2.f') size 200K reuse;
}
```

Creating a Non-Current Duplicate Database

This example assumes the following:

- Your target database PROD1 and duplicate database DUPDB are on different hosts but have the exact same file structure.
- You want to name the duplicate database files exactly like the target database files.

- You are not using a recovery catalog.
- You want to recover the duplicate database to one week ago.

```
connect target sys/change_on_install@prod1
connect auxiliary sys/sysdba@dupdb
run {
     set until time 'sysdate-7';
     allocate auxiliary channel dupdb1 type 'sbt_tape';
     allocate auxiliary channel dupdb2 type 'sbt_tape';
     duplicate target database to dupdb
     nofilenamecheck;
}
```

8

Performing Point-in-Time Recovery with Recovery Manager

This chapter describes how to use Recovery Manager (RMAN) to perform tablespace point-in-time recovery (TSPITR), and includes the following 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.

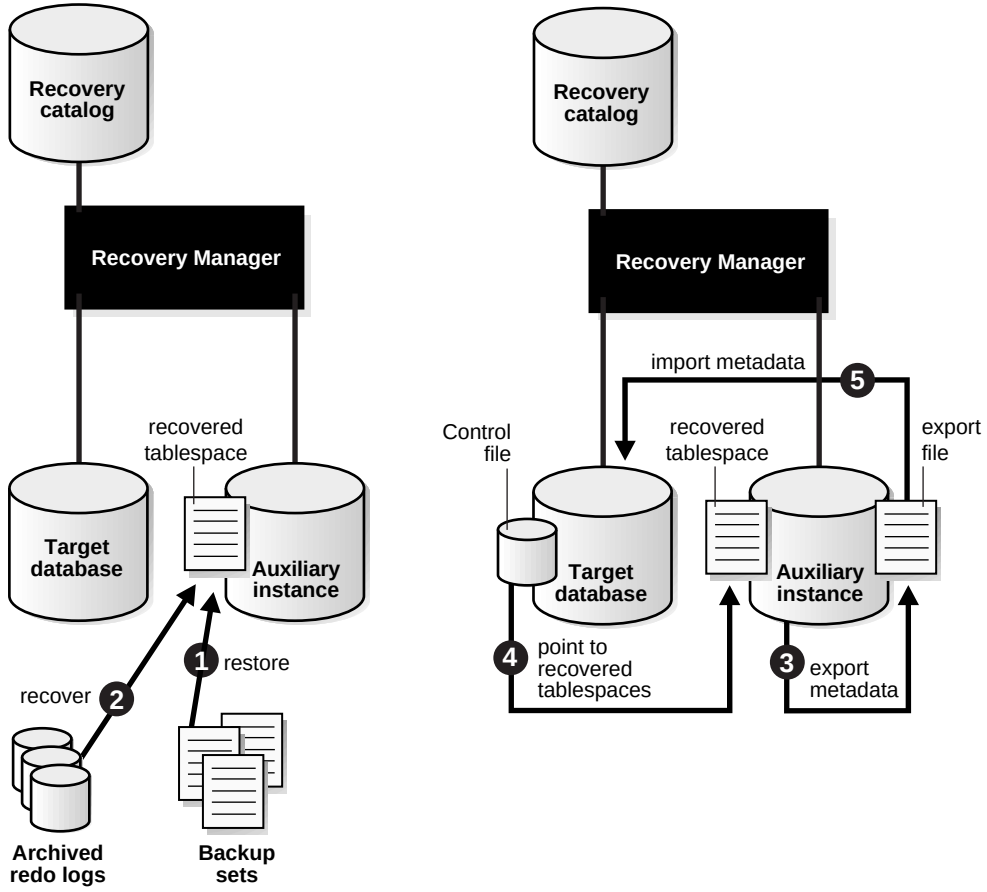
RMAN TSPITR is most useful for recovering:

- An erroneous DROP TABLE or TRUNCATE TABLE operation.
- 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 8–1 illustrates, Recovery Manager does the following:

1. Restores the specified tablespace backups to a temporary auxiliary instance.
2. Recovers the tablespace.
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 8–1 RMAN TSPITR



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 requiring TSPITR to be performed on them.

Auxiliary Set
Any other items required for TSPITR, including:

- Backup control file
- SYSTEM tablespace
- Datafiles containing rollback segments
- Temporary tablespace (optional). A small space is required by Export for sort operations (for more information on sort space issues, see "Responding to Unsuccessful TSPITR" on page 8-12).

Planning for TSPITR

Recovery Manager 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
- Researching and Resolving Inconsistencies
- Managing Data Relationships

Note: Many of the limitations and planning steps in this chapter can also be found in the chapter on O/S TSPITR in *Oracle8i 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, note these restrictions:

- Because RMAN has no historical record of the rollback segments in TSPITR, RMAN assumes that the current rollback segments were the same segments present at the time to which recovery is performed.

- If RMAN recovers to a remote time, Oracle may have reused the control file records of the copies and backups, thus making it impossible to perform TSPITR.

Understanding General Restrictions

When performing RMAN TSPITR, you *cannot*:

- Run the target and auxiliary databases on separate nodes. The target and auxiliary databases can be in a cluster configuration, however, using 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 database—the auxiliary database is for recovery only.
- Use existing backups of the recovery set datafiles for recovery after TSPITR is complete. Instead, take fresh backups of the recovered files. If you attempt to recover using a backup taken prior to performing TSPITR, recovery fails.
- 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
  - Tables with VARRAY columns
  - Tables with nested tables
  - Tables with external files
  - Snapshot logs
  - Snapshot tables
  - Objects owned by SYS (including rollback segments)

WARNING: Do not perform RMAN TSPITR for the first time on a production system or when you have a time constraint.

Researching and Resolving Inconsistencies

The primary issue for RMAN TSPITR is the possibility of application-level inconsistencies between tables in recovered and unrecovered tablespaces due to implicit rather than explicit referential dependencies. Note the following issues and have the means to resolve possible inconsistencies before proceeding:

- RMAN Only Supports Recovery Sets Containing Whole Tables
- Recovery Manager Does Not Support Tables Containing Rollback Segments
- TS_PITR_CHECK Does Not Check for Objects Owned by SYS

RMAN Only Supports Recovery Sets Containing Whole Tables

Note: This limitation is specific to RMAN TSPITR.

RMAN TSPITR only supports recovery sets that contain whole tables. For example, if you perform RMAN TSPITR on partitioned tables and spread partitions across multiple tables, RMAN returns an error message during the export phase. Recovery sets that contain either tables without their constraints or the constraints without the table also result in errors.

Recovery Manager Does Not Support Tables Containing Rollback Segments

Note: This limitation is specific to RMAN TSPITR.

If you are performing O/S TSPITR, you can take rollback segments in the recovery set offline—thus preventing changes being made to the recovery set before recovery is complete. RMAN TSPITR does not support recovery of tablespaces containing rollback segments. For more information about TSPITR and rollback segments, see the O/S TSPITR chapter in *Oracle8i Backup and Recovery Guide*.

TS_PITR_CHECK Does Not Check for Objects Owned by SYS

The TS_PITR_CHECK view provides information on dependencies and restrictions that can prevent TSPITR from proceeding. TS_PITR_CHECK does not provide information, however, about dependencies and restrictions for objects owned by SYS.

If any objects—including rollback segments—owned by SYS are in the recovery set, then there is no guarantee that you can successfully recover these objects. TSPITR utilizes the Export and Import utilities, which do not support objects owned by SYS. To find out which recovery set objects are owned by SYS, issue the following statement:

```
SELECT object_name, object_type
FROM sys.dba_objects
WHERE tablespace_name IN ('tablespace_name_1','tablespace_name_2',
      'tablespace_name_n')
AND owner = 'SYS';
```

> **See Also:** The O/S TSPITR chapter in *Oracle8i Backup and Recovery Guide* for more details about the TS_PITR_CHECK view.

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:** The O/S TSPITR chapter in *Oracle8i Backup and Recovery Guide*.

Preparing the Auxiliary Instance for TSPITR

Satisfy the following requirements before performing RMAN TSPITR:

- Create an Oracle Password File for the Auxiliary Instance
- Create a Parameter File for the Auxiliary Instance
- Start the Auxiliary Instance
- Ensure Net8 Connectivity to the Auxiliary Instance
- Start the Recovery Manager Command Line Interface

Create an Oracle Password File for the Auxiliary Instance

For information about creating and maintaining Oracle password files, see the *Oracle8i Administrator's Guide*.

Create a Parameter File for the Auxiliary Instance

Create an `init.ora` file for the auxiliary instance and set the following required parameters:

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`.
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 using **set auxname.**
LOG_FILE_NAME_CONVERT	Patterns to convert filenames for the online redo logs of the auxiliary database.
CONTROL_FILES	A different value from the CONTROL_FILES parameter in the target parameter file.

Set other parameters as needed, including the parameters that allow you to connect as SYSDBA through Net8.

Following are examples of the `init.ora` parameter settings for the auxiliary instance.

```
DB_NAME=prod1
LOCK_NAME_SPACE=_prod1
CONTROL_FILES=/oracle/aux/cf/aux_prod_cf.f
DB_FILE_NAME_CONVERT=("/oracle/prod/datafile","/oracle/aux/datafile")
LOG_FILE_NAME_CONVERT=("/oracle/prod/redo_log","/oracle/aux/redo_log")
```

> **See Also:** "Tuning TSPITR Performance" on page 8-13 for details about DB_FILE_NAME_CONVERT, and *Net8 Administrator's Guide* for more information about Net8.

Note: After setting these parameters, ensure that you do not overwrite the `init.ora` settings for the production files at the target database.

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):

```
SQL> connect sys/aux_pwd@aux_str;
SQL> startup nomount pfile='/oracle/aux/dbs/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.

Ensure Net8 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.

Start the Recovery Manager Command Line Interface

Use one of the following methods to start the RMAN command line interface:

- Connect from the O/S Command Line
- Connect from the RMAN Prompt

Connect from the O/S Command Line To connect to the auxiliary instance, target instance, and optional recovery catalog, supply the following information when starting Recovery Manager:

```
% rman target sys/target_pwd@target_str catalog rman/cat_pwd@cat_str auxiliary  \
> sys/aux_pwd@aux_str
```

Where:

sys	User with SYSDBA privileges
rman	Owner of the recovery catalog
target_pwd	The password for connecting as SYSDBA specified in the target database's `orapwd` file
target_str	The net service name for the target database
cat_pwd	The password for user RMAN specified in the recovery catalog's `orapwd` file
cat_str	The net service name for the recovery catalog database

aux_pwd	The password for connecting as SYSDBA specified in the auxiliary database's `orapwd` file.
aux_str	The net service name for the auxiliary database.

Connect 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:

```
% rman
RMAN> connect auxiliary sys/aux_pwd@aux_str
RMAN> connect target target sys/target_pwd@target_str
RMAN> connect catalog rman/cat_pwd@cat_str
```

Performing TSPITR

After you have completed all planning requirements, perform RMAN TSPITR. Issue the following commands within **run**, where *tablespace_list* is the list of tablespace names in the recovery set and *recovery_end_time* is the point to which you want to recover:

```
allocate auxiliary channel . . .
recover tablespace tablespace_list until recovery_end_time;
```

You must allocate at least one auxiliary channel with the **allocate auxiliary channel** command.

Note: The tablespace recovery set should not contain the SYSTEM tablespace or any tablespace with rollback segments.

The following example statement performs RMAN TSPITR on tablespaces TBS_2 and TBS_3 to 8 p.m. on January 10, 1999:

```
run {
     allocate auxiliary channel dev1 type 'sbt_tape';
     recover tablespace tbs_2, tbs_3 until time 'Jan 10 1999 20:00:00';
}
```

Recovery Manager automatically performs the following steps during TSPITR:

1. Restores the datafiles to the auxiliary instance.
2. Recovers the restored datafiles to the specified time.
3. Opens the auxiliary database with the RESETLOGS option.

4. 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.
5. Closes the auxiliary database.
6. 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.
7. Imports the dictionary metadata that was exported from the auxiliary database, 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, it performs a restore operation and does not execute a switch. If you have configured names for the datafiles with the **set auxname** command, and suitable datafile copies exist in those **auxname** locations, RMAN optimizes away the restore and performs a switch to the **auxname** datafile copy.

Preparing the Target Database for Use After TSPITR

The tablespaces in the recovery set remain offline until after RMAN TSPITR completes successfully.

To prepare the target database for re-use after TSPITR:

1. Make backups of tablespaces in the recovery set before bringing these tablespaces online. Note that all previous backups of datafiles in the recovery set are no longer valid. For example, this command backs up tablespace TBS_4:

```
run {
     allocate channel ch1 type disk;
     backup tablespace tbs_4;
}
```

2. Bring the recovered tablespaces online. For example, enter:

```
sql "ALTER TABLESPACE TBS_4 ONLINE";
```

3. Because the auxiliary database is not usable after a successful RMAN TSPITR, release the memory by shutting down the database:

```
shutdown abort;
```

4. Delete the following:
   - Auxiliary set datafiles restored to temporary locations during RMAN TSPITR
   - Auxiliary database control files
   - Auxiliary database redo log files

Responding to Unsuccessful TSPITR

A variety of problems can cause TSPITR to abort. For example, if there is a conflict between the target database and the converted filename, 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.txt` file (in UNIX, it is located in `$ORACLE_HOME/admin`). This file contains the following:

```
#
# tsiptr_7: do the incomplete recovery and resetlogs.  This member is used once.
#
define tspitr_7
<<<
# make the controlfile point at the restored datafiles, then recover them
recover clone database tablespace &1&;
sql clone "alter 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, follow the procedure below.

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 as in "Performing TSPITR" on page 8-10.

Tuning TSPITR Performance

This section describes steps you can take to tune the performance of RMAN TSPITR:

- Specify a New Name for Datafiles in Auxiliary Set Tablespaces
- Set the Auxiliary Name and Use a Datafile Copy for Recovery Manager TSPITR
- Use the Converted Filename in the Auxiliary Control File
- Summary: Datafile Naming Methods

Specify a New Name for Datafiles in Auxiliary Set Tablespaces

Recovery Manager restores and recovers all datafiles belonging to the tablespaces in the recovery set and auxiliary set at the auxiliary instance. Note that the auxiliary set includes the SYSTEM tablespace plus all the tablespaces with rollback segments.

Specify a new name for any datafiles in the auxiliary set tablespace using the **set newname** Recovery Manager command. 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 `init.ora` file. For example, to rename `datafile 2` to `new_df_name.f` enter:

```
set newname for datafile 2 to '/oracle/dbs/new_df_name.f';
```

You can specify a new name for any datafiles in recovery set tablespaces. If you specify a new name, the datafiles replace the original datafiles in the target control file—so the new filenames replace the existing filenames.

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.

Set the Auxiliary Name and Use a Datafile Copy for Recovery Manager TSPITR

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.

Recovery Manager uses 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 via the following command (where *filename* is the datafile name or number, and *auxiliary_datafile_name* is the datafile auxiliary name):

   ```
   set auxname for datafile filename to auxiliary_datafile_name;
   ```

2. The datafile copy was made before the time specified in the *untilClause* using the following RMAN command (where *'filename'* is the datafile filename):

   ```
   run {
        copy datafile 'filename' to auxname;
        ...
   }
   ```

Examples

The following commands are examples of the conditions required by Recovery Manager:

```
set auxname for datafile '/oracle/prod/datafile_1_1.dbf'
  to '/oracle/prod_copy/datafile_1_1.dbf';

run {
     allocate channel ch1 type disk;
     copy datafile '/oracle/prod/datafile_1_1.dbf'
     to auxname;
}
```

Recovery Manager does not use a datafile copy if you use **set newname** for the same datafile.

If Recovery Manager uses a datafile copy and TSPITR completes successfully, the *auxiliary_datafile_name* is marked DELETED in the recovery catalog. The original

datafile at the target is replaced by this datafile copy after RMAN TSPITR completes.

Use the Converted Filename in the Auxiliary Control File

If neither a new name nor auxiliary name is set for a datafile in an auxiliary set tablespace, Recovery Manager can use the converted filename specified in the auxiliary database control file to perform the restore and recovery. Recovery Manager 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 the file at the auxiliary name is unusable, Recovery Manager uses the original location of the datafile.

Summary: Datafile Naming Methods

The following commands and parameters are used to name datafiles in the auxiliary and recovery sets during TSPITR. The order of precedence in the table goes top to bottom, so **set newname** takes precedence over **set auxname** and DB_FILE_NAME_CONVERT:

	Command/Parameter	Auxiliary Set	Recovery Set
1	**set newname**	X	X
2	**set auxname**	X	X
3	DB_FILE_NAME_CONVERT	X	

If filenames are not converted in the auxiliary set, RMAN signals an error during TSPITR.

9

Recovery Manager Troubleshooting

This chapter describes how to troubleshoot and debug Recovery Manager. It includes the following topics:

- Interpreting Message Output
- Testing the Media Management API
- Monitoring RMAN Jobs
- Terminating an RMAN Session
- Troubleshooting Scenarios

Interpreting 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 addresses the following topics:

- Identifying Types of Message Output
- Identifying Error Codes
- Interpreting RMAN Error Stacks
- Interpreting Debugging Output

Identifying Types of Message Output

Output that is useful for troubleshooting failed 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 the **log** option on the command line ■ A log file created by re-directing standard output using operators at the command line	Describes actions relevant to the RMAN job as well as error messages generated by RMAN, the server, and the media vendor.
RMAN trace file	RMAN's **debug** command	The file specified using the **trace** parameter at the command line.	Contains exhaustive output on the creation and execution of PL/SQL program units.
`alert.log`	Oracle database server	The directory specified in the USER_DUMP_DEST initialization parameter.	Contains a chronological log of errors, `init.ora` settings, and administration operations. Records values for overwritten control file records (see "Monitoring the Overwriting of Control File Records" on page 3-47).
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.

Type of Output	Produced By	Location	Description
`sbtio.log`	Third-party media management software	The directory specified in the USER_DUMP_DEST initialization parameter.	Contains information on the functioning of the media management device.
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.

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:`

Explanations for `RMAN` and `ORA` error codes are in *Oracle8i Error Messages*.

RMAN Error Message Numbers

Table 9–1 indicates the error ranges for common RMAN error messages, all of which are described in *Oracle8i Error Messages*:

Table 9–1 RMAN Error Message Ranges **(Page 1 of 2)**

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

Table 9–1 RMAN Error Message Ranges **(Page 2 of 2)**

Error Range	Cause
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
20000-20999	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:
```

Below is the list of 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 9–2 Media Manager Error Message Ranges

Cause	No.	Message
sbtopen	7000	Backup file not found (only returned for read)
	7001	File exists (only returned for write)
	7002*	Bad mode specified
	7003	Bad 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

Table 9–2 Media Manager Error Message Ranges

Cause	No.	Message
sbtclose	7020*	Bad th to sbtclose (no sbtopen done)
	7021*	Bad 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*	Bad th to sbtwrite
	7042	I/O error
	7043	O/S error
	7044*	Invalid argument(s) to sbtwrite
sbtread	7060*	Bad th to sbtread
	7061	EOF encountered
	7063	I/O error
	7064	O/S error
	7065*	Invalid argument(s) to sbtread
sbtremove	7080	Backup file not found
	7081	Backup file being used
	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:

1. Because most of the messages in the error stack are not meaningful for the purposes of troubleshooting, try to identify the one or two errors that are most important.
2. 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.
3. Read the messages from the bottom up, because this is the order in which RMAN issues the messages. The first one or two errors issued are usually the most informative.
4. Identify the basic type of error according to the error range chart in Table 9–1 and then refer to *Oracle8i Error Messages* for information on the most important messages.

Interpreting RMAN Errors: Example

You attempt the following backup of tablespace TBS_99 and receive the following message:

```
RMAN> run{allocate channel c1 type disk; backup tablespace tbs_99;}

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: c1
RMAN-08500: channel c1: sid=8 devtype=DISK

RMAN-03022: compiling command: backup
RMAN-03026: error recovery releasing channel resources
RMAN-08031: released channel: c1
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure during compilation of command
RMAN-03013: command type: backup
RMAN-06038: recovery catalog package detected an error
RMAN-20202: tablespace not found in the recovery catalog
RMAN-06019: could not translate tablespace name "TBS_99"
```

You first note that the error occurred after RMAN compiled and executed the **allocate** command, but before it could execute the **backup** command. You read the last two messages in the stack first and immediately see the problem: no tablespace

called TBS_99 appears in the recovery catalog. You conclude that either TBS_99 does not exist in the database, or it does exist in the database but the recovery catalog does not yet know about it.

Interpreting Server Errors: Example

Assume that you attempt to back up your database and receive the following error:

```
RMAN> run{allocate channel c1 type disk;backup database;}

RMAN-03022: compiling command: allocate
RMAN-03026: error recovery releasing channel resources
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure during compilation of command
RMAN-03013: command type: allocate
RMAN-06004: ORACLE error from recovery catalog database: ORA-03113: end-of-file on
communication channel
RMAN-06097: text of failing SQL statement: begin dbms_rcvman . resetAll ; dbms_rcvman .
set_package_constants ; :
RMAN-06099: error occurred in source file: krmk.pc, line: 14531
```

As suggested, you start reading from the bottom up. The first message is numbered 6099, so you know that RMAN was not able to compile the command. The third message from the bottom is more specific: the error came from the recovery catalog database. The `ORA-03113` message indicates that there was a problem communicating with the recovery catalog server.

You then query the recovery catalog database and discover that it was in the process of being shut down when you executed the **run** command. Consequently, the job failed.

Interpreting Media Management Errors: Example

Media management errors 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-03007: retryable error occurred during execution of command: allocate
RMAN-07004: unhandled exception during command execution on channel c4
RMAN-10032: unhandled exception during execution of job step 4: ORA-06512: at line 158
RMAN-10035: exception raised in RPC: ORA-19624: operation failed, retry possible
ORA-19506: failed to create sequential file, name="df_99_1", parms=""
ORA-27007: failed to open file
HP-UX Error: 1003: Unknown system error
```

```
Additional information: 7004
Additional information: 1
ORA-06512: at "SYS.DBMS_BACKUP_RESTORE", line 410
RMAN-10031: ORA-19624 occurred during call to DBMS_BACKUP_RESTORE.DEVICEALLOCATE
```

Following the suggestions for reading error message stacks, you look for the `Additional information:` line and notice:

```
Additional information: 7004
```

You discover that error 7004 means that RMAN is unable to connect to the tape device. For some reason, RMAN is not recognizing your media management software.

Note also that when you read from the bottom up, the first line says that an error occurred during a call to a PL/SQL program unit called DEVICEALLOCATE. Also, the first message below the stack banner says that there was an error executing the **allocate** command. All of this information indicates that RMAN was not able to allocate a tape channel because it was unable to recognize the tape device.

Interpreting Debugging Output

If the standard RMAN error message stack is not generating sufficient information, then use the RMAN **debug** command to generate more extensive output. Use debugging for the following purposes:

- To see the actual PL/SQL programs generated by RMAN (all backup and restore operations are performed through PL/SQL procedures and functions).
- To determine where an RMAN command is failing.

Because debugging output can be voluminous, redirect the output to a trace file to prevent overloading the current log file or standard output. The debugging output displays the following detailed information:

- All SQL statements executed during RMAN command compilation
- The result of SQL statement execution
- Information generated by the recovery catalog PL/SQL packages

Interpreting Debugging Output: Example

RMAN debugging output is so highly detailed that you may find yourself unable to distinguish the useful from the useless information. Assume that you execute the following command in debug mode:

```
run{ allocate channel c1 type disk; backup tablespace TBS_5, TBS_6; }
```

The command fails. See the following subset of the **debug** output for this failed command (note the commented lines in bold):

```
EXEC SQL AT RCVCAT begin :name := upper ( :name ) ; dbms_rcvman . translateTablespace (
:name ) ; end ;
   sqlcode=0
      :b1 = "TBS_5"

/* RMAN calls the translateTablespace procedure to translate tablespace  *
 * name TBS_5 into its constituent datafiles and then calls getDataFile  *
 * to get information about each of the datafiles in tablespace TBS_5.   */

EXEC SQL AT RCVCAT begin dbms_rcvman . getDataFile ( :fno , :crescn , :cretime:cretime_i ,
:fname:fname_i , :tsname , :status:status_i , :blksize , :kbytes:kbytes_i ,
:blocks:blocks_i , :urscn , :stopscn:stopscn_i , :read_only ) ; end ;
   sqlcode=0
      :b1 = 14
      :b2 = 33410
      :b3 = "11-JUN-99"
      :b4 = "/vobs/oracle/dbs/tbs_51.f"  /* datafile name */
      :b5 = "TBS_5"
      :b6 = <EMPTY STRING>
      :b7 = 2048
      :b8 = 500
      :b9 = 250
      :b10 = 0
      :b11 = NULL
      :b12 = 0

EXEC SQL AT RCVCAT begin dbms_rcvman . getDataFile ( :fno , :crescn , :cretime:cretime_i ,
:fname:fname_i , :tsname , :status:status_i , :blksize , :kbytes:kbytes_i ,
:blocks:blocks_i , :urscn , :stopscn:stopscn_i , :read_only ) ; end ;
   sqlcode=0
      :b1 = 15
      :b2 = 33416
      :b3 = "11-JUN-99"
      :b4 = "/vobs/oracle/dbs/tbs_52.f"  /* datafile name */
      :b5 = "TBS_5"
      :b6 = <EMPTY STRING>
      :b7 = 2048
      :b8 = 500
      :b9 = 250
      :b10 = 0
      :b11 = NULL
      :b12 = 0

EXEC SQL AT RCVCAT begin dbms_rcvman . getDataFile ( :fno , :crescn , :cretime:cretime_i ,
:fname:fname_i , :tsname , :status:status_i , :blksize , :kbytes:kbytes_i ,
```

```
:blocks:blocks_i , :urscn , :stopscn:stopscn_i , :read_only ) ; end ;
   sqlcode=-1405

/* RMAN tries to translate tablespace name TBS_6 and get the datafiles, *
 * but is unable to identify any datafiles for this tablespace.         */

EXEC SQL AT RCVCAT begin :name := upper ( :name ) ; dbms_rcvman . translateTablespace (
:name ) ; end ;
   sqlcode=0
      :b1 = "TBS_6"

EXEC SQL AT RCVCAT begin dbms_rcvman . getDataFile ( :fno , :crescn , :cretime:cretime_i ,
:fname:fname_i , :tsname , :status:status_i , :blksize , :kbytes:kbytes_i ,
:blocks:blocks_i , :urscn , :stopscn:stopscn_i , :read_only ) ; end ;
   sqlcode=-20202
krmicomp: error 6038 signalled during compilation
RMAN-03026: error recovery releasing channel resources
krmkdps: this_db_key=1
krmkdps: this_dbinc_key=2
krmkdps: until_scn=
krmkdps: until_time=
krmkdps: completed_after=
krmkdps: completed_before=
krmkdps: like_pattern=
krmkdps: RA_kindMask=255
krmkdps: all_flag=0
krmqclean: the compiled command tree is:
CMD type=cleanup id=1 status=NOT STARTED
```

As you can see, the debugging output is voluminous—and the sample included is just one portion of the total output. Nevertheless, the output is often useful for pinpointing which PL/SQL package, procedure, or function was unable to execute successfully.

Testing the Media Management API

On specified platforms, Oracle provides a diagnostic tool called `sbttest`. This utility performs a simple test of the tape library by acting as the Oracle database server and attempting to communicate with the media manager.

Obtaining the 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 SunSolaris, you do not have to relink when using `sbttest`. On other platforms, relinking may be necessary.

Obtaining Online Documentation

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 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.
non-0	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, included in your system path, and linked with Oracle by typing `sbttest` at the command line:

   ```
   % sbttest
   ```

   If the program is operational, 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, 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
   ```

Monitoring RMAN Jobs

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:

View	Description
V$PROCESS	Identifies currently active processes.
V$SESSION	Identifies currently active sessions. Use this view to determine which Oracle database server sessions correspond to which RMAN allocated channels.
V$SESSION_LONGOPS	Provides progress reports on long-running operations.
V$SESSION_WAIT	Lists the events or resources for which sessions are waiting.

RMAN allows you to perform the following checks:

- Correlating Server Sessions with Channels
- Monitoring Job Progress
- Monitoring Job Performance

Correlating Server Sessions with Channels

To identify which server sessions correspond to which RMAN channels, use the **set** command with the **command id** parameter. The **command id** parameter enters the specified string into the CLIENT_INFO column of the V$SESSION dynamic performance view. Join V$SESSION with V$PROCESS to correlate the server session with the channel.

The CLIENT_INFO column of V$SESSION contains information for each Recovery Manager server session. The data appears in one of the following formats:

- **id=***string*

  This form appears for the first connection to the target database established by RMAN.

- **id=***string*, **ch=***channel_id*

  This form appears for all allocated channels.

The SPID column of V$PROCESS identifies the operating system process number.

> **See Also:** *Oracle8i Reference* for more information on V$SESSION and V$PROCESS.

To correlate a process with a channel during a backup:

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. Set the **command id** parameter *after* allocating the channels and then back up the desired object. For example, enter:

```
run {
     allocate channel t1 type disk;
     allocate channel t2 type disk;
     set command id to 'rman';
     backup
       incremental level 0
       filesperset 5
       tablespace 'SYSTEM';
     # optionally, issue a host command to access the operating system prompt
     host;
     sql 'ALTER SYSTEM ARCHIVE LOG ALL';
}
```

3. 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=rman%';

SID        SPID      CLIENT_INFO
---------- --------- ----------------------------------------------------------------
8          21973     id=rman
16         22057     id=rman
17         22068     id=rman,ch=t1
18         22070     id=rman,ch=t2
```

See Also: "set_run_option" on page 10-142 for **set command id** syntax.

Monitoring Job Progress

Monitor the progress of backups, copies, and restores by querying the view V$SESSION_LONGOPS.

Each server session performing a backup, restore, or copy reports its progress compared to the total amount of work required for that particular part of the

restore. For example, if you perform a restore using two channels, and each channel has two backup sets to restore (a total of 4 sets), then each server session reports its progress through a single set. When that set is completely restored, RMAN starts reporting progress on the next set to restore.

1. Start RMAN and connect to the target database and, optionally, the recovery catalog database. For example, enter:

```
% rman target / catalog rman/rman@rcat
```

2. Start an RMAN job. For example, enter:

```
run {
     allocate channel t1 type disk;
     backup database;
}
```

3. 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 backup 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
```

4. 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. Query V$SESSION_WAIT to determine which events are being waited for. For example, enter:

```
SQL> SELECT sid, seconds_in_wait AS sec_wait, event FROM v$session_wait
  2  WHERE wait_time = 0
  3  ORDER BY sid;

       SID   SEC_WAIT EVENT
---------- ---------- ----------------------------------------------
         1  368383335 pmon timer
         2       1097 rdbms ipc message
         3     387928 rdbms ipc message
         4          0 rdbms ipc message
         5       1408 smon timer
         6     386114 rdbms ipc message
         7     387626 rdbms ipc message
         8       1060 SQL*Net message from client
         9       1060 SQL*Net message from client
        12       1060 SQL*Net message from client
        13       2366 SQL*Net message from client
        14       2757 SQL*Net message from client
12 rows selected.
```

Note: The V$SESSION_WAIT view shows only Oracle events, not media manager events.

See Also:

- *Oracle8i Reference* for more information on V$SESSION_LONGOPS
- *Oracle8i Designing and Tuning for Performance* to learn how to use the V$BACKUP_SYNC_IO and V$BACKUP_ASYNC_IO views for detailed progress and performance information on individual files
- *Oracle8i Reference* for descriptions of V$SESSION_LONGOPS, V$BACKUP_SYNC_IO, and V$BACKUP_ASYNC_IO

Monitoring Job Performance

Monitor backup and restore performance by querying V$BACKUP_SYNC and V$BACKUP_ASYNC_IO. For a complete description of the contents of these views and how you can use them to tune backup performance, see *Oracle8i Designing and Tuning for Performance*.

> **See Also:** *Oracle8i Reference* for more information on these V$ views.

Terminating an RMAN Session

You may sometimes need to kill an RMAN job that is hanging. Often, hung jobs occur when RMAN is interfacing with a media manager. The best way to stop RMAN when the connections for the allocated channels are hung in the media manager is to kill the Oracle process of the connections. Be careful when executing this operation because killing the Oracle process may cause problems for the media manager.

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—if using a recovery catalog, none otherwise.
- The connection to the target database, also called the *default channel*.
- A *polling connection* to the target database used for RPC testing of each different connect string used in the **allocate channel** command. By default there is no connect string in **allocate channel** and so there is only one RPC connection.
- 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 default channel, and the polling connections. Target connections that are not hung in the media manager code also terminate: only the target connection 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 is media manager-dependent.

Terminating the catalog connection does not cause RMAN to finish because RMAN is not performing catalog operations. Removing default channel and polling connections cause 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 above.

Terminating an RMAN Session

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 identify and terminate an oracle process that is hung in the media manager code:

This procedure is system-specific. See your operating system-specific documentation for the relevant commands.

1. Obtain the current stack trace for the desired process id using a system-specific utility. For example, on SunSolaris you can use the command `pstack` located in `/usr/proc/bin` to obtain the stack.
2. After the stack is obtained, look for the process with `SBTxxxx` (normally `sbtopen`) as one of its top calls. Note that other layers may appear on top of it.
3. Obtain the stack again after a few minutes. If the same stack trace is returned, then you have identified the hung process.
4. Kill the hung process using a system-specific utility. For example, on SunSolaris execute a `kill -9` command.
5. Repeat this procedure for all hung channels in the media management code.

6. Check that the media manager also clears its processes, otherwise the next backup or restore may still hang due to the previous hang. In some media managers, the only solution is too shut down and restart the media manager daemons. If the documentation from the media manager is unhelpful, ask the media manager technical support for the correct solution.

Troubleshooting Scenarios

This section describes the most common problems encountered when using RMAN:

After Linking to the Media Manager on UNIX, RMAN Fails to Back Up to Tape

In this scenario, you link the media manager with Oracle but still cannot make RMAN back up to tape. You see either of these error messages:

```
# error 1
ORA-19511: SBT error = 4110, errno = 0, BACKUP_DIR environment variable is not set
# error 2
RMAN-008526: channel channel_name: WARNING: Oracle Test Disk API
```

Diagnosis of the Cause

When you install Oracle8*i*, the server kernel is linked with a shared library whose name and location differs depending on your operating system. For example, the

library on SunSolaris in version 8.1 is called `$ORACLE_HOME/lib/libobk.so`. This is a symbolic link that points to the so-called *dummy API* that Oracle links to by default. On SunSolaris, this library is called `libdsbtsh8.so`.

This dummy shared library allows you to use RMAN to test writing to disk so long as you specify BACKUP_DIR in the **parms** parameter of the **allocate channel** command.

Executable	Shared Library
oracle.exe	libobk.so -> libdsbtsh8.so

An SBT error of `4110` for Oracle version 8.1 or an `Additional information:` message of `4110` for Oracle version 8.0 indicates that Oracle is not linked with the media manager API. Instead, Oracle is linked with Oracle's own dummy API. These errors occur because the BACKUP_DIR environment variable is not specified for the channel servicing the backup. One way to set the BACKUP_DIR location is by using the **parms** parameter of the **allocate channel** command.

In Oracle version 8.1, if the `RMAN-8526` message states that the disk API was used, then the BACKUP_DIR environment variable was successfully resolved by the Oracle channel and RMAN made the backup using the disk API. If you do *not* see the `4110` or `RMAN-008526` errors, but RMAN is not making backups to the media manager, then follow the procedure below to determine whether Oracle is using the dummy API.

To determine whether Oracle is using the dummy API:

1. Navigate to $ORACLE_HOME/bin. For example, enter:

   ```
   % cd $ORACLE_HOME/bin
   ```

2. List the dynamic dependencies, that is, the shared objects linked at runtime when you execute Oracle. The command differs depending on your version of UNIX. For example, on Solaris issue:

   ```
   % ldd oracle | grep libobk.so
   ```

3. Examine the output to see whether `libobk.so` is symbolically linked to `libdsbtsh8.so`:

   ```
   libobk.so  ->  libdsbtsh8.so
   ```

   If so, then Oracle is linking to the dummy API instead of your media management API.

4. Test further by trying a backup using the dummy API. Note that you must set the **parms** parameter of the allocate command so that the ENV variable specifies a valid pathname for the BACKUP_DIR destination.

   For example, you can issue:

```
run {
     allocate channel c1 type 'sbt_tape'
       parms="ENV=(BACKUP_DIR=/oracle/work)";
     backup tablespace system;
}
```

   If you see `RMAN-08526` and `RMAN-08525` in the output, then the backup to disk using the dummy API was successful:

```
RMAN-08526: channel c1: WARNING: Oracle Test Disk API
RMAN-08525: backup set complete, elapsed time: 00:00:25
```

Solution

If Oracle is linked to the dummy API instead of your media manager's shared library, then you must tell Oracle to link to the media manager's shared library instead.

> **Note:** Database backups using the disk API are not supported.

To relink to your media management API:

1. Shut down all databases that are using the Oracle executable before removing the old `libobk.so` link. For example, enter:

```
SQL> SHUTDOWN IMMEDIATE
```

2. Delete the old symbolic link for `libobk.so`:

```
% rm $ORACLE_HOME/lib/libok.so
```

3. Create a symbolic link between `libobk.so` and the shared library that you want to use. For example, on SunSolaris you can create the symbolic link `libobk.so` to a shared library such as `liblsm.so` using the `ln` command:

```
% ln  -s  $ORACLE_HOME/lib/libobk.so  $ORACLE_HOME/lib/liblsm.so
```

4. Check that the link is successful by using the `sbttest` test program to back up a file. For example, enter:

```
% sbttest  -testfile
```

After Installing the Media Manager on NT, RMAN Fails to Back Up to Tape

In this scenario, you install the media manager on the Windows platform, but cannot make RMAN back up to tape. Instead, RMAN does one or more of the following:

- Backs up to the `$ORACLE_HOME/database` directory on disk
- Returns an SBT error of `4110` for Oracle version 8.1 or an additional information message of `4110` for Oracle version 8.0
- Fails to perform the backup

Diagnosis of the Cause

If you specify '*sbt_tape*' but RMAN does not use the media management API, then the following scenarios are possible:

- Oracle cannot find the library to load, so it uses the dummy API instead (see "After Linking to the Media Manager on UNIX, RMAN Fails to Back Up to Tape" for a discussion of the dummy API). Windows NT searches for the library called `orasbt.dll` in its default directory path.
- An error occurs in the vendor's DLL. Either the `DllMain()` function returned an error, or the vendor did not implement all of the SBT functions.

To diagnose the cause of the failure:

1. Examine the RMAN message output to determine if the following message appears:

   ```
   RMAN-008526: channel channel_name: WARNING: Oracle Test Disk API
   ```

   If you see this information, then Oracle is linking to the dummy API instead of your media management API. If not, proceed to the next step.

2. If Oracle is not using the dummy API and the backup fails to write to tape, then the problem is probably in your media manager's library. Contact the support team for your media management product and ask them help diagnose why the DLL cannot be loaded.

Solution

If Oracle is using the dummy API successfully, then it cannot find the media management API in its default directory path. You must configure the system so that Oracle can find the correct API.

Note: Database backups using the disk API are not supported.

Follow this procedure:

1. Shut down the Oracle services using the Services dialogue box. If the services are running when the media manager is installed, the new DLL will not be used until the machine is rebooted or the service is stopped and started again.
2. On Windows NT, there are two sets of environment variables: the USER and SYSTEM variables. Place `orasbt.dll` in the SYSTEM path so that the Oracle service can find it.
3. Restart the Oracle service so that the Oracle executable uses the correct DLL.

Backup Job Is Hanging

In this scenario, an RMAN backup job starts as normal and then pauses inexplicably:

```
Recovery Manager: Release 8.1.5.0.0 - Production
RMAN-06005: connected to target database: TORPEDO
RMAN-06008: connected to recovery catalog database
RMAN> run {
2> allocate channel t1 type "SBT_TAPE";
3> backup
4> tablespace system,users; }
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: t1
RMAN-08500: channel t1: sid=16 devtype=SBT_TAPE
RMAN-03022: compiling command: backup
RMAN-03023: executing command: backup
RMAN-08008: channel t1: starting datafile backupset
RMAN-08502: set_count=15 set_stamp=338309600
RMAN-08010: channel t1: including datafile 2 in backupset
RMAN-08010: channel t1: including datafile 1 in backupset
RMAN-08011: channel t1: including current controlfile in backupset
# Hanging here for 30 min now
```

Diagnosis of the Cause

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.

Your first task is to try to determine which of these scenarios is the most likely cause.

To determine the cause of the hang:

1. If you are using a media manager, examine media manager process, log, and trace files for signs of abnormal termination or other errors (see the description of message files in "Identifying Types of Message Output" on page 9-2). If this information is not helpful, proceed to the next step.
2. Restart RMAN and turn on debugging, making sure to specify a trace file to contain the output. For example, enter:

   ```
   % rman target / catalog rman/rman@catdb debug trace = /oracle/log
   ```

3. Re-execute the job:

   ```
   run {
        allocate channel c1 type 'sbt_tape';
        backup tablespace system;
   }
   ```

4. Examine the debugging output to determine where RMAN is hanging (see "Interpreting Debugging Output" on page 9-8 for tips on reading this output). The output will most likely indicate that the last RPC sent from the client to the server was SYS.DBMS_BACKUP_RESTORE.BACKUPPIECECREATE, which is the call that causes the server to interact with the media manager to write the backup data:

   ```
   krmxrpc: xc=6897512 starting longrunning RPC #13 to target: DBMS_BACKUP_RESTORE.
   BACKUPPIECECREATE
   krmxr: xc=6897512 started long running rpc
   ```

5. Check to see what the server processes performing the backup are doing. How many processes are hanging? If only one, check to see what it is doing by querying V$SESSION_WAIT. For example, to determine what process 12 is doing, enter:

   ```
   SELECT * FROM v$session_wait WHERE wait_time = 0 AND sid = 12;
   ```

Solution

Because the causes of a hung backup job can be varied, so are the solutions. The best practice is to look for the simplest solutions first. For example, it is quite common for backup jobs to hang simply because the tape device has completely filled the current cassette and is waiting for a new tape to be inserted.

To learn how to kill an RMAN session that is hanging, see "Terminating an RMAN Session" on page 9-18.

RMAN Fails to Start RPC Call

In this scenario, you run a backup job and receive message output similar to the following:

```
RMAN-08010: channel c8: including datafile number 47 in backupset
RMAN-10030: RPC call appears to have failed to start on channel c9
RMAN-10036: RPC call ok on channel c9
RMAN-08010: channel c3: including datafile number 18 in backupset
```

Diagnosis of Cause

The `RMAN-10030` error message does not usually indicate a problem. The `RMAN-10030` error indicates one of the following:

- The target database instance is slow.
- A timing problem occurred.

Timing problems occur in this way. When RMAN posts a long-term 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 `RMAN-10030` error message.

If the `RMAN-10036` error message does not appear in the output along with `RMAN-10030`, then the backup job encountered a problem. For example, sometimes the `RMAN-10030` message appears when a backup to tape hangs at the beginning of the job.

To determine the cause of the hung backup:

1. If a backup to tape stalls at the beginning, issue the following query:

```
SELECT * FROM V$SESSION_WAIT
WHERE compnam = 'dbms_backup_restore';
```

If Oracle returns no information, then the PL/SQL program performing the backup is hung.

2. Check the media manager to ensure that it did not hang or terminate abnormally. The media manager does not pass all failures back to its `libobk` client. If the client does not receive the information, then Oracle does not get notified that the backup terminated due to MML problems.

Backup Fails with Invalid RECID Error

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
```

Diagnosis of Cause

You probably restored a backup control file created through a non-Oracle mechanism, and then opened the database without performing a RESETLOGS operation. If you had created the backup control file through the RMAN **backup** command or the ALTER DATABASE BACKUP CONTROLFILE statement, then Oracle would have required you to reset the 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 using 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.

Solution

You can follow either of these procedures, although the first procedure is safer and is strongly recommended:

To reset the database using RMAN:

1. Connect to the target database using SQL*Plus:

   ```
   % sqlplus sys/oracle@prod1
   ```

2. Mount the database if it is not already mounted:

   ```
   SQL> ALTER DATABASE MOUNT;
   ```

3. Start cancel-based recovery using the backup control file, then cancel it:

```
SQL> ALTER DATABASE RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE;
SQL> ALTER DATABASE RECOVER CANCEL;
```

4. Open the database with the RESETLOGS option:

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

5. Use RMAN to connect to the target database and recovery catalog:

```
% rman target / rman/rman@rcat
```

6. Issue the RMAN **reset database** command:

```
reset database;
```

7. Take new backups so that you can recover the database if necessary:

```
run {
     allocate channel c1 type disk;
     backup database;
}
```

To create the control file using SQL*Plus:

1. Connect to the target database using SQL*Plus:

```
% sqlplus sys/oracle@prod1
```

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 trace file looks something like the following:

```
*** SESSION ID:(8.1) 1998.12.09.13.26.36.000
*** 1998.12.09.13.26.36.000
# The following statements 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 "PROD1" NORESETLOGS ARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 1
```

```
    MAXLOGHISTORY 1012
LOGFILE
  GROUP 1 '/oracle/dbs/t1_log1.f'  SIZE 200K,
  GROUP 2 '/oracle/dbs/t1_log2.f'  SIZE 200K
DATAFILE
  '/oracle/dbs/tbs_01.f',
  '/oracle/dbs/tbs_02.f',
  '/oracle/dbs/tbs_11.f',
  '/oracle/dbs/tbs_12.f',
  '/oracle/dbs/tbs_21.f',
  '/oracle/dbs/tbs_22.f',
 CHARACTER SET WE8DEC
;
# Configure snapshot controlfile filename
EXECUTE SYS.DBMS_BACKUP_RESTORE.CFILESETSNAPSHOTNAME('/oracle/dbs/snapcf_prod1.f');
# 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;
# No tempfile entries found to add.
```

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 ALL;
ALTER DATABASE OPEN ...;
```

WARNING: 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 logs 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 RMAN cannot detect the problem.

Backup Fails Because of Control File Enqueue

In this scenario, a backup job fails because RMAN cannot make a snapshot control file. The message stack is as follows:

```
RMAN-08502: set_count=11 set_stamp=333299261
RMAN-08010: channel dev1: including datafile 1 in backupset
RMAN-08512: waiting for snapshot controlfile enqueue
RMAN-08512: waiting for snapshot controlfile enqueue
RMAN-20029: cannot make a snapshot controlfile
RMAN-03026: error recovery releasing channel resources
RMAN-08031: released channel: dev1
RMAN-00569: ================error message stack follows================
RMAN-03006: non-retryable error occurred during execution of command: backup

RMAN-07004: unhandled exception during command execution on channel dev1
RMAN-10032: unhandled exception during execution of
job step 1: ORA-06512: at line 90
RMAN-10035: exception raised in RPC: ORA-00230:
operation disallowed: snapshot controlfile enqueue unavailable
ORA-06512: at "SYS.DBMS_BACKUP_RESTORE", line 1826
RMAN-10031: ORA-230 occurred during call to
DBMS_BACKUP_RESTORE.CFILEMAKEANDUSESNAPSHOT
```

Diagnosis of Cause

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:

```
RMAN-08512: 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.

To determine which job is holding the conflicting enqueue:

1. After you see the first `RMAN-08512: waiting for snapshot controlfile enqueue` message, start a new SQL*Plus session on the target database:

   ```
   % sqlplus sys/sys_pwd@prod1
   ```

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-99
```

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 once 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, 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

In this scenario, your database archives automatically to two directories: `/arc_dest` and `arc_dest2`. You tell RMAN to perform a backup and delete the input archived redo logs afterwards:

```
run {
     allocate channel c1 type 'sbt_tape';
     backup database;
     backup archivelog all delete input;
}
```

You then perform a crosscheck to make sure that the logs are gone and discover the following:

```
RMAN> change archivelog all crosscheck;

RMAN-03022: compiling command: change
RMAN-06158: validation succeeded for archived log
RMAN-08514: archivelog filename=/oracle/arch/dest2/arcr_1_964.arc recid=19 stamp=368726072
```

RMAN deleted one set of logs but not the other.

Diagnosis of Cause

This problem is not an error. RMAN deletes only one copy of each input log, so even if you archive to five destinations, RMAN deletes logs from only one directory.

Solution

To force RMAN to delete all existing archived redo logs, allocate multiple channels and specify that each channel back up and delete logs from a different archiving destination. For example, enter:

```
run {
     allocate channel t1 type 'sbt_tape';
     allocate channel t2 type 'sbt_tape';
     backup
       archivelog like '/oracle/arch/dest1/%' channel t1 delete input
       archivelog like '/oracle/arch/dest2/%' channel t2 delete input;
}
```

Backup Fails Because RMAN Cannot Locate an Archived Log

In this scenario, you schedule regular incremental backups of your database. Because RMAN can use incremental backups instead of archived redo logs to perform recovery, you use an operating system utility to delete all archived logs after each backup. The next time you take a backup, you receive this error:

```
RMAN-6089:  archive log NAME not found or out of sync with catalog
```

Diagnosis of Cause

This problem occurs when you delete the archived logs using 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 it thinks still exists.

Solution

The easiest solution is to specify the **delete input** option when backing up archived logs. For example, enter:

```
run {
     allocate channel c1 type 'sbt_tape';
     backup archivelog all delete input;
}
```

The second easiest solution is to issue the following command at the RMAN prompt after deleting the logs using an operating system utility:

```
change archivelog all crosscheck;
```

If the **compatible** parameter in the catalog is set to 8.1.5 or lower, RMAN marks all archived logs it cannot located as having the status DELETED. If **compatible** is set to 8.1.6 or higher, then RMAN removes the records from the repository.

> **See Also:** "configure" on page 10-47 to learn how to set the **compatible** parameter.

RMAN Issues Character Set Errors When You Attempt to Connect to the Target

In this scenario, you are running an 8.1.5 version of RMAN and trying to connect to a version 8.0.4 target database. You receive the following error messages when you try to connect to the target database:

```
% rman catalog rman/rman@rcat

Recovery Manager: Release 8.1.5.0.0 - Production
RMAN-06008: connected to recovery catalog database

RMAN> connect target sys/oscar123@nc0d
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===
RMAN-00571: ===========================================================
RMAN-04005: error from target database: ORA-06550: line 1, column 7:
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
```

Diagnosis of Cause

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.

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> @$ORACLE_HOME/rdbms/admin/dbmsbkrs.sql
```

```
SQL> @$ORACLE_HOME/rdbms/admin/prvtbkrs.plb
```

RMAN Denies Logon to Target Database

Recovery Manager fails with the following errors when attempting to connect to the target database:

```
% rman
Recovery Manager: Release 8.1.5.0.0 - Production

RMAN> connect target sys/change_on_install@inst1

RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-04005: error from target database:
ORA-01017: invalid username/password; logon denied
```

Diagnosis of Cause

Recovery Manager automatically requests a connection to the target database as SYSDBA. In order to connect to the target database as SYSDBA, you must either:

- Be part of the operating system DBA group with respect to the target database. This means that you have the ability to CONNECT INTERNAL to the target database without a password.
- Have created a password file. This requires the use of the `orapwd` command and the initialization parameter REMOTE_LOGIN_PASSWORDFILE.

If the target database does not have a password file, the user you are logged in as must be validated using operating system authentication.

Solution

Either create a password file for the target database or add yourself to the administrator list in your operating system. To learn how to create a password file, see *Oracle8i Administrator's Guide.*

Database Duplication Fails with RMAN-20240

In this scenario, you attempt to duplicate a database to the same host (although it could also be a remote host) using the **duplicate** command, but get the following error stack during compilation of the **recover** command:

```
RMAN-03022: compiling command: recover(3)
RMAN-03023: executing command: recover(3)
```

```
RMAN-08054: starting media recovery
RMAN-08060: unable to find archivelog
RMAN-08510: archivelog thread=1 sequence=6

RMAN-03022: compiling command: recover(4)
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00601: fatal error in recovery manager
RMAN-03012: fatal error during compilation of command
RMAN-03028: fatal error code: 3015
RMAN-03013: command type: Duplicate Db
RMAN-03015: error occurred in stored script Memory Script
RMAN-03002: failure during compilation of command
RMAN-03013: command type: recover
RMAN-03002: failure during compilation of command
RMAN-03013: command type: recover(4)
RMAN-06038: recovery catalog package detected an error
RMAN-20242: specification does not match any archivelog in the recovery catalog
```

Diagnosis of Cause

The problem is probably that the backup of the datafiles is not consistent, that is, the following SQL statement was not issued after the datafile backup:

```
ALTER SYSTEM ARCHIVE LOG CURRENT
```

Consequently, the **duplicate** command is attempting to read the online redo logs for the necessary redo records.

Solution

When creating the duplication script, use the **set until** command to specify a log sequence number for incomplete recovery. For example, to stop recovery at log sequence 5 enter:

```
run {
     set until logseq 5 thread 1;
     allocate auxiliary channel dupdb1 type disk;
     duplicate target database to dupdb;
}
```

> **See Also:** "Creating a Non-Current Duplicate Database" on page 7-19 for more information about performing incomplete recovery during the duplication operation.

UNKNOWN Database Name Appears in Recovery Catalog

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       SKDHRA   4052472287  YES  1           Sep 03 1998 06:45:51
1       19       UNKNOWN  4141147584  NO   1           Jan 08 1999 14:47:28
1       2        SKDHRC   4141147584  YES  14602       Jan 15 1999 15:32:57
```

Diagnosis of Cause

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, 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.

Solution

The UNKNOWN name entry is expected behavior: you should *not* attempt to remove UNKNOWN entries from the recovery catalog. Also, the backups of this incarnation are not usable—at least with this recovery catalog—so RMAN cannot restore them even if you issue a **reset database to incarnation** command.

Part II

Recovery Manager Reference

10

Recovery Manager Command Syntax

This chapter describes, in alphabetical order, Recovery Manager commands and sub-clauses.

Conventions Used in this Reference

This section explains the conventions used in this book including:

- Text
- Syntax Diagrams and Notation
- Code Examples

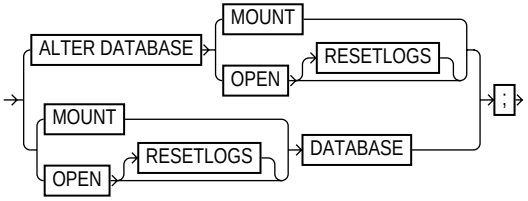
Text

The text in this reference adheres to the following conventions:

UPPERCASE	Uppercase text calls attention to SQL commands and keywords, filenames, column headings in tables and views, and initialization parameters.
bold	Bold text calls attention to Recovery Manager keywords.
italics	Italicized text calls attention to definitions of terms, the names for Recovery Manager parameters and options that are not keywords (for example, *integer*), and sample values for Recovery Manager parameters (for example, **datafile** *tbs_01.f*).

Syntax Diagrams and Notation

Syntax Diagrams This reference uses syntax diagrams to show Recovery Manager commands. These syntax diagrams use lines and arrows to show syntactic structure, as shown here:



This section describes the components of syntax diagrams and gives examples of how to write Recovery Manager commands. Syntax diagrams are made up of these items:

Keywords Keywords have special meanings in Recovery Manager syntax. In the syntax diagrams, keywords appear in square boxes and an uppercase font. When described in text, RMAN keywords appear in lowercase bold, for example, **backup database**. You must use keywords in your RMAN statements exactly as they appear in the syntax diagram, except that they can be either uppercase or lowercase.

The RMAN language is free-form. Keywords must be separated by at least one white-space character, but otherwise there are no restrictions. A command may span multiple lines.

Parameters Parameters act as placeholders in syntax diagrams. In the syntax diagrams, they appear in ovals. When described in text, RMAN parameters appear in lowercase italics, for example, *'filename'*. Parameters are usually:

- Names of database objects (*tablespace_name*)
- Oracle data type names (*date_string*)
- Sub-clauses (*datafileSpec*)

When you see a parameter in a syntax diagram, substitute an object or expression of the appropriate type in your RMAN statement. For example, to write a **duplicate target database to** command, use the name of the duplicate database you want to create, such as *dupdb*, in place of the *database_name* parameter in the syntax diagram.

Some parameter values are enclosed in required or optional quotes. The syntax diagrams show single quotes, though in all cases double quotes are also legal. For example, you specify either '*filename*' or "*filename*". For the **sql** command, it is recommended that you use double quotes.

This lists shows parameters that appear in the syntax diagrams and provides examples of the values you might substitute for them in your statements:

Parameter	Description	Examples
quoted strings such as *'filename'*, *'tablespace_name'*, *'channel_name'*, *'channel_parms'*	A string of characters contained in either single or double quotes, for example, '*filename*' or "*filename*". A quoted string may contain whitespace, punctuation, and RMAN and SQL keywords.	`"?/dbs/cf.f"` `'dev1'`

Parameter	Description	Examples
non-quoted strings such as *channel_id*, *tag_name*, *date_string*	A sequence of characters containing no white-space and no punctuation characters and starting with an alphabetic character.	`ch1`
integer	Any sequence of characters containing only number characters.	`67843`

Reserved Words Table 10–1 is a list of RMAN reserved words. If you use one of these words by itself without surrounding it in quotes, then RMAN generates an error. These are examples of correct and incorrect entries:

```
allocate channel backup type disk;        # incorrect
allocate channel 'backup' type disk;      # correct
backup database tag full;                 # incorrect
backup database tag 'full';               # correct
```

Table 10–1 RMAN Reserved Words **(Page 1 of 2)**

abort	affinity	after	all	allocate	alter	and
append	archivelog	at	auxiliary	auxname	available	backslash
backup	backuppiece	backupset	before	beginline	between	cancel
catalog	change	channel	channel_id	check	clone	clone_cf
clonename	cmdfile	check	clone	clone_cf	clonename	cmdfile
command	completed	connect	controlfile	controlfilecopy	copy	create
crosscheck	cumulative	current	database	datafile	datafilecopy	days
dba	dbid	debug	define	delete	destination	device
disk	diskratio	drop	dump	duplex	duplicate	echo
equal	execute	exit	expired	filesperset	for	force
forever	format	from	full	group	high	host
id	inaccessible	incarnation	include	incremental	input	integer
immediate	job	k	kbytes	level	libparm	library
like	limit	list	log	logfile	logical	logseq
low	maxcorrupt	maxopenfiles	msglog	mask	msgno	maintenance
mount	m	name	need	newname	nochecksum	nocatalog
newline	noredo	normal	nomount	nofilenamecheck	null	of

Table 10–1 RMAN Reserved Words **(Page 2 of 2)**

offline	orphan	obsolete	open	on	off	only
parms	plsql	print	pfile	proxy	pool	pipe
rcvcat	release	reload	replace	replicate	report	recoverable
reset	restart	restore	resync	rman	rpctestrun	readonly
readrate	recover	redundancy	register	reuse	schema	scn
script	send	set	setlimit	setsize	shutdown	size
skip	slaxdebug	sql	startup	step	tablespace	tag
target	test	thread	time	timeout	to	trace
transactional	type	unavailable	uncatalog	underscore	unrecoverable	until
upgrade	validate					

Code Examples

This reference contains many examples of RMAN commands. These examples show you how to use elements of RMAN. The following example shows a **backup** command:

```
run {
     allocate channel ch1 type disk;
     backup database;
}
```

Note that examples appear in a different font from the text.

Command Entries

The description of each command or sub-clause contains the following sections:

Syntax shows the keywords and parameters that make up the statement.

Note: Not all keywords and parameters are valid in all circumstances. Be sure to refer to the "Keywords and Parameters" section of each statement to learn about any restrictions on the syntax.

Purpose describes the basic uses of the statement.

Requirements	lists any requirements and restrictions for proper use of the command.
Keywords and Parameters	describes the purpose of each keyword and parameter. Restrictions and usage notes also appear in this section.
Examples	shows how to use various clauses and options of the statement. Usage notes: Optional sections following the examples provide more information on how and when to use the statement.

Summary of RMAN Commands

The following table provides a functional summary of RMAN commands. Note that all release 8.0 commands still work with the release 8.1 RMAN executable.

Table 10–2 Recovery Manager Commands

Command	Purpose
"allocate" on page 10-10	Establish a *channel*, which is a connection between RMAN and a database instance.
"allocateForMaint" on page 10-10	Allocate a channel in preparation for issuing maintenance commands such as **change**.
"alterDatabase" on page 10-10	Mount or open a database.
"archivelogRecordSpecifier" on page 10-18	Specify a range of archived redo logs files for use in backup, restore, and maintenance operations as well as queries to the recovery catalog.
"backup" on page 10-22	Back up a database, tablespace, datafile, or archived redo log file.
"catalog" on page 10-35	Add information about a datafile copy, archived redo log, or control file copy to the recovery catalog and control file. Catalog a datafile copy as a level 0 backup, which enables you to use it as part of an incremental backup strategy. Record the existence of file copies created before RMAN was installed or generated through means other than RMAN.

Table 10–2 Recovery Manager Commands

Command	Purpose
"change" on page 10-38	Mark a backup piece, image copy, or archived redo log as having the status UNAVAILABLE or AVAILABLE.
	Delete a backup piece, image copy, or archived redo log from the operating system and remove its recovery catalog record.
	Check whether backup pieces, datafile copies, or archived redo logs are available and, if they are not, mark them as EXPIRED.
"cmdLine" on page 10-42	Connect to the target, recovery catalog, or auxiliary database.
	Specify that you are using RMAN without a recovery catalog.
	Specify a command file, which is a user-defined file containing RMAN commands.
	Specify the file in which RMAN records the results of processed commands.
	Add to rather than overwrite the contents of the command file.
	Generate debugging output and specify its location.
"completedTimeSpec" on page 10-45	A sub-clause that specifies a time range during which the backup or copy completed.
"configure" on page 10-47	Set the compatibility level of the recovery catalog.
"connect" on page 10-51	Establish a connection between RMAN and a target, auxiliary (duplicated or auxiliary instance used for TSPITR), or recovery catalog database.
"connectStringSpec" on page 10-53	Specify the username, password, and net service name for connecting to a target, recovery catalog, or auxiliary database. The connection is necessary to authenticate the user and identify the database.
"copy" on page 10-55	Create an image copy of a file.
"createCatalog" on page 10-59	Create a schema for the recovery catalog.
"createScript" on page 10-61	Create a stored script and store it in the recovery catalog for future reference.
"crosscheck" on page 10-64	Determine whether backup sets stored on disk or tape still exist.
"datafileSpec" on page 10-66	Specify a datafile by filename or absolute file number.
"debug" on page 10-68	Turn RMAN's debugging feature off and on.
"deleteExpired" on page 10-69	Delete backup sets marked EXPIRED by the **crosscheck** command and remove references to them from the recovery catalog and control file.
"deleteScript" on page 10-69	Delete a stored script from the recovery catalog.

Table 10–2 Recovery Manager Commands

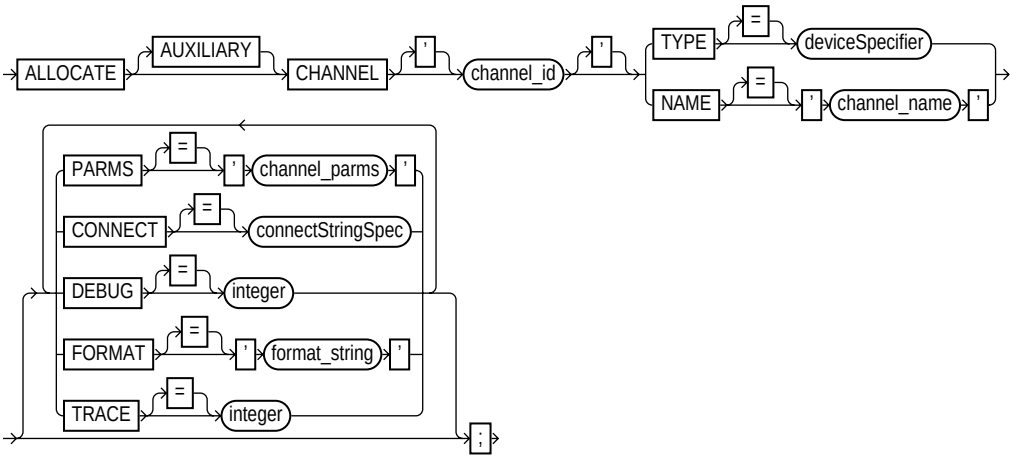
Command	Purpose
"deviceSpecifier" on page 10-72	Specify the type of storage for a backup or copy.
"dropCatalog" on page 10-74	Remove the schema from the recovery catalog.
"duplicate" on page 10-76	Use backups of the target database to create a duplicate database that you can use for testing purposes.
"host" on page 10-81	Invoke an operating system command-line sub-shell from within RMAN.
"list" on page 10-83	Produce a detailed report about a specified group of backup sets or copies recorded in the recovery catalog or target control file.
"listObjList" on page 10-92	Specify a database or one or more tablespaces, control files, datafiles, or archived redo logs.
"printScript" on page 10-94	Print a stored script to the RMAN message log file. Specify the log filename with the **log** argument at the command line (see "connect" on page 10-51).
"recover" on page 10-96	Apply redo logs or incremental backups to a restored backup set or copy in order to update it to a specified time.
"register" on page 10-101	Register the target database in the recovery catalog so that RMAN can access it.
"release" on page 10-103	Release a sequential I/O device while maintaining the connection to the target database instance.
"releaseForMaint" on page 10-104	Release a sequential I/O device specified in an **allocate channel** command with the **for delete** or **for maintenance** option.
"replaceScript" on page 10-105	Replace an existing script stored in the recovery catalog. If the script does not exist, **replace script** creates it.
"replicate" on page 10-108	Copy the control file to multiple destinations.
"report" on page 10-110	Perform detailed analyses of the content of the recovery catalog.
"reset" on page 10-118	Create a new database incarnation record in the recovery catalog.
"restore" on page 10-120	Restore files from backup sets or from copies on disk to the current location, overwriting the files with the same name.
"resync" on page 10-127	Perform a full *resynchronization*, which creates a snapshot control file and then compares the recovery catalog to either the current control file of the target database or the snapshot control file and updates it with information that is missing or changed.
"rmanCommand" on page 10-130	Execute *stand-alone* commands, which are commands you run from the command-line interpreter (CLI), that is, the RMAN prompt.

Table 10–2 Recovery Manager Commands

Command	Purpose
"run" on page 10-133	Compile and execute *job commands*, which are one or more statements executed within the braces of **run**.
"send" on page 10-136	Send a vendor-specific quoted string to one or more specific channels.
"set" on page 10-138	Specify the auxiliary filenames for target datafiles. This operation is useful when performing TSPITR.
	Display executed RMAN commands in the message log.
	Specify a database's db identifier.
	Set the filename of the snapshot control file.
"set_run_option" on page 10-142	Specify new filenames for datafiles.
	Specify a limit for the number of permissible block corruptions.
	Override default archived redo log destinations.
	Specify that backups should be duplexed.
	Determine which server process corresponds to which channel.
	Limit the number of buffers that will be read from each input datafile on a specified channel.
	Limit the number of input files that a **backup** operation can have open at any given time for a specified channel.
	Limit the size of the backup pieces for a specified channel.
"shutdown" on page 10-147	Shut down the target database without exiting RMAN. This command is equivalent to the SQL*Plus SHUTDOWN command.
"sql" on page 10-150	Execute a SQL statement from within Recovery Manager.
"startup" on page 10-152	Start up the database from within the RMAN environment. This command is equivalent to the SQL*Plus STARTUP command.
"switch" on page 10-154	Specify that a datafile copy is now the *current datafile*, that is, the datafile pointed to by the control file.
"upgradeCatalog" on page 10-158	Upgrade the recovery catalog schema from an older version to the version required by the RMAN executable.
"validate" on page 10-160	Examine a backup set and report whether its data is intact. RMAN scans all of the backup pieces in the specified backup sets and looks at the checksums to verify that the contents can be successfully restored if necessary.

allocate

Syntax



Purpose

To establish a *channel*, which is a connection between RMAN and a database instance. Each connection initiates an Oracle server session on the target instance: this server session performs the work of backing up, restoring, or recovering backup sets and copies.

Each channel operates on one backup set at a time (for **backup**, **restore**, or **recover**) or one image copy at a time (for **copy**). RMAN automatically releases the channel at the end of the job.

Control the degree of parallelism within a job by the number of allocated channels. You can allocate multiple channels simultaneously, thus allowing a single job to read or write multiple backup sets or copies in parallel. If you establish multiple connections, then each connection operates on a separate backup set or file copy.

Whether **allocate channel** causes operating system resources to be allocated depends on your operating system. On some platforms, operating system resources are allocated at the time the command is issued. On other platforms, operating system resources are not allocated until you open a file for reading or writing.

Note: When you specify **type disk**, no operating system resources are allocated other than for the creation of the server session.

Requirements

- Execute **allocate** only within the braces of a **run** command.
- You cannot allocate a channel to a multi-threaded server (MTS) session.
- Allocate a channel before executing a **backup**, **duplicate** (see "duplicate" on page 10-76), **copy**, **restore**, **recover**, or **validate** command.
- When duplexing backups, execute the **set duplex** command (see "set_run_option" on page 10-142) before allocating a channel for a **backup** command.

Keywords and Parameters

auxiliary

specifies a connection between RMAN and an auxiliary database instance. An auxiliary instance is used when executing the **duplicate** command or performing TSPITR. An auxiliary database can reside in the same host as its parent or in a different host. When specifying this option, the auxiliary database must be mounted but not open.

See Also: "duplicate" on page 10-76 to learn how to duplicate a database, and "connect" on page 10-51 to learn how to connect to a duplicate database.

channel *channel_id*

specifies a connection between RMAN and the target database instance. Each connection initiates an Oracle server session on the database instance: this server session performs the work of backing up, restoring, and recovering backups and copies.

Specify a channel id, which is the name of the channel, after the **channel** keyword. Oracle uses the identifier with the **release channel** command and to report I/O errors.

type *deviceSpecifier*

specifies the type of storage device (see "deviceSpecifier" on page 10-72).

Note: If you do not specify the **type** parameter, then you must specify the **name** parameter to identify a particular sequential I/O device. Query the V$BACKUP_DEVICE view for information about available device types and names.

name *'channel_name'*

specifies the name of a sequential I/O device. If you do not specify a device name, then the system uses any available device of the specified type. Do not use this parameter in conjunction with the **type** parameter.

Currently, no platform supports the **name** parameter.

parms *'channel_parms'*

specifies parameters for the device to allocate. Do not use this port-specific string if you have specified **type disk**.

If you use **parms** in conjunction with **type** *'sbt_tape'*, you can specify environment variables. Following are models for acceptable syntax:

```
PARMS="ENV=(var1=value1,var2=value2,var3=value3 . . . )"
PARMS="BLKSIZE=integer"
```

For example, you can specify:

```
PARMS="BLKSIZE=16384,ENV=(NSR_SERVER=tape_server,NSR_CLIENT=oracleclnt,
                          NSR_GROUP=oracle_tapes)"
```

The maximum length of the quoted string is 1000 bytes.

connect *connectStringSpec*

specifies a connect string to the database instance where RMAN should conduct the backup or restore operations (see "connectStringSpec" on page 10-53). Use this parameter when you want to spread the work of backup or restore operations across different instances in an OPS configuration.

If you do not specify this parameter, and you did not specify the **auxiliary** option, then RMAN conducts all operations on the target database instance specified by the command-line parameter (see "cmdLine" on page 10-42) or the instance connected to when you issued the **connect** command. Typically, you should not use the **connect** parameter in conjunction with the **auxiliary** option.

debug *integer*

specifies that Oracle should log debugging information about copy, backup, and restore operations performed on this channel by the target or auxiliary database to a trace file. Use this parameter only at the direction of Oracle Support: they will tell you which integer to use.

format *'format_string'*

specifies the format to use for the names of backup pieces that are created on this channel. If you do not specify a format, RMAN uses %U by default, which guarantees a unique identifier. For available **format** parameters, see the **backup** command.

This parameter is useful if you allocate multiple disk channels and want each channel to write to a different filesystem. If you specify the **format** parameter in the **backup** command, then it overrides the **format** parameter specified in **allocate channel**.

trace *integer*

specifies an integer whose meaning is determined by the media management software. Typically, this parameter controls how much diagnostic trace data is produced by the media manager.

Examples

Allocating a Single Channel for a Backup This command allocates a tape channel for a whole database backup:

```
run {
     allocate channel dev1 type 'sbt_tape';
     backup database;
}
```

Spreading a Backup Set Across Multiple Disks When backing up to disk, you can spread your backup across several disk drives. Allocate one **type disk** channel per disk drive and specify the format string so that the filenames are on different disks:

```
run{
     allocate channel disk1 type disk format '/disk1/%d_backups/%U';
     allocate channel disk2 type disk format '/disk2/%d_backups/%U';
     allocate channel disk3 type disk format '/disk3/%d_backups/%U';
     backup database;
}
```

Duplexing a Backup Set When duplexing backup sets, specify the **set duplex** command (see "set_run_option" on page 10-142) before allocating a channel. The following example generates four identical backups of `datafile 1`:

```
run {
     set duplex = 4;
     allocate channel dev1 type 'sbt_tape';
     backup datafile 1;
}
```

Allocating an Auxiliary Channel When creating a duplicate database (see "duplicate" on page 10-76), allocate a channel using the **auxiliary** option:

```
run {
     allocate auxiliary channel c1 type disk;
     allocate auxiliary channel c2 type disk;
     duplicate target database to ndbnewh
       logfile
         '/oracle/dbs/log_1.f' size 200K,
         '/oracle/dbs/log_2.f' size 200K
     skip readonly
     nofilenamecheck;
}
```

Related Topics

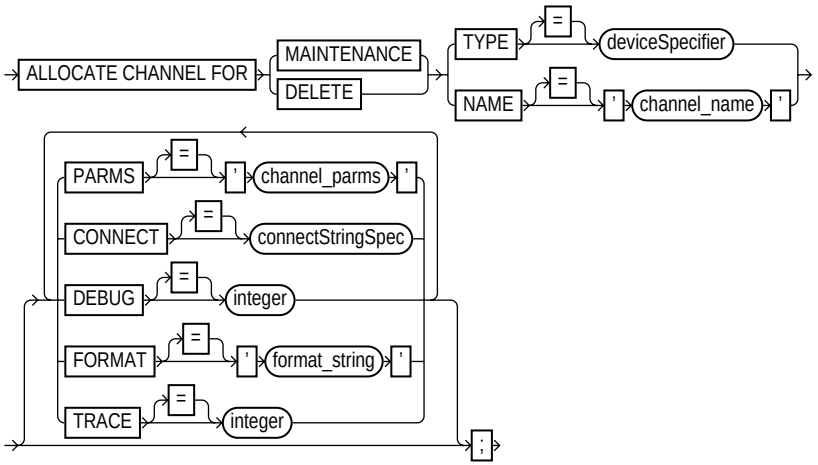
"allocateForMaint" on page 10-14

"duplicate" on page 10-76

"cmdLine" on page 10-42

allocateForMaint

Syntax



Purpose

To allocate a channel in preparation for issuing a **change** or **crosscheck** command. The **maintenance** and **delete** options, which are completely synonymous, specify the device type appropriate for the file whose status you are changing.

Requirements

- Execute this command only at the RMAN prompt.
- Issue an **allocate channel for delete** or **allocate channel for maintenance** command before issuing the following:
  - **change backupset ... delete**
  - **change backuppiece ... delete**
  - **change backupset ... crosscheck**
  - **change backuppiece ... crosscheck**
  - **crosscheck**
- Do not specify a channel id.

- You must release an allocated channel before you can allocate a new one, that is, you cannot allocate multiple maintenance channels.
- You cannot allocate a maintenance channel to a multi-threaded server (MTS) session.

Keywords and Parameters

See "allocate" on page 10-10.

Examples

Deleting a Backup Piece This example deletes a backup piece from the media management catalog:

```
allocate channel for maintenance type 'sbt_tape';
change backuppiece '/oracle/dbs/01aj3q5012' delete;
release channel;
```

Marking a File Unavailable This example crosschecks the backup set with primary key 828:

```
allocate channel for maintenance type disk;
change backupset 828 crosscheck;
release channel;
```

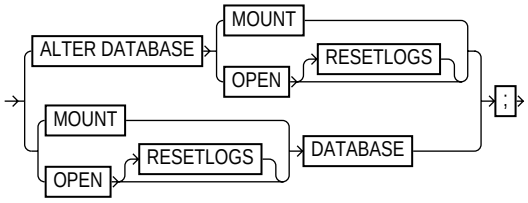
Related Topics

"allocate" on page 10-10

"change" on page 10-38

alterDatabase

Syntax



Purpose

To mount or open a database.

See Also: *Oracle8i SQL Reference* for ALTER DATABASE syntax.

Requirements

- Execute this command either within the braces of a **run** command or at the RMAN prompt.
- Execute the command only if an Oracle instance is already started.

Keywords and Parameters

alter database allows you to either mount or open the database.

- **mount** mounts the database without opening it. This option is equivalent to the SQL statement ALTER DATABASE MOUNT.
- **open** opens the database. If you specify **resetlogs**, then RMAN opens the database with the RESETLOGS option.

  The RMAN **resetlogs** option is equivalent to the SQL statement ALTER DATABASE OPEN RESETLOGS. If you use a recovery catalog, then RMAN performs an implicit **reset databas**e after the database is opened to make this new incarnation the current one in the catalog.

mount database mounts the database without opening it. This option is equivalent to the SQL statement ALTER DATABASE MOUNT.

open database	opens the database. If you specify **resetlogs**, then RMAN opens the database with the RESETLOGS option. The RMAN **resetlogs** option is equivalent to the SQL statement ALTER DATABASE OPEN RESETLOGS. If you use a recovery catalog, then RMAN performs an implicit **reset databas**e after the database is opened to make this new incarnation the current one in the catalog.

Examples

Opening the Database after a Backup This example mounts the database, takes a whole database backup, then opens the database. At the RMAN prompt enter:

```
startup mount;
run {
     allocate channel ch1 type disk;
     backup database;
     # now that the backup is complete, open the database.
     alter database open;
}
```

Mounting the Database after Restoring the Control File To restore the control file to its default location enter the following:

```
startup nomount;
run {
     allocate channel ch1 type 'sbt_tape';
     restore controlfile;
}
alter database mount;
```

Performing Incomplete Recovery and Opening with RESETLOGS This example performs incomplete recovery and then resets the online redo logs:

```
run {
     allocate channel ch1 type 'sbt_tape';
     set until scn 1024;
     restore database;
     recover database;
     alter database open resetlogs;
}
```

Related Topics

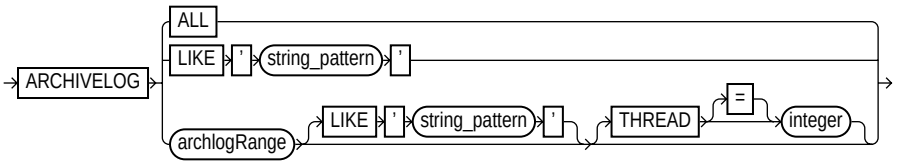
"sql" on page 10-150

"startup" on page 10-152

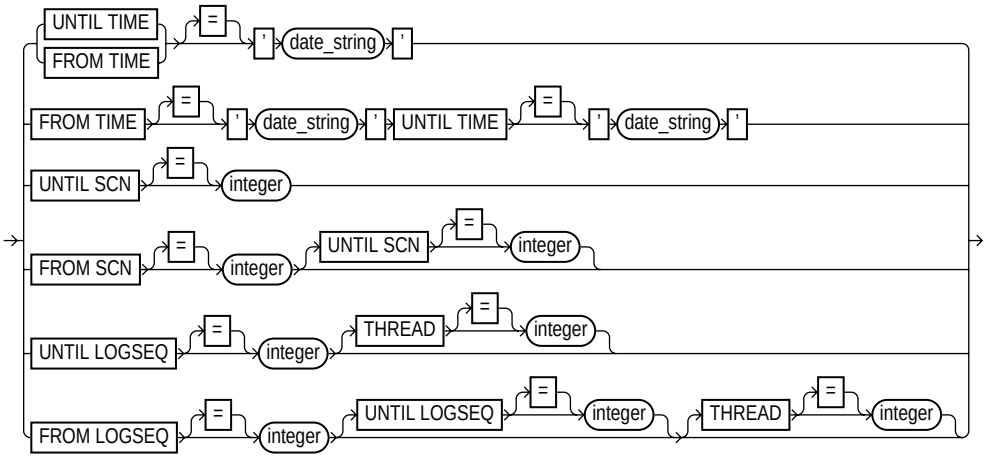
"shutdown" on page 10-147

archivelogRecordSpecifier

Syntax



archlogRange::=



Purpose

A sub-clause used to specify a range of archived redo logs files for use in backup, restore, and maintenance operations.

Requirements

Use this clause only with the following commands:

- **backup**
- **change**

- **crosscheck**
- **deleteExpired**
- **list**
- **restore**

Specifying a range of archived redo logs does not guarantee that RMAN includes all redo data in the range: for example, the last available archived log file may end before the end of the range, or an archived log file in the range may be missing. RMAN includes the archived redo logs it finds and does not issue a warning for missing files.

Keywords and Parameters

Query the V$ARCHIVED_LOG view to determine the timestamps, SCNs, and log sequence numbers for an archived log. For information on using the NLS_LANG and NLS_DATE_FORMAT environment variables to specify the format for the time, see the *Oracle8i Reference*.

all

specifies exactly one copy of each distinct log sequence number. For example, if you execute **backup archivelog all** and you archive your logs to multiple destinations, RMAN backs up *one* copy of each log sequence number—not each archived copy of each log sequence number.

like *'string_pattern'*

specifies a pathname for archived redo log files. Use this parameter when operating in OPS mode to specify which file system RMAN should access. For example, if nodes A, B, and C each archives locally, use the **like** parameter to inform the channel allocated on each node to back up only those logs that contain the specified pathname.

See Also: "Backing Up in an OPS Environment" on page 5-9 to learn how to make archived log backups in an OPS configuration, and *Oracle8i Parallel Server Documentation Set: Oracle8i Parallel Server Concepts; Oracle8i Parallel Server Setup and Configuration Guide; Oracle8i Parallel Server Administration, Deployment, and Performance* for more information about the Oracle Parallel Server configuration.

until time *'date_string'*

specifies the end date for a sequence of archived redo log files. The time specified in the string must be formatted according to the NLS date format specification currently in effect.

If you do not specify the **from time** parameter, the beginning time for the sequence will be the earliest available archived redo log. Query the V$ARCHIVED_LOG data dictionary view to determine the timestamps for the first and last entries in a log.

See Also: *Oracle8i Reference* for information on using the NLS_LANG and NLS_DATE_FORMAT environment variables to specify the format for the time.

from time *'date_string'*	specifies the beginning date for a sequence of archived redo log files. If you do not specify the **until time** parameter, RMAN will include all available log files beginning with the date specified in the **from time** parameter. Use the V$ARCHIVED_LOG data dictionary view to determine the timestamps for the first and last entries in a log file. **See Also:** *Oracle8i Reference* for information on using the NLS_LANG and NLS_DATE_FORMAT environment variables to specify the format for the time.
until SCN *integer*	specifies the ending SCN for a sequence of archived redo log files. If you do not specify the **from SCN** parameter, RMAN will use the lowest available SCN to begin the sequence.
from SCN *integer*	specifies the beginning SCN for a sequence of archived redo log files. If you do not specify the **until SCN** parameter, RMAN will include all available log files beginning with SCN specified in the **from SCN** parameter.
until logseq *integer*	specifies the terminating log sequence number for a sequence of archived redo log files. If you do not specify the **from logseq** parameter, RMAN uses the lowest available log sequence number to begin the sequence.
from logseq *integer*	specifies the beginning log sequence number for a sequence of archived redo log files. If you do not specify the **until logseq** parameter, RMAN will include all available log files beginning with log sequence number specified in the **from logseq** parameter. **Note:** You can specify all log sequence numbers in a thread by using the following syntax, where *thread_number* is an integer referring to the thread: `archivelog from logseq 0 thread thread_number`
thread *integer*	specifies the thread containing the archived redo log files you wish to include. You need only specify this parameter when running your database in an OPS configuration. Use the V$ARCHIVED_LOG data dictionary view to determine the thread number for an archived redo log record.

Specifying Records by Time This example deletes all archived redo logs older than two weeks:

```
change archivelog until time 'SYSDATE-14' delete;
```

Specifying Records by SCN This example restores backup archived redo log files from tape that fall within a range of SCNs:

```
run {
     allocate channel dev1 type 'sbt_tape';
     restore archivelog
          from SCN 500 until SCN 700;
     release channel dev1;
}
```

Specifying Records by Log Sequence Number This example backs up all archived logs from sequence # 288 to sequence # 301 on thread 1 and deletes the archived logs after the backup is complete. If the backup fails, the logs are not deleted.

```
run {
     allocate channel dev1 type 'sbt_tape';
     backup archivelog
          from logseq 288 until logseq 301 thread 1
          # delete original archived redo logs after backup completes
          delete input;
}
```

Specifying All Log Sequence Numbers in a Thread This example crosschecks all archived redo logs in thread 1:

```
change archivelog from logseq 0 thread 1 crosscheck;
```

Related Topics

"catalog" on page 10-22

"change" on page 10-38

"crosscheck" on page 10-64

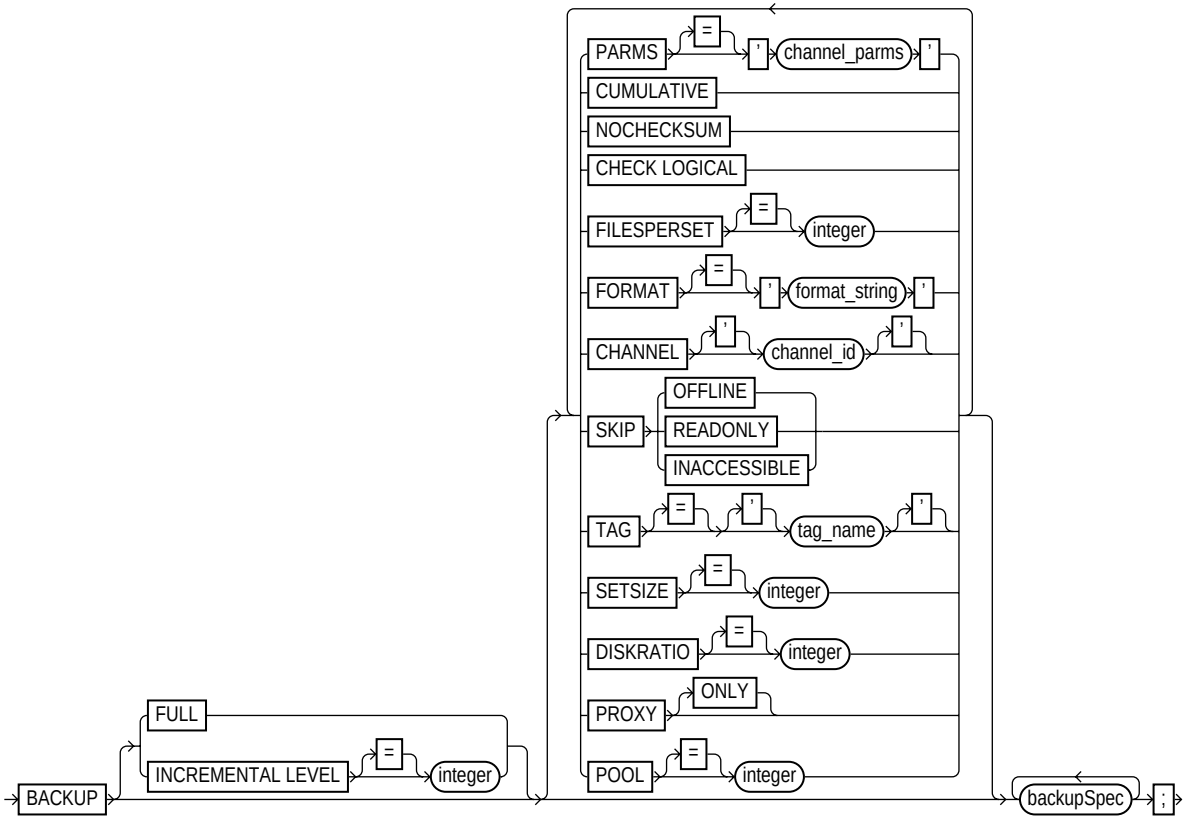
"deleteExpired" on page 10-69

"list" on page 10-83

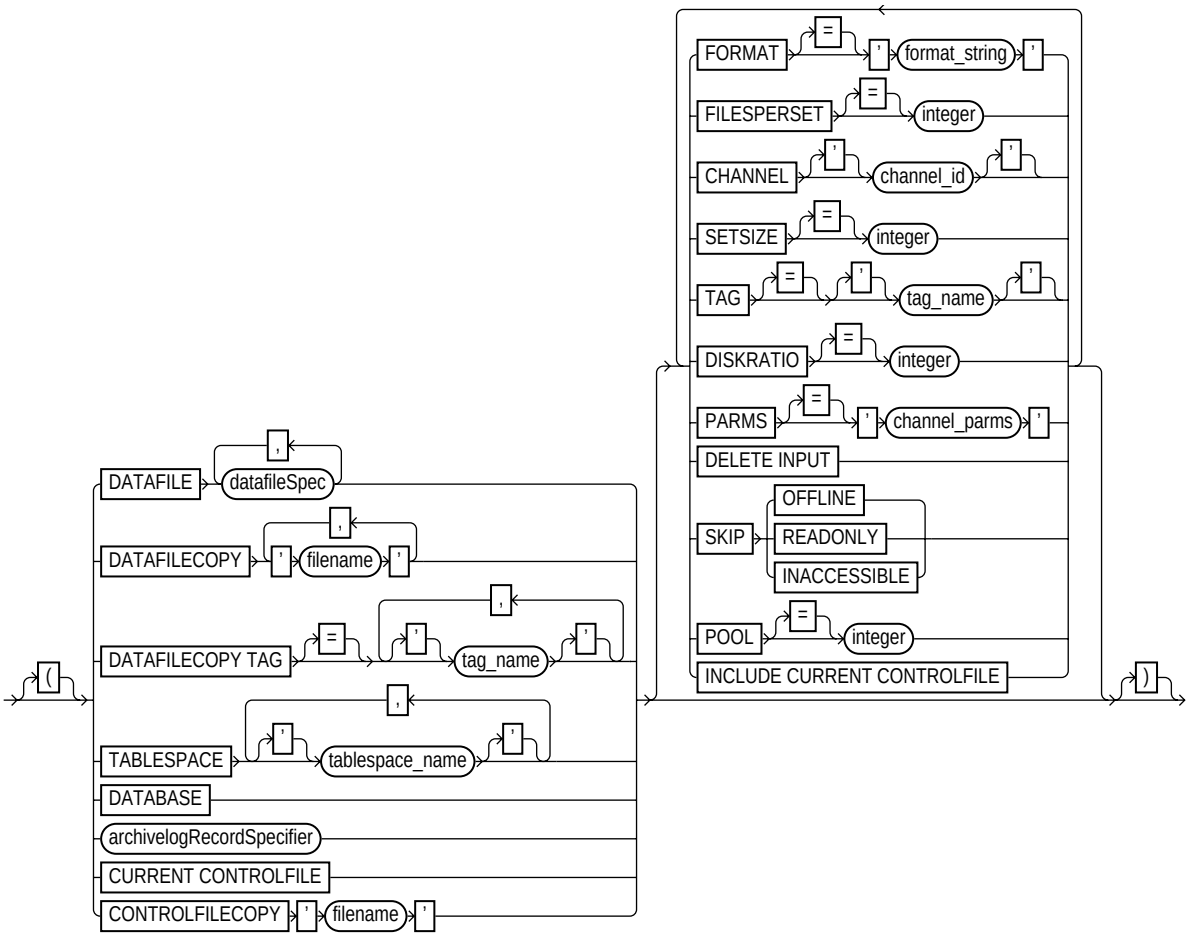
"restore" on page 10-120

backup

Syntax



backupSpec::=



Purpose

To back up a database, tablespace, datafile, control file, or archived redo log file. When performing a backup, specify the files that you want to back up. RMAN puts the input files into a backup set, which is an RMAN-specific logical structure. Each backup set contains at least one backup piece. You can also use the **backup** command to generate a proxy copy, which is a backup created by a media manager.

You control the number of backup sets that Oracle produces as well as the number of input files that RMAN places into a single backup set. Any I/O errors when reading files or writing backup pieces cause Oracle to abort the jobs.

> **See Also:** "Backup Sets" on page 1-32 for a conceptual overview of RMAN backup sets, and Chapter 5, "Making Backups and Copies with Recovery Manager" to learn how to back up files.

Requirements

When using the **backup** command you must:

- Mount or open the target database. RMAN allows you to make an *inconsistent backup* if the database is in ARCHIVELOG mode, but you must apply redo logs to make the backups consistent for use in restore operations.
- Use a current control file.
- Execute the **backup** command within the braces of a a **run** command.
- Allocate a disk or tape channel for each execution of the **backup** command.

> **Note:** Backups using the disk API are not supported (see "After Linking to the Media Manager on UNIX, RMAN Fails to Back Up to Tape" on page 9-19). Instead, allocate a channel of **type disk**.

- Give each backup piece a unique name.
- Back up onto valid media. If you specify **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 a valid backup path name. If you specify **type** *'sbt_tape'*, then you can back up to any media supported by the media management software.

You *cannot* do the following:

- Make an open backup in NOARCHIVELOG mode.
- Stripe a single backup across multiple channels.
- Stripe a single input file across multiple backup sets.
- Combine archived redo log files and datafiles into a single backup.

- Execute the **set duplex** command (see "set_run_option" on page 10-142) *after* you **allocate** a channel for a **backup** command. The **set duplex** command must precede all allocated channels or you receive an error.
- Specify the number of backup pieces that should go in a backup set.
- Create a backup set containing more than 100 backup pieces.
- Automate the creation of unique tag names for each backup. To create unique tag names every time, write a backup script and then edit it before execution using an operating system utility.
- Make the length of a backup piece filename longer than the port-specific length limit. If you use a media manager, then the limit is partially governed by the version of the media management API. Vendors using:
  - SBT 1.1 must support filenames up to 14 characters, but may support longer filenames.
  - SBT 2.0 must support filenames up to 512 characters, but may support longer filenames.

Keywords and Parameters

full

copies all blocks into the backup set, skipping only datafile blocks that have never been used. RMAN makes full backups by default if neither **full** nor **incremental** is specified. The server session does not skip blocks when backing up archived redo logs or control files.

A full backup has no effect on subsequent incremental backups, so it is not considered a part of the incremental backup strategy.

incremental level *integer*

copies only those data blocks that have changed since the last incremental *n* backup, where *n* is any integer from 1 to 4. For example, in a level 2 backup RMAN backs up all blocks used since the most recent level2, level 1, or level 0 backup.

This type of incremental backup is also called a *differential backup* to distinguish it from a cumulative backup. An incremental backup at level 0 is identical in content to a full backup, but unlike a full backup the level 0 backup is considered a part of the incremental strategy.

Oracle performs checks when attempting to create an incremental backup at a level greater than 0. These checks ensure that the incremental backup will be usable by a subsequent **recover** command. Among the checks performed are:

- A level 0 backup set must exist, or level 0 datafile copies must exist for each datafile in the **backup** command. These backup sets must not be marked UNAVAILABLE (see "change" on page 10-38).
- Sufficient incremental backups taken since the level 0 must exist and be available such that the incremental backup to be created is usable.

If you specify **incremental**, then in the *backupSpec* you must set one of the following parameters: **datafile**, **datafilecopy**, **tablespace**, or **database**. RMAN does not support incremental backups of control files, archived redo logs, or backup sets.

parms *'channel_parms'*

specifies a quoted string containing operating system-specific information. RMAN passes the string to the OSD layer each time a backup piece is created. Currently, no **parms** settings are available when specified in the **backup** command, although you can specify **parms** in the **allocate channel** command.

cumulative

copies the data blocks used since the most recent backup at level *n*-1 or lower. For example, in a cumulative level 2 backup RMAN backs up all blocks used since the most recent level 1 or level 0 backup.

nochecksum

suppresses block checksums. A *checksum* is a number that is computed from the contents of a data block. If the DB_BLOCK_CHECKSUM initialization parameter is TRUE, Oracle computes a checksum for each block and stores it in the block before writing the block to disk. When Oracle reads the block from disk later, it makes sure that the block generates the same checksum. If it does not, then the block is damaged.

Unless you specify the **nochecksum** option, Oracle computes a checksum for each block and stores it in the backup. The checksum is verified when restoring from the backup and written to the datafile when restored. If the database is already maintaining block checksums, then this flag has no effect. The checksum is always verified and stored in the backup in this case.

See Also: *Oracle8i Reference* for more information about the DB_BLOCK_CHECKSUM initialization parameter.

check logical 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, then it logs the block in the `alert.log` and server session trace file.

Provided the sum of physical and logical corruptions detected for a file remain below its **maxcorrupt** setting, the RMAN command completes and Oracle populates V$BACKUP_CORRUPTION with corrupt block ranges. If **maxcorrupt** is exceeded, then the command terminates without populating the views.

Note: For **copy** and **backup** the **maxcorrupt** setting represents the total number of physical and logical corruptions permitted on a file.

filesperset *integer* specifies the maximum number of input files per backup set. If you set **filesperset** = *n*, then RMAN never includes more than *n* files in a backup set. The default for **filesperset** is the lesser of these two values: 64, number of input files / number of channels. For example, if you back up 100 datafiles using 2 channels, RMAN sets **filesperset** to 50.

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 two datafiles when three channels are allocated and **filesperset** = 1, then one channel is necessarily idle.

See Also: The **setsize** parameter, which limits backup sets by total bytes rather than number of files included.

format *'format_string'* specifies the filename to use for the backup piece. Any name that is legal as a sequential filename on the platform is allowed, provided that each backup piece has a unique name. If backing up to disk, then any legal disk filename is allowed, provided it is unique. If you do not specify the **format** parameter, RMAN stores the backup pieces in a port-specific directory (`$ORACLE_HOME/dbs` on UNIX).

Specify the **format** parameter in any of these places:

- The *backupSpec* clause
- The **backup** command
- The **allocate channel** command

If specified in more than one of these places, RMAN searches for the **format** parameter in the order shown above.

The following substitution variables are available in format strings to aid in generating unique filenames:

%c specifies the copy number of the backup piece within a set of duplexed backup pieces. If you did not issue the **set duplex** command, then this variable will be 1 for regular backup sets and 0 for proxy copies. If you issued **set duplex**, the variable identifies the copy number: 1, 2, 3, or 4.

%p specifies the backup piece number within the backup set. This value starts at 1 for each backup set and is incremented by 1 as each backup piece is created.

	%s	specifies the backup set number. This number is a counter in the control file that is incremented for each backup set. The counter value starts at 1 and is unique for the lifetime of the control file. If you restore a backup control file, then duplicate values can result. Also, CREATE CONTROLFILE initializes the counter back to 1.
	%d	specifies the database name.
	%n	specifies the database name, padded on the right with 'x' characters to a total length of 8 characters. For example, if PROD1 is the database name, then PROD1xxx is the padded database name.
	%t	specifies the backup set timestamp, which is a 4-byte value derived as the number of seconds elapsed since a fixed reference time. The combination of %s and %t can be used to form a unique name for the backup set.
	%u	specifies an 8-character name constituted by compressed representations of the backup set number and the time the backup set was created.
	%U	specifies a convenient shorthand for **%u_%p_%c** that guarantees uniqueness in generated backup filenames. If you do not specify a format, RMAN uses **%U** by default.

channel *channel_id*

specifies the name of a channel to use when creating the backup sets. Use any name that is meaningful, for example, *ch1* or *dev1*. Oracle uses the channel id with the **release channel** command and to report I/O errors. If you do not specify this parameter, then RMAN dynamically assigns the backup sets to any available channels during job execution.

Note: You can also specify this parameter in the *backupSpec* clause.

skip

excludes datafiles or archived redo logs from the backup set.

Note: You can also specify this option in the *backupSpec* clause.

offline	specifies that offline datafiles should be excluded from the backup set.
readonly	specifies that read-only datafiles should be excluded from the backup set.
inaccessible	specifies that datafiles or archived redo logs that cannot be read due to I/O errors should be excluded from the backup set. Note that a datafile is only considered inaccessible if it cannot be read. Some offline datafiles can still be read because they still exist on disk. Others have been deleted or moved and so cannot be read, making them inaccessible.

tag *tag_name*

creates a user-specified tag for the backup set. Typically, a tag is a meaningful name such as *monday_evening_backup* or *weekly_full_backup*. Tags must be 30 characters or less. Note that tags are reusable, so that backup set 100 can have the tag *monday_evening_backup* one week while backup set 105 has the same tag the next week.

You can also specify the tag at the *backupSpec* level. If you specify the tag at:

- The command level, then all backup sets created by this command are given this tag.
- The *backupSpec* level, then backup sets created as a result of different backup specifications can have different tags.
- Both levels, then the tag in the *backupSpec* takes precedence.

Note: You cannot automatically assign a different tag name to each backup. The easiest way to give each backup a new tag is to write a backup script and then edit it with an operating system utility before each execution.

setsize *integer*

specifies a maximum size for a backup set in units of 1K (1024 bytes). Thus, to limit a backup set to 3Mb, specify **setsize** = 3000. RMAN attempts to limit all backup sets to this size. You can use **setsize** to configure backup sets so that each fits on one tape volume rather than spans multiple tape volumes. Otherwise, if one tape of a multi-volume backup set fails, then you lose the data on all the tapes rather than just one.

Because archived logs are located on one disk and so do not create an I/O distribution problem, the **setsize** parameter is easier to use than **filesperset** when you make archived redo log backups. The **filesperset** parameter is more useful for managing I/O distribution for backups of datafiles on multiple disks.

Note: Because **filesperset** has a default, both **setsize** and **filesperset** take effect when **setsize** is set. RMAN attempts to limit the size in bytes of the backup sets according to the **setsize** parameter, treating **filesperset** as an upper limit for the number of files to include in each set.

diskratio *integer*

directs RMAN to assign datafiles (only) to each backup set and spread them across the specified number of drives. For example, assume that you use 10 disks, the disks supply data at 10 bytes/second, and the tape drive requires 50 bytes/second to keep streaming. You can set **diskratio** to 5 to spread the backup load across 5 disks for each backup set.

If you set **filesperset** but not **diskratio**, then **diskratio** defaults to the same value as **filesperset**. If you specify neither parameter, **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 is 4 and the datafiles are on 3 disks, then RMAN attempts to spread the backup load for each set among 3 disks.

The **diskratio** parameter is easier for datafile backups when your datafiles are striped or reside on separate disk spindles and you either:

- Use a high-bandwidth tape drive that requires several datafiles to be multiplexed in order to keep the tape drive streaming.
- Make backups while the database is open and you want to spread the I/O load across several disk spindles in order to leave bandwidth for online operations.

Note: Do not spread I/O over more than the minimum number of disks to keep the tape streaming. Otherwise, you increase restore time for a file without increasing performance.

proxy backs up the specified files using the proxy copy functionality, which gives the media management software control over the data transfer between storage devices and the Oracle datafiles on disk. The media manager—not RMAN—decides how and when to move data.

When you execute a backup command with the proxy option, RMAN performs these steps:

1. Searches for a channel of type '*sbt_tape*' that is proxy-capable. If no such channel is found, then RMAN issues an error message. RMAN does not attempt a conventional (that is, non-proxy) backup of the specified files.
2. If RMAN locates a proxy-capable channel, it calls the media manager to determine whether it can proxy copy the file. If the media manager cannot proxy copy the file, then RMAN uses conventional backup sets to back up the file.

only causes Oracle to issue an error message when it cannot proxy copy rather than creating conventional backup sets.

pool *integer* specifies the media pool in which the backup should be stored. Consult your media management documentation to see whether the **pool** option is supported.

backupSpec A *backup_specification_list* contains a list of one or more *backupSpec* clauses. A *backupSpec* clause minimally contains a *backup_object_list*, which is a list of one or more objects to be backed up.

Each *backupSpec* clause generates one or more backup sets. A *backupSpec* clause will generate multiple backup sets if the number of datafiles specified in or implied by its *backup_object_list* exceeds the **filesperset** limit.

datafile *datafileSpec* specifies a list of one or more datafiles (see "datafileSpec" on page 10-66).

Note: If you back up `datafile 1`, which is the first file of the SYSTEM tablespace, RMAN automatically includes the control file in the backup set.

datafile copy *'filename'* specifies the filenames of one or more datafile image copies.

datafile copy tag *tag_name* specifies a list of one or more datafile copies, identified by tag. If multiple datafile copies with this tag exist, then Oracle backs up only the most current datafile copy of any particular datafile.

tablespace *tablespace_name* specifies the names of one or more tablespaces. RMAN backs up all datafiles that are currently part of the tablespaces.

This keyword is provided merely as a convenience; Oracle translates the tablespace name internally into a list of datafiles.

database specifies the control file and all datafiles in the database. This keyword is provided merely as a convenience; Oracle translates the tablespace name internally into a list of datafiles.

archivelogRecord-Specifier clause specifies a range of archived redo logs. See "archivelogRecordSpecifier" on page 10-18.

current controlfile	specifies the current control file.
controlfile copy *'filename'*	specifies the filename of a control file copy.
parms *'channel_parms'*	specifies a quoted string containing operating system-specific information. RMAN passes the string to the OSD layer each time a backup piece is created. Currently, no **parms** settings are available when specified in the **backup** command, although you can specify **parms** in the **allocate channel** command.
format *'format_string'*	Specifies the filename for the backup piece. See the description of the **format** parameter at the command level.
filesperset *integer*	specifies the maximum number of datafiles to place in one backup set. See the discussion of **filesperset** at the command level.
channel *channel_id*	specifies the name of a channel to use when creating the backup set for this *backupSpec* clause. See the discussion of **channel** at the command level.
setsize *integer*	specifies a maximum size for a backup set in units of 1K (1024 bytes). See the description of the **setsize** parameter at the command level.
tag *tag_name*	creates a tag for the backup set. See the discussion of the **tag** parameter at the command level for more information.
diskratio *integer*	specifies the number of disks involved in the backup. See the discussion of the **diskratio** parameter at the command level for more information.
delete input	deletes the input files upon successful creation of the backup set. Specify this option only when backing up archived logs or datafile copies. It is equivalent to issuing **change ... delete** for all of the input files. **Note**: The **backup** command only backs up one copy of each distinct log sequence number, so if the **delete input** option is requested, RMAN only deletes the copy of the file that it backs up. **See Also**: "configure" on page 10-47 for information on the effect of recovery catalog compatibility on this command.
skip	skips datafiles that are **offline**, **readonly**, or **inaccessible**. See the description of the **skip** option at the command level.
pool	specifies the media pool in which the backup should be stored. See the description of **pool** at the command level.
include current controlfile	creates a snapshot of the current control file and places it into each backup set produced by this clause.

Examples

Backing up a Database This command backs up the database to tape and then backs up the control file that contains the record of the database backup:

```
run {
     allocate channel dev1 type 'SBT_TAPE';
     backup database;
     backup current controlfile;
}
```

Backing up Tablespaces and Datafiles This command uses two *backupSpec* clauses to back up tablespaces and datafiles and lets RMAN perform automatic parallelization of the backup:

```
run {
     allocate channel dev1 type disk;
     allocate channel dev2 type disk;
     backup
       (tablespace system,sales1,sales2,sales3
       filesperset 20 skip readonly)
       (datafile 12, 14, 15);
}
```

Backing Up Multiple Copies of Archived Redo Logs This example backs up the archived redo logs in `/oracle/arch/dest1` to one set of tapes and the logs from `/oracle/arch/dest2` to another set of tapes. This scenario assumes that you have two tape drives available.

```
run {
     allocate channel t1 type 'sbt_tape';
     allocate channel t2 type 'sbt_tape';
     backup
       filesperset=20
       format='al_%d/%t/%s/%p'
       (archivelog like '/oracle/arch/dest1/%' channel t1 delete input)
       (archivelog like '/oracle/arch/dest2/%' channel t2 delete input);
}
```

Performing a Cumulative Incremental Backup of a Database This example backs up all blocks changed in the database since the most recent level 0 or level 1 backup:

```
run {
     allocate channel dev1 type 'sbt_tape';
     backup
       incremental level 2 cumulative
       # do not include inaccessibile datafiles in the backup
       skip inaccessible
```

```
      database;
}
```

Duplexing a Backup Set When duplexing backup sets, specify the **set duplex** command before allocating a channel:

```
run {
     # generate four identical backup sets of datafile 1
     set duplex=4;
     allocate channel dev1 type 'sbt_tape';
     backup datafile 1;
}
```

Specifying How Channels Divide a Workload This example parallelizes a backup operation by specifying which channels should back up which files and to which location:

```
run {
     allocate channel ch1 type 'SBT_TAPE';
     allocate channel ch2 type disk;
     allocate channel ch3 type 'SBT_TAPE';
     backup
       # channel ch1 backs up datafiles to tape drive #1
       (datafile 1,2,3,4
       channel ch1)
       # channel ch2 backs up control file copy to disk
       (controlfilecopy '/oracle/copy/cf.f'
       channel ch2)
       # channel ch3 backs up archived redo logs to tape drive #2
       (archivelog from time 'SYSDATE-14'
       channel ch3);
}
```

Performing an OPS Backup The following script distributes datafile and archived redo log backups across two nodes in an Oracle Parallel Server environment:

```
run {
     allocate channel node_1 type 'SBT_TAPE' connect 'sys/sys_pwd@node_1';
     allocate channel node_2 type 'SBT_TAPE' connect 'sys/sys_pwd@node_2';
        backup filesperset 1
          (tablespace system, rbs, data1, data2
           channel node_1)
          (tablespace temp, reccat, data3, data4
           channel node_2);
        backup filesperset 20
          (archivelog until time 'SYSDATE' like '/node1/arc/%'
           delete input
           channel node_1);
          (archivelog until time 'SYSDATE' like '/node2/arc/%'
```

```
          delete input
          channel node_2);
}
```

Checking for Corruption This example backs up `datafile 3` and specifies that no more than 2 blocks with physical or logical corruption will be tolerated:

```
run {
     set maxcorrupt for datafile 3 to 2;
     allocate channel dev1 type 'sbt_tape';
     backup check logical
       datafile 3;
}
```

Related Topics

"allocate" on page 10-10

"archivelogRecordSpecifier" on page 10-18

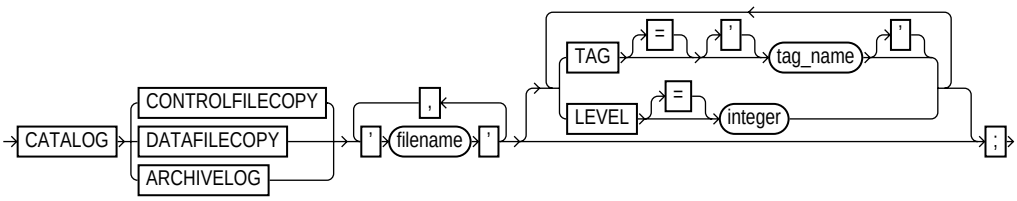
"configure" on page 10-47

"printScript" on page 10-94

"set_run_option" on page 10-142

catalog

Syntax



Purpose

Use the **catalog** command to:

- Add information about an operating system datafile copy, archived redo log, or control file copy to the recovery catalog and control file.
- Catalog a datafile copy as a level 0 backup, which enables you to use it as part of an incremental backup strategy.
- Record the existence of backups of version 8 databases created before RMAN was installed.
- Record the existence of Oracle7 backups of read-only or offline normal files made before migrating to Oracle8 or Oracle8*i*.

See Also: "Maintaining the RMAN Repository" on page 3-8.

Requirements

- Execute the **catalog** command only at the RMAN prompt.
- Operate RMAN with a recovery catalog.
- For an operating system backup to be cataloged, it must be:
  - Accessible on disk.
  - A complete image copy of a single file.
  - A consistent or inconsistent whole database, tablespace, datafile, control file, or archived redo log backup. If inconsistent, it must have been created using the BEGIN BACKUP / END BACKUP statements. If a control file

backup, it should have been made using the ALTER DATABASE BACKUP CONTROLFILE statement.

RMAN treats all such operating system backups as datafile copies.

You *cannot* use **catalog** to perform the following operations:

- Catalog archived redo logs and control file copies that were created in Oracle7 unless the file belongs to a tablespace that was offline normal or read-only when you migrated the database to Oracle version 8.0 or later.
- Re-catalog backup pieces or backup sets.

Keywords and Parameters

controlfilecopy *'filename'*	specifies the filename of a control file copy to be added to or updated in the recovery catalog and control file.
datafilecopy *'filename'*	specifies the filename of a datafile copy to be added to or updated in the recovery catalog and control file.
archivelog *'filename'*	specifies the filename of an archivelog copy to be added to or updated in the recovery catalog and control file.
tag *tag_name*	specifies the tag of the input file, for example, *Sunday_PM_Backup*.
level *integer*	indicates that the file copy should be recorded as an incremental backup at the specified level, typically level 0. You can perform incremental backups using a datafile copy as the base level 0 backup.

Examples

Cataloging an Archived Redo Log This statement catalogs the archived redo logs `log1`, `log2`, and `log3`:

```
catalog archivelog 'log1', 'log2', 'log3';
```

Cataloging a File Copy as an Incremental Backup The following example catalogs datafile copy `tbs_2.c` as an incremental level 0 backup:

```
catalog datafile '/oracle/copy/tbs_2.c' level 0;
```

Cataloging an Operating System Copy The following makes an operating system copy of a datafile using the RMAN **host** command and then catalogs the copy (sample output included):

```
host 'cp $ORACLE_HOME/dbs/sales.f $ORACLE_HOME/dbs/sales.bak';
catalog datafilecopy '$ORACLE_HOME/dbs/sales.bak';
```

```
RMAN-03022: compiling command: catalog
RMAN-03023: executing command: catalog
RMAN-08050: cataloged datafile copy
RMAN-08513: datafile copy filename=/oracle/dbs/sales.bak recid=121 stamp=342972501
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
```

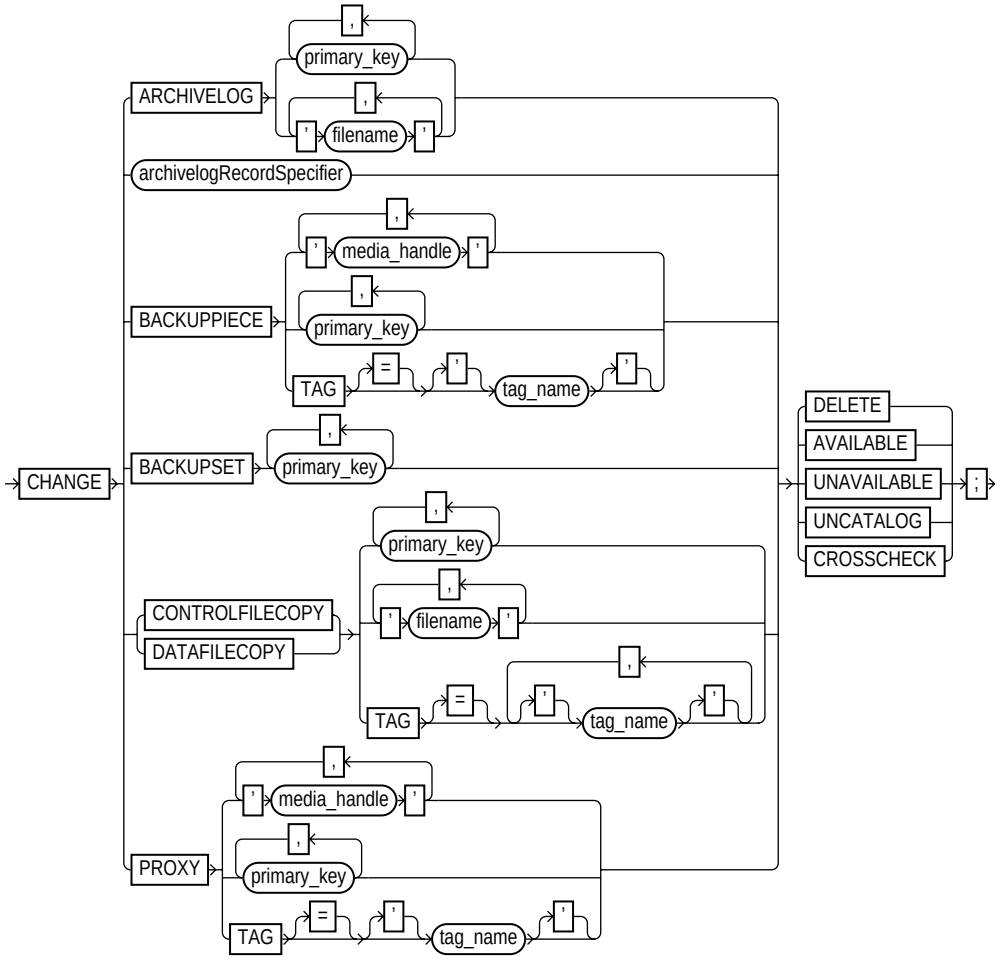
Related Topics

"list" on page 10-83

"report" on page 10-110

change

Syntax



Purpose

To check or remove physical backups, copies, and archived logs as well as update or remove their corresponding repository records. Use this command to:

- Mark a backup or copy as having the status UNAVAILABLE or AVAILABLE in the repository.
- Delete a backup or image copy from the operating system and completely remove its records from the repository.
- Delete an archived redo log and update its status to DELETED in the repository (if **configure** is set to 8.1.5 or lower) or remove its record from the repository (if **configure** is set to 8.1.6 or higher).
- Check whether backups, image copies, and archived redo logs are available and, if they are not, mark them as EXPIRED in the repository.

See Also: "Maintaining the RMAN Repository" on page 3-8.

Requirements

- Execute the **change** command at the RMAN prompt or within a **run** command.
- Use the command only on files that are recorded in the RMAN repository and belong to the current database incarnation.
- The following options require a recovery catalog:
  - **available**
  - **unavailable**
  - **uncatalog**
  - **crosscheck** (only required for backup sets and backup pieces)
- Issue an **allocate channel for delete** or **allocate channel for maintenance** command before issuing the following:
  - **change backupset ... delete**
  - **change backuppiece ... delete**
  - **change backupset ... crosscheck**
  - **change backuppiece ... crosscheck**

Keywords and Parameters

To obtain the primary keys of the records whose status you want to change, issue a **list** command or query the recovery catalog views.

archivelog specifies an archived redo log by either *primary_key* or *'filename'*.

archivelogRecordSpecifier clause specifies a range of archived redo logs. See "archivelogRecordSpecifier" on page 10-18.

backuppiece specifies a backup piece by *primary_key*, *'media_handle'*, or *tag_name*.

backupset *primary_key* specifies a backup set by *primary_key*.

controlfilecopy specifies a control file copy by *primary_key*, *'filename'*, or *tag_name*. If you crosscheck a control file copy, you must specify a filename rather than a primary key.

datafilecopy specifies a datafile copy by either *primary_key*, *'filename'*, or *tag_name*.

proxy specifies a proxy copy by *primary_key* or *'filename'*, or *tag_name*.

delete physically deletes the specified files from the operating system. If the file is a backup or image copy, RMAN also removes its repository records. If the file is an archived log, and if recovery catalog compatibility is set to 8.1.6 or higher, RMAN also removes the associated record from the repository. If compatibility is set to 8.1.5 or lower, RMAN updates the associated record to status DELETED.

See Also: "Setting Recovery Catalog Compatibility" on page 3-4 and "configure" on page 10-47 for more information about catalog compatibility.

available marks a backup or copy as having the status AVAILABLE**.** View the status in the **list** output.

unavailable marks a backup or copy as having the status UNAVAILABLE. View the status in the **list** output. This option is provided for cases when the file cannot be found or has migrated offsite. A file that is marked UNAVAILABLE will not be used in a **restore** or **recover** command. If the file is later found or returns to the main site, then you can use the **available** option to reflect this change.

uncatalog removes references to a datafile copy or archived redo log (but not a backup piece or backup set) from the recovery catalog. Use this command to notify RMAN when a file is deleted by some means other than a **change ... delete** command. If you attempt to use the **uncatalog** option on a backup piece or backup set, RMAN returns an error message.

Note: To remove all records with DELETED status at once, execute the `prgrmanc.sql` script located in the `$ORACLE_HOME/admin` directory. See "Deleting Backups and Copies and Updating Their Status in the RMAN Repository" on page 3-18 for instructions.

crosscheck	checks whether the specified backups and copies exist. If RMAN cannot find backup pieces, it marks them as having the status EXPIRED. It marks all other types of absent files—image copies and archived redo logs—as DELETED. If the files are on disk, RMAN queries the file headers. For other device types, RMAN queries the media manager to see whether the file exists in the media management catalog. **Note:** RMAN considers archived redo logs as copies, so issue the **change archivelog all crosscheck** command if one or more logs become unavailable. If the archived logs become unavailable again, you must issue **catalog archivelog** to re-catalog them. **Note:** If you crosscheck a control file copy, specify a filename rather than a primary key.

Examples

Deleting a Backup Piece This example deletes a backup piece from tape:

```
allocate channel for delete type 'sbt_tape';
change backuppiece '$ORACLE_HOME/dbs/testdb_87fa39e0' delete;
release channel;
```

Marking a Backup Set as Unavailable This example marks a backup set as having the status UNAVAILABLE. You do not need to allocate a maintenance channel:

```
change backupset 100 unavailable;
```

Crosschecking Files This example checks to see whether all of the registered archived redo logs still exist; if not, RMAN changes their status to EXPIRED:

```
allocate channel for maintenance type disk;
change archivelog all crosscheck;
release channel;
```

Uncataloging an Archived Redo Log This example deletes an archived log from the operating system using an O/S utility, then uncatalogs the record:

```
% rm /oracle/arc/log1_100.arc
% rman target / catalog rman/rman@rcat
RMAN> change archivelog '/oracle/arc/log1_100.arc' uncatalog;
```

Related Topics

"allocateForMaint" on page 10-14

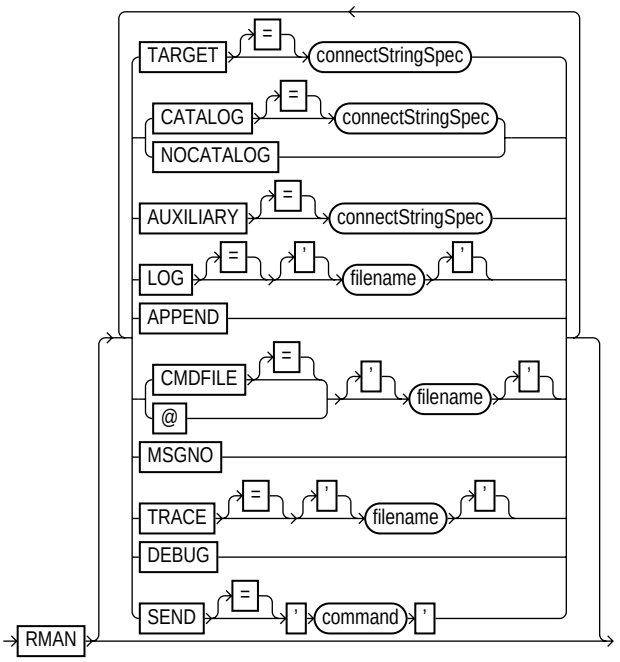
"archivelogRecordSpecifier" on page 10-18

"crosscheck" on page 10-64

"deleteExpired" on page 10-69

cmdLine

Syntax



Purpose

To start RMAN from the operating system command line. Use these arguments to:

- Connect to the target, recovery catalog, or auxiliary database.

  Note: On some platforms, you may choose to connect at the command line because the password is visible to other users on the system. The **connect** command is an alternative method that avoids this problem.

- Specify that you are using RMAN without a recovery catalog.
- Run a command file, which is a user-defined file containing RMAN commands.

- Specify the file in which RMAN records the results of processed commands.
- Add to rather than overwrite the contents of the command file.
- Generate debugging output and specify its location.
- Send a command to the media manager.
- Cause RMAN to print message numbers in the **list** output.

See Also: "Connecting to RMAN" on page 2-8.

Requirements

Use these arguments at the operating system command line rather than at the RMAN prompt.

Keywords and Parameters

target *connectStringSpec*	specifies a connect string to the target database, for example, **target** *sys/change_on_install@inst1*. See "connectStringSpec" on page 10-53.
catalog *connectStringSpec*	specifies a connect string to the database containing the recovery catalog, for example, **catalog** *rman/rman@inst2*. See "connectStringSpec" on page 10-53.
nocatalog	indicates that you are using RMAN without a recovery catalog.You must use this argument when starting Recovery Manager without a recovery catalog.
auxiliary *connectStringSpec*	specifies a connect string to an auxiliary database, for example, **auxiliary** *sys/change_on_install@dupdb*. See "connectStringSpec" on page 10-53.
log *filename*	specifies the file where Recovery Manager will record RMAN output, that is, the commands that were processed and their results. If you do not specify this argument, then Recovery Manager writes its message log file to standard output.
append	causes new output to be appended to the end of the message log file. If you do not specify this parameter and a file with the same name as the message log file already exists, RMAN overwrites it.
cmdfile *filename*	runs a file containing a user-defined list of RMAN commands. If the first character of the filename is alphabetic, then you can omit the quotes around the filename. The contents of the command file should be identical to commands entered at the RMAN prompt. For example, the following file contents will cause RMAN to connect to a target database and recovery catalog RCAT: `connect target; connect catalog rman/rman@rcat;` RMAN terminates after running the command file.
@*filename*	equivalent to **cmdfile**.

msgno	causes RMAN to print message numbers, that is, RMAN-xxxx, for the output of the **list** command. By default, **list** does not print the RMAN-xxxx prefix.
trace *filename*	specifies the name of the file in which RMAN logs debugging information. You must also specify the **debug** option to generate debugging output. If you specify **debug** without also specifying **trace**, then RMAN writes the debugging output to standard output or the message log if one is specified.
debug	activates the debugging feature. Use this option only for problem diagnosis under the direction of Oracle World Wide Support.
send '*command*'	sends a vendor-specific command string to all allocated channels. See your media management documentation to determine whether this feature is supported. **See Also:** "send" on page 10-136 to send a string to specific channels.

Examples

Connecting without a Recovery Catalog This example connects to the target database PROD1 without a recovery catalog:

```
% rman target sys/sys_pwd@prod1 nocatalog
```

Connecting to an Auxiliary Instance This example connects to the target database PROD1, the recovery catalog database RCAT, and the auxiliary instance AUX1:

```
% rman target sys/sys_pwd@prod1 catalog rman/rman@rcat auxiliary sys/aux_pwd@aux1
```

Specifying a Command File This example connects to the target database PROD1 and the recovery catalog database RCAT, and then runs the command file `b_whole_10.rcv`:

```
% rman target sys/sys_pwd@prod1 catalog rman/rman@rcat @'/oracle/dbs/b_whole_l0.rcv'
```

Specifying a Message Log in Append Mode This example connects to the target database PROD1 without a recovery catalog and then specifies that RMAN should append messages to the message log:

```
% rman target sys/sys_pwd@prod1 nocatalog log = $ORACLE_HOME/dbs/log/msglog.f append
```

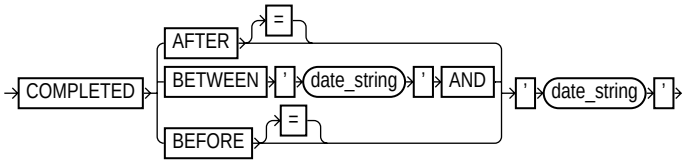
Related Topics

"connect" on page 10-51

"rmanCommand" on page 10-130

completedTimeSpec

Syntax



Purpose

A sub-clause that specifies when a backup or copy completed.

Requirements

All date strings must be either:

- Formatted according to the NLS date format specification currently in effect.
- Created by a SQL expression that returns a DATE value, for example, 'SYSDATE-30'.

Use this sub-clause in conjunction with the following commands:

- **crosscheck**
- **deleteExpired**
- **list**

> **Note:** The **from time** and **until time** parameters no longer work with commands that use *completedTimeSpec*. If you run a script that specifies these parameters, the job will fail.

Keywords and Parameters

after '*date_string*'	specifies the time after which the backup was completed.
between '*date_string*' **and** '*date_string*'	specifies a time range during which the backup was completed.
before '*date_string*'	specifies the time before which the backup was completed.

Examples

Crosschecking Backups within a Time Range This example crosschecks the backup sets of the database made last month:

```
crosscheck backup of database between 'SYSDATE-62' and 'SYSDATE-31';
```

Deleting Expired Backups This example deletes expired backup sets of `datafile 1` made in the last two weeks:

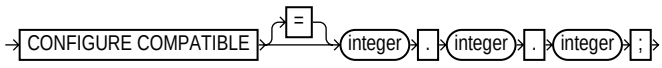
```
delete expired backup of datafile 1 after 'SYSDATE-14';
```

Listing Copies This example lists image copies of `/oracle/dbs/tbs_22.f` made before December 13, 1998:

```
list copy of datafile '/oracle/dbs/tbs_22.f' before 'Dec 13 1998 20:31:10';
```

configure

Syntax



Purpose

To control the compatibility of the recovery catalog packages with the RMAN executable.

The compatibility level of the recovery catalog is determined by a parameter that is stored in the catalog itself. It specifies the minimum acceptable release of the RMAN executable that can function with the catalog. For example, if the recovery catalog compatibility is set to 8.1.4, then only an RMAN executable of release 8.1.4 or later can connect to the catalog.

> **Note:** RMAN fails with error `RMAN-06191` if you try to run an old RMAN executable against a release 8.1.6 recovery catalog with compatibility set to 8.1.6.

The following table illustrates possible compatibility scenarios for a release 8.1.6 RMAN executable and a recovery catalog:

If you...	Then the recovery catalog...	And catalog compatibility is automatically set to...
Execute the **create catalog** command	Is created as a release 8.1.6 recovery catalog	8.1.6. **Note:** You cannot use the 8.1.6 catalog with a pre-8.1.6 release of the RMAN executable.
Execute the **upgrade catalog** command	Is upgraded from a pre-8.1.6 release to a release 8.1.6 catalog	8.0.4. **Note:** The 8.1.6 catalog is backwards compatible with older releases of the RMAN executable.

If you...	Then the recovery catalog...	And catalog compatibility is automatically set to...
Execute the **create catalog** command	Is created as a release 8.1.6 recovery catalog	8.1.6. **Note:** You cannot use the 8.1.6 catalog with a pre-8.1.6 release of the RMAN executable.

The compatibility level of the catalog is important because it affects the way that repository records are updated and deleted, as explained in the following table:

Table 10–3 Effect of Compatibility on Repository Record Removal

If compatibility is...	Then change ... delete and backup ... delete input...	And change ... uncatalog and prgrmanc.sql...
8.1.6 or higher	Delete backup sets, image copies, and archived logs and remove their records from the recovery catalog	Can remove archived log records from the catalog without creating problems when you attempt to restore archived logs whose records have been removed
8.1.5 or lower	▪ Delete backup sets and image copies and remove the records from the catalog ▪ Delete archived logs but change catalog records to status DELETED (and not remove them)	Can remove archived log records from the catalog while sometimes creating problems when you attempt to restore logs whose records have been removed

If you are using the release 8.1.6 RMAN executable and do not intend to use a pre-8.1.6 release against the recovery catalog, then you can raise the compatibility level of the catalog as follows:

```
configure compatible = 8.1.6;
```

After you issue this command, RMAN can delete rows from the AL recovery catalog table as archived redo logs are deleted through either the **change ... delete** command or the **backup archivelog ... delete input** command. Also, RMAN does not run into problems when attempting to restore archived logs whose records are no longer in the AL table.

Note: Existing AL table rows with status DELETED remain in the catalog. The `$ORACLE_HOME/rdbms/admin/prgrmanc.sql` script can delete these rows.

Requirements

- This command is only applicable if you are using a recovery catalog.
- Execute this command at the RMAN prompt.
- You cannot lower the compatibility level of the recovery catalog: it can only be increased.
- The specified version number must be within the ranges of versions supported by the DBMS_RCVMAN and DBMS_RCVCAT packages. The packages maintain behavior that is backwards compatible with this version of RMAN.
- If you issue **create catalog** using release 8.1.6 of the RMAN executable, then the recovery catalog is *not* backwards compatible with older releases of the RMAN executable. If you execute **upgrade catalog** using release 8.1.6 of the RMAN executable, then the recovery catalog *is* backwards compatible to release 8.0.4. To maintain this compatibility, RMAN cannot delete any rows from the AL recovery catalog table.
- To create an 8.1.6 catalog that is backwards compatible with pre-8.1.6 RMAN executables, you must issue **create catalog** with the older RMAN version first (or run `catrman.sql` if using a pre-8.1.5 release), and then execute **upgrade catalog** with the 8.1.6 RMAN.

Keywords and Parameters

integer.integer.integer	specifies a three-digit Oracle release number, for example, 8.1.6 or 8.0.4.

Examples

Creating an 8.1.6 Catalog for Use with Pre-8.1.6 RMAN Executables This example creates a release 8.1.6 recovery catalog that is compatible with RMAN executables released before 8.1.6. The scenario uses a release 8.1.5 RMAN executable to create the recovery catalog, and a release 8.1.6 RMAN executable to upgrade it:

```
% rman target / catalog rman/rman@rcat
```

```
Recovery Manager: Release 8.1.5.0.0

RMAN-06005: connected to target database: RMAN (DBID=1237603294)
RMAN-06008: connected to recovery catalog database

RMAN> create catalog;
RMAN> exit

% rman target / catalog rman/rman@rcat

Recovery Manager: Release 8.1.6.0.0

RMAN-06005: connected to target database: RMAN (DBID=1237603294)
RMAN-06008: connected to recovery catalog database

RMAN> upgrade catalog;
RMAN> upgrade catalog;

RMAN> exit
```

Increasing the Compatibility Level of the Catalog Assume the previous example, in which you have an 8.1.6 recovery catalog that can function with an 8.1.5 RMAN executable. If you want to specify that this recovery catalog can only function with an 8.1.6 or later release of RMAN, then issue the following:

```
RMAN> configure compatible = 8.1.6;
```

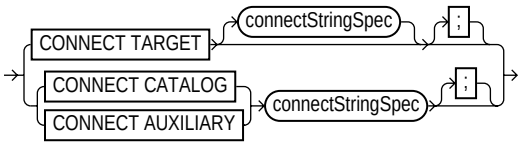
Related Topics

"createCatalog" on page 10-59

"upgradeCatalog" on page 10-158

connect

Syntax



Purpose

To establish a connection between RMAN and a target, auxiliary, or recovery catalog database.

> **Note:** When connecting from the command line, the password may be visible to other users on the system. The **connect** command avoids this problem.

See Also: "cmdLine" on page 10-42 for command line connection options.

Requirements

You can only use the **connect** command if you are at the RMAN prompt and if you are not already connected.

Keywords and Parameters

connect target *connectStringSpec*	establishes a connection between RMAN and the target database. See "connectStringSpec" on page 10-53.
connect catalog *connectStringSpec*	establishes a connection between RMAN and the recovery catalog database. See "connectStringSpec" on page 10-53.
connect auxiliary *connectStringSpec*	establishes a connection between RMAN and an auxiliary instance. An auxiliary instance can be used with the **duplicate** command or used during TSPITR. See "connectStringSpec" on page 10-53.

Examples

Connecting Without a Recovery Catalog This example starts RMAN and then connects to the target database with a Net8 service name `prod1`:

```
% rman nocatalog
RMAN> connect target sys/change_on_install@prod1;
```

Connecting with a Recovery Catalog This example starts RMAN and then connects to the target database PROD1 using operating system authentication and the recovery catalog database RCAT using a password file:

```
% rman
RMAN> connect target /; connect catalog rman/rman@rcat;
```

Connecting to Target, Recovery Catalog, and Duplicate Databases This example connects to three different databases specifying a username and password for each:

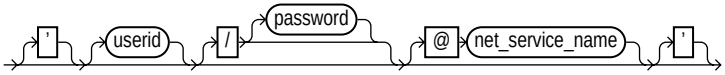
```
% rman
RMAN> connect target sys/sysdba@prod1;
RMAN> connect catalog rman/rman@rcat;
RMAN> connect auxiliary sys/sysdba@dupdb;
```

Related Topics

"cmdLine" on page 10-42

connectStringSpec

Syntax



Purpose

A sub-clause specifying the username, password, and net service name for connecting to a target, recovery catalog, or auxiliary database. The connection is necessary to authenticate the user and identify the database.

Requirements

- You must have SYSDBA privileges on the target and auxiliary databases.
- Do not connect to the recovery catalog database as user SYS.

Keywords and Parameters

/

if you do not specify a userid or password when connecting to the target database, a forward slash establishes a connection as SYS using operating system authentication. For example, enter the following to connect to the target database:

```
% rman target /
```

Note: The forward slash depends on the ORACLE_SID environment variable to know which database you want to connect to. The ORACLE_SID can point to either the auxiliary or target database, but not both at the same time.You cannot connect to the recovery catalog database using only the forward slash.

userid

establishes a connection to the database for the specified user. If you do not specify a password, RMAN obtains the password interactively by displaying a prompt. The characters will not be echoed to the terminal.

You must have SYSDBA authority when connecting to the target database, but must *not* connect as SYS to the recovery catalog database.

Note: The connect string must not contain any white space, but it can contain punctuation characters such at "/" and "@".

/password

establishes a connection for the specified user using a password. If the target database is not open, then a password file must exist.

@net_service_name	establishes a connection to the database using an optional Net8 net service name. The service name must be valid as specified in the `tnsnames.ora` file.

Examples

Connecting Without a Recovery Catalog This example connects to the target database using a password and the Net8 service name PROD1:

```
% rman target sys/change_on_install@prod1 nocatalog
```

Entering the Password Interactively This example connects to the target database as user SYS but without specifying a password at the command line:

```
% rman target sys

Recovery Manager: Release 8.1.5.0.0

target database Password:
```

Connecting with Operating System Authentication This example starts RMAN and then connects to the target database PROD1 using operating system authentication and the recovery catalog database RCAT using a password file:

```
% rman
RMAN> connect target /
RMAN> connect catalog rman/rman@rcat
```

Connecting to a Target Database, Recovery Catalog, and Auxiliary Instance This example connects to three different databases from the command line, specifying a username, password, and net service name for each:

```
% rman target sys/sysdba@prod1 catalog rman/rman@rcat auxiliary sys/sysdba@dupdb
```

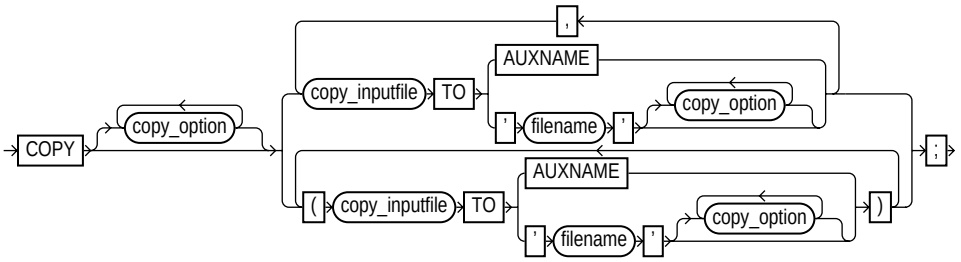
Related Topics

"cmdLine" on page 10-42

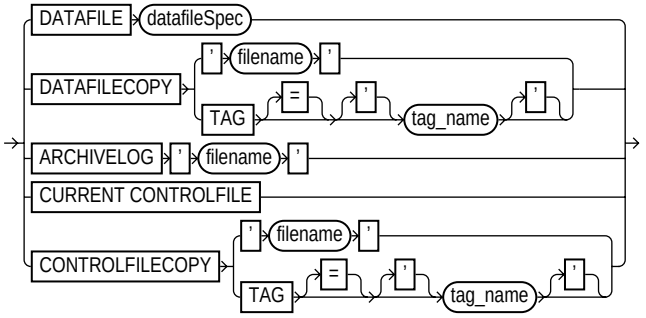
"connect" on page 10-51

copy

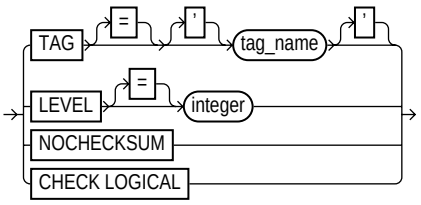
Syntax



copy_inputfile::=



copy_option::=



Purpose

Create an image copy of a file. The output file is always written to disk. You can copy the following types of files:

- Datafiles (current or copies)
- Archived redo logs
- Control files (current or copies)

In many cases, copying datafiles is more beneficial than backing them up, since the output is 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 backs up only one datafile and contains a single backup piece.

See Also: "Making Image Copies" on page 5-13.

Requirements

- Execute the command from within the braces of a **run** command.
- Precede a **copy** command with at least one **allocate channel** command specifying the **type disk** option.
- You cannot make incremental copies.

Keywords and Parameters

copy_option specifies optional parameters affecting either the input or output files or both.

tag *tag_name*	specifies the tag of the input file or output file copy.
level *integer*	includes the input file or output file copy in the incremental backup strategy by making it serve as a basis for subsequent incremental backup sets. Typically, you specify **level 0**. If you do not use the **level** option, then the datafile copy has no impact on the incremental backup strategy.
nochecksum	suppresses block checksums. Unless you specify this option, Oracle computes a checksum for each block. RMAN verifies the checksum when restoring the copy. If the database is already maintaining block checksums, then this flag has no effect.

	check logical	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, then it logs the block in the `alert.log` and server session trace file. Provided the sum of physical and logical corruptions detected for a file remain below its **maxcorrupt** setting, the RMAN command completes and Oracle populates V$COPY_CORRUPTION with corrupt block ranges. If **maxcorrupt** is exceeded, then the command terminates without populating the views. **Note:** For **copy** and **backup** the **maxcorrupt** setting represents the total number of physical and logical corruptions permitted on a file.
copy_inputfile clause	specifies the type of input file, that is, the file that you want to copy.	
	datafile *datafileSpec*	specifies a list of one or more datafiles as input. See "datafileSpec" on page 10-66. **Note:** If you specify a filename, then it must be the name of a current datafile as listed in the control file.
	datafilecopy	specifies a list of one or more datafile copies as input. Specify the datafile copies by *'filename'* or **tag =** *tag_name*. The filename must *not* be the name of a current datafile listed in the control file. The existing copy may have been created by either a previous **copy** command or by an external operating system utility.
	archivelog *'filename'*	specifies the filename of an input archived redo log. The archived log may have been created by the Oracle archiving session or by a previous **copy** command. Specify the archived redo log by filename.
	current controlfile	specifies the current control file.
	controlfilecopy *'filename'*	specifies the filename of a control file copy. You can also set **tag =** *tag_name* to specify a list of one or more control file copies. **Note:** The control file copy is marked as a backup control file, so media recovery will be necessary if you mount the control file copy. This command is equivalent to the ALTER DATABASE BACKUP CONTROLFILE TO '...' statement.
to '*filename*'	specifies the filename of the output file copy.	
to auxname	specifies that Oracle should copy the input datafile to the filename specified in an earlier **set auxname** command for the input datafile.	

Examples

Copying a Datafile This example copies the datafile `tbs_01.f` with the **nochecksum** option to the output file `temp3.f`, marking it as a level 0 backup:

```
run {
     allocate channel dev1 type disk;
     copy
       nochecksum
       datafile '$ORACLE_HOME/dbs/tbs_01.f'
         to '$ORACLE_HOME/copy/temp3.f'
       level 0;
}
```

Copying the Control File This example copies the current control file and gives the copy the tag *weekly_cf_copy*:

```
run {
     allocate channel dev1 type disk;
     copy
       current controlfile
         to '$ORACLE_HOME/copy/cf1.f'
       tag = 'weekly_cf_copy';
}
```

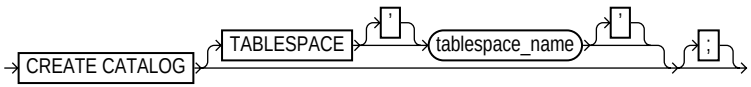
Related Topics

"allocate" on page 10-10

"backup" on page 10-22

createCatalog

Syntax



Purpose

To create a schema for the recovery catalog. Typically, you create this schema in a separate recovery catalog database.

Note: In releases prior to 8.1.5, you created the recovery catalog schema by connecting to the recovery catalog database and executing the `catrman.sql` script.

See Also: "Creating the Recovery Catalog" on page 3-2 to learn how to create the recovery catalog.

Requirements

- Execute this command only at the RMAN prompt.
- The recovery catalog owner must be granted the RECOVERY_CATALOG_OWNER role, and also be granted space privileges in the tablespace where the recovery catalog tables will reside.
- RMAN must be connected to the recovery catalog database either through the **catalog** command-line option (see "cmdLine" on page 10-42) or the **connect catalog** command.
- Do not create the recovery catalog in the SYS schema.

See Also: *Oracle8i Administrator's Guide* for more information about the RECOVERY_CATALOG_OWNER role.

Keywords and Parameters

tablespace *tablespace_name*	specifies the tablespace in which to store the recovery catalog schema. The catalog owner must be granted quota privileges. If you do not specify a tablespace, RMAN stores the recovery catalog in the SYSTEM tablespace.

Examples

Creating a Catalog Schema This example creates a user RMAN, grants RMAN the RECOVERY_CATALOG_OWNER role, then creates the recovery catalog in the schema RMAN.CATTBS of the database RCAT:

```
% sqlplus sys/change_on_install@rcat;

SQL> CREATE USER rman IDENTIFIED BY rman
  2> DEFAULT TABLESPACE cattbs QUOTA UNLIMITED ON cattbs;
SQL> GRANT recovery_catalog_owner TO rman;
SQL> exit

% connect catalog rman/rman@rcat;
RMAN> create catalog tablespace cattbs;
```

Related Topics

"connect" on page 10-51

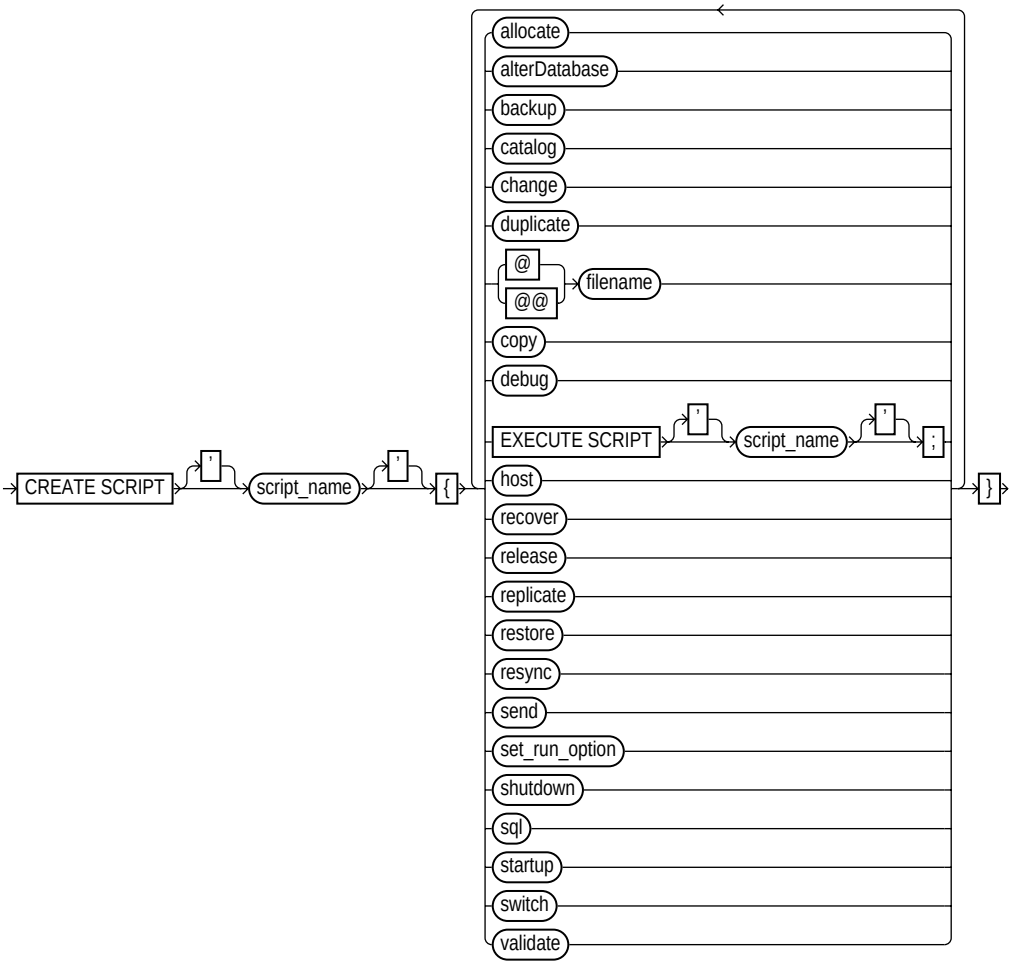
"configure" on page 10-47

"dropCatalog" on page 10-74

"upgradeCatalog" on page 10-158

createScript

Syntax



Purpose

To create a script and store it in the recovery catalog for future reference. Stored scripts provide a common repository for frequently executed collections of RMAN commands: use any command legal within a **run** command in the script. The script is not executed immediately; use the **execute script** command (see "run" on page 10-133) to run it.

See Also: "Storing Scripts in the Recovery Catalog" on page 3-27.

Requirements

Note the following restrictions:

- Execute **create script** only at the RMAN prompt.
- You must be connected to the recovery catalog.
- You cannot execute a **run** command within a stored script. When you run an **execute script** command within a **run** command, RMAN places the contents of the script between the braces of **run**. For this reason, you should not allocate a channel at the **run** command level if you already allocated it in the script.

Keywords and Parameters

For descriptions of the individual commands that you can use in a stored script, see the appropriate entry, for example, "backup" on page 10-22. Note that the **@@** command exhibits special behavior when you execute it within a script. For information on the **execute script** command, see "run" on page 10-133.

script_name	creates a stored script with the specified name. The statements allowable within the parentheses of the **create script** '*script_name*' **(...)** command are the same allowable within the **run** command. The statements within the braces constitute the *job_command_list*. **Note:** To execute the stored script, use the **execute script** command within the braces of the **run** command.
@*filename*	executes a series of RMAN commands stored in an operating system file with the specified full pathname, for example, `@$ORACLE_HOME/dbs/cmd/cmd1.f`. Do not use quotes around the string or leave whitespace between the @ and filename. RMAN processes the specified file as if its contents had appeared in place of the @ command. **Note:** The file must contain only complete Recovery Manager commands. A syntax error results if the file contains a partial command.

@@filename	specifies the relative filename of an operating system file containing a series of RMAN commands, for example, `cmd1.f`. The command file specified by **@@** is assumed to be in the same directory as the parent script. Do not use quotes around the string or leave whitespace between the @**@** and filename. RMAN processes the specified file as if its contents had appeared in place of the @**@** command. **Note:** The file must contain only complete Recovery Manager commands. A syntax error will result if the file contains a partial command

Examples

Creating a Script This example creates a script called B_WHOLE_10 that backs up the database and archived redo logs, then executes it:

```
create script b_whole_l0 {
    allocate channel d1 type disk;
    allocate channel d2 type disk;
    allocate channel d3 type disk;
    backup
        incremental level 0
        tag b_whole_l0
        filesperset 6
        database;
    sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
    backup
        filesperset 20
        archivelog all
        delete input;
}

RMAN-03022: compiling command: create script
RMAN-03023: executing command: create script
RMAN-08085: created script b_whole_10

run { execute script b_whole_10; }
```

Related Topics

"deleteScript" on page 10-71

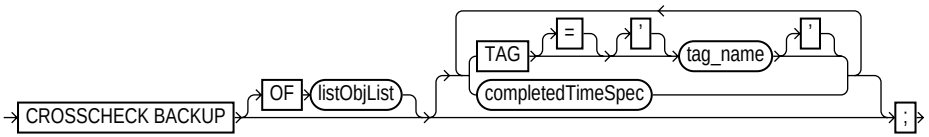
"printScript" on page 10-94

"replaceScript" on page 10-105

"run" on page 10-133

crosscheck

Syntax



Purpose

To determine whether backups stored on disk or tape exist. Backups are either backup sets or media-managed proxy copies.

The **crosscheck** command checks only backup sets marked AVAILABLE or EXPIRED, either by examining the backup pieces on disk when the channel is **type disk**, or by querying the media manager when the channel is **type** *'sbt_tape'*. It only processes backups created on the specified channel.

RMAN does not delete any backup pieces that it is unable to find, but updates their repository records to EXPIRED status. If some backup pieces were erroneously marked as EXPIRED, for example, because the media manager was misconfigured, then after ensuring that the files really do exist in the media manager, run the **crosscheck backup** command again to restore those files to AVAILABLE status.

See Also: "Maintaining the RMAN Repository" on page 3-8.

Requirements

- Execute **crosscheck backup** only at the RMAN prompt.
- Allocate a maintenance channel before issuing **crosscheck backup**.

Keywords and Parameters

of *listObjList*	restricts the list of objects operated on to the object type specified in the *listObjList* clause. If no objects are specified, the command checks all objects: **of database controlfile archivelog all**. See "listObjList" on page 10-92.
tag *tag_name*	specifies the tag for the backup set.
completedTimeSpec	specifies a time range for backup completion. See "completedTimeSpec" on page 10-92.

Examples

Crosschecking All Backups The following example queries the status of all backups on disk and includes sample output:

```
RMAN> allocate channel for maintenance type disk;

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: delete
RMAN-08500: channel delete: sid=15 devtype=DISK

RMAN> crosscheck backup;

RMAN-03022: compiling command: XCHECK
RMAN-03023: executing command: XCHECK
RMAN-08517: backup piece handle=/vobs/oracle/dbs/07a9ck4t_1_1 recid=7 stamp=3453
RMAN-08074: crosschecked backup piece: found to be 'AVAILABLE'
RMAN-08517: backup piece handle=/vobs/oracle/dbs/08a9cl23_1_1 recid=8 stamp=3453
RMAN-08074: crosschecked backup piece: found to be 'EXPIRED'
RMAN-08517: backup piece handle=/vobs/oracle/dbs/09a9cl2b_1_1 recid=9 stamp=3453
RMAN-08074: crosschecked backup piece: found to be 'AVAILABLE'

RMAN> release channel;

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
RMAN-08031: released channel: delete
```

Crosschecking Within a Range of Dates The following example queries the media manager for the status of the backup sets for `datafile 3` in a given six-month range. Note that RMAN uses the date format specified in the NLS_DATE_FORMAT parameter, which is 'DD-MON-YY' in this example:

```
allocate channel for maintenance type 'sbt_tape';
crosscheck backup of datafile 3 device type 'sbt_tape' completed between '01-JAN-98' and
     '01-JUL-98';
release channel;
```

Related Topics

"allocateForMaint" on page 10-14

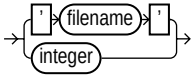
"change" on page 10-38

"list" on page 10-83

"report" on page 10-110

datafileSpec

Syntax



Purpose

A sub-clause that specifies a datafile by filename or absolute file number.

Keywords and Parameters

'*datafile*'	specifies the datafile using either the full path or a relative filename. If you specify a relative filename, the filename is qualified in a port-specific manner by the target database.
integer	specifies the datafile using its absolute file number. Obtain the file number from the V$DATAFILE, V$DATAFILE_COPY, or V$DATAFILE_HEADER views or **report schema** command output.

Examples

Specifying a Datafile by Filename This example copies datafile `/oracle/dbs/tbs_12` to disk, specifying it by filename:

```
run {
     allocate channel ch1 type disk;
     copy datafile '/oracle/dbs/tbs_12.f'
       to '/oracle/copy/tbs_1.copy';
}
```

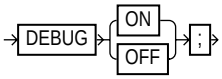
Specifying a Datafile by Absolute File Number This example copies datafile `/oracle/dbs/tbs_31.f` to disk, specifying it by file number:

```
RMAN> report schema;

RMAN-03022: compiling command: report
Report of database schema
File K-bytes    Tablespace           RB segs Name
---- ---------- -------------------- ------- ------------------
1         47104 SYSTEM               YES     /vobs/oracle/dbs/tbs_01.f
2           978 SYSTEM               YES     /vobs/oracle/dbs/tbs_02.f
```

```
3           978 TBS_1                NO      /vobs/oracle/dbs/tbs_11.f
4           978 TBS_1                NO      /vobs/oracle/dbs/tbs_12.f
5           978 TBS_2                NO      /vobs/oracle/dbs/tbs_21.f
6           978 TBS_2                NO      /vobs/oracle/dbs/tbs_22.df
7           500 TBS_1                NO      /vobs/oracle/dbs/tbs_13.f
8           500 TBS_2                NO      /vobs/oracle/dbs/tbs_23.f
9           500 TBS_2                NO      /vobs/oracle/dbs/tbs_24.f
10          500 TBS_3                NO      /vobs/oracle/dbs/tbs_31.f

run {
     allocate channel ch1 type disk;
     copy datafile 10
     to '/oracle/copy/tbs_31.copy';
}
```

debug

Syntax



Purpose

To turn RMAN's debugging feature off and on. Because its purpose is to diagnose RMAN problems, use this feature under the guidance of Oracle Support.

See Also: "Interpreting Debugging Output" on page 9-8.

Requirements

Execute this command at the RMAN prompt or within the braces of a **run** command.

Keywords and Parameters

on	activates the debugging feature on.
off	deactivates the debugging feature.

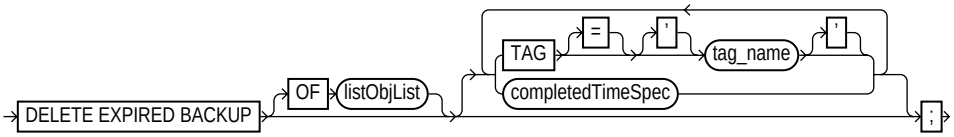
Examples

Activating the Debugging Feature In this example, debug output will be displayed during the backup of datafile 3, but not datafile 4:

```
run {
     allocate channel c1 type disk;
     debug on;
     backup datafile 3;
     debug off;
     backup datafile 4;
}
```

deleteExpired

Syntax



Purpose

To remove backup set records with status EXPIRED from the recovery catalog. This command operates only on the recovery catalog records for the backup pieces marked EXPIRED by the **crosscheck** command. Use the **list** command or query the recovery catalog views to obtain the status of backup sets.

> **Note:** If for some reason a backup set marked EXPIRED exists when you run the **delete expired backup** command, RMAN deletes the physical files.

Requirements

- You must be using a recovery catalog.
- Execute **delete expired backup** only at the RMAN prompt.
- Precede **delete expired backup** with an **allocate channel for delete** or an **allocate channel for maintenance** command (see "allocateForMaint" on page 10-14).

Keywords and Parameters

of *listObjList*	restricts the list of objects to the object type specified in the *listObjList* clause. If no objects are specified, the command deletes all expired objects: **of database controlfile archivelog all**. See "listObjList" on page 10-92.
tag *tag_name*	specifies the tag for the backup set.
completedTimeSpec	specifies a time range for backup completion. See "completedTimeSpec" on page 10-92.

Examples

Deleting Expired Backups The following example checks the media manager for expired backups of the tablespace USER_DATA that are more than one month old and removes their catalog records:

```
allocate channel for delete ch1 type 'sbt_tape';
crosscheck backup of tablespace user_data completed before 'SYSDATE-31';
delete expired backup of tablespace user_data' completed before 'SYSDATE-31';
release channel;
```

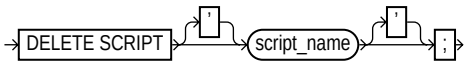
Related Topics

"allocateForMaint" on page 10-14

"crosscheck" on page 10-64

deleteScript

Syntax



Purpose

To delete a stored script from the recovery catalog.

Requirements

- Execute **delete script** only at the RMAN prompt.
- You must be using a recovery catalog.

Keywords and Parameters

script_name	deletes the specified script. The script name must be one of the names specified in a previous **create script** or **replace script** command (see "createScript" on page 10-61).

Examples

Deleting a Script This example deletes the script `b_whole_10`:

```
delete script 'b_whole_10';
```

Related Topics

"createScript" on page 10-61

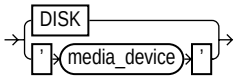
"printScript" on page 10-94

"replaceScript" on page 10-105

"run" on page 10-133

deviceSpecifier

Syntax



Purpose

A sub-clause specifying the type of storage for a backup or copy.

Keywords and Parameters

disk	specifies disk storage.
'*media_device*'	specifies a sequential I/O device or access method for storage. The syntax and semantics of sequential I/O device types are platform-specific. Currently, the only available value is '*sbt_tape*', which functions with a third-party tape sub-system interface.

Examples

Allocating a Tape Channel This example allocates a maintenance channel for a media management device:

```
allocate channel for maintenance type 'sbt_tape';
```

Backing Up to Disk This example backs up the database to disk:

```
run {
     allocate channel ch1 type disk;
     backup database;
}
```

Restoring from Disk and Tape This example recovers the database using backups from disk and tape:

```
run {
     allocate channel d1 type disk;
     allocate channel t1 type 'sbt_tape'
     restore database;
     recover database;
}
```

Related Topics

"allocate" on page 10-10

"allocateForMaint" on page 10-14

"list" on page 10-83

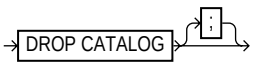
"releaseForMaint" on page 10-104

"report" on page 10-110

"restore" on page 10-120

dropCatalog

Syntax



Purpose

To remove the schema from the recovery catalog.

WARNING: This command deletes all information from the recovery catalog; if you have no backups of the recovery catalog, then all backups of all databases managed by this recovery catalog become unusable after you execute this command.

See Also: "Dropping the Recovery Catalog" on page 3-44.

Requirements

- Execute this command only at the RMAN prompt.
- You must be connected to the recovery catalog database through the **catalog** command-line option (see "cmdLine" on page 10-42) or the **connect catalog** command.
- Enter the command twice to confirm that you want to drop the schema.

Examples

Deleting the Catalog This example drops the schema from the recovery catalog (you must enter the command twice to confirm):

```
RMAN> drop catalog

RMAN-06435: recovery catalog owner is rman
RMAN-06436: enter DROP CATALOG command again to confirm catalog removal
RMAN> drop catalog
```

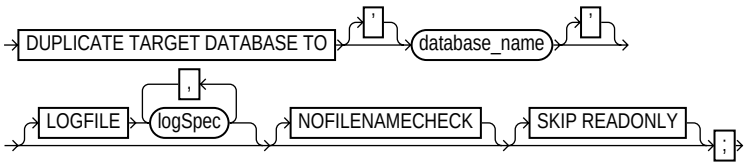
Related Topics

"createCatalog" on page 10-59

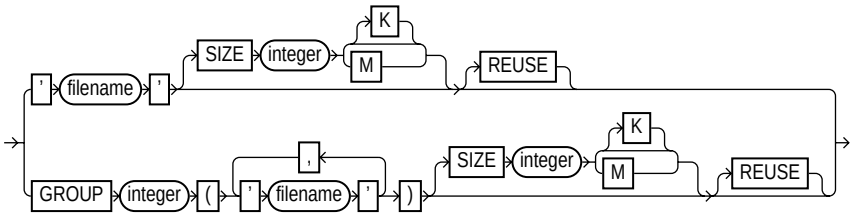
"upgradeCatalog" on page 10-158

duplicate

Syntax



logSpec::=



Purpose

To use backups of the target database to create a duplicate database. A duplicate database provides a safe environment for testing backup and recovery procedures.

When duplicating a database that is currently in NOARCHIVELOG mode, recovery occurs using the **noredo** option. Hence, if incremental backups exist, RMAN applies only these backups to the restored files during recovery. For databases in ARCHIVELOG mode, **duplicate** recovers by default up to the last archived redo log generated at the time the command was executed—unless the **set until** clause is specified, in which case recovery is bounded by the parameter settings.

> **See Also:** Chapter 7, "Creating a Duplicate Database with Recovery Manager" to learn how to duplicate a database.

Requirements

- Execute this command only within the braces of a **run** command.
- This command does not require the use of a recovery catalog.

- Issue one or more **allocate auxiliary channel** commands before executing the **duplicate** command. If you are duplicating from image copies or disk backups, then the more channels you allocate, the faster the duplicating operation.
- You must be connected to both the target database and auxiliary instance.
- The auxiliary instance must be started with the NOMOUNT option.
- The target database must be open.
- If your target and duplicate databases reside on the same host, set the CONTROL_FILES parameter appropriately so that the duplicate database control file does not overwrite the target control file. See the *Oracle8i Reference* for a description of this parameter.
- If your target and duplicate databases share the same host, set all *_PATH and *_DEST initialization parameters appropriately so that your target database files are not overwritten by the duplicate database files. See the *Oracle8i Reference* for descriptions of these parameters.
- If your target and duplicate databases reside on different hosts, then you must do one of the following:
  - Move disk copies and backups from the target host to the duplicate host and re-catalog them.
  - Make sure that all backups and copies on the target host are remotely accessible from the duplicate host.
- Specify new filenames or convert target filenames for the datafiles and online redo logs. If you do not specify filenames, RMAN reuses the target datafile names. You *must* use **nofilenamecheck** in this case.
- The **duplicate** command automatically restores the appropriate backup or copy of each target datafile. If any target datafile does not have a backup, then the duplicate operation will abort.

Keywords and Parameters

database_name specifies the name of the duplicate database. You must specify the database name because the new database is started but not mounted, so the database name cannot be obtained. The name should match the name in the `init.ora` file of the duplicate database or Oracle will signal an error when creating the control file.

Note: You can use the same database name for the target and duplicate databases since RMAN generates a new DBID for the duplicate database.

logfile *logSpec* specifies the online redo logs. The syntax is the same used in the LOGFILE option of the CREATE DATABASE statement.

If you do not specify the logfile clause, then RMAN uses LOG_FILE_NAME_CONVERT if it is set. If neither logfile nor LOG_FILE_NAME_CONVERT is specified, RMAN uses the original target redo log filenames for the duplicate files. You must use the nofilenamecheck option in this case.

See Also: *Oracle8i SQL Reference* for more about the CREATE DATABASE statement.

'filename'	specifies the filename of the online redo log.
size *integer*	specifies the size of the file in kilobytes (**K**) or megabytes (**M**). If you omit this parameter, the file must already exist.
reuse	allows Oracle to reuse an existing file. If the file already exists, Oracle verifies that its size matches the value of the **size** parameter. If the file does not exist, Oracle creates it. If you omit the **size** parameter, the file must already exist. The **reuse** option is significant only when used in conjunction with the **size** parameter. If you omit the **size** parameter, Oracle expects the file to exist already.
group *integer* ('*filename*', ...)	specifies a redo log group containing one or more members. Each filename specified within the parentheses indicates a member of the group.

nofilenamecheck prevents RMAN from checking whether target datafiles sharing the same names as the duplicated files are in use. The user is responsible for determining that the duplicate operation will not overwrite useful data.

This option is necessary when you are creating a duplicate database in a different host that has the same disk configuration, directory structure, and filenames as the host of the target database. For example, imagine a small database that in the `/dbs` directory of HOST1:

```
/oracle/dbs/system_prod1.dbf
/oracle/dbs/users_prod1.dbf
/oracle/dbs/tools_prod1.dbf
/oracle/dbs/rbs_prod1.dbf
/oracle/dbs/users2_prod1.dbf
```

Assume that you want to duplicate the database in machine HOST2, which happens to have the same file system `/oracle/dbs/*`, and you want to use the same filenames in the duplicate as in the primary. In this case, use the **nofilenamecheck** option to avoid renaming all the files. Because RMAN is not aware of the different hosts, RMAN cannot determine automatically that it should not check the filenames.

skip readonly excludes the datafiles in read-only tablespaces from the duplicate database.

Note: A record for the skipped read-only tablespace still appears in DBA_TABLESPACES. This feature allows you to activate the read-only tablespace later. For example, you can store the read-only tablespace data on a CD-ROM instead of on disk, then mount the CD-ROM later and view the data.

Examples

Setting New Filenames Manually This example assumes that your target database is on HOST1 and you wish to duplicate your database to NEWDB on `host2` with the file structure `/oracle/dbs/*`. Because the filenames in HOST1 are irregularly named and located in various sub-directories, you use **set newname** commands to re-name the files consistently. The **duplicate** command uses backup sets stored on tape to duplicate the target database to database NEWDB:

```
connect target;
connect catalog rman/rman@rmancat;
connect auxiliary sys/change_on_install@newdb;
run {
    allocate auxiliary channel newdb1 type 'sbt_tape';
    allocate auxiliary channel newdb2 type 'sbt_tape';
    allocate auxiliary channel newdb3 type 'sbt_tape';
    allocate auxiliary channel newdb4 type 'sbt_tape';
    set newname for datafile 1 TO '$ORACLE_HOME/dbs/newdb_data_01.f';
    set newname for datafile 2 TO '$ORACLE_HOME/dbs/newdb_data_02.f';
    set newname for datafile 3 TO '$ORACLE_HOME/dbs/newdb_data_11.f';
    set newname for datafile 4 TO '$ORACLE_HOME/dbs/newdb_data_12.f';
    set newname for datafile 5 TO '$ORACLE_HOME/dbs/newdb_data_21.f';
    set newname for datafile 6 TO '$ORACLE_HOME/dbs/newdb_data_22.f';
    duplicate target database to newdb logfile
      group 1 ('$ORACLE_HOME/dbs/newdb_log_1_1.f',
               '$ORACLE_HOME/dbs/newdb_log_1_2.f') size 200K,
      group 2 ('$ORACLE_HOME/dbs/newdb_log_2_1.f',
               '$ORACLE_HOME/dbs/newdb_log_2_2.f') size 200K reuse;
}
```

Reusing the Target Filenames This example assumes that you are restoring to a new host and that:

- The target host and duplicate host have the same file structure.
- You wish to name the duplicate files exactly like the target database files.
- You are not using a recovery catalog.
- You do not want to duplicate read-only tablespaces.
- You want to prevent RMAN from checking whether files on the target database that have the same names as the duplicated files are in use.

```
connect target
connect auxiliary sys/aux_pwd@newdb
run {
     allocate auxiliary channel ndbnewh1 type disk;
     allocate auxiliary channel ndbnewh2 type disk;
```

```
    duplicate target database to ndbnewh
    logfile
      '$ORACLE_HOME/dbs/log_1.f' size 200K,
      '$ORACLE_HOME/dbs/log_2.f' size 200K
    skip readonly
    nofilenamecheck;
}
```

Related Topics

"allocate" on page 10-10

"connect" on page 10-51

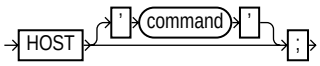
"copy" on page 10-55

"set" on page 10-138

"startup" on page 10-152

host

Syntax



Purpose

To invoke an operating system command-line sub-shell from within RMAN.

Requirements

Execute this command at the RMAN prompt or within the braces of a **run** command.

Keywords and Parameters

host enables you to execute an operating system command. Use this parameter:

- with a *'command'*, in which case RMAN runs the command in the specified string and then continues.
- without a *'command'*, in which case RMAN displays a command prompt and resumes after you exit the sub-shell.

Examples

Executing an Operating System Copy Within RMAN This example shuts down the database, makes a backup of datafile `tbs_01.f` using a media manager, then makes an image copy of the same file on disk using a UNIX command. The database needs to be shut down cleanly to prevent fractured blocks:

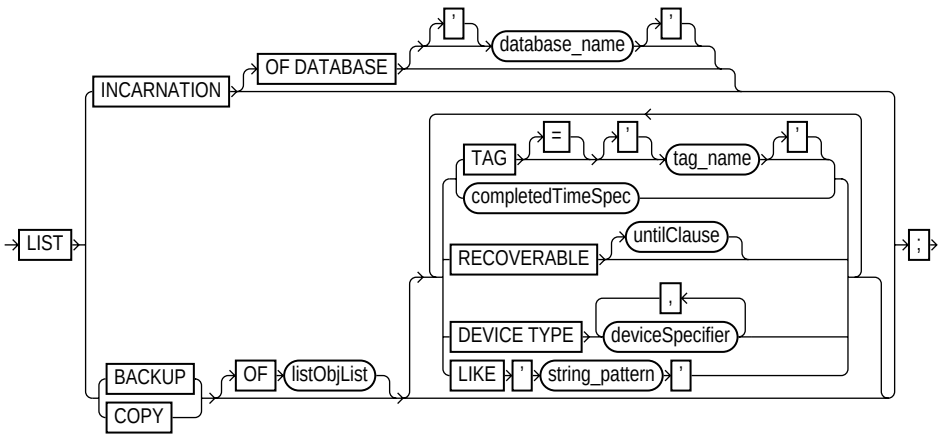
```
shutdown immediate;
run {
     allocate channel ch1 type disk;
     allocate channel ch2 type 'sbt_tape';
     backup datafile '$ORACLE_HOME/dbs/tbs_01.f' channel ch2;
     host 'cp $ORACLE_HOME/dbs/tbs_01.f $ORACLE_HOME/dbs/copy/tbs_01.f';
}
```

Hosting to the Operating System Within a Copy Job This example makes an image copy of datafile 3, hosts out to the UNIX prompt to check that the copy is in the directory, then resumes the **run** job:

```
RMAN> run {
2> allocate channel c1 type disk;
3> copy datafile 3 to 'df.3';
4> host;
5> release channel c1;
6> }

RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: c1
RMAN-08500: channel c1: sid=17 devtype=DISK

RMAN-03022: compiling command: copy
RMAN-03023: executing command: copy
RMAN-08000: channel c1: copied datafile 3
RMAN-08501: output filename=/oracle/dbs/df.3 recid=102 stamp=352745706
RMAN-03023: executing command: partial resync
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete

RMAN-03022: compiling command: host
% ls df.3
df.3
% exit
exit
RMAN-06134: host command complete

RMAN-03022: compiling command: release
RMAN-03023: executing command: release
RMAN-08031: released channel: c1
```

list

Syntax



Purpose

To produce a detailed listing of specified backups (either backup sets or media-managed proxy copies) or image copies recorded in the recovery catalog or target control file. RMAN records the output to either standard output or the message log (see "cmdLine" on page 10-42), but not to both at the same time. Use this command to list:

- Backups or image copies of a specified list of datafiles.
- Backups or image copies of any datafile that is a member of a specified list of tablespaces.
- All backups or image copies of all datafiles in the database, optionally restricted by time, datafile copy filename, device name, recoverability, or tag.
- Backups or copies of archived redo logs with a specified name and/or within a specified range.
- Incarnations of a specified database or of all databases known to the recovery catalog.

See Also: Chapter 4, "Generating Lists and Reports with Recovery Manager" to learn how to make lists and reports.

Requirements

- Execute **list** only at the RMAN prompt.
- The **incarnation** option requires the use of a recovery catalog.
- You must be connected to the target database. If you use a recovery catalog, you must also be connected to it.
- The **list** command will not show any records with DELETED status. To see the records with DELETED status, query the recovery catalog views. See Chapter 11, "Recovery Catalog Views".
- If you generate a list of backups and copies limited by time, SCN, or log sequence number, you must specify the *completedTimeSpec* rather than the *untilClause*. The *untilClause* no longer works with the **list** command for release 8.1.5 and higher.
- You cannot use the **like** option with the **list ... archivelog** command.

Keywords and Parameters

incarnation displays information about the incarnations of a database. See Table 10–10 for an explanation of the column headings of the **list incarnation** output table.

The listing includes the primary keys of all database incarnation records for the specified database name. Use the key in a **reset database** command to change the incarnation that RMAN considers to be current to a previous incarnation. If you do not specify the **of database** option, then the command lists all databases registered in the recovery catalog.

of database *database_name* specifies the name of the database.

copy displays information about datafile copies, archived redo logs, and image copies of archived redo logs. By default, **list** will display copies of all files in the database. Both usable and unusable image copies are included in the output, even those that cannot be restored or are expired or unavailable.

See Also: Table 10–8 and Table 10–9 for an explanation of the column headings of the **list copy** output tables.

backup	displays information about backups: backup sets, backup pieces, and proxy copies. The output displays a unique key for each. By default, backups of the whole database are listed. Both usable and unusable backups are included in the output, even those that cannot be restored, are expired or unavailable, or are incremental backups that cannot be restored because their parent full backup or copy no longer exists. **See Also:** Table 10–4, Table 10–5, Table 10–6, and Table 10–7 for an explanation of the column headings of the **list backup** output tables. Use the KEY column of the output to obtain the primary key usable in the **change** and **delete expired backupset** commands.
of *listObjList*	restricts the list of objects operated on to the object type specified in the *listObjList* clause. See "listObjList" on page 10-92. If you do not specify an object, **list** defaults to **of database**.
completedTimeSpec	specifies a range of time for completion of the backup or copy. See "completedTimeSpec" on page 10-45.
tag *tag_name*	restricts the datafile copies and backups by specifying the tag of the copy or backup. If you specify **tag**, only copies or backups with the specified tag will be listed.
recoverable	specifies only backups or copies of datafiles that are available and can possibly be used in a restore operation. To be a candidate for restore operations a backup must meet these criteria. If the backup is: ▪ Incremental, then a valid parent must exist to which this incremental can be applied. ▪ In a prior incarnation, then there must be no further changes to the files in that incarnation. In other words, the files must be offline and must not have come online again in that incarnation. *untilClause* specifies an end time, SCN, or log sequence number. See "untilClause" on page 10-156.
device type *deviceSpecifier*	lists only backup sets residing on one of the specified device types (see "deviceSpecifier" on page 10-72). If not specified, all available backup sets will be listed. This option applies only to the **list backup** command.
like *string_pattern*	restricts datafile copies by specifying a filename pattern. The pattern can contain Oracle pattern matching characters '%' and '_'. RMAN lists only files whose name matches the pattern. **Note:** You cannot use the **like** option with the **list ... archivelog** command.

List Output

The status information that appears in the output is shown in Table 10–4:

Table 10–4 List of Backup Sets **(Page 1 of 2)**

Column	Indicates
KEY	a unique key identifying this backup set. **Note:** If the target database control file is used instead of the recovery catalog, then this field is a unique identifier that specifies this backup set in the target database control file (and is equal to the RECID, which serves this purpose when a recovery catalog is not used). Use this key in a **change ... backupset** statement to change the status of the backup set.
RECID	when combined with the STAMP column, a unique key that identifies this backup set in the target database control file. The RECID will be invalid when a new control file record occupies the space used by the old record. For this reason, issue **resync** commands often so that the new records are copied to the recovery catalog as soon as possible.
STAMP	when combined with the RECID column, a unique key that identifies this backup set in the target database control file.
LV	the level of the backup: NULL for non-incrementals, level 0-4 for incrementals.
SET STAMP	when combined with the SET COUNT column, a unique key that identifies this backup set in the target database control file. Use these values to access the control file records in the V$BACKUP_SET, V$BACKUP_PIECE, V$BACKUP_DATAFILE, and V$BACKUP_REDOLOG views. The SET STAMP value is valid at all times, both in the control file (when not using a recovery catalog) and when using a recovery catalog. SET STAMP values are never entered by a user because they are part of a two-value key. Oracle World Wide Support may request this value if your database requires recovery when no recovery catalog exists and control file records are gone. **See Also:** *Oracle8i Reference* for more information about data dictionary views.

Table 10–4 List of Backup Sets **(Page 2 of 2)**

SET COUNT	when combined with the SET STAMP column, a unique key that identifies this backup set in the target database control file. Use these values to access the control file records in the V$BACKUP_SET, V$BACKUP_PIECE, V$BACKUP_DATAFILE, and V$BACKUP_REDOLOG views. The SET COUNT value is valid at all times, both in the control file (when not using a recovery catalog) and when using a recovery catalog. SET COUNT values are never entered by a user because they are part of a two-value key. Oracle World Wide Support may request this value if your database requires recovery when no recovery catalog exists and control file records are gone.
COMPLETION TIME	the date and time that the backup set completed. Note that the format of this field depends on the NLS_LANG and NLS_DATE_FORMAT environment settings.

Table 10–5 List of Backup Pieces

Column	Indicates
KEY	a unique identifier for this backup piece in the recovery catalog or target database control file. **Note:** The values for KEY in the recovery catalog and the control file are different.
PC#	the piece number of this backup piece within the backup set.
CP#	the copy number of this backup piece in a duplexed backup. For example, if **set duplex** = 4, then CP# will range from 1 to 4. **Note:** If the backup is not duplexed, then CP# = 1.
STATUS	the backup piece status: AVAILABLE, UNAVAILABLE, or EXPIRED (see the **change** command for an explanation of each status).
COMPLETION TIME	the date and time when the piece was created.
PIECE NAME	the name of the backup piece.

Table 10–6 Controlfile Included

Column	Indicates
CKP SCN	the SCN of the backup control file checkpoint. All database changes recorded in the redo records before the specified SCN are reflected in this control file.
CKP TIME	the time of the backup control file checkpoint. All database changes recorded in the redo records before the specified time are reflected in this control file.

Table 10–7 List of Datafiles Included

Column	Indicates
FILE	the number of the file that was backed up.
NAME	the location where this file would be restored now if it were restored from this backup set and no **set newname** command (see "set_run_option" on page 10-142) was entered.
LV	the level of the backup: NULL for non-incrementals, level 0-4 for incrementals.
TYPE	whether the backup was FULL or INCR (incremental).
CKP SCN	the checkpoint of the datafile at the time it was backed up. All database changes prior to the SCN have been written to the file; changes after the specified SCN have not been written to the file.
CKP TIME	the checkpoint of the datafile at the time it was backed up. All database changes prior to the time have been written to the file; changes after the specified time have not been written to the file.

Table 10–8 List of Datafile Copies **(Page 1 of 2)**

Column	Indicates
KEY	the unique identifier for the datafile copy. Use this value in a **change** command to alter the status of the datafile copy. **Note:** The values for KEY in the recovery catalog and the control file are different.
FILE	the file number of the original datafile from which this copy was made.
S	the backup piece status: AVAILABLE, UNAVAILABLE, or EXPIRED (see the **change** command for an explanation of each status).

Table 10–8 List of Datafile Copies **(Page 2 of 2)**

COMPLETION TIME	the date and time that the copy completed. Note that the value of this field is sensitive to the NLS_LANG and NLS_DATE_ FORMAT environment variables.
CKP SCN	the checkpoint of this datafile when it was copied. All database changes prior to the SCN have been written to this file.
CKP TIME	the checkpoint of this datafile when it was copied. All database changes prior to the time have been written to this file.
NAME	the filename of the datafile copy.

Table 10–9 List of Archived Log Copies

Column	Indicates
KEY	the unique identifier for this archived redo log copy. Use this value in a **change** command to alter the status of the copy. **Note:** The values for KEY in the recovery catalog and the control file are different.
THRD	the redo log thread number.
SEQ	the log sequence number.
COMPLETION TIME	the date and time that the copy completed. Note that the value of this field is sensitive to the NLS_LANG and NLS_DATE_ FORMAT environment variables.
NAME	the filename of the archived redo log copy.

Table 10–10 List of Database Incarnations **(Page 1 of 2)**

Column	Indicates
DB KEY	when combined with the INC KEY, the unique key by which RMAN identifies the database incarnation in the recovery catalog. Use this key to unregister a database, that is, delete all the rows associated with that database from the recovery catalog.
INC KEY	when combined with DB KEY, the unique key by which RMAN identifies the database incarnation in the recovery catalog. Use this key in the **reset database to incarnation** command, which you must use if you want to recover your database to a time prior to the most recent RESETLOGS.
DB NAME	the database name as listed in the DB_NAME parameter.

Table 10–10 List of Database Incarnations **(Page 2 of 2)**

DB ID	the database identification number, which Oracle generates automatically at database creation.
CUR	whether the incarnation is the current incarnation of the database.
RESET SCN	the SCN at which the incarnation was created.
RESET TIME	the time at which the incarnation was created.

Examples

Listing Copies The following example lists datafile copies and archived redo logs recorded in the recovery catalog:

```
list copy of database archivelog all;

List of Datafile Copies
Key     File S Completion time Ckp SCN    Ckp time    Name
------- ---- - --------------- ---------- ----------  ------
1262    1    A 18-AUG-98       219859     14-AUG-98   /vobs/oracle/dbs/copy/tbs_01.f

List of Archived Log Copies
Key     Thrd Seq     S Completion time Name
------- ---- ------- - --------------- ----------------------------------
789     1    1       A 14-JUL-98       /vobs/oracle/work/arc_dest/arcr_1_1.arc
790     1    2       A 11-AUG-98       /vobs/oracle/work/arc_dest/arcr_1_2.arc
791     1    3       A 12-AUG-98       /vobs/oracle/work/arc_dest/arcr_1_3.arc
```

Listing Backups The following example lists backups of two datafiles recorded in the recovery catalog:

```
list backup of datafile '/oracle/dbs/tbs_01.f', '/oracle/dbs/tbs_02.f';

List of Backup Sets
Key     Recid      Stamp      LV Set Stamp  Set Count  Completion Time
------- ---------- ---------- -- ---------- ---------- ---------------------
1174    12         341344528  0  341344502  16         14-SEP-98

List of Backup Pieces
Key     Pc# Cp# Status     Completion Time    Piece Name
------- --- --- ---------- ------------------ ---------------------------
1176    1   1   AVAILABLE  14-AUG-98          /vobs/oracle/dbs/0ga5h07m_1_1

Controlfile Included
Ckp SCN    Ckp time
---------- ---------------
219857     14-AUG-98
```

```
List of Datafiles Included
File Name                                  LV Type Ckp SCN    Ckp Time
---- ------------------------------------- -- ---- ---------- -------------
1    /oracle/dbs/tbs_01.f                  0  Full 199843     14-AUG-98
2    /oracle/dbs/tbs_02.f                  0  Full 199843     14-AUG-98
```

Listing Database Incarnations The following example lists all database incarnations recorded in the recovery catalog:

```
list incarnation;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         PROD1     1224038686  NO     1            02-JUL-98
1       582       PROD1     1224038686  YES    59727        10-JUL-98
```

Related Topics

"crosscheck" on page 10-64

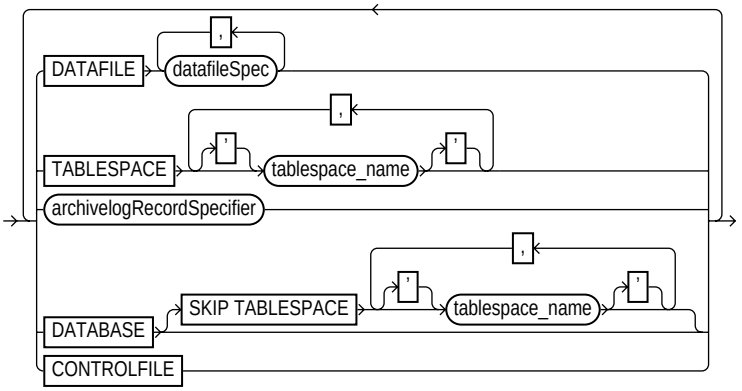
"listObjList" on page 10-92

"report" on page 10-110

"validate" on page 10-160

listObjList

Syntax



Purpose

A sub-clause used to specify database files and archived redo logs.

Requirements

Use this clause in the following commands:

- **list**
- **crosscheck**
- **deleteExpired**

Keywords and Parameters

datafile *datafileSpec*	specifies datafiles by filename for file number. The clause specifies datafile image copies or backup sets that contain at least one of the datafiles. See "datafileSpec" on page 10-66.
tablespace *tablespace_name*	specifies tablespace names. The clause specifies datafile image copies or backup sets that contain at least one of the datafile from the specified tablespace.
archivelogRecord-Specifier	specifies a range of archived redo logs. See "archivelogRecordSpecifier" on page 10-18.

database	specifies backup sets or image copies of all files in the current database. **skip tablespace** *tablespace_name* omits the specified tablespaces from the **database** specification.
controlfile	specifies the current control file.

Examples

Listing Datafile Copies The following command lists image copies of all the files in the database, skipping the TEMP tablespace:

```
list copy of database skip tablespace temp;
```

Crosschecking Archived Redo Logs The following example queries the media manager for the status of archived redo log backup sets created in the last 90 days:

```
allocate channel for maintenance type 'sbt_tape';
crosscheck
   backup
   of archivelog
   from time 'SYSDATE-90';
release channel;
```

Deleting Expired Control File Backup Sets The following command deletes expired backups of the control file:

```
delete expired backup of controlfile;
```

Related Topics

"archivelogRecordSpecifier" on page 10-18

"crosscheck" on page 10-64

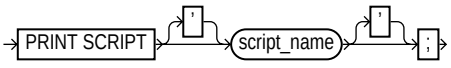
"datafileSpec" on page 10-66

"deleteExpired" on page 10-69

"list" on page 10-83

printScript

Syntax



Purpose

To print a stored script to standard output or the RMAN message log. Specify the log filename with the **log** argument at the command line. If you do not specify this argument, Recovery Manager writes message output to standard output.

> **Note:** You can also display the individual lines of your stored scripts by querying the RC_STORED_SCRIPT_LINE recovery catalog view.

Requirements

- Use this command only at the RMAN prompt.
- You must be using a recovery catalog.

Keywords and Parameters

script_name prints a stored script with the specified name to standard output or a message log. To obtain a listing of all stored scripts, use SQL*Plus connect to the recovery catalog as the catalog owner and issue the following query:

```
select * from rc_stored_script;
```

Note: To run the script, use **execute script** within the braces of the **run** command.

See Also: "RC_STORED_SCRIPT" on page 11-25 to learn about RC_STORED_SCRIPT.

Examples

Printing a Script to the Message Log This example connects to the target database PROD1 and the recovery catalog database RCAT, and directs the RMAN log output to a message log file. It then creates the `backup_db` script and prints it to `rman_log`. Finally, it executes the script:

```
rman target sys/change_on_install@prod1 catalog rman/rman@rcat log rman_log
create script backup_db {
       allocate channel d1 type disk;
       backup database;
}
print script backup_db;

run{ execute script backup_db;};
```

Printing a Script to the Screen This example prints a stored script to the screen:

```
print script tbs1_b;

RMAN-03027: printing stored script: tbs1_b
{
allocate channel ch1 type disk;
backup tablespace tbs1;
}
```

Related Topics

"cmdLine" on page 10-42

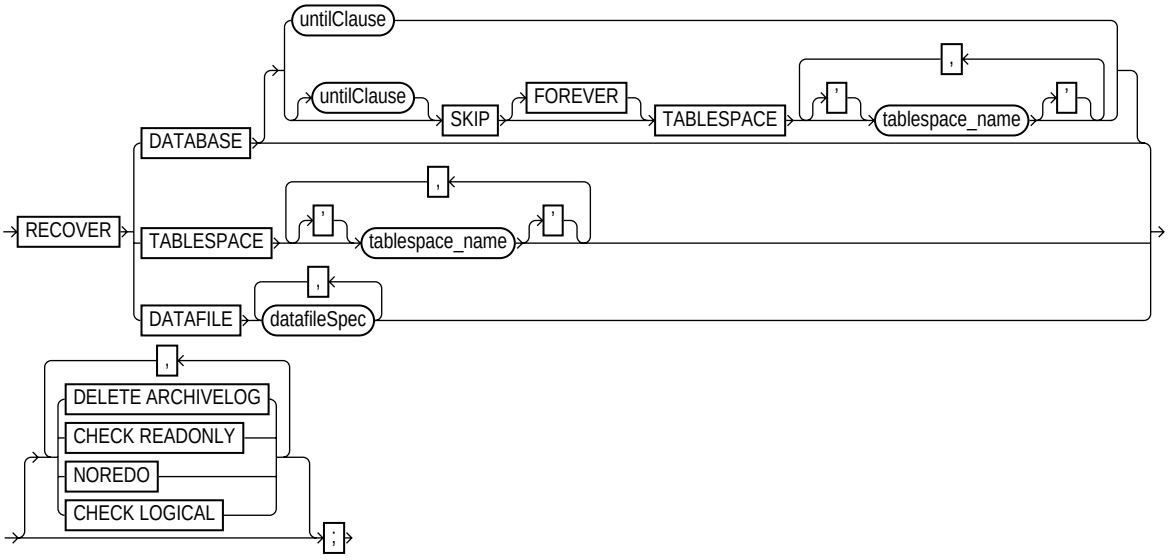
"createScript" on page 10-61

"deleteScript" on page 10-71

"run" on page 10-133

recover

Syntax



Purpose

To apply redo logs or incremental backups to one or more restored datafiles in order to update them to a specified time.

RMAN uses online redo records and restores backup sets of archived redo logs as needed to perform the media recovery. RMAN first looks for the original archived logs or image copies, and if none are available, restores backups.

If RMAN has a choice between applying an incremental backup or applying redo, then it always chooses the incremental backup. If overlapping levels of incremental backup are available, RMAN automatically chooses the one covering the longest period of time.

> **Note:** When RMAN applies incremental backups, it recovers changes to objects created with the NOLOGGING option. Applying archived redo logs to datafiles does not recover these changes.

See Also: Chapter 6, "Restoring and Recovering with Recovery Manager" to learn how to recover datafiles.

Requirements

- Execute this command only within the braces of a **run** command.
- You can use this command without a recovery catalog, but only if control file recovery is not required. RMAN cannot operate when neither the recovery catalog nor the target database control file are available.
- For datafile and tablespace recovery, the target database must be mounted. If it is open, then the datafiles or tablespaces to be recovered must be offline. For database recovery, the database must be mounted but not open.
- At least one **allocate channel** command must precede **recover** unless you do not need to restore archived redo log or incremental datafile backup sets.
- Allocate the appropriate type of device for the backups that you need to restore. If the appropriate type of device is not available, then the **recover** command aborts.
- Only the current datafiles may be recovered or have incremental backups applied to them.
- Typically, you should enter a **set until** command before both the **restore** and **recover** command. If you specify a **set until** command *after* a **restore** and *before* a **recover**, you may not be able to perform media recovery on the database to the time required because the restored files may have timestamps later than the specified time.
- The **recover database** command will not recover any files that are offline normal or read-only at the point in time to which the files are being recovered. RMAN omits offline normal files with no further checking. If **check readonly** is specified, then RMAN checks each read-only file on disk to ensure that it is already current at the desired point in time. If **check readonly** is not specified, then RMAN omits read-only files.
- Open with the RESETLOGS option after incomplete recovery or recovery with a backup control file.

Keywords and Parameters

database	specifies that the entire database is to be recovered. You can specify an optional *untilClause* that causes the recovery to stop when the specified condition has been reached.
untilClause	specifies a non-current time, SCN, or log sequence number for the **recover** command. See "untilClause" on page 10-156.
skip [**forever**] **tablespace** *tablespace_name*	lists tablespaces that should not be recovered, which is useful for avoiding recovery of tablespaces containing only temporary data or for postponing recovery of some tablespaces. The **skip** clause takes the datafiles in the specified tablespaces offline before starting media recovery. These files are left offline after the media recovery is complete. If you perform incomplete recovery, then **skip** is not allowed. Instead, use **skip forever**, with the intention of dropping the skipped tablespaces after opening the database with the RESETLOGS option. The **skip forever** clause causes RMAN to take the datafiles offline using the DROP option. Only use **skip forever** when the specified tablespaces will be dropped after opening the database.
tablespace *tablespace_name*	specifies tablespaces by tablespace name.
datafile *datafileSpec*	specifies a list of one or more datafiles to recover. Specify datafiles by filename using a quoted string or absolute datafile number using an *integer* (see "datafileSpec" on page 10-66). If you are using the control file as the exclusive repository for RMAN metadata, then the filename must be the name of the datafile as known in the control file. If you are using a recovery catalog, then the filename of the datafile must be the most recent name recorded in the catalog. For example, assume that a datafile was renamed in the control file. The database then crashes before you can resynchronize the catalog. Specify the old name of the datafile in the **recover** command, since this is the name recorded in the catalog.
delete archivelog	deletes restored archived logs that are no longer needed. RMAN does not delete archived logs that were already on disk before the **restore** command started.
check readonly	checks the headers of read-only files to ensure that they are current before omitting them from the recovery.
noredo	suppresses the application of redo logs—only incremental backups are applied. This option is intended for recovery of NOARCHIVELOG databases using incremental backups. If you do not specify **noredo** when recovering a NOARCHIVELOG database, Oracle aborts a recovery and issues an error.

Examples

Recovering a Tablespace in an Open Database The following example takes tablespace TBS_1 offline, restores and recovers it, then brings it back online:

```
run {
     allocate channel dev1 type 'sbt_tape';
     sql "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE";
     restore tablespace tbs_1;
     recover tablespace tbs_1;
     sql "ALTER TABLESPACE tbs_1 ONLINE";
}
```

Recovering Datafiles Restored to New Locations The following example allocates one disk channel and one media management channel to use datafile copies on disk and backups on tape, and restores one of the datafiles in tablespace TBS_1 to a different location:

```
run {
     allocate channel dev1 type disk;
     allocate channel dev2 type 'sbt_tape';
     sql "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE";
     set newname for datafile 'disk7/oracle/tbs11.f'
       to 'disk9/oracle/tbs11.f';
     restore tablespace tbs_1;
     switch datafile all;
     recover tablespace tbs_1;
     sql "ALTER TABLESPACE tbs_1 ONLINE";
}
```

Performing Incomplete Recovery Using a Backup Control File Assume that both the database and archived redo log 1234 were lost due to a disk crash. Because you do not have incremental backups, you need to recover the database using available archived redo logs. There is no need to restore tablespace READONLY1 because it has not changed since log 1234.

```
run {
     # Recover database until log sequence 1234
     allocate channel dev1 type disk;
     allocate channel dev2 type 'sbt_tape';
     set until logseq 1234 thread 1;
     restore controlfile to '/vobs/oracle/dbs/cf1.f' ;
     # Because you specified a restore destination, you must manually replicate the
     # control file. The restore command replicates automatically when no destination is
     # specified.
     replicate controlfile from '/vobs/oracle/dbs/cf1.f';
     alter database mount;
```

```
    restore database skip tablespace temp1, readonly1;
    recover database skip forever tablespace temp1;
    sql "ALTER DATABASE OPEN RESETLOGS";
    sql "DROP TABLESPACE temp1";
    sql "CREATE TABLESPACE temp1 DATAFILE '/vobs/oracle/dbs/temp1.f' SIZE 10M";
    release channel dev1;
    release channel dev2;
}
```

Related Topics

"allocate" on page 10-10

"set_run_option" on page 10-142

"restore" on page 10-120

"untilClause" on page 10-156

register

Syntax

Purpose

To register the target database in the recovery catalog so that RMAN can access it. RMAN obtains all information it needs to register the target database from the target database itself.

Note: If you perform a RESETLOGS operation on a database and later register it in the recovery catalog, the catalog records the DB_NAME for the old incarnations as UNKNOWN because the old incarnations were not previously registered. You should not try to remove these records.

See Also: "Registering a Database with the Recovery Catalog" on page 3-9.

Requirements

- Execute this command only at the RMAN prompt.
- You can only register a database once in a given recovery catalog.
- You must be using a recovery catalog.
- The target database must be mounted.
- The **register database** command fails when RMAN detects duplicate database identifiers. This situation can arise when databases are created by copying files from an existing database rather than using the **duplicate** command.

  If this failure occurs, you can create a second recovery catalog in another user's schema by executing **create catalog** (see "createCatalog" on page 10-59) using a different user id. Then, register the database with the duplicate database identifier into the newly created recovery catalog in the new schema.

Note: If you are using RMAN with different target databases that have the same database name and identifier, be extremely careful to always specify the correct recovery catalog schema when invoking Recovery Manager.

Examples

Registering a Database This example registers the target database, catalogs an existing datafile copy, then opens the database for use:

```
connect target / catalog rman/rman@rcat;
startup mount;
register database;
catalog datafilecopy '/vobs/oracle/dbs/foo.f';
sql 'alter database open';
```

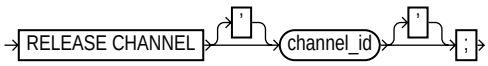
Related Topics

"catalog" on page 10-35

"duplicate" on page 10-76

release

Syntax



Purpose

To release a channel while maintaining the connection to the target database instance. Specify the channel name with the same identifier used in the **allocate channel** command. This command is optional; RMAN automatically releases all channels allocated when the **run** command terminates.

Requirements

Execute this command only within the braces of a **run** command.

Keywords and Parameters

channel_id	specifies the channel id used in the **allocate channel** command.

Examples

Releasing a Channel This example makes three identical backup sets of `datafile 1` to tape, then releases the tape channel. RMAN then makes three identical backups of `datafile 2` to disk and then releases the disk channel:

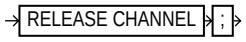
```
run {
     set duplex=3;
     allocate channel ch1 type 'SBT_TAPE';
     allocate channel ch2 type disk;
     backup channel ch1 datafile 1;
     release channel ch1;
     backup datafile 2;
}
```

Related Topics

"allocate" on page 10-10

releaseForMaint

Syntax



Purpose

To release a sequential I/O device specified in an **allocate channel** command with the **for delete** or **for maintenance** options. Note that maintenance channels are unaffected by **allocate channel** and **release channel** command issued within a **run** command.

Requirements

- Execute this command only at the RMAN prompt.
- You must have a maintenance channel allocated in order to release it.

Examples

Releasing a Maintenance Channel After a Delete Operation This example allocates and then releases a maintenance channel to the media manager:

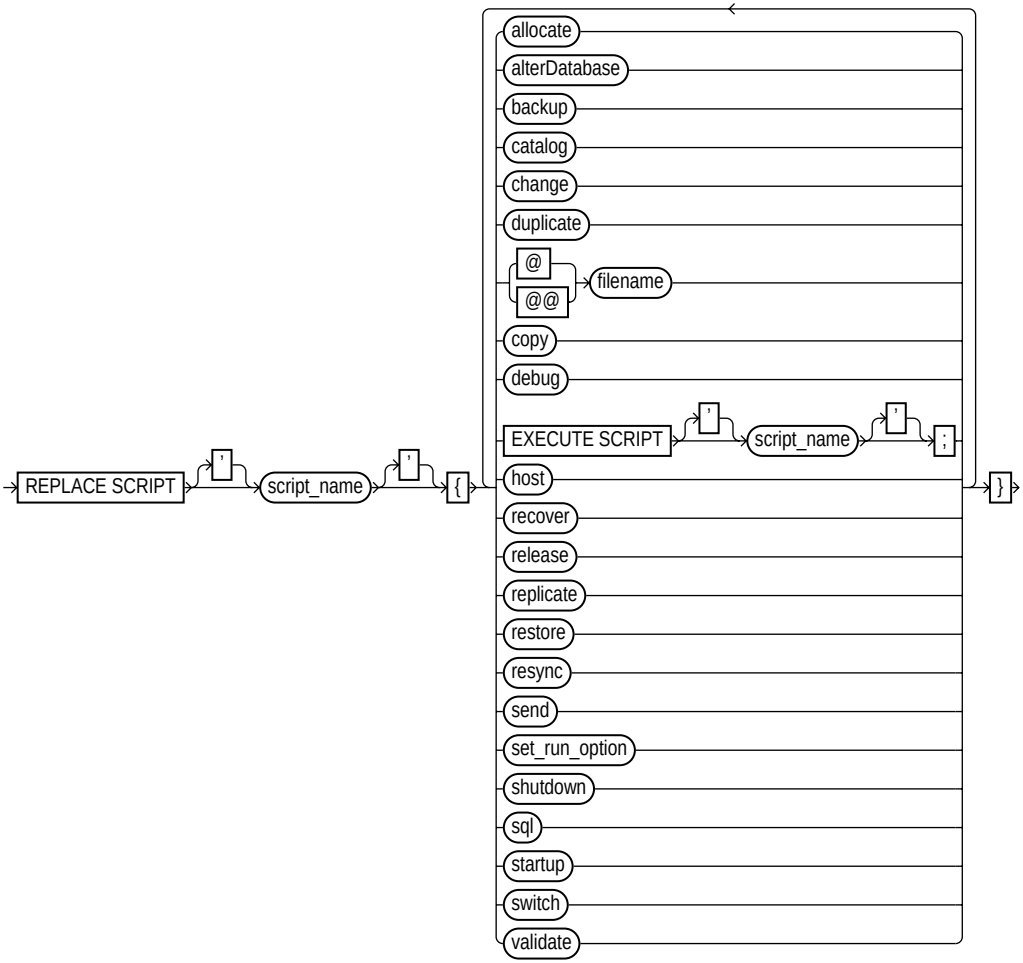
```
allocate channel for delete type 'sbt_tape';
change backuppiece 100 delete;
run {
     allocate channel ch1 type disk;
     backup datafile 1;
     release channel ch1; # releases run channel
}
release channel; # releases maintenance channel
```

Related Topics

"allocateForMaint" on page 10-14

replaceScript

Syntax



Purpose

To replace an existing script stored in the recovery catalog. If the script does not exist, **replace script** creates it.

The stored script feature is provided primarily to provide a common repository for frequently executed collections of RMAN commands: use any command legal within a **run** command in the script. The script is not executed immediately; use the **execute script** command (see "run" on page 10-133) to run it.

See Also:

- For descriptions of the individual commands that you can use in a stored script, see the appropriate entry, for example, "backup" on page 10-22.
- For information about the **@** and **@@** arguments, see "createScript" on page 10-61.
- For information about the **execute script** command, see "run" on page 10-133.

Requirements

- Execute **replace script** only at the RMAN prompt.
- You must be using a recovery catalog.

Keywords and Parameters

replace script *script_name*	replaces the specified stored script with the new commands. The statements allowable within the parentheses of the **replace script** '*filename*' **(...)** command are the same allowable within the **run** command. To obtain a listing of all stored scripts, use SQL*Plus to connect to the recovery catalog database as the catalog owner and issue the following query: `select * from rc_stored_script;` **Note:** To run the script, issue **execute script** within the braces of the **run** command. **See Also:** "RC_STORED_SCRIPT" on page 11-25 for more information about RC_STORED_SCRIPT.

Examples

Replacing a Script This example creates a script called `backup_full`, replaces it with a different script, and then executes it:

```
create script backup_full {
   allocate channel ch1 type 'SBT_TAPE';
   allocate channel ch2 type 'SBT_TAPE';
   allocate channel ch3 type 'SBT_TAPE';
   backup database;
}
replace script backup_full {
   allocate channel ch1 type disk;
   backup database;
}
run { execute script backup_full; }
```

Related Topics

"createScript" on page 10-61

"deleteScript" on page 10-71

"printScript" on page 10-94

"run" on page 10-133

replicate

Syntax

Purpose

To copy a control file to the locations specified in the CONTROL_FILES initialization parameter of the target database.

After restoring the control file, you can use the **replicate controlfile** statement to prepare the database for mounting. This operation is equivalent to executing multiple **copy controlfile** statements.

> **Note:** The **restore** command will automatically replicate the control file to all CONTROL_FILES locations if no restore destination is specified.

Requirements

- Execute **replicate controlfile** only within the braces of a **run** command.
- At least one **allocate channel** statement specifying the **type disk** option must precede a **replicate controlfile** statement.

Keywords and Parameters

'filename'	specifies the location of the control file to be replicated. For example, if you restore a control file backup to `/oracle/temp/cf.bak`, then you would also specify this filename in the **replicate** command.

Examples

Replicating a Restored Control File This example restores a control file and then replicates it:

```
startup nomount;
run {
     set until time 'Jun 18 1998 16:32:36';
     allocate channel ch1 type disk;
     # restore a backup controlfile to a temporary location.
     restore controlfile to '/tmp/cf.tmp';
     replicate controlfile from '/tmp/cf.tmp';
     startup force mount;
}
```

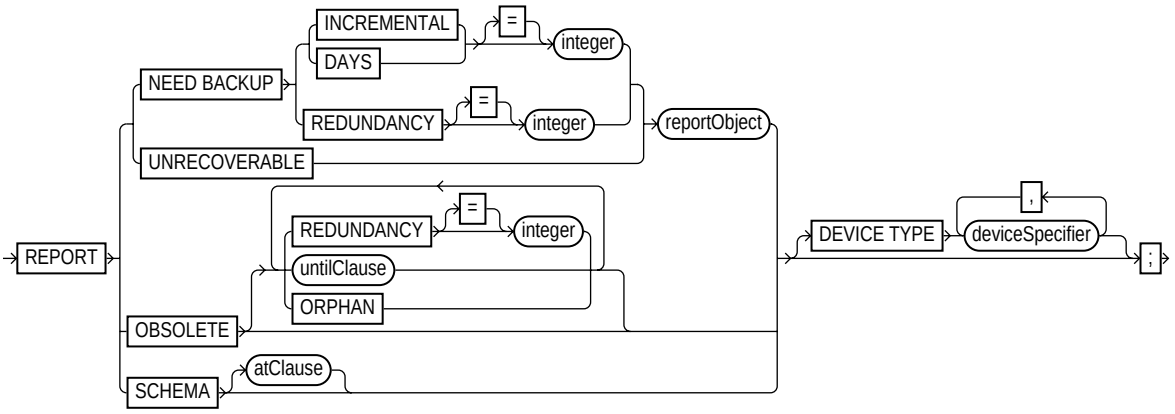
Related Topics

"copy" on page 10-55

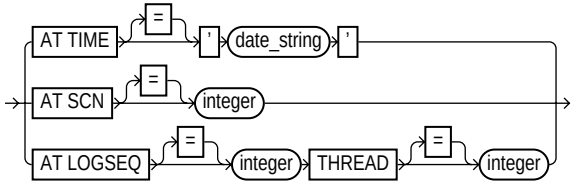
"restore" on page 10-120

report

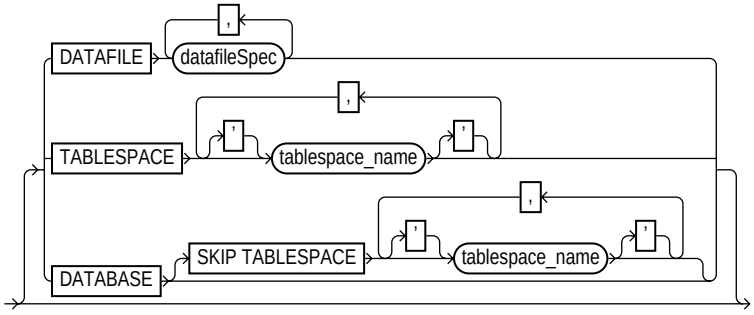
Syntax



atClause::=



reportObject::=



Purpose

To perform detailed analyses of the RMAN repository. Oracle writes the output from the **report** command to standard output or the message log file (see "connect" on page 10-51).

Use the **report** command to answer questions such as the following:

- Which files need a backup?
- Which files have not had a backup in a while?
- Which files are not recoverable due to unrecoverable operations?
- Which backup files can be deleted?
- What was the physical schema of the database at a previous time?

> **See Also:** "Generating Reports" on page 4-5.

Requirements

- Execute this command only at the RMAN prompt.
- You must use a recovery catalog when issuing a **report schema** command with the **at time**, **at scn**, or **at logseq** options. Otherwise, a recovery catalog is not required for the **report** command.

Keywords and Parameters

need backup lists all datafiles in need of a new backup. The report assumes that you will use the most recent backup for restore operations.

incremental *integer* specifies a threshold number of incremental backups. If complete recovery of a datafile requires more than the specified number of incremental backups, then the datafile requires a new full backup. The **report** command, like the **recover** command, uses the lowest level of incremental backup whenever there is a choice. This is the same strategy that RMAN would use if the file were actually being recovered by the **recover** command.

Note: Files for which no backups exist will not appear in this list: issue the **report need backup redundancy** command to display them.

days *integer* specifies a threshold number of days of redo log files that need application during recovery of this file. For example, **report need backup days 7 database** shows the datafiles whose recovery requires more than one week's worth of archived redo logs.

If the target database control file is mounted and current, RMAN makes the following optimizations to this report:

- Files that are offline and whose most recent backup contains all changes to the file will not be included.
- Files that were offline and are now online, and whose most recent backup contains all changes up to the offline time, will only be reported if they have been online for more than the specified number of days.

redundancy *integer* specifies the minimum number of backups or copies that must exist for a datafile to be considered *not* in need of a backup. In other words, a datafile needs a backup if there are fewer than *integer* backups or copies of this file. For example, **redundancy** *2* means that if there are fewer than two copies or backups of a datafile, then it needs a new backup.

unrecoverable lists all unrecoverable datafiles. A datafile is considered unrecoverable if an unrecoverable operation has been performed against an object residing in the datafile since the last backup of the datafile.

Note: The non-existence of any backup of a datafile is not sufficient reason to consider it unrecoverable. Such datafiles can be recovered through the use of the CREATE DATAFILE command, provided that redo logs starting from when the file was created still exist.

reportObject clause specifies the datafiles to be included in the report. The report can include the entire database (optionally skipping certain tablespaces), a list of tablespaces, or a list of datafiles.

datafile *datafileSpec* lists the specified datafiles. RMAN reports on backups or datafile copies that contain at least one of the specified datafiles.

tablespace *tablespace_name* — lists datafiles in the specified tablespace. RMAN reports on backups or datafile copies that include at least one datafile from a specified tablespace.

database — lists backups or datafile copies of all files in the current database.

skip tablespace *tablespace_name* — excludes the specified tablespaces from the **database** specification.

obsolete — lists full backups and datafile copies recorded in the RMAN repository that can be deleted because they are no longer needed. If you do not specify further parameters, **redundancy** defaults to 1. If you use this option in conjunction with **device type**, RMAN only considers backups and copies on the specified type.

redundancy *integer* — specifies the minimum level of redundancy considered necessary for a backup or copy to be obsolete. A datafile copy is obsolete if there are at least *integer* more recent backups or image copies of this file; a datafile backup set is obsolete if there are at least *integer* more recent backups or image copies of each file contained in the backup set. For example, **redundancy** *2* means that there must be at least two more recent backups or copies of a datafile for any other backup or copy to be obsolete.

untilClause — specifies that no backup or copy will be considered obsolete if there are at least *n* (where *n* is the value for **redundancy**) backups or copies that are more recent but do not contain changes later than the specified time, SCN, or log sequence number. For example, **obsolete redundancy** *2* **until** *'SYSDATE-7'* means that a backup or copy is obsolete if there are at least two backups or copies that are more recent and those copies were checkpointed more than a week ago.

This clause is useful if the database must be recoverable to non-current time, SCN, or log sequence number. See "untilClause" on page 10-156.

orphan — specifies as obsolete those backups and copies that are unusable because they belong to incarnations of the database that are not direct ancestors of the current incarnation. Note that **report obsolete orphan** displays orphaned backups *in addition to* the normal display of obsolete backups. For an explanation of orphaned backups, see "Reporting on Orphaned Backups" on page 1-27.

schema — lists the names of all datafiles and tablespaces at the specified point in time.

atClause — specifies a point in time as a time, an SCN, or a log sequence number.

at time *date_string* — specifies a date. The NLS_LANG and NLS_DATE_FORMAT environment variables specify the format for the time.

at scn *integer* — specifies an SCN.

at logseq *integer* — specifies a log sequence number for a specified redo thread. The integer indicates the time when the specified log and thread were first opened.

device type *deviceSpecifier*	specifies the type of storage device. RMAN only considers backups and copies available on the specified device for its report. See "deviceSpecifier" on page 10-72.

Report Output

The fields in each report are described below:

Table 10–11 Report of Database Schema

Column	Indicates
FILE	the absolute datafile number.
K-BYTES	the size of the file in kilobytes.
TABLESPACE	the tablespace name.
RB SEGS	YES if rollback segments exist in the tablespace and NO if they do not (only if connected to the recovery catalog). If RMAN is not connected to the catalog, then '***' is displayed.
NAME	the filename of the datafile.

Table 10–12 Report of Obsolete Backups and Copies

Column	Indicates
TYPE	whether the object is a backup set, backup piece, proxy copy, or datafile copy.
KEY	a unique key that identifies this backup in the target database control file.
COMPLETION TIME	the time that the backup or copy completed.
FILENAME/HANDLE	the filename or media handle of the backup or datafile copy.

Table 10–13 Report of Files that Need Backup Due to Unrecoverable Operations

Column	Indicates
FILE	the absolute number of the datafile that needs a new backup due to unrecoverable operations.
TYPE OF BACKUP REQUIRED	FULL or INCREMENTAL, depending on which type of backup is necessary to ensure the recoverability of all of the data in this file. If FULL, then create a full backup, level-0 backup, or a datafile copy. If INCREMENTAL, then a full or incremental backup will also suffice.
NAME	the name of the datafile.

Table 10–14 Report of Files with Less Than n Redundant Backups

Column	Indicates
FILE	the absolute datafile number of a datafile with less than *n* redundant backups.
#BKPS	the number of backups that exist for this file.
NAME	the name of the file.

Table 10–15 Report of Files Whose Recovery Needs More Than n Days of Archived Logs

Column	Indicates
FILE	the absolute file number of a datafile that requires more than *n* days of archived redo logs for recovery.
DAYS	the number of days of archived redo data required for recovery.
NAME	the name of the datafile.

Table 10–16 Report of Files That Need More than n Incrementals During Recovery

Column	Indicates
FILE	the absolute file number of a datafile that requires more than *n* incrementals for complete recovery.
INCREMENTALS	the number of incremental backups required for complete recovery.
NAME	the name of the datafile.

Examples

Reporting Database Schema This example reports the names of all datafiles and tablespaces in the database one week ago:

```
report schema at time 'SYSDATE-7';

Report of database schema
File K-bytes    Tablespace           RB segs Name
---- ---------- -------------------- ------- ------------------
1         47104 SYSTEM               YES     /vobs/oracle/dbs/tbs_01.f
2           978 SYSTEM               YES     /vobs/oracle/dbs/tbs_02.f
3           978 TBS_1                NO      /vobs/oracle/dbs/tbs_11.f
4           978 TBS_1                NO      /vobs/oracle/dbs/tbs_12.f
5           978 TBS_2                NO      /vobs/oracle/dbs/tbs_21.f
6           978 TBS_2                NO      /vobs/oracle/dbs/tbs_22.f
7           500 TBS_3                NO      /vobs/oracle/dbs/tbs_31.f
8           500 TBS_3                NO      /vobs/oracle/dbs/tbs_32.f
9          5120 SYSTEM               YES     /vobs/oracle/dbs/tbs_03.f
```

Reporting Datafiles Needing Incremental Backups This example reports all datafiles in the database that require the application of five or more incremental backups to be recovered to their current state:

```
report need backup incremental 5 database;

Report of files that need more than 5 incrementals during recovery
File Incrementals Name
---- ------------ -----------------------------------------------
1    9            /vobs/oracle/dbs/tbs_01.f
2    9            /vobs/oracle/dbs/tbs_02.f
3    9            /vobs/oracle/dbs/tbs_11.f
4    9            /vobs/oracle/dbs/tbs_12.f
5    9            /vobs/oracle/dbs/tbs_21.f
6    9            /vobs/oracle/dbs/tbs_22.f
7    9            /vobs/oracle/dbs/tbs_23.f
8    9            /vobs/oracle/dbs/tbs_03.f
```

Reporting Datafiles Needing Backups The following example reports all datafiles from tablespace SYSTEM that will need more than two days of archived redo logs to be applied during recovery after being restored from the most recent backup:

```
report need backup days 2 tablespace system;

Report of files whose recovery needs more than 2 days of archived logs
File Days  Name
---- ----- -----------------------------------------------------
1    3     /vobs/oracle/dbs/tbs_01.f
```

```
2    3     /vobs/oracle/dbs/tbs_02.f
16   3     /vobs/oracle/dbs/tbs_03.f
```

Reporting Unrecoverable Datafiles The following example reports all datafiles that cannot be recovered from existing backups because redo may be missing:

```
report unrecoverable;

Report of files that need backup due to unrecoverable operations
File Type of Backup Required Name
---- ----------------------- -----------------------------------
4     FULL                   /vobs/oracle/dbs/tbs_12.f
```

Reporting Obsolete Backups and Copies The following example reports obsolete backups and copies with a redundancy of 1:

```
report obsolete;

Report of obsolete backups and copies
Type                 Key    Completion Time    Filename/Handle
-------------------- ------ ------------------ --------------------
Backup Set           836    04-DEC-98
Backup Piece         839    04-DEC-98          /vobs/oracle/dbs/05aetj6b_1_1
Backup Set           807    04-DEC-98
Backup Piece         810    04-DEC-98          /vobs/oracle/dbs/03aetj1f_1_1
Backup Set           835    04-DEC-98
Backup Piece         838    04-DEC-98          /vobs/oracle/dbs/04aetj6b_1_1
```

Related Topics

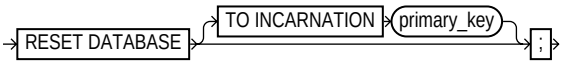
"list" on page 10-83

"untilClause" on page 10-156

"validate" on page 10-160

reset

Syntax



Purpose

To create a new database incarnation record in the recovery catalog. RMAN considers the new incarnation as the current incarnation of the database. All subsequent backups and redo log archiving operations performed by the target database will be associated with the new database incarnation.

Requirements

- Execute **reset database** only at the RMAN prompt.
- You must be using a recovery catalog.
- You must issue a **reset database** command before you can use RMAN with a target database that has been opened with the RESETLOGS option. If you do not, then RMAN refuses to access the recovery catalog because it cannot distinguish between a RESETLOGS operation and an accidental restore of an old control file. The **reset database** command gives confirmation to RMAN that you issued a RESETLOGS command.

Keywords and Parameters

to incarnation *primary_key*	changes the incarnation that RMAN considers to be current to an older incarnation. This option is useful in the rare circumstance in which you want to undo the effects of a RESETLOGS by restoring backups of a prior incarnation of the database. Specify the primary key of the DBINC record for the database incarnation. Obtain the key value using the **list incarnation of database** command. After you issue the **reset database to incarnation** command, issue **restore** and **recover** commands to restore the database files from the prior incarnation and recover them.

Examples

Resetting a Database After RESETLOGS The following example resets a database after performing incomplete media recovery:

```
run {
```

```
     allocate channel dev1 type disk;
     set until logseq 1234 thread 1;
     restore database skip tablespace readonly;
     recover database;
     sql "ALTER DATABASE OPEN RESETLOGS";
     release channel dev1;
}
reset database;
```

Resetting an Old Incarnation The following command makes an old incarnation of database PROD1 current again:

```
# obtain primary key of old incarnation
list incarnation of database prod1;

List of Database Incarnations
DB Key  Inc Key   DB Name   DB ID       CUR    Reset SCN    Reset Time
------- -------   -------   ------      ---    ----------   ----------
1       2         PROD1     1224038686  NO     1            02-JUL-98
1       582       PROD1     1224038686  YES    59727        10-JUL-98

shutdown immediate;
# reset database to old incarnation
reset database to incarnation 2;
# recover it
run {
     allocate channel dev1 type disk;
     restore controlfile;
     startup mount;
     restore database;
     recover database;
     sql "ALTER DATABASE OPEN RESETLOGS";
     release channel dev1;
}
```

Related Topics

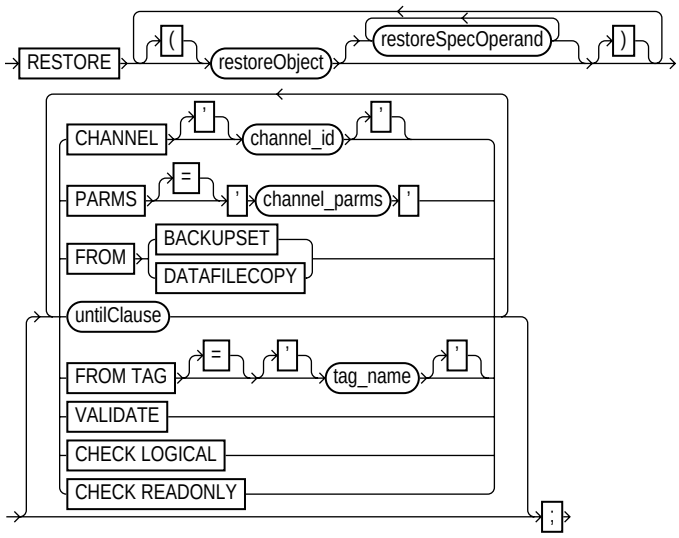
"list" on page 10-83

"restore" on page 10-120

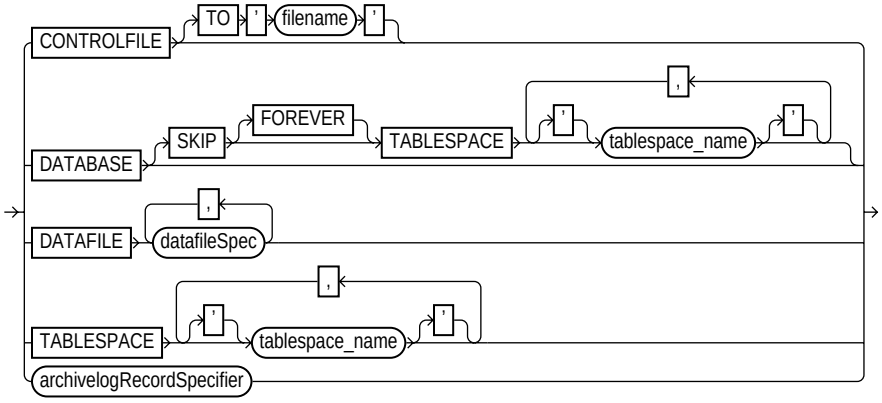
"recover" on page 10-96

restore

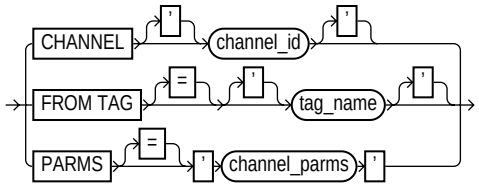
Syntax



restoreObject::=



restoreSpecOperand::=



Purpose

To restore files from backups or image copies to the current location, overwriting the files with the same name. You can use **set newname** command to restore files to non-default locations. RMAN restores backups from disk or tape and restores images copies from disk only.

RMAN considers restored files as datafile copies. If you restore to the default location, RMAN creates records for the datafile copies in the repository and immediately updates them to status DELETED. If you restore to a new location, issue **set newname** commands to rename the files, and issue a **switch** command to make the restored files current, RMAN also updates the records to DELETED. If you do not issue **switch** commands, then RMAN considers the restored files as valid copies for use in future restore operations.

Typically, you restore when a media failure has damaged a current datafile, control file, or archived redo log or prior to performing a point-in-time recovery. This command restores full backups, incremental backups (level 0 only), or copies of:

- Database

- Tablespaces
- Datafiles
- Control files
- Archived redo logs

> **Note:** Because the **recover** command automatically restores archived redo logs as needed, you should seldom need to restore archived logs with the **restore** command. Possible reasons for manually restoring archived redo logs are to speed up recovery or stage the logs to multiple destinations.

Note that when you perform a restore operation using a backup control file and use a recovery catalog, RMAN automatically adjusts the control file to reflect the structure of the restored backup.

> **See Also:** Chapter 6, "Restoring and Recovering with Recovery Manager" to learn how to restore files.

Requirements

- Execute **restore** only within the braces of a **run** command.
- To restore datafiles to their current location, the database must be either:
  - Mounted
  - Open with the datafiles offline

  If the entire database is to be restored, then it must be mounted.
- To restore to a new location, use the **set newname** commands to rename the datafiles and use **switch** commands to make the restored files into the current database files.
- At least one **allocate channel** command must precede a **restore**.
- If you use the **from datafilecopy** option, then the allocated channels must be of **type disk**.
- If you use the **from backupset** operand, then the appropriate type of storage devices must be allocated for the backup sets that need to be restored. If the appropriate device is not allocated, then you may not be able to find a candidate backup set or copy to restore, and the **restore** command fails.

- RMAN only restores backups that were created on the same type of channels that are allocated for the **restore** command.

  For example, if you made some backups of a datafile to **disk** channels and others to *'sbt_tape'* channels, and only a **disk** channel is allocated for the **restore** command, then RMAN will not restore from any backups that were created on *'sbt_tape'* channels.

- If datafile filenames are symbolic links, that is, files that point to other files, then the control file contains the filenames of the link files but RMAN performs I/O on the datafiles pointed to by the link files. If a link file is lost and you **restore** a datafile without first re-creating the symbolic link, however, then RMAN restores the datafile to the location of the link file rather than to the location pointed to by the link.
- Open the database with the RESETLOGS option after restoring with a backup control file.
- You can only restore from a previous incarnation when restoring the whole database. For example, you cannot restore one datafile of a previous incarnation while the current database is in a different incarnation.
- Do not specify a datafile more than once in a restore job. For example, the following command is illegal because `datafile 1` is both specified explicitly and implied by the SYSTEM tablespace:

  ```
  restore
    tablespace system
    datafile 1;
  ```

- If you restore a pre-8.1.6 control file on Windows NT that has not been normalized, you must normalize it before mounting the database by following the procedure described in *Oracle8i Migration*. A flawed mechanism in releases prior to release 8.1.6 on Windows NT could allow two different filenames to refer to the same physical file.

Keywords and Parameters

restoreObject specifies the objects to be restored.

controlfile restores the current control file and automatically replicates it to all CONTROL_FILES locations in the parameter file. If you specify a new pathname with the **to** *'filename'* option, RMAN restores the control file to the new location: you must replicate it manually using the **replicate** command.

	database	restores all datafiles in the database except those that are offline or read-only. Unlike **backup database**, **restore database** does *not* automatically include the control file—you must issue an additional **restore** command to perform this operation. If you specify the **check readonly** option, then RMAN examines the headers of all read-only files and restores any that need restoring. Use an optional **skip ... tablespace** argument to avoid restoring specified tablespaces, which is useful when you want to avoid restoring tablespaces containing temporary data.
	datafile *datafileSpec*	restores the datafiles specified by filename or absolute datafile number. See "datafileSpec" on page 10-66.
	tablespace *tablespace_name*	restores all datafiles in the specified tablespaces.
	archivelogRecord-Specifier clause	restores the specified range of archived redo logs. See "archivelogRecordSpecifier" on page 10-18.
restoreSpec-Operand	specifies options for the *restoreObject* clause. **Note:** These parameters override the parameters with the same name at the **restore** command level.	
	channel *channel_id*	specifies the name of a channel to use for this restore operation. If you do not specify a channel, **restore** uses any available channel allocated with the correct device type.
	from tag *tag_name*	overrides the default selection of the most recent backups or file copy available. The tag restricts the automatic selection to backup sets or file copies that have the specified tag. If multiple backup sets or file copies have a matching tag, then RMAN selects the most recent one.
	parms *channel_parms*	specifies a quoted string containing operating system-specific information. The string is passed to the OSD layer each time a backup piece is restored.
channel *channel_id*	See the *restoreSpecOperand* clause.	
from tag *tag_name*	See the *restoreSpecOperand* clause.	
parms *channel_parms*	See the *restoreSpecOperand* clause.	
from	specifies whether RMAN should restore from a **datafilecopy** on disk or a **backupset**. By default **restore** chooses the most recent backup set or file copy, that is, the file copy or backup set that needs the least media recovery.	
untilClause	limits the selection to those backup sets or file copies that would be suitable for performing a point-in-time recovery. In the absence of any other criteria, RMAN selects the most current file copy or backup set to restore. See "untilClause" on page 10-156.	

validate	causes RMAN to decide which backup sets, datafile copies, and archived logs need to be restored and then scans them to verify their contents. This operation creates no output files. Specify this option periodically to verify that the copies and backup sets required to restore the specified files are intact and usable.
check logical	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 logs the block in the `alert.log` and server session trace file. Provided the sum of physical and logical corruptions detected for a file remain 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:** The **maxcorrupt** setting represents the total number of physical and logical corruptions permitted on a file.
check readonly	checks the datafiles to make sure they exist, are readable, and have the appropriate checkpoint. If any of these conditions is not met, then RMAN restores the files—whether or not they are read-only. By default, RMAN does not restore read-only files when you issue the **restore database** command.

Examples

Restoring a Tablespace This example takes a tablespace offline, restores it, then performs media recovery:

```
run {
     # recover tablespace tbs_1 while the database is open
     allocate channel ch1 type 'sbt_tape';
     sql "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE" ;
     restore tablespace tbs_1 ;
     recover tablespace tbs_1 ;
     sql "ALTER TABLESPACE tbs_1 ONLINE" ;
     release channel ch1 ;
}
```

Restoring the Control File This example restores the control file to its default location, replicates it to all CONTROL_FILES locations, and mounts the database:

```
startup nomount;
run {
     allocate channel ch1 type 'sbt_tape';
     restore controlfile;
     alter database mount;
}
```

Restoring the Control File Using a Tag This example restores the control file specified by a tag, replicates it to all CONTROL_FILES locations, and then mounts the database:

```
startup nomount;
run {
     allocate channel ch1 type 'sbt_tape';
     restore controlfile from tag 'monday_cf_backup';
     alter database mount;
}
```

Restoring the Database Using a Backup Control File This example restores the control file to a new location, replicates it to all control file locations specified in the parameter file, and then mounts the control file in order to restore the database:

```
startup nomount;
run {
     allocate channel ch1 type 'sbt_tape';
     restore controlfile to '/oracle/dbs/cf1.ctl';
     replicate controlfile from '/oracle/dbs/cf1.ctl';
     alter database mount;
     restore database;
}
```

Restoring Archived Redo Logs to a New Location This example restores all archived redo logs to the `/oracle/temp_restore` directory:

```
run {
      set archivelog destination to '/oracle/temp_restore';
      allocate channel ch1 type disk;
      restore archivelog all;
}
```

Related Topics

"allocate" on page 10-10

"recover" on page 10-96

"untilClause" on page 10-156

resync

Syntax

Purpose

To perform a full *resynchronization* of the recovery catalog. Resynchronizations can be *full* or *partial*.

In a full resynchronization, RMAN updates all changed records for the *physical schema*: datafiles, tablespaces, redo threads, and online redo logs. If the database is open, RMAN also obtains information about rollback segments.

In a partial resynchronization, RMAN reads the current control file to update changed information, but does not resynchronize metadata about the physical schema or rollback segments.

When resynchronizing, RMAN creates a snapshot control file in order to obtain a read-consistent view of the control file, then updates the catalog with any new information from the snapshot.

The **resync catalog** command updates these classes of records:

Record Type	Description
Log history	records that are created whenever a redo log switch occurs. Note that log history records describe an online log switch, not a log archival.
Archived redo logs	Records associated with archived logs that were created by archiving an online redo log, copying an existing archived redo log, or restoring backups of archived redo logs.
Backups	Records associated with backup sets, backup pieces, backup set members, proxy copies, and image copies.
Physical schema	Records associated with datafiles and tablespaces. If the target database is open, then rollback segment information is also updated.

The following commands update the recovery catalog automatically when the target database control file is mounted and the recovery catalog database is available at command execution:

- **backup**
- **change**
- **copy**
- **crosscheck**
- **deleteExpired**
- **duplicate**
- **restore**
- **switch**
- **recover**
- **list**
- **report**

When you run these commands, RMAN automatically executes a full or partial resynchronization as needed. RMAN reads the current control file and does not resynchronize metadata about physical schema unless it determines that this information has changed. If RMAN does detect a change, it performs a full resynchronization.

Use **resync catalog** to perform manual full resynchronizations when:

- The recovery catalog is unavailable when you issue any of the commands that automatically perform a resynchronization.
- You are running in ARCHIVELOG mode, since the recovery catalog is *not* updated automatically when a log switch occurs or when an online redo log is archived.
- You have made changes to the physical structure of the target database such as adding or dropping a tablespace. As with archive operations, the recovery catalog is *not* updated automatically when the physical schema changes.

Requirements

- Execute **resync catalog** at the RMAN prompt or within the braces of a **run** command.
- You must be using a recovery catalog.
- RMAN updates physical schema information in the recovery catalog only when the target database 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 RMAN does not update physical schema information in the recovery catalog.

Keywords and Parameters

from controlfilecopy *'filename'*	specifies the name of the control file copy to use for reynchronization. Physical schema information is not updated when you use this option.

Examples

Resynchronizing After a Structural Change This example adds datafile `sales.f` to tablespace TBS_1 and then resynchronizes the recovery catalog to reflect the physical database change:

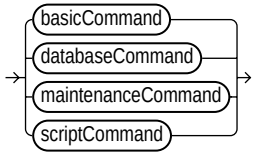
```
startup mount;
sql "ALTER TABLESPACE tbs_1 ADD DATAFILE ''sales.f'' NEXT 10K MAXSIZE 100K";
resync catalog;
```

Resynchronizing in ARCHIVELOG Mode This example performs a manual full resync for an ARCHIVELOG database after archiving all unarchived redo logs:

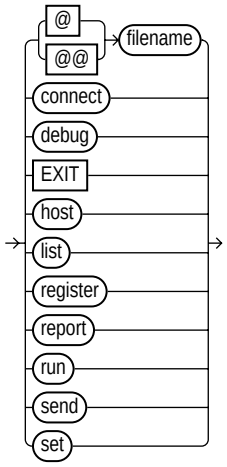
```
sql "ALTER SYSTEM ARCHIVE LOG ALL";
resync catalog;
```

rmanCommand

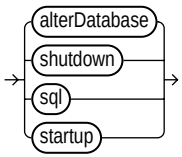
Syntax



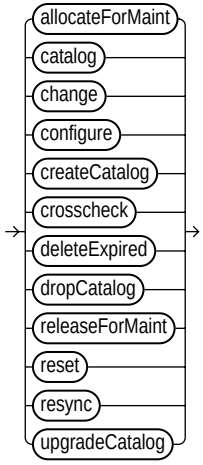
basicCommand::=



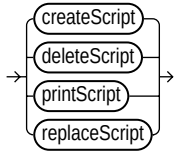
databaseCommand::=



maintenanceCommand::=



scriptCommand::=



Purpose

To execute *stand-alone* commands, which are run from the RMAN prompt. The basic categories of stand-alone commands are:

Command Type	Purpose
Basic commands	Perform the most fundamental RMAN operations: connecting, exiting, running jobs, making reports, and so forth.
Database commands	Start, shut down, and open the database, and execute SQL.
Maintenance commands	Manage the RMAN repository.
Script commands	Manage stored RMAN scripts.

Requirements

Refer to individual entries in this chapter for information about RMAN commands.

Keywords and Parameters

@filename	executes a series of RMAN commands stored in an operating system file with the specified full pathname, for example, `@$ORACLE_HOME/dbs/cmd/cmd1.rman`. If you do not specify the full pathname, the current working directory is assumed, for example, `@cmd1.rman`. Do not use quotes around the string or leave whitespace between the @ and filename. RMAN processes the specified file as if its contents had appeared in place of the @ command. **Note:** The file must contain complete RMAN commands; partial commands generate syntax errors.
@@filename	is identical to @filename unless used within a script. If contained in a script, *@@filename* directs RMAN to look for the specified filename in the same path as the command file from which it was called. For example, assume that your working directory on UNIX is `$ORACLE_HOME`, and you invoke RMAN as follows: `% rman @$ORACLE_HOME/rdbms/admin/dba/scripts/cmd1.rman` Assume that the command `@@cmd2.rman` appears inside the `cmd1.rman` script. In this case, the @@ command directs RMAN to look for the file `cmd2.rman` in the directory `$ORACLE_HOME/rdbms/admin/dba/scripts`.
exit	exits Recovery Manager.

Examples

Running a Command File This example connects to the target database and recovery catalog from the command line, then runs the command file `cmd1.f`:

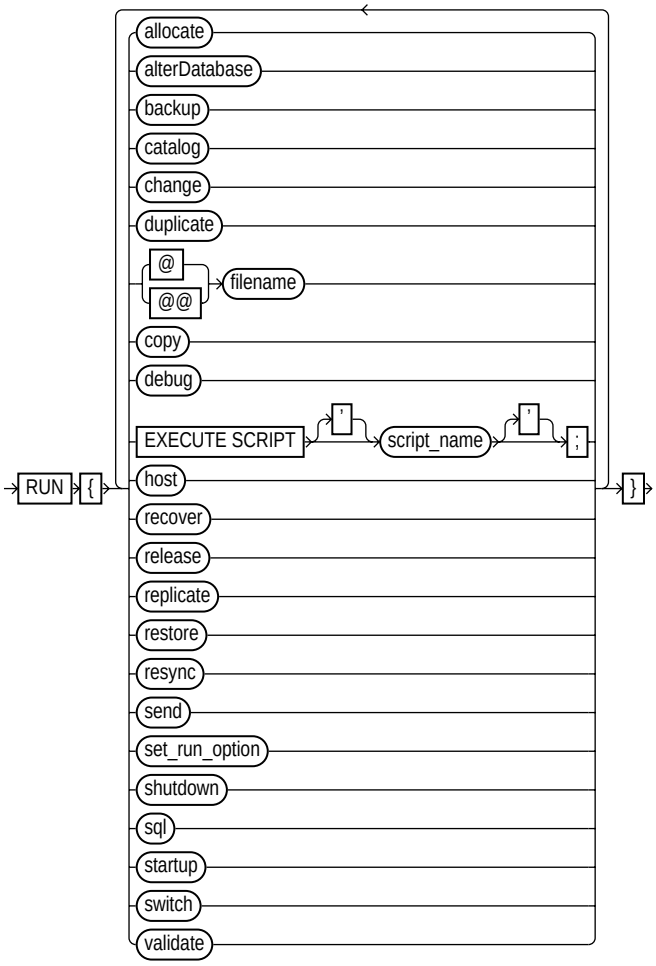
```
% rman target / catalog rman/rman@rcat
RMAN> @$ORACLE_HOME/dbs/cmd/cmd1.f
```

Related Topics

"run" on page 10-133

run

Syntax


RUN
{
allocate
alterDatabase
backup
catalog
change
duplicate
@
@@
filename
copy
debug
EXECUTE SCRIPT
'
script_name
'
;
host
recover
release
replicate
restore
resync
send
set_run_option
shutdown
sql
startup
switch
validate
}

Purpose

To compile and execute *job commands*, which are one or more statements executed within the braces of **run**. The **run** command compiles the list of job commands into one or more job steps and then executes them immediately. RMAN compiles and executes each command before processing the next one.

Requirements

- Execute this command only at the RMAN prompt.
- You must precede and follow the list of job commands with an opening and closing brace.

Keywords and Parameters

Refer to individual entries for information about commands that you can run from the RMAN prompt.

@filename	executes a series of RMAN commands stored in an operating system file with the specified full pathname, for example, `@$ORACLE_HOME/dbs/cmd/cmd1.rman`. If you do not specify the full pathname, the current working directory is assumed, for example, `@cmd1.rman`. Do not use quotes around the string or leave whitespace between the @ and filename. RMAN processes the specified file as if its contents had appeared in place of the @ command. **Note:** The file must contain only complete Recovery Manager commands. A syntax error will result if the file contains a partial command.
@@filename	is identical to @filename unless used within a script. If contained in a script, *@@filename* directs RMAN to look for the specified filename in the same path as the command file from which it was called. For example, assume that your working directory on UNIX is `$ORACLE_HOME`, and you invoke RMAN from the command line as follows: `% rman @$ORACLE_HOME/rdbms/admin/dba/scripts/cmd1.rman` Assume that the `@@cmd2.rman` command appears inside the `cmd1.rman` script. In this case, the @@ command directs RMAN to look for the file `cmd2.rman` in the directory `$ORACLE_HOME/rdbms/admin/dba/scripts`.
execute script *script_name*	runs the specified stored script. To obtain a listing of all stored scripts, use SQL*Plus to connect to the recovery catalog database as the catalog owner and issue the following query: `select * from rc_stored_script;` **See Also:** "RC_STORED_SCRIPT" on page 11-25 for more information about RC_STORED_SCRIPT, and "createScript" on page 10-61 for information about creating scripts.

Examples

Making a Backup This example backs up a database using a single server process to perform the backup:

```
run{
     allocate channel c1 type disk;
     backup database;
}
```

Restoring a Tablespace This example takes tablespace `tbs_1` offline, restores it, then performs complete media recovery:

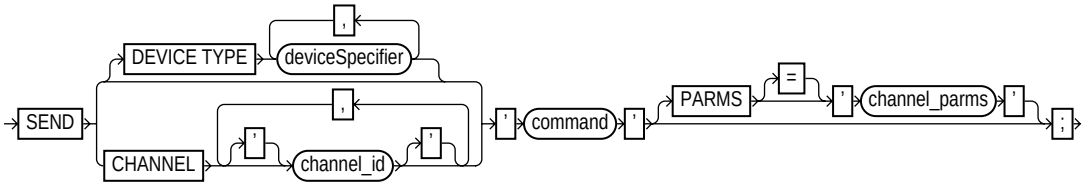
```
run {
     allocate channel ch1 type 'sbt_tape';
     sql "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE" ;
     restore tablespace tbs_1 ;
     recover tablespace tbs_1 ;
     sql "ALTER TABLESPACE tbs_1 ONLINE" ;
     release channel ch1 ;
}
```

Executing a Script This example executes the stored script `backupdb`:

```
run {  execute script backupdb; }
```

send

Syntax



Purpose

To send a vendor-specific quoted string to one or more specific channels. See your media management documentation to determine which commands are supported.

Requirements

- Execute **send** at the RMAN prompt or within the braces of a **run** command.
- You must use a media manager to use **send** and only with the commands supported by the media manager. The contents of the quoted string are not interpreted by Oracle, but are passed unaltered to the media management sub-system.

Keywords and Parameters

command	specifies a vendor-specific media management command. See your media management documentation to determine which commands are supported.
device type *deviceSpecifier*	specifies the type of storage device and sends the command to all channels of the specified type. See "deviceSpecifier" on page 10-72.
channel *channel_id*	specifies which channel to use. If you do not specify this keyword, RMAN uses all allocated channels. You must specify a channel id, which is the name of the channel, after the **channel** keyword. Oracle uses the channel id with the **release channel** command and also to report I/O errors.

parms *'channel_parms'*	specifies parameters affecting the device you have allocated. Do not use this port-specific string if you have specified **type disk**. If you use **parms** in conjunction with **type** *'sbt_tape'*, then you can specify environment variables with the following syntax: `PARMS = "ENV = (var1=value1, var2=value2, var3=value3 . . . )"` The maximum length of the quoted string is 1000 bytes.

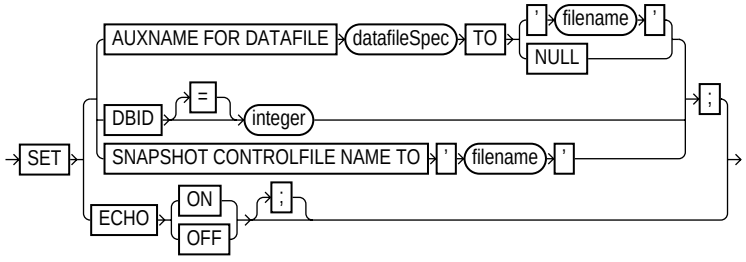
Examples

Sending a String to the Media Manager This example sends a vendor-specific command to a media manager:

```
send device type 'sbt_tape' 'A223dr';
```

set

Syntax



Purpose

Use **set** to:

- Specify the filenames for the auxiliary database during TSPITR.
- Specify new datafile filenames for use in the **duplicate** command.
- Display executed RMAN commands in the message log.
- Specify a database's db identifier.
- Set the filename of the snapshot control file.

Requirements

- Use **set** only at RMAN prompt.
- Use **set dbid** only if you have not yet connected to the target database.
- You must have a recovery catalog to use the **auxname for datafile to** *'filename'* option.

Keywords and Parameters

auxname for datafile *datafileSpec* **to** *'filename'*

sets the target database datafile to its new filename on the auxiliary database. If you are performing TSPITR or using the **duplicate** command, setting this option allows you to pre-configure the filenames for use on the auxiliary database without manually specifying the auxiliary filenames during the procedure.

For example, use this command during TSPITR if your datafiles are on raw disks and you want to restore auxiliary datafiles to raw disk for performance reasons. Typically, you set the **auxname** parameter in TSPITR for the datafiles of the SYSTEM tablespace and the tablespaces containing rollback segments. Take care not to overlay the files that are in use by the production database and which can be discarded after TSPITR completes. In essence, the **auxname** of a datafile is the location where TSPITR can create a temporary copy of it.

When renaming files with the **duplicate** command, **set auxname** is an alternative to **set newname**. The difference is that once you set the **auxname**, you do not need to reset it when you issue another **duplicate** command: it remains in effect until you issue **set auxname ... to null**. In contrast, you must reissue the **set newname** command every time you rename files.

See Also: Chapter 8, "Performing Point-in-Time Recovery with Recovery Manager" to learn how to perform RMAN TSPITR, and Chapter 7, "Creating a Duplicate Database with Recovery Manager"to learn how to duplicate a database.

to null unspecifies the current value for **auxname** for the specified datafile.

dbid *integer*

specifies the *db identifier*, which is a unique 32-bit identification number computed when the database is created. The DBID column of the V$DATABASE data dictionary view displays the identifier. The DBID is also stored in the DB table of the recovery catalog.

The **set dbid** command is useful for restoring the control file when each of these conditions is met:

- The control file has been lost and must be restored from a backup.
- You are using a recovery catalog.
- Multiple databases registered in the recovery catalog share a database name.
- You receive the `RMAN-20005: target database name is ambiguous` message when you attempt to restore the control file.

If these conditions are *not* met, RMAN will correctly identify the control file to restore, so you do not need to use the **set dbid** command.

RMAN accepts **set dbid** only if you have not yet connected to the target database, that is, **set dbid** must precede the **connect target** command. If the target database is mounted, then RMAN verifies that the user-specified DBID matches the DBID from the database; it not, RMAN signals an error. If the target database is not mounted, RMAN uses the user-specified DBID to restore the control file. Once you have restored the control file, you can mount the database to restore the rest of the database.

echo [**on** | **off**]	controls whether RMAN commands are displayed in the message log. When reading commands from a command file, RMAN automatically echoes those commands to the message log. When reading commands from STDIN, RMAN does not echo those commands to the message log unless the **set echo on** command is used. The command is useful only when `stdin` and `stdout` have been redirected. For example, in UNIX you can re-direct RMAN's input and output in this manner: `% rman target sys/sys_pwd@prod1 catalog rman/rman@rcat < input_file > output_file` By specifying **set echo on**, you enable the commands contained in `input_file` to be visible in `output_file`.
snapshot controlfile name to *'filename'*	sets the snapshot control file name in the target database to the specified location. RMAN uses the snapshot control file to resynchronize the recovery catalog. For more information about snapshot control files, see "Determining the Snapshot Control File Location" on page 2-3.

Examples

Restoring the Control File This example uses the user-specified DBID to restore the control file. Once you have restored the control file, you can mount the database to restore the rest of the database.

```
set dbid = 862893450;
connect target;
startup nomount;
run {
    allocate channel dev1 type disk;
    # restoring the control file from its default location automatically replicates it
    restore controlfile;
    alter database mount;
}
```

Specifying the Snapshot Control File Location This example specifies a new location for the snapshot control file and then resynchronizes the recovery catalog.

```
set snapshot controlfile name to '/oracle/dbs/snap.cf';
resync catalog;
```

Specifying the Snapshot Control File Location This example duplicates a database to a remote host with a different directory structure, using **set auxname** to specify new filenames for the datafiles:

```
# set auxiliary names for the datafiles
set auxname for datafile 1 to '/oracle/auxfiles/aux_1.f';
set auxname for datafile 2 to '/oracle/auxfiles/aux_2.f';
set auxname for datafile 3 to '/oracle/auxfiles/aux_3.f';
set auxname for datafile 4 to '/oracle/auxfiles/aux_4.f';
```

```
run {
     allocate auxiliary channel dupdb1 type disk;
     duplicate target database to dupdb
     logfile
       group 1 ('$ORACLE_HOME/dbs/dupdb_log_1_1.f',
                '$ORACLE_HOME/dbs/dupdb_log_1_2.f') size 200K,
       group 2 ('$ORACLE_HOME/dbs/dupdb_log_2_1.f',
                '$ORACLE_HOME/dbs/dupdb_log_2_2.f') size 200K reuse;
}
# Un-specify the auxiliary names for your datafiles so that they will not be overwritten
# by mistake:
set auxname for datafile 1 to null;
set auxname for datafile 2 to null;
set auxname for datafile 3 to null;
set auxname for datafile 4 to null;
```

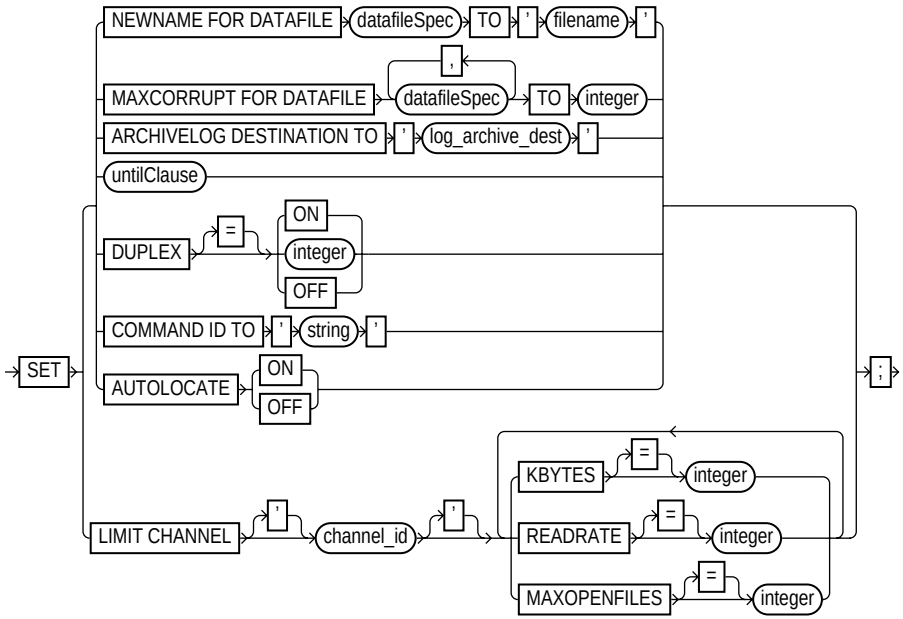
Related Topics

"set_run_option" on page 10-142

"duplicate" on page 10-76

set_run_option

Syntax



Purpose

To set attributes for a **run** command that persist until the end of the job. The specified attributes affect all statements within **run** that follow the **set** command. Use **set** to:

- Specify new filenames for datafiles.
- Specify a limit for the number of permissible block corruptions.
- Override default archived redo log destinations.
- Set an end time, SCN, or log sequence number for recovery.
- Specify that backups should be duplexed.
- Determine which server session corresponds to which channel.

- Limit the number of buffers per second that will be read from each input datafile on a specified channel.
- Limit the number of input files that a **backup** operation can have open at any given time for a specified channel.
- Limit the size of the backup pieces for a specified channel.

Requirements

- Execute this command only within the braces of a **run** command.
- The **set duplex** command affects all channels allocated after issuing the command and is in effect until explicitly disabled or changed. The **set duplex** command does not affect previously allocated channels.
- Issue **set autolocate on** only in an OPS configuration and only when the media management server is not cluster-aware.
- Issue **set autolocate on** *before* **restore** and **recover** commands.
- Issue **set limit channel** after allocating the affected channel.

Keywords and Parameters

newname for datafile *datafileSpec* **to** '*filename*'	sets the default name for all subsequent **restore** or **switch** commands that affect the specified datafile (see "datafileSpec" on page 10-66). If you do not issue this command before the datafile restore operation, then RMAN restores the file to its default location.
maxcorrupt for datafile *datafileSpec* **to** *integer*	sets a limit on the number of previously undetected physical block corruptions that Oracle will allow in a specified datafile or list of datafiles (see "datafileSpec" on page 10-66). If a **backup** or **copy** command detects more than the specified number of corruptions, the command aborts. The default limit is zero, meaning that RMAN tolerates no corrupt blocks. **Note:** If you specify **check logical**, then the **maxcorrupt** limit applies to logical corruptions as well.

autolocate forces RMAN to automatically discover which nodes of an OPS cluster contain the backups that you want to restore. Set to **on** or **off** (default).

This option forces RMAN to hunt for backups on all allocated channels and to restore backups only from those channels that locate the backups on tape or on a filesystem. For example, assume that nodes A, B, and C are in an OPS configuration. If node A backs up a datafile to a tape drive or local filesystem, you must tell RMAN not to attempt to restore from nodes A or B. The **set autolocate** command performs this function.

Issue the **set autolocate on** command only if:

- The command precedes **restore** or **recover** commands.
- Channels are allocated on different nodes of an OPS cluster.
- The media management servers do not already offer cluster-wide service.
- It is necessary (the command incurs system overhead).

archivelog destination to *'log_archive_dest'* overrides the LOG_ARCHIVE_DEST or LOG_ARCHIVE_DEST_1 initialization parameter in the target database when forming names for restored archive logs during subsequent **restore** and **recover** commands. RMAN restores the logs to the destination specified in '*log_archive_dest*'. Use this parameter to restore archived redo logs that are not already on disk.

Use this command to stage many archived logs to different locations while a database restore is occurring. RMAN knows where to find the newly restored archive logs; it does not require them to be in the destination specified by LOG_ARCHIVE_DEST_1 or LOG_ARCHIVE_DEST. For example, if you specify a different destination from the one in the parameter file and restore archived redo log backups, subsequent restore and recovery operations will detect this new location. RMAN always looks for archived redo logs on disk first before restoring them from backup sets.

untilClause specifies an end time, SCN, or log sequence number for a subsequent **restore** or **recover** command. See "untilClause" on page 10-156.

duplex specifies the number of copies of each backup piece that the channels should create: 1, 2, 3, or 4. The **set duplex** command, which affects only the **backup** command, affects all channels allocated after issuing the command and is in effect until explicitly disabled (**off**) or changed during the session. By default duplex is **off**, that is, RMAN produces a single backup set. If you specify **on**, RMAN produces two identical backup sets.

command id to *'string'*

enters the specified string into the V$SESSION.CLIENT_INFO column of all channels. Use this information to determine which Oracle server sessions correspond to which RMAN channels.

The V$SESSION.CLIENT_INFO column contains information for each RMAN server session. The data appears in one of the following formats:

- id=*string*
- id=*string*, ch=*channel_id*

The first form appears in the RMAN target database connection. The second form appears in all allocated channels. When the current job is complete, the V$SESSION.CLIENT_INFO column will be cleared.

See Also: *Oracle8i Reference* for more information on V$SESSION.CLIENT_INFO.

limit channel *channel_id*

sets parameters that specify limits applying to any **backup** or **copy** command that executes using the allocated channel.

kbytes *integer*

specifies the maximum size in kilobytes of the backup pieces created on this channel.

readrate *integer*

specifies the maximum number of buffers (each of size DB_BLOCKSIZE * DB_FILE_DIRECT_IO_COUNT) per second read for **backup** or **copy** operations from each of the input datafiles. By default, this parameter is not set. Use this parameter to "throttle back" RMAN, that is, set an upper limit for block reads so that RMAN does not consume excessive disk bandwidth and thereby degrade online performance.

See Also: *Oracle8i Designing and Tuning for Performance* for more information.

maxopenfiles *integer*

controls the maximum number of input files that a **backup** command can have open at any given time. Use this parameter to prevent "Too many open files" operating system error messages when backing up a large number of files into a single backup set. If you do not specify **maxopenfiles**, then a maximum of 32 input files can be open concurrently.

Examples

Setting the Command ID This example sets the command ID, backs up the DATA_1 tablespace, hosts out to the operating system, then archives the online redo logs:

```
run {
     set command id to 'rman';
     allocate channel t1 type 'SBT_TAPE'
     allocate channel t2 type 'SBT_TAPE';
     backup
          incremental level 0
```

```
          filesperset 5
          tablespace data_1;
     host;
     sql 'ALTER SYSTEM ARCHIVE LOG ALL';
}
```

Duplexing a Backup Set This example makes two identical backup sets of `datafile 1`:

```
run {
     set duplex = ON;
     allocate channel dev1 type disk;
     backup
          filesperset 1
          datafile 1;
}
```

Setting Channel Limits This example allocates three channels and sets the maximum size for backup pieces created on each channel. It also makes three identical backups of the database:

```
startup mount;
run {
     set duplex=3;
     allocate channel ch1 type 'sbt_tape';
     allocate channel ch2 type 'sbt_tape';
     allocate channel ch3 type 'sbt_tape';

     set limit channel ch1 kbytes 2097150;
     set limit channel ch2 kbytes 2097150;
     set limit channel ch3 kbytes 2907150;

     backup
          filesperset 5
          database;
     alter database open;
}
```

Related Topics

"recover" on page 10-96

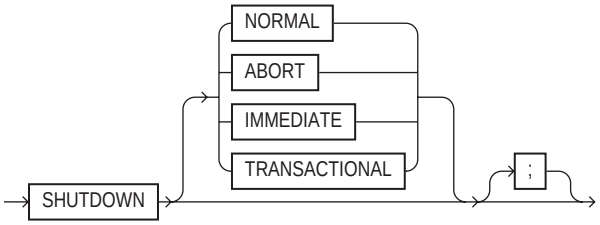
"restore" on page 10-120

"set" on page 10-138

"untilClause" on page 10-156

shutdown

Syntax



Purpose

To shut down the target database without exiting RMAN. This command is equivalent to using the SQL*Plus SHUTDOWN statement.

> **See Also:** *Oracle8i Administrator's Guide* for information on how to start up and shut down a database, and *SQL*Plus User's Guide and Reference* for SHUTDOWN syntax.

Requirements

- Execute this command at the RMAN prompt or within the braces of a **run** command.
- You cannot use the RMAN **shutdown** command to shut down the recovery catalog database. To shut down this database, start a SQL*Plus session and issue a SHUTDOWN statement.
- The **normal**, **transactional**, and **immediate** options all perform a clean close of the database. The **abort** option does not cleanly close the database; Oracle will perform instance recovery at startup.
- If your database operates in NOARCHIVELOG mode, then you must shut down the database cleanly and then issue a **startup mount** before a making a backup.

Keywords and Parameters

normal shuts down the database with normal priority (default option), which means:

- No new connections are allowed after the statement is issued.
- Before the database is shut down, Oracle waits for all currently connected users to disconnect from the database.
- The next startup of the database will not require instance recovery.

abort aborts the target instance, with the following consequences:

- All current client SQL statements are immediately terminated.
- Uncommitted transactions are not rolled back until next startup.
- Oracle disconnects all connected users.
- Oracle will perform crash recovery on the database at next startup.

immediate shuts down the target database immediately, with the following consequences:

- Current client SQL statements being processed by Oracle are allowed to complete.
- Uncommitted transactions are rolled back.
- All connected users are disconnected.

transactional shuts down the target database while minimizing interruption to clients, with the following consequences:

- Clients currently conducting transactions are allowed to complete, that is, either commit or abort before shutdown.
- No client can start a new transaction on this instance; any client attempting to start a new transaction is disconnected.
- After all transactions have either committed or aborted, any client still connected is disconnected.

Examples

Shutting Down a Database in Immediate Mode This example waits for current SQL transactions to be processed before shutting down, then mounts the database:

```
shutdown immediate;
startup mount;
```

Shutting Down a Database in NOARCHIVELOG Mode This example backs up a database running in NOARCHIVELOG mode:

```
shutdown abort;
startup dba;
```

```
shutdown;
startup mount;
# executing the above commands ensures that database is in proper state for noarchivelog
# backup
run {
     set duplex = 4;
     allocate channel dev1 type 'sbt_tape';
     backup database;
     alter database open;
}
```

Related Topics

"alterDatabase" on page 10-16

"startup" on page 10-152

sql

Syntax

Purpose

To execute a SQL statement from within Recovery Manager. For valid SQL syntax, see the *Oracle8i SQL Reference*.

Requirements

- Execute the **sql** command at the RMAN prompt or within the braces of a **run** command.
- If the string that RMAN passes to PL/SQL contains a filename, then the filename must be enclosed in duplicate *single* quotes and the entire string following the **sql** parameter must be enclosed in *double* quotes. For example, use the following syntax:

  ```
  sql "CREATE TABLESPACE temp1 DATAFILE ' '$ORACLE_HOME/dbs/temp1.f' ' "
  ```

  If you attempt to use single quotes for the string following the SQL parameter or single quotes for the filename, then the command fails.
- You cannot execute SELECT statements.

Keywords and Parameters

'command'	specifies a SQL statement for execution. For example, issuing the following: `sql 'ALTER SYSTEM ARCHIVE LOG ALL';` at the RMAN prompt archives the online redo logs.

Examples

Making an Operating System Copy of an Online Tablespace This example hosts out to the operating system to make an operating system copy of online tablespace TBS_1 and then catalogs it:

```
sql 'ALTER TABLESPACE tbs_1 BEGIN BACKUP';
host 'cp $ORACLE_HOME/dbs/tbs_1.f/dbs/tbs_1.f $ORACLE_HOME/copy/temp3.f';
sql 'ALTER TABLESPACE tbs_1 END BACKUP';
catalog datafilecopy '$ORACLE_HOME/copy/temp3.f';
sql 'ALTER SYSTEM ARCHIVE LOG ALL';
```

Specifying a Filename within a Quoted String This example specifies a filename using duplicate single quotes within the context of a double-quoted string:

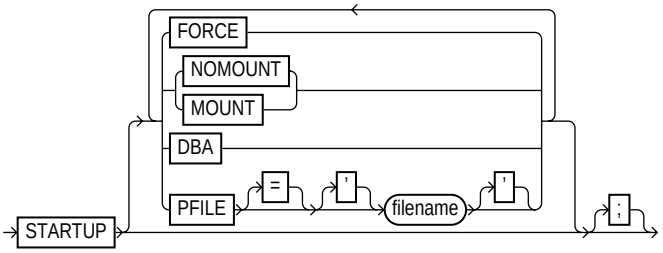
```
sql "ALTER TABLESPACE tbs_1 ADD DATAFILE ''/oracle/dbs/tbs_7.f'' NEXT 10K MAXSIZE 100k;"
```

Opening with the RESETLOGS Option This example performs incomplete recovery and opens the database with the RESETLOGS option:

```
run {
     set until scn 1000;
     allocate channel c1 type 'sbt_tape';
     restore database;
     recover database;
     sql 'ALTER DATABASE OPEN RESETLOGS';
     reset database;
}
```

startup

Syntax



Purpose

To start the database from within the RMAN environment. This command is equivalent to using the SQL*Plus STARTUP command. You can:

- Start the instance without mounting a database.
- Start the instance and mount the database, but leave it closed.
- Start the instance, and mount and open the database in:
  - unrestricted mode (accessible to all users).
  - restricted mode (accessible to DBAs only).

See Also: *Oracle8i Administrator's Guide* to learn how to start up and shut down a database, and *SQL*Plus User's Guide and Reference* for SQL*Plus STARTUP syntax.

Requirements

- Execute this command either at the RMAN prompt or within the braces of a **run** command.
- You cannot use the RMAN **startup** command to open the recovery catalog database. To start this database, start a SQL*Plus session and execute a STARTUP statement.

Keywords and Parameters

If you do not specify any options, RMAN mounts and opens the database.

force	executes either of these operations: ■ If the database is open, this option first shuts down the database with a **shutdown abort** statement before re-opening it. ■ If the database is closed, this option opens the database.
nomount	starts the instance without mounting the database.
mount	starts the instance, then mounts the database without opening it. If you specify neither the **mount** nor **nomount** options, the **startup** command opens the database.
dba	restricts access to the database to users with the RESTRICTED SESSION privilege.
pfile = *filename*	specifies the filename of the `init.ora` file for the target database. If this parameter is not specified, the default `init.ora` filename is used.

Examples

Opening the Database Using the Default Parameter File This example starts and opens the database:

```
startup;
```

Mounting the Database While Specifying the Parameter File This example forces a **shutdown abort** and then mounts the database with restricted access, specifying a non-default parameter file location:

```
startup force mount dba pfile=t_init1.ora;
```

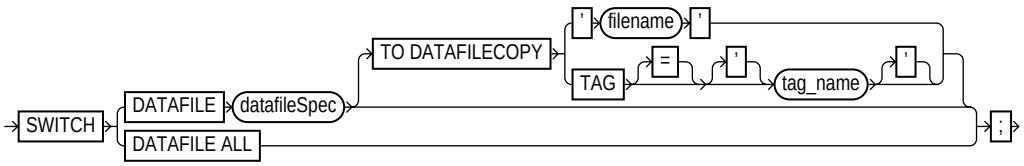
Related Topics

"alterDatabase" on page 10-16

"shutdown" on page 10-147

switch

Syntax



Purpose

To specify that a datafile copy is now the *current datafile*, that is, the datafile pointed to by the control file. A switch is equivalent to using the ALTER DATABASE RENAME DATAFILE statement: Oracle renames the files in the control file but does not actually rename them on your operating system. Note that this command updates the records for the datafile copy in the recovery catalog and the control file to status AVAILABLE.

Requirements

- Execute **switch** within the braces of a **run** command.
- If your control file is a restored backup control file, then **switch** adds the datafile to the control file if it is not there already. You can only add datafiles through **switch** that were created *after* the backup control file was created.

Keywords and Parameters

datafile
datafileSpec

specifies the datafile that you wish to rename. After the switch, the control file no longer views the specified file as current. For example, this command points the control file from `tbs_1.f` to `cp1.f`:

```
switch datafile '$/dbs/tbs_1.f' to datafilecopy '$/dbs/copies/cp1.f';
```

If you do not specify a **to** option, then RMAN uses the filename specified on a prior **set newname** command (see "set_run_option" on page 10-142) for this file number as the switch target.

	to datafilecopy	specifies the input copy file for the switch, that is, the datafile copy that you wish to rename. For example, if you issue:
		`switch datafile 2 to datafilecopy '/oracle/dbs/df2.copy';`
		The control file will list `df2.copy` as the filename for `datafile 2`.
		You can also specify the datafile copy by tag. If the tag is ambiguous, then the most current copy is used, that is, the one that requires the least media recovery.
		When the tag is not unique, then RMAN understands the tag to refer to the most recently created copy. Thus, the command:
		`switch datafile 3 to datafilecopy tag mondayPMcopy;`
		switches `datafile 3` to the most recently created Monday evening copy.

datafile all	specifies that all datafiles for which a **set newname for datafile** command (see "set_run_option" on page 10-142) has been issued in this job are switched to their new name.	

Examples

Switching After a Restore This example allocates one disk device and one tape device to allow RMAN to restore both from disk and tape.

```
run {
     allocate channel dev1 type disk;
     allocate channel dev2 type 'sbt_tape';
     sql "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE";
     set newname for datafile 'disk7/oracle/tbs11.f'
       to 'disk9/oracle/tbs11.f';
     restore tablespace tbs_1;
     switch datafile all;
     recover tablespace tbs_1;
     sql "ALTER TABLESPACE tbs_1 ONLINE";
}
```

Related Topics

"printScript" on page 10-94

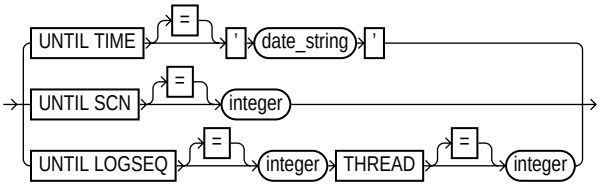
"restore" on page 10-120

"run" on page 10-133

"set" on page 10-138

untilClause

Syntax



Purpose

A sub-clause that specifies an upper limit by time, SCN, or log sequence number for various RMAN operations.

Requirements

All date strings must be either:

- Formatted according to the NLS date format specification currently in effect.
- Created by a SQL expression that returns a DATE value, for example, 'SYSDATE-30'.

Use this sub-clause in conjunction with the following commands:

- **recover**
- **report**
- **restore**
- **set_run_option**

Keywords and Parameters

until time *'date_string'*	specifies a time as an upper limit.
until scn *integer*	specifies an SCN as an upper limit.
until logseq *integer*	specifies a redo log sequence number as an upper limit.

thread *integer*	indicates the thread number for the redo log in question.

Examples

Performing Incomplete Recovery Until a Log Sequence Number This example assumes that log sequence 1234 was lost due to a disk crash and the database needs to be recovered using available archived redo logs.

```
run {
     allocate channel ch1 type disk;
     allocate channel ch2 type 'sbt_tape';
     set until logseq 1234 thread 1;
     restore controlfile to '$ORACLE_HOME/dbs/cf1.f' ;
     replicate controlfile from '$ORACLE_HOME/dbs/cf1.f';
     alter database mount;
     restore database;
     recover database;
     sql "ALTER DATABASE OPEN RESESTLOGS";
}
```

Performing DBPITR to a Specified SCN This example recovers the database until a specified SCN:

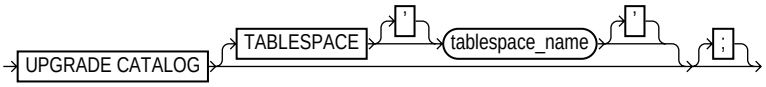
```
startup mount;
run{
     allocate channel ch1 type disk;
     restore database;
     recover database until scn 1000;
     sql "ALTER DATABASE OPEN RESESTLOGS";
}
```

Reporting Obsolete Backups This example assumes that you want to be able to recover to any point within the last seven days. It considers backups made more than a week ago as obsolete:

```
report obsolete until time 'SYSDATE-7';
```

upgradeCatalog

Syntax



Purpose

To upgrade the recovery catalog schema from an older version to the version required by the RMAN executable. For example, if you use an 8.0 recovery catalog with an 8.1 version of RMAN, then you must upgrade the catalog.

Requirements

- RMAN must be connected to the recovery catalog.
- You must enter the **upgrade** command twice in a row to confirm the upgrade.
- You will receive an error if the recovery catalog is already at a version greater than needed by the RMAN executable. RMAN permits the command to be run if the recovery catalog is already current, however, so that the packages can be re-created if necessary. RMAN displays all error messages generated during the upgrade in the message log.

Keywords and Parameters

tablespace *tablespace_name*	specifies the tablespace in which to store the recovery catalog. If not specified, then no **tablespace** parameter will be used in the CREATE TABLE statements used to upgrade the recovery catalog, which means that the catalog will be stored in the default tablespace.

Examples

Upgrading a Catalog This example connects to recovery catalog database RECDB and then upgrades it to a more current version:

```
rman catalog rcat/rcat@recdb

RMAN-06008: connected to recovery catalog database
RMAN-06186: PL/SQL package rcat.DBMS_RCVCAT version 08.00.04 in RCVCAT
```

```
            database is too old

RMAN> upgrade catalog

RMAN-06435: recovery catalog owner is rcat
RMAN-06442: enter UPGRADE CATALOG command again to confirm catalog upgrade

RMAN> upgrade catalog

RMAN-06408: recovery catalog upgraded to version 08.01.03
RMAN-06452: DBMS_RCVMAN package upgraded to version 08.01.05
RMAN-06452: DBMS_RCVCAT package upgraded to version 08.01.03
```

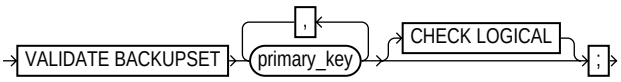
Related Topics

"createCatalog" on page 10-59

"dropCatalog" on page 10-74

validate

Syntax



Purpose

To examine a backup set and report whether it can be restored. RMAN scans all of the backup pieces in the specified backup sets and looks at the checksums to verify that the contents are intact so that the backup can be successfully restored if necessary.

> **Note:** The **validate backupset** command tests whether the backup sets can be restored, whereas **change ... crosscheck** merely examines the headers of the specified files if they are on disk or queries the media management catalog if they are on tape.

Use this command when you suspect that one or more backup pieces in a backup set are missing or have been damaged. Use **validate backupset** to specify which backups to test; use the **validate** option of the **restore** command to let RMAN choose which backups to validate.

Requirements

- Use this command only within the braces of a **run** command.
- Allocate at least one channel before executing a **validate backupset** statement.

Keywords and Parameters

primary_key specifies the backup sets to be validated by *primary_key*. Obtain the primary keys of backup sets by executing a **list** statement or, if you use a recovery catalog, by querying the RC_BACKUP_SET fixed view.

check logical	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 logs the block in the `alert.log` and server session trace file. The RMAN command completes and Oracle populates V$BACKUP_CORRUPTION and V$COPY_CORRUPTION with corrupt block ranges. **Note: validate** does not use **maxcorrupt**.

Examples

Validating a Backup Set This example validates the status of the backup set whose primary key is 12:

```
run{
     allocate channel ch1 type disk;
     validate backupset 12;
}
# As the output indicates, RMAN determines whether it is possible to restore the
# specified backup set.
RMAN-03022: compiling command: allocate
RMAN-03023: executing command: allocate
RMAN-08030: allocated channel: c1
RMAN-08500: channel ch1: sid=10 devtype=DISK

RMAN-03022: compiling command: validate
RMAN-03023: executing command: validate
RMAN-08016: channel ch1: starting datafile backupset restore
RMAN-08502: set_count=16 set_stamp=341344502 creation_time=14-AUG-98
RMAN-08023: channel ch1: restored backup piece 1
RMAN-08511: piece handle=/oracle/dbs/0ga5h07m_1_1 params=NULL
RMAN-08024: channel ch1: restore complete
RMAN-08031: released channel: ch1
```

Related Topics

"restore" on page 10-120

11

Recovery Catalog Views

This chapter contains descriptions of recovery catalog views. You can only access these views if you have created a recovery catalog.

Note: These views are not normalized, but are optimized for RMAN usage. Hence, most catalog views have redundant values that result from the join of several tables.

RC_ARCHIVED_LOG

This view lists historical information about all archived redo logs. It corresponds to the V$ARCHIVED_LOG view.

Oracle inserts an archived redo log record after the online redo log is successfully archived or cleared (NAME column is NULL if the log was cleared). If the log is archived multiple times (the maximum is 5), then there will be multiple archived log records with the same THREAD#, SEQUENCE#, and FIRST_CHANGE#, but with a different name. An archived log record is also inserted when an archived log is restored from a backup set or a copy.

Note that an archived redo log can have no log history record if its log history record is written over in the control file or after a RESETLOGS operation.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
AL_KEY	NUMBER	NOT NULL	The primary key of the archived redo log in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The archived redo log RECID from V$ARCHIVED_LOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The archived redo log stamp from V$ARCHIVED_LOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
THREAD#	NUMBER	NOT NULL	The number of the redo thread.
SEQUENCE#	NUMBER	NOT NULL	The log sequence number.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS when the record was created.
FIRST_CHANGE#	NUMBER	NOT NULL	The SCN generated when Oracle switched into the redo log.

Column	Datatype	NULL	Description
FIRST_TIME	DATE	NOT NULL	The time when Oracle switched into the redo log.
NEXT_CHANGE#	NUMBER	NOT NULL	The first SCN of the next redo log in the thread.
NEXT_TIME	DATE		The first timestamp of the next redo log in the thread.
BLOCKS	NUMBER	NOT NULL	The number of operating system blocks written (also the size of the archived log when the copy was made).
BLOCK_SIZE	NUMBER	NOT NULL	The size of the block in bytes.
COMPLETION_TIME	DATE	NOT NULL	The time when the redo log was archived or copied.
ARCHIVED	VARCHAR2(3)		Indicates whether the log was archived: YES (archived redo log) or NO (inspected file header of online redo log and added record to V$ARCHIVED_LOG). Inspecting the online logs creates archived log records for them, which allows them to be applied during RMAN recovery. Oracle sets ARCHIVED to NO to prevent online logs from being backed up.
STATUS	VARCHAR2(1)	NOT NULL	The status of the archived redo log: A (available), U (unavailable), or D (deleted).

RC_BACKUP_CONTROLFILE

This view lists information about control files in backup sets. Note that a backup datafile record with file number 0 represents the backup control file in the V$BACKUP_DATAFILE view.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
BCF_KEY	NUMBER	NOT NULL	The primary key of the control file backup in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The RECID value from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.

Column	Datatype	NULL	Description
STAMP	NUMBER	NOT NULL	The STAMP value from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	NOT NULL	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to form a join with RC_BACKUP_SET.
SET_STAMP	NUMBER	NOT NULL	The SET_STAMP value from V$BACKUP_SET. SET_ STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	NOT NULL	The SET_COUNT value from V$BACKUP_SET. SET_ STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS when the record was created.
CHECKPOINT_CHANGE#	NUMBER	NOT NULL	The control file checkpoint SCN.
CHECKPOINT_TIME	DATE	NOT NULL	The control file checkpoint time.
CREATION_TIME#	DATE	NOT NULL	The control file creation time.
BLOCK_SIZE	NUMBER	NOT NULL	The size of the blocks in bytes.
OLDEST_OFFLINE_ RANGE	NUMBER	NOT NULL	Internal use only.
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup set: A (available), U (unavailable), or D (deleted).

RC_BACKUP_CORRUPTION

This view lists corrupt block ranges in datafile backups. It corresponds to the V$BACKUP_CORRUPTION view in the control file. Note that corruptions are not tolerated in control file and archived redo log backups.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	NOT NULL	The record identifier from V$BACKUP_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The stamp propagated from V$BACKUP_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	NOT NULL	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to form a join with RC_BACKUP_SET.
SET_STAMP	NUMBER	NOT NULL	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	NOT NULL	The SET_COUNT value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
PIECE#	NUMBER	NOT NULL	The backup piece to which the block belongs.
BDF_KEY	NUMBER	NOT NULL	The primary key for the datafile backup or copy in the recovery catalog. Use this key to join with RC_BACKUP_DATAFILE. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
BDF_RECID	NUMBER	NOT NULL	The RECID value from V$BACKUP_DATAFILE.
BDF_STAMP	NUMBER	NOT NULL	The STAMP value from V$BACKUP_DATAFILE.
FILE#	NUMBER	NOT NULL	The absolute file number for the datafile.
CREATION_CHANGE#	NUMBER	NOT NULL	The SCN at backup creation.

Column	Datatype	NULL	Description
BLOCK#	NUMBER	NOT NULL	The block number of the first corrupted block in the file.
BLOCKS	NUMBER	NOT NULL	The number of corrupted blocks found beginning with BLOCK#.
CORRUPTION_CHANGE#	NUMBER		The SCN at which corruption was detected.
MARKED_CORRUPT	VARCHAR2(3)		YES if this corruption was not previously detected by Oracle or NO if it was already known by Oracle.

RC_BACKUP_DATAFILE

This view lists information about datafiles in backup sets. It corresponds to the V$BACKUP_DATAFILE view. A backup datafile is uniquely identified by BDF_KEY.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
BDF_KEY	NUMBER	NOT NULL	The primary key of the datafile backup in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The backup datafile RECID from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The backup datafile stamp from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	NOT NULL	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to form a join with RC_BACKUP_SET.
SET_STAMP	NUMBER	NOT NULL	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.

Column	Datatype	NULL	Description
SET_COUNT	NUMBER	NOT NULL	The SET_COUNT value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
BS_RECID	NUMBER	NOT NULL	The RECID from V$BACKUP_SET.
BS_STAMP	NUMBER	NOT NULL	The STAMP from V$BACKUP_SET.
BACKUP_TYPE	VARCHAR2(1)	NOT NULL	The type of the backup: D (full or level 0 incremental) or I (incremental level 1 or higher).
INCREMENTAL_LEVEL	NUMBER		The level of the incremental backup: NULL or 0 - 4.
COMPLETION_TIME	DATE		The completion time of the backup.
FILE#	NUMBER	NOT NULL	The absolute file number of the datafile.
CREATION_CHANGE#	NUMBER	NOT NULL	The SCN at the creation of the backup.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS in the datafile header.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS in the datafile header.
INCREMENTAL_CHANGE#	NUMBER	NOT NULL	The SCN that determines whether a block will be included in the incremental backup. A block is only included if the SCN in the block header is greater than or equal to INCREMENTAL_CHANGE#. The range of redo covered by the incremental backup begins with INCREMENTAL_CHANGE# and ends with CHECKPOINT_CHANGE#.
CHECKPOINT_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent datafile checkpoint.
CHECKPOINT_TIME	DATE	NOT NULL	The time of the last datafile checkpoint.
ABSOLUTE_FUZZY_CHANGE#	NUMBER		The absolute fuzzy SCN.
DATAFILE_BLOCKS	NUMBER	NOT NULL	The number of data blocks in the datafile.
BLOCKS	NUMBER	NOT NULL	The number of data blocks written to the backup.
BLOCK_SIZE	NUMBER	NOT NULL	The size of the data blocks in bytes.
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup set: A (all pieces available), D (all pieces deleted), O (some pieces are available but others are not, so the backup set is unusable).

RC_BACKUP_PIECE

This view lists information about backup pieces. This view corresponds to the V$BACKUP_PIECE view. Each backup set contains one or more backup pieces.

Multiple copies of the same backup piece can exist, but each copy has its own record in the control file and its own row in the view.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DB_ID	NUMBER	NOT NULL	The database identifier.
BP_KEY	NUMBER	NOT NULL	The primary key for the backup piece in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The backup piece RECID from V$BACKUP_PIECE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The backup piece stamp propagated from V$BACKUP_ PIECE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	NOT NULL	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to form a join with RC_BACKUP_SET.
SET_STAMP	NUMBER	NOT NULL	The SET_STAMP value from V$BACKUP_SET. SET_ STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	NOT NULL	The SET_COUNT value from V$BACKUP_SET. SET_ STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
BACKUP_TYPE	VARCHAR2(1)	NOT NULL	The type of the backup: D (full or level 0 incremental), I (incremental level 1 or higher), L (archived redo log).
INCREMENTAL_LEVEL	NUMBER		The level of the incremental backup: NULL or 0 - 4.
PIECE#	NUMBER	NOT NULL	The number of the backup piece. The first piece has the value of 1.
COPY#	NUMBER	NOT NULL	The copy number of the backup piece.
DEVICE_TYPE	VARCHAR2(255)	NOT NULL	The type of backup device: DISK or SBT_TAPE (sequential media).

Column	Datatype	NULL	Description
HANDLE	VARCHAR2(1024)	NOT NULL	The filename of the backup piece. It is the value that RMAN passes to the OSD layer that identifies the file.
COMMENTS	VARCHAR2(255)		Comments about the backup piece.
MEDIA	VARCHAR2(80)		A comment that contains further information about the media manager that created this backup.
MEDIA_POOL	NUMBER		The number of the media pool in which the backup is stored.
CONCUR	VARCHAR2(3)		Specifies whether backup media supports concurrent access: YES or NO.
TAG	VARCHAR2(32)		The user-specified tag for the backup piece.
START_TIME	DATE	NOT NULL	The time when RMAN started to write the backup piece.
COMPLETION_TIME	DATE	NOT NULL	The time when the backup piece was completed.
ELAPSED_SECONDS	NUMBER		The duration of the creation of the backup piece.
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup piece: A (available), U (unavailable), D (deleted), or X (expired).

RC_BACKUP_REDOLOG

This view lists information about archived redo logs in backup sets. It corresponds to the V$BACKUP_REDOLOG view.

You cannot back up online logs directly; you must first archive them to disk and then back them up. An archived log backup set contains one or more archived logs.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
BRL_KEY	NUMBER	NOT NULL	The primary key of the archived redo log in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.

Column	Datatype	NULL	Description
RECID	NUMBER	NOT NULL	The record identifier propagated from V$BACKUP_ REDOLOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The stamp from V$BACKUP_REDOLOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	NOT NULL	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to form a join with RC_BACKUP_SET.
SET_STAMP	NUMBER	NOT NULL	The SET_STAMP value from V$BACKUP_SET. SET_ STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	NOT NULL	The SET_COUNT value from V$BACKUP_SET. SET_ STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
BACKUP_TYPE	VARCHAR2(1)	NOT NULL	The type of the backup: L (archived redo log).
COMPLETION_TIME	DATE	NOT NULL	The time when the backup completed.
THREAD#	NUMBER	NOT NULL	The thread number of the redo log.
SEQUENCE#	NUMBER	NOT NULL	The log sequence number.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS when the record was created.
FIRST_CHANGE#	NUMBER	NOT NULL	The SCN generated when Oracle switched into the redo log.
FIRST_TIME	DATE	NOT NULL	The time when Oracle switched into the redo log.
NEXT_CHANGE#	NUMBER	NOT NULL	The first SCN of the next redo log in the thread.
NEXT_TIME	DATE	NOT NULL	The first timestamp of the next redo log in the thread.
BLOCKS	NUMBER	NOT NULL	The number of operating system blocks written to the backup.
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup set: A (all pieces available), D (all pieces deleted), O (some pieces are available but others are not, so the backup set is unusable).

RC_BACKUP_SET

This view lists information about backup sets for all incarnations of the database. It corresponds to the V$BACKUP_SET view. A backup set record is inserted after the backup has successfully completed.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DB_ID	NUMBER	NOT NULL	The unique database identifier.
BS_KEY	NUMBER	NOT NULL	The primary key of the backup set in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The backup set RECID from V$BACKUP_SET. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file. Use either RECID and STAMP or SET_STAMP and SET_COUNT to access V$BACKUP_SET.
STAMP	NUMBER	NOT NULL	The backup set STAMP from V$BACKUP_SET. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file. Use either RECID and STAMP or SET_STAMP and SET_COUNT to access V$BACKUP_SET.
SET_STAMP	NUMBER	NOT NULL	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies this record in the target database control file. Use either RECID and STAMP or SET_STAMP and SET_COUNT to access V$BACKUP_SET.
SET_COUNT	NUMBER	NOT NULL	The SET_COUNT value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies this record in the target database control file. Use either RECID and STAMP or SET_STAMP and SET_COUNT to access V$BACKUP_SET.
BACKUP_TYPE	VARCHAR2(1)	NOT NULL	The type of the backup: D (full backup or level 0 incremental), I (incremental of level 1 or higher), L (archived redo log).
INCREMENTAL_LEVEL	NUMBER	NOT NULL	The level of the incremental backup: NULL or 0 - 4.
PIECES	NUMBER	NOT NULL	The number of backup pieces in the backup set.
START_TIME	DATE	NOT NULL	The time when the backup began.
COMPLETION_TIME	DATE	NOT NULL	The time when the backup completed.
ELAPSED_SECONDS	NUMBER		The duration of the backup in seconds.

Column	Datatype	NULL	Description
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup set: A (all backup pieces available), D (all backup pieces deleted), O (some backup pieces are available but others are not, so the backup set is unusable).

RC_CHECKPOINT

This view lists recovery catalog resynchronization information. It does not give information about database checkpoints. In most cases, you should use the RC_RESYNC view instead.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8(	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
CKP_KEY	NUMBER	NOT NULL	The primary key for the checkpoint.
CKP_SCN	NUMBER	NOT NULL	The control file checkpoint SCN.
CKP_CF_SEQ	NUMBER	NOT NULL	The control file sequence number.
CKP_TIME	DATE		The time at which the catalog was checkpointed.
CKP_TYPE	VARCHAR2(7)	NOT NULL	The type of catalog resynchronization: FULL or PARTIAL.

RC_CONTROLFILE_COPY

This view lists information about control file copies on disk. A datafile copy record with a file number of 0 represents the control file copy in V$DATAFILE_COPY.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
CCF_KEY	NUMBER	NOT NULL	The primary key of the control file copy in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The record identifier from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The stamp from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
NAME	VARCHAR2(1024)	NOT NULL	The control file copy filename.
TAG	VARCHAR2(32)		The tag of the control file copy. NULL if no tag used.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS when the record was created.
CHECKPOINT_CHANGE#	NUMBER	NOT NULL	The control file checkpoint SCN.
CHECKPOINT_TIME	DATE	NOT NULL	The control file checkpoint time.
CREATION_TIME	DATE	NOT NULL	The control file creation time.
BLOCK_SIZE	NUMBER	NOT NULL	The block size in bytes.
MIN_OFFR_RECID	NUMBER	NOT NULL	Internal use only.
OLDEST_OFFLINE_RANGE	NUMBER	NOT NULL	Internal use only.
COMPLETION_TIME	DATE	NOT NULL	The time when the copy was generated.
STATUS	VARCHAR2(1)	NOT NULL	The status of the copy: A (available), U (unavailable), or D (deleted).

RC_COPY_CORRUPTION

This view lists corrupt block ranges in datafile copies. It corresponds to the V$COPY_CORRUPTION view.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	NOT NULL	The record identifier from V$COPY_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The stamp from V$COPY_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
CDF_KEY	NUMBER	NOT NULL	The primary key of the datafile copy in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output. Use this column to form a join with RC_DATAFILE_COPY.
COPY_RECID	NUMBER	NOT NULL	The RECID from RC_DATAFILE_COPY. This value is propagated from the control file.
COPY_STAMP	NUMBER	NOT NULL	The STAMP from RC_DATAFILE_COPY. This value is propagated from the control file.
FILE#	NUMBER	NOT NULL	The absolute file number of the datafile.
CREATION_CHANGE#	NUMBER	NOT NULL	The SCN recorded at the creation of the copy.
BLOCK#	NUMBER	NOT NULL	The block number of the first corrupted block in the file.
BLOCKS	NUMBER	NOT NULL	The number of corrupted blocks found beginning with BLOCK#.
CORRUPTION_CHANGE#	NUMBER		The SCN at which corruption was detected.
MARKED_CORRUPT	VARCHAR2(3)		YES if this corruption was not previously detected by the database server or NO if it was already known by the database server.

RC_DATABASE

This view gives information about the databases registered in the recovery catalog. It corresponds to the V$DATABASE view.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER		The primary key for the current incarnation. Use this column to form a join with RC_DATABASE_ INCARNATION.
DBID	NUMBER	NOT NULL	Unique identifier for the database obtained from V$DATABASE.
NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database for the current incarnation.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS operation when the record was created.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS operation when the record was created.

RC_DATABASE_INCARNATION

This view lists information about all database incarnations registered in the recovery catalog. Oracle creates a new incarnation whenever you open a database with the RESETLOGS option.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the database. Use this column to form a join with almost any other catalog view.
DBID	NUMBER	NOT NULL	Unique identifier for the database.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation.
NAME	VARCHAR2(8)	NOT NULL	The DB_NAME for the database at the time of the RESETLOGS operation.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the RESETLOGS operation that created this incarnation.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the RESETLOGS operation that created this incarnation.
CURRENT_INCARNATION	VARCHAR2(3)		YES if it is the current incarnation; NO if it is not.
PARENT_DBINC_KEY	NUMBER		The DBINC_KEY of the previous incarnation for this database. The value is NULL if it is the first incarnation recorded for the database.

RC_DATAFILE

This view lists information about all datafiles registered in the recovery catalog. It corresponds to the V$DATAFILE view. A datafile is shown as dropped if its tablespace was dropped.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
TS#	NUMBER	NOT NULL	The tablespace identifier in the target database. The TS# may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
TABLESPACE_NAME	VARCHAR2(30)	NOT NULL	The tablespace name. The name may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
FILE#	NUMBER	NOT NULL	The absolute file number of the datafile. The same datafile number may exist multiple times in the same incarnation if the datafile is dropped and re-created.
CREATION_CHANGE#	NUMBER	NOT NULL	The SCN at datafile creation.
CREATION_TIME	DATE		The time of datafile creation.
DROP_CHANGE#	NUMBER		The SCN recorded when the datafile was dropped. If a new datafile with the same file number is discovered then the DROP_CHANGE# is set to CREATION_CHANGE# for the datafile; otherwise the value is set to RC_CHECKPOINT.CKP_SCN.
DROP_TIME	DATE		The time when the datafile was dropped. If a new datafile with the same file number is discovered then the DROP_TIME is set to CREATION_TIME for the datafile; otherwise the value is set to RC_CHECKPOINT.CKP_TIME.
BYTES	NUMBER		The size of the datafile in bytes.
BLOCKS	NUMBER		The number of blocks in the datafile.
BLOCK_SIZE	NUMBER	NOT NULL	The size of the data blocks.
NAME	VARCHAR2(1024)		The datafile filename.
STOP_CHANGE#	NUMBER		SCN for datafile if offline normal or read-only.
READ_ONLY	NUMBER	NOT NULL	1 if STOP_CHANGE# is read-only; otherwise 0.

RC_DATAFILE_COPY

This view lists information about datafile copies on disk. It corresponds to the V$DATAFILE_COPY view.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
CDF_KEY	NUMBER	NOT NULL	The primary key of the datafile copy in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The datafile copy record from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The datafile copy stamp from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
NAME	VARCHAR2(1024)	NOT NULL	The filename of the datafile copy.
TAG	VARCHAR2(32)		The tag for the datafile copy.
FILE#	NUMBER	NOT NULL	The absolute file number for the datafile.
CREATION_CHANGE#	NUMBER	NOT NULL	The SCN for the datafile copy creation.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS in the datafile header.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS in the datafile header.
INCREMENTAL_LEVEL	NUMBER		The incremental level of the copy: 0 or NULL.
CHECKPOINT_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent datafile checkpoint.
CHECKPOINT_TIME	DATE	NOT NULL	The time of the most recent datafile checkpoint.
ABSOLUTE_FUZZY_CHANGE#	NUMBER		The highest SCN in any block of the file, if known.

Column	Datatype	NULL	Description
RECOVERY_FUZZY_ CHANGE#	NUMBER		The SCN to which recovery must proceed for the file to become not fuzzy. If not NULL, this file must be recovered at least to the specified SCN before the database can be opened with this file.
RECOVERY_FUZZY_TIME	DATE		The time to which recovery must proceed for the file to become not fuzzy. If not NULL, this file must be recovered at least to the specified time before the database can be opened with this file.
ONLINE_FUZZY	VARCHAR2(3)		YES / NO. If set to YES, this copy was made after a crash or OFFLINE IMMEDATE (or is a copy of a copy that was taken improperly while the database was open). Recovery will need to apply all redo up to the next crash recovery marker to make the file consistent.
BACKUP_FUZZY	VARCHAR2(3)		YES / NO. If set to YES, this is a copy taken using the BEGIN BACKUP / END BACKUP technique. To make this copy consistent, Recovery needs to apply all redo up to the marker that is placed in the redo stream when the ALTER TABLESPACE END BACKUP command is used.
BLOCKS	NUMBER	NOT NULL	The number of blocks in the datafile copy (also the size of the datafile when the copy was made).
BLOCK_SIZE	NUMBER	NOT NULL	The size of the blocks in bytes.
COMPLETION_TIME			The time when the copy completed.
STATUS	VARCHAR2(1)	NOT NULL	The status of the copy: A (available), U (unavailable), or D (deleted).

RC_LOG_HISTORY

This view lists historical information about the online redo logs. It corresponds to the V$LOG_HISTORY view.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	NOT NULL	The redo log history RECID from V$LOG_HISTORY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.

Column	Datatype	NULL	Description
STAMP	NUMBER	NOT NULL	The redo log history stamp from V$LOG_HISTORY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
THREAD#	NUMBER	NOT NULL	The thread number of the online redo log.
SEQUENCE#	NUMBER	NOT NULL	The log sequence number of the redo log.
FIRST_CHANGE#	NUMBER	NOT NULL	The SCN generated when switching into the redo log.
FIRST_TIME	DATE	NOT NULL	The timestamp when switching into the redo log.
NEXT_CHANGE#	NUMBER	NOT NULL	The first SCN of the next redo log in the thread.
CLEARED	VARCHAR2(3)		'?' if the redo log was cleared with the ALTER DATABASE CLEAR LOGFILE statement; otherwise, NULL. This statement allows a log to be dropped without archiving it first.

RC_OFFLINE_RANGE

This view lists the offline ranges for datafiles. It corresponds to the V$OFFLINE_RANGE view.

An offline range is created for a datafile when its tablespace is first altered to be offline normal or read-only, and then subsequently altered to be online or read-write. Note that no offline range is created if the datafile itself is altered to be offline or if the tablespace is altered to be offline immediate.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	NOT NULL	The record identifier for the offline range from V$OFFLINE_RANGE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The stamp for the offline range from V$OFFLINE_RANGE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.

Column	Datatype	NULL	Description
FILE#	NUMBER	NOT NULL	The absolute file number of the datafile.
CREATION_CHANGE#	NUMBER	NOT NULL	The SCN at datafile creation.
OFFLINE_CHANGE#	NUMBER	NOT NULL	The SCN taken when the datafile was taken offline.
ONLINE_CHANGE#	NUMBER	NOT NULL	The online checkpoint SCN.
ONLINE_TIME	DATE	NOT NULL	The online checkpoint time.
CF_CREATE_TIME	DATE		The time of control file creation.

RC_PROXY_CONTROLFILE

This view contains descriptions of control file backups that were taken using the proxy copy functionality. It corresponds to the V$PROXY_DATAFILE view.

In a proxy copy, the media manager takes over the operations of backing up and restoring data. Each row represents a backup of one control file.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
XDF_KEY	NUMBER	NOT NULL	The proxy copy primary key in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The proxy copy record identifier from V$PROXY_ DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The proxy copy stamp from V$PROXY_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
TAG	VARCHAR2(32)		The tag for the proxy copy.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS when the record was created.

Column	Datatype	NULL	Description
CHECKPOINT_CHANGE#	NUMBER	NOT NULL	Checkpoint SCN when the copy was made.
CHECKPOINT_TIME	DATE	NOT NULL	Checkpoint time when the copy was made.
CREATION_TIME	DATE	NOT NULL	The control file creation time.
BLOCK_SIZE	NUMBER	NOT NULL	The block size for the copy in bytes.
MIN_OFFR_RECID	NUMBER	NOT NULL	Internal use only.
OLDEST_OFFLINE_ RANGE	NUMBER	NOT NULL	Internal use only.
DEVICE_TYPE	VARCHAR2(255)	NOT NULL	The type of sequential media device.
HANDLE	VARCHAR2(1024)	NOT NULL	The filename for the proxy copy. This is the value that RMAN passes to the operating system-dependent layer that identifies the file.
COMMENTS	VARCHAR2(255)		Comments about the proxy copy.
MEDIA	VARCHAR2(80)		A comment that contains further information about the media manager that created this backup.
MEDIA_POOL	NUMBER		The number of the media pool in which the proxy copy is stored.
START_TIME	DATE	NOT NULL	The time when proxy copy was initiated.
COMPLETION_TIME	DATE	NOT NULL	The time when the proxy copy was completed.
ELAPSED_SECONDS	NUMBER		The duration of the proxy copy.
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup set: A (available), U (unavailable), X (expired), or D (deleted).

RC_PROXY_DATAFILE

This view contains descriptions of datafile backups that were taken using the proxy copy functionality. It corresponds to the V$PROXY_DATAFILE view.

In a proxy copy, the media manager takes over the operations of backing up and restoring data. Each row represents a backup of one database file.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
XDF_KEY	NUMBER	NOT NULL	The proxy copy primary key in the recovery catalog. If you issue the **list** command while connected to the recovery catalog, this value appears in the KEY column of the output.
RECID	NUMBER	NOT NULL	The proxy copy record identifier from V$PROXY_ DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	NOT NULL	The proxy copy stamp from V$PROXY_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
TAG	VARCHAR2(32)		The tag for the proxy copy.
FILE#	NUMBER	NOT NULL	The absolute file number of the datafile that is proxy copied.
CREATION_CHANGE#	NUMBER	NOT NULL	The datafile creation SCN.
RESETLOGS_CHANGE#	NUMBER	NOT NULL	The SCN of the most recent RESETLOGS in the datafile header.
RESETLOGS_TIME	DATE	NOT NULL	The timestamp of the most recent RESETLOGS in the datafile header.
INCREMENTAL_LEVEL	NUMBER		0 if this copy is part of an incremental backup strategy, otherwise NULL.
CHECKPOINT_CHANGE#	NUMBER	NOT NULL	Checkpoint SCN when the copy was made.
CHECKPOINT_TIME	DATE	NOT NULL	Checkpoint time when the copy was made.
ABSOLUTE_FUZZY_ CHANGE#	NUMBER		The highest SCN in any block of the file, if known.

Column	Datatype	NULL	Description
RECOVERY_FUZZY_ CHANGE#	NUMBER		The SCN to which recovery must proceed for the file to become not fuzzy. If not NULL, this file must be recovered at least to the specified SCN before the database can be opened with this file.
RECOVERY_FUZZY_TIME	DATE		The timestamp to which recovery must proceed for the file to become not fuzzy. If not NULL, this file must be recovered at least to the specified time before the database can be opened with this file.
ONLINE_FUZZY	VARCHAR2(3)		YES / NO. If set to YES, this copy was made after a crash or offline immediate (or is a copy of a copy which was taken improperly while the database was open). Recovery will need to apply all redo up to the next crash recovery marker to make the file consistent.
BACKUP_FUZZY	VARCHAR2(3)		YES / NO. If set to YES, this is a copy taken using the BEGIN BACKUP / END BACKUP backup method. To make this copy consistent, Recovery will need to apply all redo up to the marker that is placed in the redo stream when the ALTER TABLESPACE END BACKUP statement is issued.
BLOCKS	NUMBER	NOT NULL	Size of the datafile copy in blocks (also the size of the datafile when the copy was made).
BLOCK_SIZE	NUMBER	NOT NULL	The block size for the copy in bytes.
DEVICE_TYPE	VARCHAR2(255)	NOT NULL	The type of sequential media device.
HANDLE	VARCHAR2(1024)	NOT NULL	The filename for the proxy copy. This is the value that RMAN passes to the operating system-dependent layer that identifies the file.
COMMENTS	VARCHAR2(255)		Comments about the proxy copy.
MEDIA	VARCHAR2(80)		A comment that contains further information about the media manager that created this backup.
MEDIA_POOL	NUMBER		The number of the media pool in which the proxy copy is stored.
START_TIME	DATE	NOT NULL	The time when proxy copy was initiated.
COMPLETION_TIME	DATE	NOT NULL	The time when the proxy copy was completed.
ELAPSED_SECONDS	NUMBER		The duration of the proxy copy.
STATUS	VARCHAR2(1)	NOT NULL	The status of the backup set: A (available), U (unavailable), X (expired), or D (deleted).

RC_REDO_LOG

This view lists information about the online redo logs for all incarnations of the database.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
THREAD#	NUMBER	NOT NULL	The number of the redo thread.
GROUP#	NUMBER	NOT NULL	The number of the online redo log group.
NAME	VARCHAR2(1024)	NOT NULL	The name of the online redo log file.

RC_REDO_THREAD

This view lists data about all redo threads for all incarnations of the database.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
THREAD#	NUMBER	NOT NULL	The redo thread number for the database incarnation.
STATUS	VARCHAR2(1)	NOT NULL	The status of the redo thread: D (disabled), E (enabled), or O (open).
SEQUENCE#	NUMBER	NOT NULL	The last allocated log sequence number.
ENABLE_CHANGE#	NUMBER		The SCN at which this thread was enabled.
ENABLE_TIME	DATE		The time at which this thread was enabled.
DISABLE_CHANGE#	NUMBER		The SCN of the last disabled thread.
DISABLE_TIME	DATE		The time of the last disabled thread.

RC_RESYNC

This view lists information about recovery catalog resynchronizations. Every full resynchronization takes a snapshot of the target database control file and resynchronizes the recovery catalog from the snapshot.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
RESYNC_KEY	NUMBER	NOT NULL	The primary key for the resynchronization.
CONTROLFILE_ CHANGE#	NUMBER	NOT NULL	The control file checkpoint SCN from which the catalog was resynchronized.
CONTROLFILE_TIME	DATE		The control file checkpoint timestamp from which the catalog was resynchronized.
CONTROLFILE_ SEQUENCE#	NUMBER	NOT NULL	The control file sequence number.
CONTROLFILE_ VERSION	DATE	NOT NULL	The creation time for the version of the control file from which the catalog was resynchronized.
RESYNC_TYPE	VARCHAR2(7)	NOT NULL	The type of resynchronization: FULL or PARTIAL.
DB_STATUS	VARCHAR2(7)		The status of the target database: OPEN or MOUNTED.
RESYNC_TIME	DATE	NOT NULL	The time of the resynchronization.

RC_STORED_SCRIPT

This view lists information about scripts stored in the recovery catalog. The view contains one row for each stored script.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the database that owns this script. Use this column to form a join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
SCRIPT_NAME	VARCHAR2(100)	NOT NULL	The name of the script.

RC_STORED_SCRIPT_LINE

This view lists information about lines of the scripts stored in the recovery catalog. The view contains one row for each line of each stored script.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the database that owns this script. Use this column to form a join with almost any other catalog view.
SCRIPT_NAME	VARCHAR2(100)	NOT NULL	The name of the stored script.
LINE	NUMBER	NOT NULL	The number of the line in the script. The line of a script is uniquely identified by SCRIPT_NAME and LINE.
TEXT	VARCHAR2(1024)	NOT NULL	The text of the line of the script.

RC_TABLESPACE

This view lists all tablespaces registered in the recovery catalog, all dropped tablespaces, and tablespaces that belong to old incarnations. It corresponds to the V$TABLESPACE view. The current value is shown for tablespace attributes.

Column	Datatype	NULL	Description
DB_KEY	NUMBER	NOT NULL	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	NOT NULL	The primary key for the incarnation of the target database. Use this column to form a join with RC_ DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	NOT NULL	The DB_NAME of the database incarnation to which this record belongs.
TS#	NUMBER	NOT NULL	The tablespace identifier in the target database. The TS# may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
NAME	VARCHAR2(30)	NOT NULL	The tablespace name. The name may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
CREATION_CHANGE#	NUMBER	NOT NULL	The creation SCN (from the first datafile).
CREATION_TIME	DATE		The creation time of the tablespace. NULL for offline tablespaces after creating the control file.

Column	Datatype	NULL	Description
DROP_CHANGE#	NUMBER		The SCN recorded when the tablespace was dropped. If a new tablespace with the same TS# is discovered then the DROP_CHANGE# is set to CREATION_CHANGE# for the tablespace; otherwise, the value is set to RC_CHECKPOINT.CKP_SCN.
DROP_TIME	DATE		The date when the tablespace was dropped.

Glossary

advancing the checkpoint

The action that occurs when the redo log entry marking the checkpoint changes. For example, the CKPT process logs redo record 356 as the checkpoint, then three seconds later records redo record 358 as the checkpoint.

See Also: checkpoint, redo record

archived redo log

A copy of one of the filled members of an online redo log group made when the database is in ARCHIVELOG mode. As the LGWR process fills each online redo log with redo records, Oracle copies the log to one or more offline archive log destinations. This copy is the archived redo log, also known as the *offline redo log*.

ARCHIVELOG mode

The mode of the database in which Oracle copies filled online redo logs to disk. Specify the mode at database creation or by using the ALTER DATABASE command. You can enable automatic archiving either dynamically using the ALTER SYSTEM command or by setting the initialization parameter LOG_ARCHIVE_START to TRUE.

Running your database in ARCHIVELOG mode has several advantages over NOARCHIVELOG mode. You can:

- Back up your database while it is open and being accessed by users.
- Recover your database to any desired point in time.

To protect your ARCHIVELOG mode database in case of failure, back up your archived logs.

See Also: archived redo log, NOARCHIVELOG mode

archiving

The operation in which the ARC*n* background process copies filled online redo logs to offline destinations. You must run the database in ARCHIVELOG mode to archive redo logs.

ATL (automated tape library)

A unit that contains one or more tape drives, a robotic arm, and a shelf of tapes. The ATL, also called a *tape silo*, is able to load and unload tapes into the tape drive from the shelf without operator intervention. More sophisticated tape libraries are able to identify each tape; for example, the robotic arm can use a bar-code reader to scan each tape's barcode and identify it.

See Also: media manager

auxiliary database

(1) A database created from target database backups using the RMAN **duplicate** command.

(2) A temporary database that is restored to a new location and then started up with a new instance name during tablespace point-in-time recovery (TSPITR). A TSPITR auxiliary database contains the recovery set and auxiliary set.

See Also: TSPITR, recovery set, auxiliary set

auxiliary set

In TSPITR, the set of files that is not in the recovery set but which must be restored in the clone database for the TSPITR set to be successful. These auxiliary files include:

- Backup control file
- SYSTEM tablespace
- Any datafiles containing rollback segments
- Temporary tablespace (optional)

See Also: auxiliary database, recovery set, TSPITR

backup

(1) A copy of data, that is, a database, tablespace, table, datafile, control file, or archived redo log. You can make a backup by:

- Making a copy of one or more tables via the Export utility.

- Using Recovery Manager to back up one or more datafiles, control files, or archived redo logs.
- Making a copy either to disk or to tape using operating system utilities (such as `cp`, `tar`, `dd`).

(2) An RMAN command that creates a backup set. The output of a backup command is only usable by RMAN; the output of the RMAN **copy** command can be used without additional processing.

See Also: copy, backup set, multiplexing, RMAN

backup, closed

See closed backup

backup, whole database

See whole database backup

backup control file

A backup of the control file. Make the backup by:

- Using the Recovery Manager **backup** or **copy** command. Never create a backup control file by using operating system commands.
- Using the SQL command ALTER DATABASE BACKUP CONTROLFILE TO '*filename*'.

Typically, you restore backup control files when all copies of the current control file are damaged; sometimes you restore them before performing certain types of point-in-time recovery.

See Also: control file

backup piece

A backup piece is a physical file in an RMAN-specific format that belongs to only one backup set. A backup set usually contains only one backup piece. The only time RMAN creates more than one backup piece is when you limit the piece size a **set limit kbytes** command. Use this command when the storage or media manager you are writing your backup to is not able to support writing a file larger than a certain size.

See Also: backup, backup set, RMAN

backup set

An RMAN-specific logical grouping of one or more physical files called *backup pieces*. The output of the RMAN **backup** command is a backup set. Extract the files in a backup set by using the RMAN **restore** command. You can multiplex files into a backup set, that is, intermingle blocks from input files into a single backup set.

There are two types of backup sets:

- Datafile backup sets, which are backups of any datafiles or a control file. This type of backup set is *compressed*, which means that it only contains datafile blocks that have been used; unused blocks are omitted.
- Archivelog backup sets, which are backups of archived redo logs.

See Also: backup piece, compression, multiplexing, RMAN

breaking a mirror

The termination of a disk mirroring procedure so that a mirror image is no longer kept up-do-date. You can create operating system database backups by placing the tablespaces in the database in hot backup mode and then breaking the mirror. After taking the tablespaces out of hot backup mode, back up the broken mirror side to tape. After the backup is complete, you can *resilver* the mirror.

See Also: hot backup mode, mirroring, resilvering a mirror

buffer cache

The portion of the SGA that holds copies of data blocks read from datafiles. All user processes concurrently connected to the instance share access to the database buffer cache.

The buffers in the cache are organized in two lists: the dirty list and the least recently used (LRU) list. The dirty list holds dirty buffers, which contain data that has been modified but has not yet been written to disk. The least recently used (LRU) list holds free buffers (unmodified and available), pinned buffers (currently being accessed), and dirty buffers that have not yet been moved to the dirty list.

See Also: SGA (System Global Area)

cancel-based recovery

A type of incomplete media recovery in which you use the RECOVER command with the UNTIL CANCEL clause. Recovery proceeds until you issue the CANCEL command.

See Also: incomplete recovery, media recovery

change vector

A single change to a single data block. A change vector is the smallest unit of change recorded in the redo log.

See Also: redo record

change-based recovery

A type of incomplete media recovery that recovers up to a specified SCN. You can also perform cancel-based recovery, which recovers until you issue the CANCEL command, and time-based recovery, which recovers to a specified time.

See Also: cancel-based recovery, incomplete recovery, media recovery, system change number (SCN), time-based recovery

channel

A connection between Recovery Manager and the target database. Each allocated channel starts a new Oracle server session; the session then performs backup, restore, and recovery operations. The type of channel determines whether the Oracle server process will attempt to read or write and whether it will work through a third-party media manager. If the channel is of type:

- **disk**, the server process attempts to read backups from or write backups to disk.
- '*sbt_tape*', the server process attempts to read backups from or write backups to a third-party media manager.

Channels are always able to read and write datafiles to and from disk, no matter what their type.

See Also: channel limits, media manager, target database

channel limits

RMAN parameters that allow you to control backups and copies created by a specified channel. You can set the following limits:

- **kbytes**, which specifies the maximum size of a backup piece created on a channel.
- **readrate**, which specifies a maximum number of buffers per second read from each input datafile.
- **maxopenfiles**, which determines the maximum number of input files that a backup or copy can have open at a given time.

See Also: channel

checkpoint

A pointer indicating that all changes prior to the SCN specified by a redo record have been written to the datafiles by DBW*n*. Each redo record in the redo log describes a change or a set of atomic changes to database blocks; a checkpoint for a redo entry confirms that the changes described in previous redo entries have been written to disk, not just to memory buffers. The background process CKPT automatically records a checkpoint in the control file every three seconds.

See Also: control file, redo record

checksum

A numeric value that is mathematically derived from the contents of an Oracle data block. The checksum allows Oracle to validate the consistency of the block.

See Also: data block

circular reuse records

Control file records containing non-critical information used by RMAN for backups and recovery operations. These records are arranged in a logical ring. When all available record slots are full, Oracle either expands the control file to make room for a new records or overwrites the oldest record.

See Also: non-circular reuse records

clean shutdown

A database shut down with the IMMEDIATE, TRANSACTIONAL, or NORMAL options of the SHUTDOWN command. A database shut down cleanly does not require recovery; it is already in a consistent state.

closed backup

A backup of one or more database files taken while the database is closed. Typically, closed backups are also whole database backups. If you closed the database cleanly, then all the files in the backup are consistent. If you shut down the database using a SHUTDOWN ABORT or the instance terminated abnormally, then the backups are inconsistent.

See Also: clean shutdown, consistent backup

closed database

A database that is not available to users for queries and updates. When the database is closed you can start the instance and optionally mount the database.

See Also: open database

cold backup

See closed backup

command file

A file containing a sequence of RMAN commands that you can run from the command line. The contents of the command file should be identical to commands entered at the command line.

complete recovery

The recovery of a database by applying all online and archived redo generated since the restored backup. Typically, you perform complete media recovery when media failure damages one or more datafiles or control files. You fully recover the damaged files using all redo generated since the restored backup was taken. If you use RMAN, you can also apply incremental backups during complete recovery.

See Also: incomplete recovery, media recovery

compression

The process of copying only used data blocks into RMAN backup sets. A newly created datafile contains many never-used blocks. When RMAN creates backup sets, it only includes blocks that have been used; it follows that RMAN does not write never-used blocks into backup sets.

consistent backup

A whole database backup that you can open with the RESETLOGS option without performing media recovery. In other words, you do not need to apply redo to any datafiles in this backup for it to be consistent. All datafiles in a consistent backup must:

- Have the same checkpoint SCN in their headers, unless they are datafiles in tablespaces that are read-only or offline normal (in which case they will have a clean SCN that is earlier than the checkpoint SCN).
- Contain no changes past the checkpoint SCN, that is, are not fuzzy.
- Match the datafile checkpoint information stored in the control file.

You can only take consistent backups after a database has been shut down cleanly. The database must not be opened until the backup has completed.

See Also: clean shutdown, fuzzy file, inconsistent backup, system change number (SCN), whole database backup

control file

A binary file associated with a database that maintains the physical structure and timestamps of all files in that database. Oracle updates the control file continuously during database use and must have it available for writing whenever the database is mounted or open.

See Also: backup control file, current control file

copy

(1) To replicate data. You make copies of Oracle datafiles, control files, and archived redo logs in two ways:

- Using operating system utilities (for example, the UNIX `cp` or `dd`).
- Using the Recovery Manager **copy** command.

(2) A Recovery Manager command that makes a replica of a database's datafiles, control file, or archived redo logs. This replica is made by an Oracle server process, allocated to a Recovery Manager channel, which reads the Oracle file and writes a replica out to disk. Recovery Manager can copy the files of an open database without putting the tablespaces into hot backup mode.

See Also: backup, hot backup mode

current datafile

In RMAN, the datafile in the target database pointed to by the control file. You can make a backup datafile current again by executing a **switch** command.

corrupt block

An Oracle block that is not in a recognized Oracle format, or whose contents are not internally consistent. Oracle identifies corrupt blocks as one of two types:

- Logically corrupted, that is, the block was corrupted by the application of redo.
- Media corrupted, that is, the block format is not correct. The block may have:
  - An impossible format
  - A wrong data block address
  - An impossible block type

You can only repair a media corrupted block by:

- Replacing the block and initiating recovery. Replace the block by restoring the datafile or applying an incremental backup.

- Renewing the block. Renew a block by dropping the table (or other database object) that contains the corrupt block so that its blocks are reused for another object

If media corruption is due to faulty hardware, neither solution will work until the hardware fault is corrected.

See Also: data block, fractured block

corrupt datafile

A datafile that contains one or more corrupt blocks.

See Also: corrupt block

crash recovery

The automatic application of online redo records to a database after either a single-instance database crashes or all instances of an OPS database crash. Crash recovery only requires redo from the online logs: archived redo logs are not required.

In crash recovery, an instance automatically recovers the database before opening it. In general, the first instance to open the database after a crash or SHUTDOWN ABORT automatically performs crash recovery.

See Also: recovery, redo record

crosscheck

A check to determine whether files on disk or in the media management catalog correspond to the information in the recovery catalog (if used) and the control file. Because the media manager can mark tapes as expired or unusable, and because files can be deleted from disk or otherwise become corrupted, the recovery catalog and control file can contain outdated information about backups and image copies.

Use **change ... crosscheck** when you want to provide a list of backup sets or pieces to check; use **crosscheck backupset** when you wish to restrict the crosscheck to a specified device type, object type, or date range and let RMAN generate the list of backup sets or pieces. To determine whether you can restore a file, use **validate** or **restore ... validate**.

See Also: media manager, recovery catalog, validation

cumulative backup

An incremental backup that backs up all the blocks changed since the most recent backup at level *n*-1 or lower. For example, in a cumulative level 2 backup, RMAN

determines which level 1 or level 0 backup is most recent and then backs up all blocks changed since that backup.

See Also: data block, differential backup, incremental backup, multi-level incremental backups

current control file

The control file on disk; it is the most recently modified control file for the current incarnation of the database. For a control file to be considered current during recovery, it must not have been restored from backup.

See Also: control file

current online redo log

The online redo log file in which the LGWR background process is currently logging redo records. Those files to which LGWR is not writing are called inactive.

Every database must contain at least two online redo log files. If you are *multiplexing* your online redo log, LGWR concurrently writes the same redo data to multiple files. The individual files are called *members* of an online redo log *group*.

See Also: online redo log, redo log, redo log buffer, redo log groups

data block

The smallest unit of data in an Oracle database, the size of which is determined by the parameter DB_BLOCK_SIZE at database creation.

See Also: corrupt block

database point-in-time recovery (DBPITR)

The recovery of a database to a specified non-current time, SCN, or log sequence number.

See Also: incomplete recovery, tablespace point-in-time recovery (TSPITR)

datafile

A datafile is a physical operating system file on disk that was created by Oracle and contains data structures such as tables and indexes. A datafile can only belong to one database.

See Also: inaccessible datafile

datafile copy

A copy of a datafile on disk produced by either:

- The Recovery Manager **copy** command.
- An operating system utility.

See Also: backup, copy

datafile header

See **file header**

db identifier

An internal, uniquely generated number that differentiates databases. Oracle creates this number automatically when you create the database.

differential backup

A type of incremental backup that 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 2, level 1, or level 0 backup is most recent and then backs up all blocks changed since that backup. Differential backups, also called *non-cumulative incremental backups*, are the default type of incremental backup.

See Also: cumulative backup, incremental backup, multi-level incremental backups

duplicate database

A database created from target database backups using the RMAN duplicate command.

See Also: auxiliary database

export

The extraction of logical data (that is, not physical files) from a database using the Export utility. You can then use the Import utility to import the data into a database.

See Also: full export

file header

The first block of an Oracle datafile. The file header contains bookkeeping information related to the file, including the checkpoint SCN. Oracle requires media recovery when the checkpoint SCN in the datafile header does not match the file header information stored in the control file.

See Also: checkpoint

fractured block

A type of media corruption that can occur when DBW*n* is writing a block at the same time an operating system utility is reading the block for backup. The block that the operating system reads can be *split*, that is, the top of the block is written at one point in time while the bottom of the block is written at another point in time. If you restore a file containing a fractured block and Oracle reads the block, then the block is considered *corrupt*.

The potential for fractured blocks necessitates putting tablespaces in hot backup mode before operating system online backups. A database in hot backup mode writes whole Oracle data blocks to the redo log, so that if a block is split during the backup, you can repair it by using redo. Recovery Manager does not experience this problem because the server process performing the backup or copy reads each block to determine whether it is split and re-reads the block until it gets a consistent version.

See Also: corrupt block, corrupt datafile, hot backup mode

full backup

A non-incremental RMAN backup. Note that "full" does not refer to how much of the database is backed up, but to the fact that the backup is not incremental. Consequently, you can make a full backup of one datafile.

The only difference between a full backup and an incremental level 0 backup is that the full backup will not affect the number of blocks backed up by any subsequent incremental backup.

See Also: incremental backup

full export

An export of the whole database.

See Also: export

full resynchronization

A Recovery Manager operation that updates the recovery catalog with all changed information in the database's control file. You can initiate full catalog resynchronizations by issuing the RMAN command **resync catalog**. Recovery Manager initiates resync operations as needed when executing certain commands.

See Also: control file, recovery catalog, resynchronization

fuzzy file

A datafile that contains at least one block with an SCN more recent than the checkpoint SCN in its header. For example, this situation occurs when Oracle updates a datafile that is in hot backup mode. A fuzzy file that is restored always requires recovery.

See Also: checkpoint, hot backup mode

hot backup

See open backup

hot backup mode

The database mode initiated when you issue the ALTER TABLESPACE *tablespace_name* BEGIN BACKUP command before taking an open backup. You take a tablespace out of hot backup mode when you issue the ALTER TABLESPSACE *tablespace_name* END BACKUP command.

You must use this command when you make an operating system backup of one or more datafiles in an online tablespace. Recovery Manager does not require you to put the database in hot backup mode. Updates to tablespaces in hot backup mode create more than the usual amount of redo because each change causes Oracle to write the entire block rather than just the changed data to the redo log.

See Also: corrupt block, fractured block, open backup

image copy

A copy of a single datafile, archived redo log file, or control file that is:

- Usable as-is to perform recovery (unlike a backup set, which is in an RMAN-specific format).
- Generated using the RMAN **copy** command or an operating system command such as the UNIX dd.

See Also: copy

inaccessible datafile

A datafile that Oracle is attempting to read, but cannot find. Attempts to access an inaccessible file result in errors. Typically, a file is inaccessible because the media on which it is stored is faulty or the file has been moved or deleted.

See Also: datafile, media failure

inactive redo log

A redo log file that is not required for instance recovery because the changes contained in its redo records have already been applied to the database. The current redo log file is never inactive. If you operate your database in ARCHIVELOG mode, the ARC*n* process archives inactive redo log files.

See Also: current online redo log, online redo log, redo log, redo log buffer, redo log groups

incarnation

A separate version of a physical database. The incarnation of the database changes when you open it with the RESETLOGS option. Make a whole database backup of all files that are not offline-clean or read-only after opening with the RESETLOGS option. If using RMAN, issue the **reset database** command after opening in RESETLOGS mode.

incomplete recovery

The recovery of a database in which you do not apply all of the changes generated since you created the restored backup.

Incomplete recovery is usually performed when:

- The online logs are lost due to hardware failure. In this case, you recover the database until the last archived log generated before the failure.
- A user error necessitates recovery up until just before the error occurred.

  The requirement is to recover up until some point in time before an incorrect action occurred in the database. For example, a user mistakenly deletes payroll transactions before the transactions are sent to the payroll agency. In this example, the DBA will need to restore the whole database and then perform incomplete recovery up until the point just before the user deleted the transactions.
- An archived redo log required for recovery is missing

  An archived redo log which is needed for complete recovery was not backed up, or the archived redo log contents are corrupt. In this case, you only option is to recover up to the missing log.

In each case, open the database with the RESETLOGS option after performing media recovery. If you use RMAN with a recovery catalog, you must also reset the database.

See Also: complete recovery, media recovery, recovery, redo record

inconsistent backup

A backup in which some of the files in the backup contain changes that were made after the files were checkpointed. This type of backup needs recovery before it can be made consistent. Inconsistent backups are usually created by taking open database backups; that is, the database is open while the files are being backed up. You can also make an inconsistent backup by backing up datafiles while a database is closed, either:

- Immediately after an Oracle instance crashed (or all instances in an Oracle Parallel Server cluster).
- After shutting down the database using SHUTDOWN ABORT.

Note that inconsistent backups are only useful if the database is in ARCHIVELOG mode.

See Also: consistent backup, open backup, system change number (SCN), whole database backup

incremental backup

An RMAN backup in which only modified blocks are backed up. Incremental backups are classified by *level*. An incremental level 0 backup performs the same function as a full backup in that they both back up all blocks that have ever been used. The difference is that a full backup will not affect blocks backed up by subsequent incremental backups, whereas an incremental backup will affect blocks backed up by subsequent incremental backups.

Incremental backups at levels greater than 0 back up only blocks that have changed since previous incremental backups. Blocks that have not changed are not backed up.

Incremental backups are divided into two types: *differential* and *cumulative*. Differentials back up all blocks that have changed since the most recent backup at level *n* or lower. For example, a differential level 2 backup backs up all blocks modified since a previous level 2, level 1, or level 0 backup, whichever is most recent. Cumulative backups back up all the blocks used since the most recent backup at level *n*-1 or lower. For example, a cumulative level 2 backup backs up all blocks modified since a previous level 1 or level 0 backup, whichever is most recent.

See Also: cumulative backup, incremental backup

instance

An SGA, Oracle code, and background processes. Create an instance by issuing any of the following commands:

- STARTUP NOMOUNT—the instance starts, but does not mount the control files or open the database.
- STARTUP MOUNT—The instance starts, then mounts the database's control files. It does not open the database.
- STARTUP—The instance starts, mounts the database's control files, and opens the database.

An instance is stopped by issuing a SHUTDOWN statement.

See Also: SGA (System Global Area)

instance recovery

In an OPS configuration, the application of redo data to an open database by an instance when this instance discovers that another instance has crashed. A surviving instance automatically uses the redo log to recover the data in the instance's buffer cache. Oracle undoes any uncommitted transactions that were in progress on the failed instance when it crashed and then clears any locks held by the crashed instance after recovery is complete.

See Also: recovery, redo record

job

The contents of an RMAN **run** command.

See Also: job commands

job commands

RMAN commands such as **backup**, **copy**, and **recover** that you must execute within the brackets of a **run** command.

See Also: stand-alone commands

LogMiner

A utility that allows you read information contained in online or archived redo logs based on various selection criteria. For example, you can select information from the V$LOGMINER_CONTENTS view that enables you to:

- Track changes to a specific table.
- Track changes made by a specific user.
- Map data access patterns.

- View the SQL syntax for undoing or redoing a specific change made against the database.
- Use archived data for tuning and capacity planning.

See Also: archived redo log

log sequence number

A number that uniquely identifies a set of redo records in a redo log file. When Oracle fills one online redo log file and switches to a different one, Oracle automatically assigns the new file a log sequence number. For example, if you create a database with two online log files, then the first file is assigned log sequence number 1. When the first file fills and Oracle switches to the second file, it assigns log sequence number 2; when it switches back to the first file, it assigns log sequence number 3, and so forth.

See Also: log switch, redo log

log sequence recovery

For RMAN, a type of incomplete recovery that recovers up to a specified log sequence number.

See Also: incomplete recovery

log switch

The point at which LGWR stops writing to the active redo log file and switches to the next available redo log file. LGWR switches when either the active log file is filled with redo records or you force a switch manually.

If you run your database in ARCHIVELOG mode, Oracle archives the redo data in inactive log files into archived redo logs. When a log switch occurs and LGWR begins overwriting the old redo data, you are protected against data loss because the archived redo log contains the old data. If you run in NOARCHIVELOG mode, Oracle overwrites old redo data at a log switch without archiving it. Hence, you lose all old redo data.

See Also: redo log

logical backups

Backups in which the Export utility uses SQL to read database data and then export it into a binary file at the operating system level. You can then import the data back into a database using the Import utility.

Backups taken with the Export utility differ in the following ways from RMAN backups:

- Database logical objects are exported independently of the files that contain those objects.
- Logical backups can be imported into a different database, even on a different platform. RMAN backups are not portable between databases or platforms.

See Also: physical backups

managed recovery mode

A mode of a standby database in which the standby waits for archived log files from a target database and then automatically applies the redo logs once the files become available. This feature eliminates the need for you to interactively provide the recovery process with filenames of the archived redo logs.

See Also: standby database

media failure

A physical problem that arises when Oracle fails in its attempt to write or read a file that is required to operate the database. A common example is a disk head crash that causes the loss of all data on a disk drive. Disk failure can affect a variety of files, including the datafiles, redo log files, and control files. Because the database instance cannot continue to function properly, it cannot write the data in the buffer cache of the SGA to the datafiles.

See Also: buffer cache, media recovery

media manager

A utility provided by a third party vendor that is capable of actions such as loading, labelling and unloading sequential media such as tape drives. Media managers also allow you to configure media expiration and recycling, and may also have the ability to control Automated Tape Libraries (ATLs).

See Also: ATL (automated tape library)

media management interface

An Oracle published API to which media management vendors have written compatible software libraries. This software integrates with Oracle so that an Oracle server process is able to issue commands to the Media Manager to write backup files to sequential storage, and read files from sequential storage. When Oracle issues a request to backup or restore a file, the media manager handles the actions required to load, label, and unload the correct tape.

The media management interface is also called the *media management layer*, the *media management library (MML)*, and the *SBT interface*.

See Also: media manager

media recovery

The application of online or archived redo records to a restored backup to bring it current to a specified time. When performing media recovery, you can recover:

- The whole database
- A tablespace
- A datafile

If you use all redo data, you perform complete recovery; if you use only part of the redo data, you perform incomplete recovery. Typically, you perform media recovery after a media failure.

In ARCHIVELOG mode, you have the choice of complete or incomplete recovery. In NOARCHIVELOG mode, your only option is typically to restore from the most recent backup without applying redo data.

In exceptional circumstances, you can recover a datafile or database if the database is not in ARCHIVELOG mode, but only if none of the online logs has been overwritten since the backup.

See Also: complete recovery, incomplete recovery, media failure, recovery, redo record

mirroring

Using the operating system to maintain an identical copy of Oracle data. Typically, mirroring is performed on duplicate hard disks at the operating system level, so that if one of the disks becomes unavailable, the other disk can continue to service requests without interruptions. For example, you can mirror a datafile so that Oracle writes the same information to two different disk drives. You can then *break* the mirror to create a backup and later *resilver* the mirror.

When you mirror files, Oracle writes once while the operating system writes to multiple disks; when you *multiplex* files, Oracle writes the same data into multiple files.

See Also: breaking a mirror, resilvering a mirror

mounted database

An instance that is started and has the control files associated with the database open. You can mount a database without opening it; typically, you put the database in this state for maintenance or for restore and recovery operations.

See Also: instance

multi-level incremental backups

RMAN-generated incremental backups that allow you to conserve space by planning which blocks you want to back up and when. A level 0 incremental backup, which is the base for subsequent incremental backups, copies all blocks containing data. When you generate a level *n* incremental backup in which *n* is greater than 0, you back up either:

- All blocks that have changed since the most recent backup at level *n* or lower. This is the default type of incremental backup, called a *differential backup*.
- All blocks used since the most recent backup at level *n*-1 or lower. This type of backup is called a *cumulative backup*.

You can create a backup strategy in which you generate a backup at a different level each day, thereby controlling how much data you back up.

See Also: cumulative backup, differential backup, incremental backup

multiplexing

- **online redo logs**

  The automated maintenance of more than one identical copy of the online redo log. To multiplex the online logs, create multiple members in each redo log group. The degree of multiplexing is directly related to the number of members in each group.

- **control file**

  The automated maintenance of more than one identical copy of a database's control file. To multiplex the control file, create multiple entries in the CONTROL_FILES initialization parameter.

- **backup set**

  Datafile blocks included in the same RMAN backup set are multiplexed, that is, mixed together. Blocks from all datafiles in the backup set are interspersed with blocks from the other datafiles in the set.

- **archived redo logs**

The Oracle process ARC*n* is able to archive multiple copies of a redo log. You can multiplex archived redo logs by setting LOG_ARCHIVE_DEST_*n* (where *n* is an integer from 1 to 5, allowing up to four extra copies) or LOG_ARCHIVE_DUPLEX_DEST (allowing one extra copy) in your INIT.ORA file.

See Also: mirroring

multiple ARC*n* processing

Using multiple ARC*n* processes to archive online redo logs to one or more locations. Multiple ARC*n* processing prevents the bottleneck that occurs when LGWR writes to the online redo log faster than a single archive process can write to the archive destination(s). You can enable this feature at startup or at runtime by setting the initialization parameter LOG_ARCHIVE_MAX_PROCESS = *n*, where *n* is any integer from 1 to 10.

NOARCHIVELOG mode

The mode of the database in which Oracle does not require filled online redo logs to be archived to disk. Specify the mode at database creation or change it by using the ALTER DATABASE command. Oracle does not recommend running in NOARCHIVELOG mode because it severely limits the possibilities for recovery of lost data.

See Also: archived redo log, ARCHIVELOG mode

non-circular reuse records

Control file records containing critical information needed by RMAN. These records not change often and cannot be overwritten. Some examples of information in circular re-use records include:

- Datafiles
- Online redo logs
- Redo threads

See Also: circular reuse records

normal archiving transmission

The transmittal of archived redo log files to a local disk.

See Also: standby transmission

offline tablespace

A tablespace that is not available to users when the database is open. You can only take a tablespace offline while the database is open. If a tablespace is taken offline, all online datafiles contained in the tablespace are taken offline.

You can take a tablespace offline using the ALTER TABLESPACE OFFLINE statement with three different options:

- NORMAL

  All the files in the tablespace are checkpointed, then taken offline. This option is sometimes called *offline clean*. If any datafile belonging to the tablespace is not available, the tablespace cannot be taken offline normal. Datafiles in a tablespace taken offline cleanly do not need to be recovered before the tablespace is brought back online.

- TEMPORARY

  All datafiles in the tablespace that are accessible to Oracle are checkpointed, then taken offline. Files that were checkpointed by the OFFLINE TEMPORARY command do not need recovery. Datafiles that were not checkpointed because they were not accessible at the time of an OFFLINE IMMEDIATE command must be recovered before the tablespace is brought back online.

- IMMEDIATE

  All files in the tablespace are taken offline without any attempt to checkpoint the files first. All files in the tablespace must be recovered before the tablespace is brought online.

See Also: offline datafile

offline datafile

A datafile that is not available to users when the database is open. In exceptional circumstances, Oracle will automatically take a datafile offline if required. This file will need recovery before it can be brought online.

You can take a datafile offline either:

- As a consequence of an ALTER TABLESPACE OFFLINE operation.
- By issuing the command ALTER DATABASE DATAFILE *filename* OFFLINE. You must recover it before bringing it back online. You can issue this statement while the database is mounted or open.

See Also: offline tablespace

online datafile

A datafile that users can access. The database can be open or mounted when you issue the command ALTER DATABASE DATAFILE *filename* ONLINE. If the database is open, the datafile must be consistent with the rest of the database before you can bring it online. If the database is mounted, then you can bring the datafile online without being consistent with the other datafiles, but it will require recovery before the database is opened.

See Also: online tablespace

online redo log

The online redo log is a set of two or more files that record all changes made to Oracle datafiles and control files. Whenever a change is made to the database, Oracle generates a redo record in the redo buffer. The LGWR process flushes the contents of the redo buffer into the online redo log.

The c*urrent online redo log* is the one being written to by LGWR. When LGWR gets to the end of the file, is performs a *log switch* and begins writing to a new log file. If you run the database in ARCHIVELOG mode, then the ARC*n* process or processes copy the redo data into an *archived redo log*.

See Also: archived redo log

online tablespace

A tablespace that is available to users while the database is open. You can make a tablespace available for access by users by issuing the command ALTER TABLESPACE *tablespace_name* ONLINE. The database must be open to alter a tablespace online, and all files in the tablespace must be consistent with the rest of the database before the tablespace can be made online.

See Also: online datafile

open backup

A backup of one or more datafiles taken while a database is open. When you make an operating system backup while the database is open, you must put the tablespaces in hot backup mode by issuing an ALTER TABLESPACE BEGIN BACKUP command. When you make a backup using Recovery Manager while the database is open, however, you do not need to put the tablespaces in hot backup mode.

See Also: hot backup mode

open database

A database that is available to users to query and update. The database is opened either automatically through a STARTUP statement or explicitly through an ALTER DATABASE OPEN statement.

orphaned backups

Backups and copies that are unusable because they belong to incarnations of the database that are not direct ancestors of the current incarnation. For a visual depiction of orphaned backups, see "Reporting on Orphaned Backups" on page 1-27.

parallel recovery

A form of recovery in which several processes simultaneously apply changes from redo log files. Instance and media recovery can be parallelized automatically by specifying an initialization parameter or options to the SQL / SQL*Plus RECOVER command. Oracle uses one process to read the log files sequentially and dispatch redo information to several recovery processes, which apply the changes from the log files to the datafiles.

See Also: serial recovery

parallelization

Allocating multiple channels for Recovery Manager backup and recovery operations. You can parallelize:

- Backup set creation by allocating multiple channels before issuing a **backup** command.
- File copy creation by allocating multiple channels and including multiple files to be copied within a single **copy** command.
- Restore operations, with the degree of parallelism depending on the number of channels allocated as well as the distinct number of backup sets or file copies that must be read during the restore operation.
- Recovery operations when applying incremental backups, with the degree of parallelism depending on the number of channels allocated and also the distinct number of backup sets that are available to read from.

partial resync

See resynchronization

password files

A file created by the ORAPWD command. A database must use password files if you wish to connect as SYSDBA over a network. For a more comprehensive explanation, see the *Oracle8i Administrator's Guide.*

physical backups

Physical database files that have been copied from one place to another. The files can be datafiles, archived redo logs, or control files. You can make physical backups using Recovery Manager or with operating system commands such as the UNIX dd.

physical schema

The datafiles, tablespaces, redo threads, and redo logs that exist in a database at a given time. Issue the RMAN **report schema** command to obtain a list of tablespaces and datafiles.

A full resynchronization of the recovery catalog updates all changed RMAN metadata in the repository, including physical schema information. If the database is open, RMAN also gathers information about rollback segments. A partial resynchronization of the recovery catalog does not update physical schema or rollback information.

See Also: resynchronization

pluggable tablespace

See transportable tablespace

proxy copy

The functionality that enables a media manager to take over the transfer of data between the media storage device and disk during RMAN backup and restore operations.

See Also: media manager

read-only database

A database opened with the ALTER DATABASE OPEN READ ONLY command. As their name suggests, read-only databases are for queries only and cannot be modified. Oracle allows a standby database to be run in read-only mode, which means that it can be queried while still serving as an up-to-date emergency replacement for the primary database.

read-only tablespace

A tablespace whose status has been changed to prevent it from being updated. You put in read-only mode by executing the SQL statement ALTER TABLESPACE <tablespace> READ ONLY. Typically, you put a tablespace in read-only mode to reduce the frequency with which it is backed up. For example, instead of backing up the tablespace nightly, you reduce the backup frequency to once a month.

Note: The longer the duration between backups of a tablespace, the longer you will need to retain your backup media and the larger the risk of failed backup media (as you will have backed it up fewer times).

---

recover

(1) A Recovery Manager command that updates a restored datafile by the application of incremental backups (if they exist) and then by the application of archived or online redo logs.

(2) A SQL*Plus command that updates a restored file by the application of archived or online redo logs.

See Also: recovery

recovery

The application of redo data or incremental backups to database files in order to reconstruct lost changes. The three types of recovery are *instance recovery*, *crash recovery*, and *media recovery*. Oracle performs the first two types of recovery automatically using online redo records; only media recovery requires you to restore a backup and issue commands. Only Recovery Manager allows you to recover datafiles by applying incremental backups.

See Also: complete recovery, incomplete recovery, media recovery

recovery catalog

A set of Oracle tables and views used by Recovery Manager to store information about Oracle databases. Recovery Manager uses this data to manage the backup, restore, and recovery of Oracle databases. If you choose not to use a recovery catalog, RMAN uses the target database control file.

See Also: recovery catalog database

recovery catalog database

An Oracle database that contains a recovery catalog schema. You should not store the recovery catalog in your target database.

Recovery Manager

A utility that backs up, restores, and recovers Oracle databases. You can use it with or without the central information repository called a *recovery catalog*. If you do not use a recovery catalog, RMAN uses the database's control file to store information necessary for backup and recovery operations. You can use RMAN in conjunction with a media manager, which allows you to back up files to tertiary storage.

See Also: backup piece, backup set, copy, media manager, recovery catalog

recovery set

One or more tablespaces that are being recovered to an earlier point in time during TSPITR. After TSPITR, all database objects in the recovery set have been recovered to

See Also: auxiliary set, tablespace point-in-time recovery (TSPITR)

redo log

A file containing redo records. There are two types of redo logs: *online redo logs* and *archived redo logs*.

The online redo log is a set of two or more files that records all changes made to Oracle datafiles and control files. The LGWR process records the redo records in the log. The c*urrent online redo log* is the one LGWR is currently writing to.

The archived redo log, also known as the offline redo log, is a copy of the online redo log that has been copied to an offline destination. If the database is in ARCHIVELOG mode, the ARC*n* process or processes copy each online redo log to one or more archive log destinations after it is filled.

See Also: archived redo log, online redo log, redo record

redo log buffer

The memory buffer in the system global area (SGA) in which Oracle logs redo records. The background process LGWR flushes the buffers into the current online redo log.

See Also: redo record

redo log groups

Each online redo log belongs to a group. A group has one or more identical members. A multiplexed redo log is a redo log in which the redo groups have multiple members.

redo record

A group of change vectors describing a single, atomic change to the database. Oracle constructs redo records for all data block changes and saves them on disk in the current online redo log. Redo records allow changes to database blocks to be reconstructed should data loss occur.

See Also: redo log

registration

In RMAN, the execution of a **register database** command in order to record the existence of a target database in the recovery catalog.

repository

The collection of RMAN metadata about backup and recovery operations on the target database. Either the control file or the recovery catalog can function as the RMAN repository.

See Also: control file, recovery catalog

RESETLOGS option

A method for opening a database that results in a new database incarnation, the resetting of the log sequence number to 1, and the re-formatting or re-creation of the online redo logs. A database must be opened with the RESETLOGS keyword after:

- Incomplete recovery
- Recovery using a backup control file

resilvering a mirror

Informing the operating system or hardware managing the mirror that you want to refresh a broken mirror from the half that is up-to-date and then maintain both sides of the mirror.

See Also: breaking a mirror, mirroring

restore

The replacement of a lost or damaged file with a backup. You can restore files either with operating system commands such as UNIX cp or the RMAN **restore** command.

See Also: recover

resync

See resynchronization

resynchronization

The operation that updates the recovery catalog with current information from the target database control file. You can initiate a full resynchronization of the catalog by issuing a **resync catalog** command.

Partial resynchronizations transfer information to the recovery catalog about archived redo logs, backup sets and datafile copies. Partial resynchronizations will not transfer information such as:

- New datafiles
- New or removed tablespaces
- New or removed online log groups and members

If Recovery Manager determines that a full or partial resynchronization is necessary, it initiates one automatically before commands such as **backup**, **copy**, **restore**, and **recover**.

RMAN

See RMAN

rolling back

The use of rollback segments to undo uncommitted transactions applied to the database during the rolling forward stage of recovery.

See Also: recovery, rolling forward

rolling forward

The application of redo records or incremental backups to datafiles and control files in order to recover changes to those files.

See Also: recovery, rolling back

SBT

System Backup to Tape

See Also: media management interface

serial recovery

A form of recovery in which a single process applies the changes in the redo log files sequentially.

See Also: parallel recovery

SGA (System Global Area)

A group of shared memory structures that contain data and control information for one Oracle database instance. The SGA and Oracle processes constitute an Oracle instance. Oracle automatically allocates memory for an SGA whenever you start an instance and the operating system reclaims the memory when you shut down the instance. Each instance has one and only one SGA.

snapshot control file

A copy of a database's control file taken by Recovery Manager. RMAN uses the snapshot control file to read a consistent version of a control file when either resynchronizing the recovery catalog or backing up the control file. A snapshot control file is created by Recovery Manager using the same Oracle code that creates backup control files: ALTER DATABASE BACKUP CONTROL FILE TO '*location*'.

split block

See fractured block

staging

The process of restoring archived logs from tertiary storage to disk in order to allow recovery to proceed. RMAN stages the logs to disk when the **recover** command is executed. To use this feature, you must configure a media manager.

See Also: media manager

stand-alone commands

RMAN commands that you do not have to execute within the brackets of a **run** command.

See Also: job commands

standby database

An identical copy of a production database that you can use for disaster protection. You can update your standby database with archived redo logs from the production database in order to keep it current. Should a disaster destroy the production database, you can activate your standby database and make it the new production database.

standby transmission

The transmittal of archived redo log files via a network to either a local or remote standby database.

See Also: standby database

stored script

A sequence of RMAN commands stored in the recovery catalog.

See Also: recovery catalog

switch

A Recovery Manager command which converts a datafile copy into a datafile used by an Oracle database. It performs the equivalent function of the SQL statement ALTER DATABASE RENAME FILE '*original_name*' TO '*new_name*', and also marks the datafile copy as no longer available.

system change number (SCN)

A stamp that defines a committed version of a database at a point in time. Oracle assigns every committed transaction a unique SCN.

tablespace

A database is divided into one or more logical storage units called tablespaces. Each tablespace has one or more physical datafiles exclusively associated with it.

See Also: datafile

tablespace point-in-time recovery (TSPITR)

The recovery of one or more non-SYSTEM tablespaces to a point in time that is different from the database. You can use either RMAN or operating system methods to perform TSPITR.

tag

A user-specified character string that acts as a symbolic name for a backup set or image copy. You can specify a tag when executing the **restore** or **change** command. The maximum length of a tag is 30 characters.

tail of the log

The most recent redo record in the redo log file. As users make changes to the database, the tail keeps moving forward. Since the latest checkpoint is always temporally behind the tail of the log, it is said to *lag* the tail of the log. If the checkpoint lags the tail of the log significantly, recovery time increases.

tape streaming

Writing output to a tape drive fast enough to keep the tape constantly busy.

tape drive

A piece of hardware that reads and writes magnetic tapes.

tape silo

See ATL (automated tape library)

tape volume

One physical piece of tape media.

target database

In RMAN, the database that you are backing up or restoring.

thread

Each Oracle instance has its own set of online redo log groups. These groups are called a *thread* of online redo. In non-OPS environments, each database has only one thread that belongs to the instance accessing it. In OPS environments, each instance has a separate thread, that is, each instance has its own online redo log. Each thread has its own current log member.

time-based recovery

The incomplete recovery of database files to a non-current time. Time-based recovery is also known as *point-in-time recovery*. There are two types:

- Database point-in-time recovery (DBPITR), which is the incomplete recovery of all datafiles and control file to a time before the most recent time.

- Tablespace point-in-time recovery (TSPITR), which is the incomplete recovery of all datafiles in one or more tablespaces on an auxiliary database to a specific time before the most current time. The tablespace is then re-integrated into the original database.

See Also: incomplete recovery, media recovery, recovery, TSPITR

transportable tablespace

A feature that allows you to transport a set of tablespaces from one database to another. Transporting or "plugging" a tablespace into a database is like creating a tablespace with pre-loaded data. This feature is often an advantage because:

- It is faster than import/export or unload/load, since it involves only copying datafiles and integrating metadata.
- You can use it to move index data, allowing you to avoid rebuilding indexes.

TSPITR

See tablespace point-in-time recovery (TSPITR)

validation

A test that checks whether a backup set or copy can be restored. RMAN scans all of the copies or backup pieces in the specified backup sets and looks at the checksums to verify that the contents can be successfully restored.

Use the **restore ... validate** or **validate backupset** command when you suspect that one or more copies or backup pieces in a backup set are missing or have been damaged. Note that **restore ... validate** and **validate backupset** actually test whether the files can be restored, whereas **change ... crosscheck** and **crosscheck** merely examine the file headers.

See Also: crosscheck, media manager, recovery catalog

whole database backup

A backup of the control file and all datafiles that belong to a database.

See Also: backup

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W