Oracle® Database

Backup and Recovery Reference

11*g* Release 2 (11.2)

E10643-04

October 2010

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

ORACLE®

Oracle Database Backup and Recovery Reference, 11*g* Release 2 (11.2)

E10643-04

Copyright © 2003, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Lance Ashdown

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

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface ix

Audience ix
Documentation Accessibility ix
Related Documentation x
Conventions x

1 About RMAN Commands

Structure of RMAN Command Entries 1-1
RMAN Syntax Diagrams 1-2
Keywords in RMAN Syntax 1-2
Placeholders in RMAN Syntax 1-2
Quotes in RMAN Syntax 1-3
Format of RMAN Commands 1-3
RMAN Reserved Words 1-4
Summary of RMAN Commands 1-6
Summary of RMAN Subclauses 1-8

2 RMAN Commands: @ (at sign) to QUIT

@ (at sign) 2-2
@@ (double at sign) 2-4
ADVISE FAILURE 2-6
ALLOCATE CHANNEL 2-10
ALLOCATE CHANNEL FOR MAINTENANCE 2-13
ALTER DATABASE 2-17
BACKUP 2-19
CATALOG 2-52
CHANGE 2-57
CONFIGURE 2-65
CONNECT 2-86
CONVERT 2-89
CREATE CATALOG 2-102
CREATE SCRIPT 2-105
CROSSCHECK 2-109
DELETE 2-113
DELETE SCRIPT 2-119

DROP CATALOG 2-121
DROP DATABASE 2-123
DUPLICATE 2-124
EXECUTE SCRIPT 2-146
EXIT 2-149
FLASHBACK DATABASE 2-150
GRANT 2-156
HOST 2-159
IMPORT CATALOG 2-161
LIST 2-164
PRINT SCRIPT 2-185
QUIT 2-187

3 RMAN Commands: RECOVER to VALIDATE

RECOVER 3-2
REGISTER DATABASE 3-16
RELEASE CHANNEL 3-18
REPAIR FAILURE 3-21
REPLACE SCRIPT 3-25
REPORT 3-28
RESET DATABASE 3-35
RESTORE 3-37
RESYNC CATALOG 3-53
REVOKE 3-57
RMAN 3-59
RUN 3-65
SEND 3-68
SET 3-70
SHOW 3-82
SHUTDOWN 3-85
SPOOL 3-87
SQL 3-88
STARTUP 3-90
SWITCH 3-92
TRANSPORT TABLESPACE 3-96
UNREGISTER 3-102
UPGRADE CATALOG 3-105
VALIDATE 3-107

4 RMAN Subclauses

allocOperandList 4-2
archivelogRecordSpecifier 4-6
completedTimeSpec 4-11
connectStringSpec 4-13
datafileSpec 4-15
deviceSpecifier 4-16
fileNameConversionSpec 4-17

forDbUniqueNameOption 4-20
foreignlogRecordSpecifier 4-21
formatSpec 4-23
keepOption 4-26
listObjList 4-29
maintQualifier 4-32
maintSpec 4-34
obsOperandList 4-36
recordSpec 4-37
tempfileSpec 4-39
toDestSpec 4-40
Examples 4-41
untilClause 4-42

5 Recovery Catalog Views

Summary of RMAN Recovery Catalog Views 5-1
RC_ARCHIVED_LOG 5-4
RC_BACKUP_ARCHIVELOG_DETAILS 5-6
RC_BACKUP_ARCHIVELOG_SUMMARY 5-7
RC_BACKUP_CONTROLFILE 5-8
RC_BACKUP_CONTROLFILE_DETAILS 5-10
RC_BACKUP_CONTROLFILE_SUMMARY 5-11
RC_BACKUP_COPY_DETAILS 5-12
RC_BACKUP_COPY_SUMMARY 5-13
RC_BACKUP_CORRUPTION 5-14
RC_BACKUP_DATAFILE 5-15
RC_BACKUP_DATAFILE_DETAILS 5-17
RC_BACKUP_DATAFILE_SUMMARY 5-18
RC_BACKUP_FILES 5-19
RC_BACKUP_PIECE 5-22
RC_BACKUP_PIECE_DETAILS 5-24
RC_BACKUP_REDOLOG 5-26
RC_BACKUP_SET 5-28
RC_BACKUP_SET_DETAILS 5-30
RC_BACKUP_SET_SUMMARY 5-32
RC_BACKUP_SPFILE 5-33
RC_BACKUP_SPFILE_DETAILS 5-34
RC_BACKUP_SPFILE_SUMMARY 5-35
RC_CHECKPOINT 5-36
RC_CONTROLFILE_COPY 5-37
RC_COPY_CORRUPTION 5-39
RC_DATABASE 5-40
RC_DATABASE_BLOCK_CORRUPTION 5-41
RC_DATABASE_INCARNATION 5-42
RC_DATAFILE 5-43
RC_DATAFILE_COPY 5-45
RC_LOG_HISTORY 5-47

RC_OFFLINE_RANGE 5-48
RC_PROXY_ARCHIVEDLOG 5-49
RC_PROXY_ARCHIVELOG_DETAILS 5-51
RC_PROXY_ARCHIVELOG_SUMMARY 5-52
RC_PROXY_CONTROLFILE 5-53
RC_PROXY_COPY_DETAILS 5-55
RC_PROXY_COPY_SUMMARY 5-56
RC_PROXY_DATAFILE 5-57
RC_REDO_LOG 5-59
RC_REDO_THREAD 5-60
RC_RESTORE_POINT 5-61
RC_RESYNC 5-62
RC_RMAN_BACKUP_JOB_DETAILS 5-63
RC_RMAN_BACKUP_SUBJOB_DETAILS 5-65
RC_RMAN_BACKUP_TYPE 5-67
RC_RMAN_CONFIGURATION 5-68
RC_RMAN_OUTPUT 5-69
RC_RMAN_STATUS 5-70
RC_SITE 5-72
RC_STORED_SCRIPT 5-73
RC_STORED_SCRIPT_LINE 5-74
RC_TABLESPACE 5-75
RC_TEMPFILE 5-76
RC_UNUSABLE_BACKUPFILE_DETAILS 5-78

A Deprecated RMAN Syntax

B RMAN Compatibility

About RMAN Compatibility B-1
Determining the Recovery Catalog Schema Version B-1
RMAN Compatibility Matrix B-2
Cross-Version Compatibility of Recovery Catalog Exports and Imports B-3
RMAN Compatibility: Scenario B-3

C Oracle Secure Backup (OSB) Cloud Module

Backup On the Cloud C-1
Getting Started C-2
Hardware and Software Prerequisites C-2
Registering for An Oracle Technology Network (OTN) Account C-3
Signing Up For Amazon S3 - AWS Account C-3
Getting Your AWS Identifiers C-3
Installing the OSB Cloud Module Library C-3
Running the S3 Backup Installer C-5
Storing Configuration in RMAN Repository - *optional* C-6
Using the OSB Web Services Library and First Backup C-7
Securing OSB Cloud Module Backups C-7

Helpful Links C-7
Troubleshooting the OSB Cloud Module C-7

Index

Preface

This preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documentation
- Conventions

Audience

Backup and Recovery Reference is intended for database administrators who perform the following tasks:

- Use Recovery Manager (RMAN) to back up, restore, and recover Oracle databases
- Perform maintenance on RMAN backups of database files

To use this document, you must know the following:

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

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documentation

For more information, see these Oracle resources:

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

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

Conventions

The text in this reference adheres to the following conventions:

- `UPPERCASE monospace`

  Calls attention to RMAN keywords, SQL keywords, column headings in tables and views, and initialization parameters.

- `lowercase monospace`

  Calls attention to variable text in RMAN examples.

- *`italics`*

  Calls attention to RMAN or SQL placeholders, that is, text that should not be entered as-is but represents a value to be entered by the user.

> **See Also:** Chapter 1, "About RMAN Commands" for more information about RMAN conventions

1

About RMAN Commands

This chapter includes the following topics:

- Structure of RMAN Command Entries
- RMAN Syntax Diagrams
- Format of RMAN Commands
- RMAN Reserved Words
- Summary of RMAN Commands
- Summary of RMAN Subclauses

Structure of RMAN Command Entries

All RMAN commands in Chapter 2, "RMAN Commands: @ (at sign) to QUIT" are organized into the following sections:

Purpose This section provides a brief description of the command and the most common circumstances in which it is used.

Prerequisites This section lists the prerequisites for using the command.

Usage Notes This optional section provides general notes about using the command. Notes specific to a command option are included with the option description.

Syntax The syntax diagrams show the keywords and parameters that make up the statement.

> **Caution:** Not all keywords and parameters are valid in all circumstances. Be sure to refer to the "Semantics" section of each statement and clause to learn about any restrictions on the syntax.

Semantics This section describes the purpose of the keywords, parameters, and clauses that make up the syntax, as well as restrictions and other usage notes that may apply to them. The conventions for keywords and parameters used in this chapter are explained in "Conventions" on page x.

Examples This section shows how to use the various clauses and parameters of the statement.

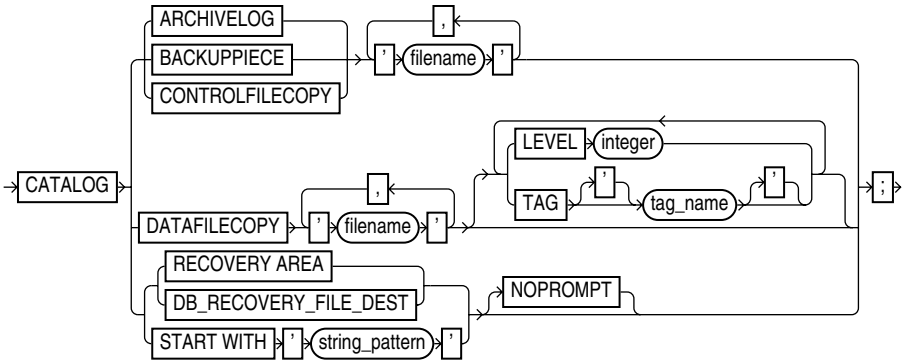
Optional sections following the examples provide more information on how and when to use the statement.

RMAN Syntax Diagrams

Syntax descriptions are provided in this book for RMAN command-line constructs in graphic form or Backus Naur Form (BNF). See *Oracle Database SQL Language Reference* for general information about how to interpret syntax diagrams and BNF notation. This section explains RMAN conventions exclusively.

Recovery Manager syntax diagrams use lines and arrows to show syntactic structure, as shown in the following example for the `CATALOG` command.

***catalog*::=**



This section describes the components of syntax diagrams and gives examples of how to write RMAN commands. Syntax diagrams are made up of these items:

- Keywords in RMAN Syntax
- Placeholders in RMAN Syntax
- Quotes in RMAN Syntax

Keywords in RMAN Syntax

Keywords have special meanings in Recovery Manager syntax. In the syntax diagrams, keywords appear in rectangular boxes and an uppercase font, like the word `CATALOG` in the example diagram. When used in text and code examples, RMAN keywords appear in uppercase, monospace font, for example, `CATALOG DATAFILECOPY`. You must use keywords in RMAN statements exactly as they appear in the syntax diagram, except that they can be either uppercase or lowercase.

Placeholders in RMAN Syntax

Placeholders in syntax diagrams indicate non-keywords. In the syntax diagrams, they appear in ovals, as in the word *integer* in the example diagram. When described in text, RMAN placeholders appear in lowercase italic, for example, *'filename'*. Placeholders are usually:

- Names of database objects (*tablespace_name*)
- Oracle data type names (*date_string*)
- Subclauses (*datafileSpec*)

When you see a placeholder in a syntax diagram, substitute an object or expression of the appropriate type in the RMAN statement. For example, to write a `DUPLICATE TARGET DATABASE TO` *'database_name'* command, use the name of the duplicate

database you want to create, such as `dupdb`, in place of the *database_name* placeholder in the diagram.

The only system-independent, valid environment variables in RMAN quoted strings are a question mark (`?`) for the Oracle home and an at-sign (`@`) for the SID. However, you can use operating system-specific environment variables on the target system within quoted strings. The environment variables are interpreted by the database server and not the RMAN client.

The following table shows placeholders that appear in the syntax diagrams and provides examples of the values you might substitute for them in your statements.

Placeholder	Description	Examples
Quoted strings such as *'filename'*, *'tablespace_name'*, *'channel_name'*, *'channel_parms'*	A string of characters contained in either single or double quotes. A quoted string may contain white space, punctuation, and RMAN and SQL keywords.	`"?/dbs/cf.f"` `'dev1'`
Nonquoted 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 only numeric characters.	`67843`

Quotes in RMAN Syntax

The RMAN syntax diagrams shows some placeholder values enclosed in required or optional quotes. The syntax diagrams show single quotes, though in all cases double quotes are also valid in RMAN syntax. For example, you may specify either *'filename'* or *"filename"*. For `SQL` commands, use double quotes, because the SQL statement may also contain quotes. Single and double quotes are not used interchangeably in SQL, and single quotes are the more common type of quote in a SQL statement.

Format of RMAN Commands

The RMAN language is free-form. Keywords must be separated by at least one white space character (such as a space, tab, or line break). An RMAN command starts with a keyword corresponding to one of the commands described in Chapter 2, "RMAN Commands: @ (at sign) to QUIT", followed by arguments and ending with a semicolon, as shown in the syntax diagrams. The following example shows an RMAN backup command:

```
BACKUP DATABASE;
```

A command can span multiple lines. For example, you can rewrite each keyword in the preceding command on a separate line as follows:

```
BACKUP
  DATABASE
;
```

You can insert a comment by using a pound (`#`) character at any point in a line. After the `#` character, the remainder of the line is ignored. For example:

```
# run this command once each day
```

```
BACKUP INCREMENTAL LEVEL 1
  FOR RECOVER OF COPY      # using incrementally updated backups
  WITH TAG "DAILY_BACKUP"  # daily backup routine
  DATABASE;
```

RMAN Reserved Words

The RMAN language contains a number of reserved words, which are or have been used in RMAN commands. In general, avoid using reserved words in ways that conflict with their primary meaning in the RMAN command language.

If you must use one of the reserved words as an argument to an RMAN command (for example, as a filename, tablespace name, tag name, and so on), then surround it with single or double quotes. Otherwise, RMAN cannot parse your command correctly and generates an error. Example 1–1 shows correct and incorrect usage of RMAN reserved words in RMAN commands.

Example 1–1 Using Reserved Words as Arguments to RMAN Commands

```
ALLOCATE CHANNEL backup DEVICE TYPE DISK;         # incorrect
ALLOCATE CHANNEL "backup" DEVICE TYPE DISK;       # correct
BACKUP DATABASE TAG full;                         # incorrect
BACKUP DATABASE TAG 'full';                       # correct
```

Table 1–1 lists all of the current reserved words.

Table 1–1 RMAN Reserved Words

A-C	C-D	D-L	L-P	P-S	S-W
,	CHANGE	DURATION	LOGSEQ	PARALLELMEDIARESTORE	SECTION
#	CHANNEL	ECHO	LOGS	PARALLEL	SEND
(	CHECKSYNTAX	ENCRYPTION	LOG	PARAMETER	SEQUENCE
)	CHECK	EXCLUDE	LOGSCN	PARAMETER_VALUE_CONVERT	SETLIMIT
\	CLEAR	EXECUTE	LOW	PARMS	SETSIZE
{	CLONENAME	EXIT	MAINTENANCE	PARTIAL	SET
}	CLONE	EXPIRED	MASK	PASSWORD	SHIPPED
<<<	CLONE_CF	EXPORT	MAXCORRUPT	PFILE	SHOW
>>>	CLOSED	FAILOVER	MAXDAYS	PIPE	SHUTDOWN
;	CMDFILE	FAILURE	MAXOPENFILES	PLATFORM	SINCE
&	COMMAND	FILESPERSET	MAXPIECESIZE	PLSQL	SIZE
_	COMMENT	FILES	MAXSEQ	PLUS	SKIP
'	COMPATIBLE	FILES	MAXSETSIZE	POINT	SLAXDEBUG
=	COMPLETED	FINAL	MAXSIZE	POLICY	SNAPSHOT
^	COMPRESSED	FLASHBACK	METHOD	POOL	SPFILE
@	COMPRESSION	FORCE	MINIMIZE	PREVIEW	SPOOL
.	CONFIGURE	FOREIGN	MISC	PRIMARY	SQL
:	CONNECT	FOREVER	MOUNT	PRINT	STANDBY
ABORT	CONSISTENT	FORMAT	MSGLOG	PRIORITY	STARTUP
ACCESSIBLE	CONTROLFILECOPY	FOR	MSGNO	PRIVILEGES	START
ACTIVE	CONTROLFILE	FROM	M	PROXY	STEP

Table 1–1 (Cont.) RMAN Reserved Words

A-C	C-D	D-L	L-P	P-S	S-W
ADVISEID	CONVERT	FULL	NAMES	PUT	SUMMARY
ADVISE	COPIES	GET	NAME	QUIT	SWITCH
AES128	COPY	GLOBAL	NEED	RATE	TABLESPACES
AES192	CORRUPTION	HIGH	NEW-LINE	RCVCAT	TABLESPACE
AES256	CREATE	GRANT	NEWNAME	RCVMAN	TAG
AFFINITY	CRITICAL	GROUP	NEW	READONLY	TARGET
AFTER	CROSSCHECK	GUARANTEE	NOCATALOG	READRATE	TDES168
ALGORITHM	CUMULATIVE	G	NOCFAU	RECALL	TEMPFILE
ALLOCATE	CURRENT	HEADER	NOCHECKSUM	RECOVERABLE	TEST
ALLOW	DATABASE	HIGH	NODEVALS	RECOVERY	THREAD
ALL	DATAFILECOPY	HOST	NODUPLICATES	RECOVER	TIMEOUT
ALTER	DATAFILE	IDENTIFIED	NOEXCLUDE	REDUNDANCY	TIMES
AND	DATAPUMP	IDENTIFIER	NOFILENAMECHECK	REGISTER	TIME
APPEND	DAYS	ID	NOFILEUPDATE	RELEASE	TO
APPLIED	DBA	IMMEDIATE	NOKEEP	RELOAD	TRACE
ARCHIVELOG	DBID	IMPORT	NOLOGS	REMOVE	TRANSACTIONAL
AREA	DB_FILE_NAME_ CONVERT	INACCESSIBLE	NOMOUNT	RENORMALIZE	TRANSPORT
AS	DB_NAME	INCARNATION	NONE	REPAIRID	TYPE
AT	DB_RECOVERY_FILE_ DEST	INCLUDE	NOPARALLEL	REPAIR	UNAVAILABLE
ATALL	DB_UNIQUE_NAME	INCLUDING	NOPROMPT	REPLACE	UNCATALOG
AUTOBACKUP	DEBUG	INCREMENTAL	NOREDO	REPLICATE	UNDO
AUTOLOCATE	DECRYPTION	INPUT	NORMAL	REPORT	UNLIMITED
AUXILIARY	DEFAULT	INSTANCE	NOT	RESETLOGS	UNRECOVERABLE
AUXNAME	DEFINE	IO	NO	RESET	UNREGISTER
AVAILABLE	DELETE	JOB	NULL	RESTART	UNTIL
BACKED	DELETION	KBYTES	OBSOLETE	RESTORE	UPGRADE
BACKUPPIECE	DESTINATION	KEEP	OFFLINE	RESTRICTED	UP
BACKUPSET	DETAIL	KRB	OFF	RESYNC	USING
BACKUPS	DEVICE	K	OF	RETENTION	VALIDATE
BACKUP	DIRECTORY	LEVEL	ONLY	REUSE	VERBOSE
BEFORE	DISKRATIO	LIBPARM	ON	REVOKE	VIRTUAL
BETWEEN	DISK	LIBRARY	OPEN	RPCTEST	WINDOW
BLOCKRECOVER	DISPLAY	LIKE	OPTIMIZATION	RPC	WITH
BLOCKS	DORECOVER	LIMIT	OPTION	RUN	
BLOCK	DROP	LIST	ORPHAN	SAVE	
BY	DUMP	LOAD	OR	SCHEMA	
CANCEL	DUPLEX	LOGFILE	PACKAGES	SCN	
CATALOG	DUPLICATE	LOGICAL	PARALLELISM	SCRIPT	

Summary of RMAN Commands

Table 1–2 provides a functional summary of RMAN commands that you can execute at the RMAN prompt, within a RUN command, or both. All commands from previous RMAN releases work with the current release, although some commands and options are now deprecated (see Appendix A, "Deprecated RMAN Syntax"). For command-line options for the RMAN client, refer to RMAN on page 3-59.

Table 1–2 Recovery Manager Commands

Command	Purpose
@ (at sign) on page 2-2	Run a command file.
@@ (double at sign) on page 2-4	Run a command file in the same directory as another command file that is currently running. The @@ command differs from the @ command only when run from within a command file.
ADVISE FAILURE on page 2-6	Display repair options.
ALLOCATE CHANNEL on page 2-10	Establish a channel, which is a connection between RMAN and a database instance.
ALLOCATE CHANNEL FOR MAINTENANCE on page 2-13	Allocate a channel in preparation for issuing maintenance commands such as DELETE.
ALTER DATABASE on page 2-17	Mount or open a database.
BACKUP on page 2-19	Back up database files, copies of database files, archived logs, or backup sets.
CATALOG on page 2-52	Add information about file copies and user-managed backups to the repository.
CHANGE on page 2-57	Mark a backup piece, image copy, or archived redo log as having the status UNAVAILABLE or AVAILABLE; remove the repository record for a backup or copy; override the retention policy for a backup or copy; update the recovery catalog with the DB_UNIQUE_NAME for the target database.
CONFIGURE on page 2-65	Configure persistent RMAN settings. These settings apply to all RMAN sessions until explicitly changed or disabled.
CONNECT on page 2-86	Establish a connection between RMAN and a target, auxiliary, or recovery catalog database.
CONVERT on page 2-89	Convert datafile formats for transporting tablespaces and databases across platforms.
CREATE CATALOG on page 2-102	Create the schema for the recovery catalog.
CREATE SCRIPT on page 2-105	Create a stored script and store it in the recovery catalog.
CROSSCHECK on page 2-109	Determine whether files managed by RMAN, such as archived logs, datafile copies, and backup pieces, still exist on disk or tape.
DELETE on page 2-113	Delete backups and copies, remove references to them from the recovery catalog, and update their control file records to status DELETED.
DELETE SCRIPT on page 2-119	Delete a stored script from the recovery catalog.
DROP CATALOG on page 2-121	Remove the schema from the recovery catalog.
DROP DATABASE on page 2-123	Delete the target database from disk and unregisters it.
DUPLICATE on page 2-124	Use backups of the target database to create a duplicate database that you can use for testing purposes or to create a standby database.
EXECUTE SCRIPT on page 2-146	Run an RMAN stored script.
EXIT on page 2-149	Quit the RMAN executable.

Table 1–2 (Cont.) Recovery Manager Commands

Command	Purpose
FLASHBACK DATABASE on page 2-150	Return the database to its state at a previous time or SCN.
GRANT on page 2-156	Grant privileges to a recovery catalog user.
HOST on page 2-159	Invoke an operating system command-line subshell from within RMAN or run a specific operating system command.
IMPORT CATALOG on page 2-161	Imports the metadata from one recovery catalog into a different recovery catalog.
LIST on page 2-164	Produce a detailed listing of backup sets or copies.
PRINT SCRIPT on page 2-185	Display a stored script.
QUIT on page 2-187	Exit the RMAN executable.
RECOVER on page 3-2	Apply redo log files and incremental backups to datafiles or data blocks restored from backup or datafile copies, to update them to a specified time.
REGISTER DATABASE on page 3-16	Register the target database in the recovery catalog.
RELEASE CHANNEL on page 3-18	Release a channel that was allocated with an ALLOCATE CHANNEL command or ALLOCATE CHANNEL FOR MAINTENANCE command.
REPAIR FAILURE on page 3-21	Repair one or more failures recorded in the automated diagnostic repository.
REPLACE SCRIPT on page 3-25	Replace an existing script stored in the recovery catalog. If the script does not exist, then REPLACE SCRIPT creates it.
REPORT on page 3-28	Perform detailed analyses of the content of the recovery catalog.
RESET DATABASE on page 3-35	Inform RMAN that the SQL statement ALTER DATABASE OPEN RESETLOGS has been executed and that a new incarnation of the target database has been created, or reset the target database to a prior incarnation.
RESTORE on page 3-37	Restore files from backup sets or from disk copies to the default or a new location.
RESYNC CATALOG on page 3-53	Perform a full resynchronization, which creates a snapshot control file and then copies any new or changed information from that snapshot control file to the recovery catalog.
REVOKE on page 3-57	Revoke privileges from a recovery catalog user.
RMAN on page 3-59	Start RMAN from the operating system command line.
RUN on page 3-65	Execute a sequence of one or more RMAN commands, which are one or more statements executed within the braces of RUN.
SEND on page 3-68	Send a vendor-specific quoted string to one or more specific channels.
SET on page 3-70	Set the value of various attributes that affect RMAN behavior for the duration of a RUN block or a session.
SHOW on page 3-82	Display the current CONFIGURE settings.
SHUTDOWN on page 3-85	Shut down the target database. This command is equivalent to the SQL*Plus SHUTDOWN command.
SPOOL on page 3-87	Write RMAN output to a log file.
SQL on page 3-88	Execute a SQL statement from within Recovery Manager.
STARTUP on page 3-90	Start up the target database. This command is equivalent to the SQL*Plus STARTUP command.

Table 1–2 (Cont.) Recovery Manager Commands

Command	Purpose
SWITCH on page 3-92	Specify that a datafile copy is now the **current datafile**, that is, the datafile pointed to by the control file. This command is equivalent to the SQL statement ALTER DATABASE RENAME FILE as it applies to datafiles.
TRANSPORT TABLESPACE on page 3-96	Create transportable tablespace sets from backup for one or more tablespaces.
UNREGISTER on page 3-102	Unregister a database from the recovery catalog.
UPGRADE CATALOG on page 3-105	Upgrade the recovery catalog schema from an older version to the version required by the RMAN executable.
VALIDATE on page 3-107	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.

Summary of RMAN Subclauses

Subclauses that are used in multiple commands are documented in a separate chapter to avoid unnecessary duplication. The descriptions of commands that use these subclauses include a cross-reference to the subclause entry in Chapter 4, "RMAN Subclauses". Table 1–3 summarizes the RMAN subclauses.

Table 1–3 Recovery Manager Subclauses

Subclause	Specifies . . .
allocOperandList::= on page 4-2	Channel control options such as PARMS and FORMAT
archivelogRecordSpecifier on page 4-6	A range of archived redo log files
completedTimeSpec on page 4-11	A time range during which the backup or copy completed
connectStringSpec on page 4-13	The user name, 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
datafileSpec on page 4-15	A datafile by filename or absolute file number
deviceSpecifier on page 4-16	The type of storage device for a backup or copy
fileNameConversionSpec on page 4-17	Patterns to transform source to target filenames during BACKUP AS COPY, CONVERT and DUPLICATE
forDbUniqueNameOption on page 4-20	All databases in a Data Guard environment or a database with the specified DB_UNIQUE_NAME
foreignlogRecordSpecifier on page 4-21	A range of foreign archived redo log file
formatSpec on page 4-23	A filename format for a backup or copy
keepOption on page 4-26	A backup or copy should or should not be exempt from the current retention policy
listObjList on page 4-29	Items that will be displayed by the LIST command
maintQualifier on page 4-32	Additional options for maintenance commands such as DELETE and CHANGE
maintSpec on page 4-34	Files operated on by maintenance commands such as CHANGE, CROSSCHECK, and DELETE
obsOperandList on page 4-36	Backups that are obsolete according to specified criteria

Table 1–3 (Cont.) Recovery Manager Subclauses

Subclause	Specifies . . .
recordSpec on page 4-37	Objects that the maintenance commands should operate on
tempfileSpec on page 4-39	A tempfile by path or by file number
untilClause on page 4-42	An upper limit by time, SCN, or log sequence number. This clause is usually used to specify the desired point in time for an incomplete recovery

2

RMAN Commands: @ (at sign) to QUIT

This chapter describes RMAN commands in alphabetical order. For a summary of the RMAN commands and command-line options, refer to "Summary of RMAN Commands" on page 1-6.

@ (at sign)

Purpose

Use the @ command to execute a series of RMAN commands stored in an operating system file with the specified path name.

> **Note:** The file must contain complete RMAN commands. Partial commands generate syntax errors.

Prerequisites

The command file must contain complete RMAN commands.

If you use the @ command within a RUN command, then the @ command must be on its own line (see Example 2–2 on page 2-3).

Usage Notes

RMAN processes the file as if its contents had appeared in place of the @ command. As shown in Example 2–3 on page 2-3, you can specify substitution variables in a command file and then pass values to the command file during execution.

> **See Also:** RMAN on page 3-59 to learn more about using substitution variables in RMAN

Syntax

@::=

Semantics

Syntax Element	Description
filename	Specifies the name of a command file, for example, `@/oracle/dbs/cmd/cmd1.rman`. If you do not specify the absolute path name, then the current working directory is assumed, for example, `@cmd1.rman`. Any file extension (or no file extension) is valid. Do not use quotes around the string or leave whitespace between the @ keyword and the file name.

Examples

Example 2–1 Running a Command File from the Operating System Command Line

This example creates an RMAN command file and then executes it from the operating system command line.

```
% echo "BACKUP DATABASE;" > backup_db.rman
% rman TARGET / @backup_db.rman
```

Example 2–2 Running a Command File Within RMAN

This example shows how you can execute a command file from the RMAN prompt and from within a `RUN` command. User-entered text appears in bold.

```
RMAN> @backup_db.rman
RMAN> RUN {
2> @backup_db.rman
3> backup database;
4> **end-of-file**
5> }
```

Example 2–3 Specifying Substitution Variables

Suppose that you use a text editor to create command file `whole_db.rman` with the following contents:

```
# name: whole_db.rman
BACKUP TAG &1 COPIES &2 DATABASE;
EXIT;
```

The following example starts RMAN from the operating system prompt and connects to the target database. The example then runs the `@` command, passing variables to the command file to create two database backups with tag `Q106`:

```
% rman TARGET /
RMAN> @/tmp/whole_db.rman Q106 2
```

@@ (double at sign)

Purpose

Use the `@@` command to execute a series of RMAN commands stored in an operating system file with the specified filename.

If `@@` is contained in a command file, then `@@`*`filename`* directs RMAN to look for the specified filename in the same directory as the command file from which it was called. If not used within a command file, the `@@` command is identical to the `@ (at sign)` command.

Prerequisites

The command file must contain complete RMAN commands.

Usage Notes

The command file is local to the RMAN client. The name resolution of the file is dependent on the operating system. For example, `@tmp/cmd1.rman` in UNIX or Windows means that `tmp` is a subdirectory of the current directory and that the file `cmd1.rman` is in this subdirectory.

To illustrate the differences between the `@` and `@@` commands, assume that you invoke RMAN as follows:

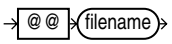
```
% rman TARGET /
RMAN> @/tmp/cmd1.rman
```

Assume that the command `@@cmd2.rman` appears inside the `cmd1.rman` script. In this case, the `@@` command directs RMAN to search for the file `cmd2.rman` in the directory `/tmp`.

As with the `@` command, you can specify substitution variables in a command file and then pass values to the command file during execution of `@@` (see Example 2–4 on page 2-4).

Syntax

@@::=



Semantics

Syntax Element	Description
filename	Specifies the name of a command file, for example, `@@cmd2.rman`.

Example

Example 2–4 Calling a Command File Within Another Command File

The following operating system commands create command files `backup_logs.rman` and `backup_db.rman`:

```
% echo "BACKUP ARCHIVELOG ALL;" > /tmp/bkup_logs.rman
```

```
% echo "BACKUP TAG &1 DATABASE;" > /tmp/bkup_db.rman
% echo "@@bkup_logs.rman" >> /tmp/bkup_db.rman
```

The following example starts RMAN from the command line and connects to the target database with operating system authentication. The `@` command executes `bkup_db.rman`, which contains the command `@@bkup_logs.rman`. The `@@` command specifies that RMAN should look for the `bkup_logs.rman` script in the same directory in which `bkup_db.rman` is located. Note that the example uses a substitution variable to specify the tag `WHOLE_DB` for the database backup.

```
% rman TARGET /
RMAN> @/tmp/bkup_db.rman whole_db
```

ADVISE FAILURE

Purpose

Use the ADVISE FAILURE command to display repair options for the specified failures. This command prints a summary of the failures identified by the Data Recovery Advisor and implicitly closes all open failures that are already fixed.

The recommended workflow is to run the following commands in an RMAN session: LIST FAILURE to display failures, ADVISE FAILURE to display repair options, and REPAIR FAILURE to fix the failures.

Prerequisites

RMAN must be connected to a target database. See the CONNECT and RMAN commands to learn how to connect to a database as TARGET.

The target database instance must be started. The target database must be a single-instance database and must not be a physical standby database, although it can be a logical standby database.

In the current release, Data Recovery Advisor only supports single-instance databases. Oracle Real Application Clusters (Oracle RAC) databases are not supported.

Usage Notes

Data Recovery Advisor verifies repair feasibility before proposing a repair strategy. For example, Data Recovery Advisor checks that all backups and archived redo log files needed for media recovery are available. The ADVISE FAILURE output indicates the repair strategy that Data Recovery Advisor considers optimal for a given set of failures. The ADVISE FAILURE command can generate both manual and automated repair options.

Manual Repair Options

Manual repair options are either mandatory or optional. The optional actions may fix the failures more quickly or easily than automated repairs. For example, Data Recovery Advisor may recommend a failover to a standby database as an alternative to a primary database repair.

In other cases, the only options are manual because automated repairs are not feasible. For example, I/O failures often cannot be repaired automatically. Also, it is sometimes impossible to diagnose a failure because insufficient data is returned by the operating system or the disk subsystem.

Automated Repair Options

Each automated repair option is either a single repair or a set of repair steps (see Table 2–1, " Automated Repair Options" for a description of command output). When a repair option has a script that contains multiple repair steps, ADVISE FAILURE generates the script so that the repair steps are in the correct order. A single repair always fixes critical failures together. You must repair critical failures, but you can also repair noncritical failures at the same time. You can repair noncritical failures in a random order, one by one, or in groups.

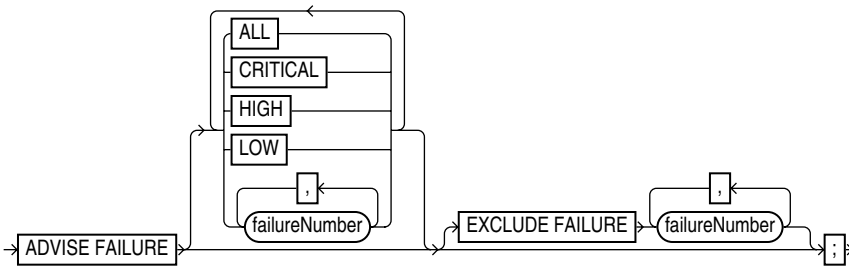
Oracle RAC and Data Recovery Advisor

If a data failure brings down all instances of an Oracle RAC database, then you can mount the database in single-instance mode and use Data Recovery Advisor to detect

and repair control file, SYSTEM datafile, and dictionary failures. You can also initiate health checks to test other database components for data failures. This approach will not detect data failures that are local to other cluster instances, for example, an inaccessible datafile.

Syntax

***advise*::=**



Semantics

advise

Syntax Element	Description
ADVISE FAILURE	Displays information for all CRITICAL and HIGH priority failures recorded in the automatic diagnostic repository. You can only use ADVISE FAILURE with no options when a LIST FAILURE command was previously executed in the current session. **Note:** If a new failure has been recorded in the diagnostic repository since the last LIST FAILURE command in the current RMAN session, then RMAN issues a warning before advising on CRITICAL and HIGH failures.
ALL	Lists options that repair all open failures together.
CRITICAL	Lists options that repair only critical failures.
HIGH	Lists options that repair only failures with HIGH priority.
LOW	Lists options that repair only failures with LOW priority.
failureNumber	Lists options that repair only the specified failures.
EXCLUDE FAILURE *failureNumber*	Excludes the specified failures from the list.

ADVISE FAILURE Command Output

The ADVISE FAILURE output includes the LIST FAILURE output, which is described in Table 2–27, " List of Failures". See Example 2–5 on page 2-8 for sample output.

RMAN presents mandatory and optional manual actions in an unordered list. If manual options exist, then they appear before automated options. Table 2–1 describes the output for automated repair options.

Table 2–1 Automated Repair Options

Column	Indicates
Option	The identifier for the automated repair option.

Table 2–1 (Cont.) Automated Repair Options

Column	Indicates
`Strategy`	A strategy to fix the failure with the `REPAIR FAILURE` command. The Data Recovery Advisor always presents an automated repair option with no data loss when possible. Automated repair options fall into the following basic categories: ▪ Repair with no data loss ▪ Repair with data loss, for example, Flashback Database **Note:** The `ADVISE` command maps a set of failures to the set of repair steps that Data Recovery Advisor considers to be optimal. When possible, Data Recovery Advisor consolidates multiple repair steps into a single repair. For example, if the database has corrupted datafile, missing control file, and lost current redo log group, then Data Recovery Advisor would recommend a single, consolidated repair plan to restore the database and perform point-in-time recovery.
`Repair Description`	A description of the proposed repair. For example, the proposed repair could be to restore and recover datafile 17.
`Repair Script`	The location of an editable script with all repair actions and comments. If you do not choose an automated repair, then you can review this script and edit it for use in a manual recovery strategy.

Examples

Example 2–5 Displaying Repair Options for All Failures

This example shows repair options for all failures known to the Recovery Data Advisor. The example indicates two failures: missing datafiles and a datafile with corrupt blocks.

```
RMAN> LIST FAILURE;

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

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

RMAN> ADVISE FAILURE;

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

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

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

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

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

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

ALLOCATE CHANNEL

Purpose

`ALLOCATE CHANNEL` manually allocates a **channel** (which is a connection between RMAN and a database instance). The `ALLOCATE CHANNEL` command must be issued within a `REPAIR FAILURE` block. It allocates a channel only in the block where the command is issued.

Prerequisites

The target instance must be started.

Usage Notes

Manually allocated channels are distinct from automatically allocated channels specified with `CONFIGURE`. Automatic channels apply to any RMAN job in which you do *not* manually allocate channels. You can override automatic channel configurations by manually allocating channels within a `RUN` command, but you cannot use `BACKUP DEVICE TYPE` or `RESTORE DEVICE TYPE` to use automatic channels after specifying manual channels with `ALLOCATE CHANNEL`.

Multiple Channels

You can allocate up to 255 channels; each channel can read up to 64 files in parallel. You can control the degree of parallelism within a job by the number of channels that you allocate. Allocating multiple channels simultaneously allows a single job to read or write multiple backup sets or disk copies in parallel, which each channel operating on a separate backup set or copy.

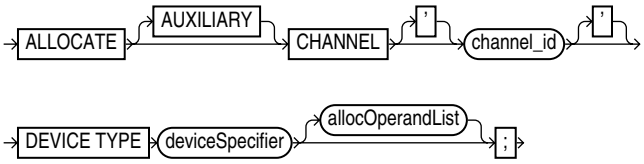
When making backups to disk, the guideline is to allocate one channel for each output device (see Example 2–7 on page 2-11). If RMAN is writing to a striped file system or an ASM disk group, however, then multiple channels can improve performance. When backing up to tape, the guideline is that the number of tape channels should equal the number of tape devices divided by the number of duplexed copies (see Example 2–8 on page 2-12).

Channels in an Oracle RAC Environment

As long as the `SYSDBA` password for all Oracle RAC instances is the same, then you never need to put passwords in the `CONNECT` option of the `ALLOCATE` or `CONFIGURE` command. If you use a connect string of the form *`user@database`*, then RMAN automatically uses the same password that was used for the `TARGET` connection when the RMAN session was started.

Syntax

***allocate*::=**



(*deviceSpecifier::=* on page 4-16, *allocOperandList::=* on page 4-2)

Semantics

Syntax Element	Description
AUXILIARY	Specifies a connection between RMAN and an auxiliary database instance. An auxiliary instance is used when executing the DUPLICATE or TRANSPORT TABLESPACE command, and when performing TSPITR with RECOVER TABLESPACE (see Example 2–9 on page 2-12). When specifying this option, the auxiliary instance must be started but not mounted. **See Also:** DUPLICATE on page 2-124 to learn how to duplicate a database, and CONNECT on page 2-86 to learn how to connect to a duplicate database instance
CHANNEL *channel_id*	Specifies a connection between RMAN and the target database instance. The *channel_id* is the case-sensitive name of the channel. The database uses the *channel_id* to report I/O errors. Each connection initiates an database server session on the target or auxiliary instance: this session performs the work of backing up, restoring, or recovering RMAN backups. You cannot make a connection to a shared server session. Whether ALLOCATE CHANNEL allocates operating system resources immediately depends on the 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. Each channel operates on one backup set or image copy at a time. RMAN automatically releases the channel at the end of the job. **Note:** You cannot prefix ORA_ to a channel name. RMAN reserves channel names beginning with the ORA_ prefix for its own use.
DEVICE TYPE *deviceSpecifier*	Specifies the type of storage for a backup. Query the V$BACKUP_DEVICE view for information about available device types and names. **Note:** When you specify DEVICE TYPE DISK, no operating system resources are allocated other than for the creation of the server session. **See Also:** *deviceSpecifier* on page 4-16
allocOperandList	Specifies control options for the allocated channel. Note that the channel parameters for sequential I/O devices are platform-specific (see Example 2–6 on page 2-11). **See Also:** *allocOperandList* on page 4-2

Examples

Example 2–6 Manually Allocating a Channel for a Backup

This example allocates a single tape channel for a whole database and archived redo log backup. The PARMS parameter specifies the Oracle Secure Backup media family named wholedb_mf.

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS 'ENV=(OB_MEDIA_FAMILY=wholedb_mf)';
  BACKUP DATABASE;
  BACKUP ARCHIVELOG ALL NOT BACKED UP;
}
```

Example 2–7 Distributing a Backup Across Multiple Disks

When backing up to disk, you can spread the backup across several disk drives. Allocate one DEVICE TYPE DISK channel for each disk drive and specify the format string so that the output files are on different disks.

```
RUN
{
  ALLOCATE CHANNEL disk1 DEVICE TYPE DISK FORMAT '/disk1/%U';
  ALLOCATE CHANNEL disk2 DEVICE TYPE DISK FORMAT '/disk2/%U';
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

Example 2–8 Creating Multiple Copies of a Backup on Tape

In this example, four tape drives are available for writing: `stape1`, `stape2`, `stape3`, and `stape4`. You use the `SET BACKUP COPIES` command to indicate that RMAN should create two identical copies of the database backup. Because the guideline is that the number of tape channels should equal the number of tape devices divided by the number of duplexed copies, you allocate two channels. Note that in this case the `BACKUP_TAPE_IO_SLAVES` initialization parameter must be set to `TRUE`.

In the `OB_DEVICE_`*n* parameter for Oracle Secure Backup, the *n* specifies the copy number of the backup piece. RMAN writes copy 1 of each backup piece to tape drives `stape1` and `stape2` and writes copy 2 of each backup piece to drives `stape3` and `stape4`. Thus, each copy of the database backup is distributed between two tape drives, so that part of the data is on each drive.

```
RUN
{
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt
    PARMS 'ENV=(OB_DEVICE_1=stape1,OB_DEVICE_2=stape3)';
  ALLOCATE CHANNEL t2 DEVICE TYPE sbt
    PARMS 'ENV=(OB_DEVICE_1=stape2,OB_DEVICE_2=stape4)';
  SET BACKUP COPIES 2;
  BACKUP DATABASE;
}
```

Example 2–9 Allocating an Auxiliary Channel for Database Duplication

This example creates a duplicate database from backups. RMAN can use configured channels for duplication even if they do not specify the `AUXILIARY` option. In this example, no SBT channel is preconfigured, so an auxiliary SBT channel is manually allocated.

```
RUN
{
  ALLOCATE AUXILIARY CHANNEL c1 DEVICE TYPE sbt;
  DUPLICATE TARGET DATABASE
    TO dupdb
  DB_FILE_NAME_CONVERT '/disk2/dbs/','/disk1/'
  SPFILE
    PARAMETER_VALUE_CONVERT   '/disk2/dbs/',
                              '/disk1/'
    SET LOG_FILE_NAME_CONVERT '/disk2/dbs/',
                              '/disk1/';
}
```

> **See Also:** *Oracle Database Reference* for more information on the parameter `DB_FILE_NAME_CONVERT`

ALLOCATE CHANNEL FOR MAINTENANCE

Purpose

Use the `ALLOCATE CHANNEL FOR MAINTENANCE` command to manually allocate a channel in preparation for issuing a `CHANGE`, `DELETE`, or `CROSSCHECK` command. You can use the `RELEASE CHANNEL` command to unallocate the channel.

> **Note:** If you `CONFIGURE` at least one channel for each device type in your configuration, then you do not need to use `ALLOCATE CHANNEL FOR MAINTENANCE`. It is recommended that you use configured channels instead of maintenance channels. You can use configured channels for all RMAN I/O to the specified device, not just the maintenance tasks supported by maintenance channels. The configured channels persist across RMAN sessions.

Prerequisites

Execute this command only at the RMAN prompt, not within a `RUN` block. The target instance must be started. You cannot allocate a maintenance channel to a shared session.

Usage Notes

As a rule, you should allocate one maintenance channel for each device. Manually allocated channels and automatic channels are never mixed. In general, you should allocate multiple maintenance channels for a single job only in these situations:

- To enable crosschecking or deletion of all backup pieces both on disk and tape, with a single command (see Example 2–11 on page 2-15)
- To make crosschecking and deleting work correctly in an Oracle RAC configuration in which each backup piece exists only on one node (see Example 2–12 on page 2-16)

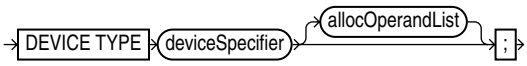
RMAN uses the following convention for naming of maintenance channels: `ORA_MAINT_`*`devicetype`*`_`*`n`*, where *`devicetype`* refers to `DISK` or `sbt` and *`n`* refers to the channel number. For example, RMAN uses these names for two manually allocated disk channels:

```
ORA_MAINT_DISK_1
ORA_MAINT_DISK_2
```

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to crosscheck and delete on multiple channels

Syntax

***allocateForMaint*::=**



(*deviceSpecifier::=* on page 4-16, *allocOperandList::=* on page 4-2)

Semantics

allocateForMaint

Syntax Element	Description
DEVICE TYPE *deviceSpecifier*	Specifies the type of storage for a backup. Query the V$BACKUP_DEVICE view for information about available device types and names. **See Also:** *deviceSpecifier* on page 4-16
allocOperandList	Specifies control options for the allocated channel. Note that the channel parameters for sequential I/O devices are platform-specific. **See Also:** *allocOperandList* on page 4-2

Examples

Example 2–10 Deleting Backup Sets

Assume that you want to recycle a set of tapes by deleting all RMAN backups. In this example, only a disk channel is configured by default. The example manually allocates an SBT channel, deletes all backups from tape, and then releases the channel.

```
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;

allocated channel: ORA_MAINT_SBT_TAPE_1
channel ORA_MAINT_SBT_TAPE_1: SID=135 device type=SBT_TAPE
channel ORA_MAINT_SBT_TAPE_1: Oracle Secure Backup

RMAN> DELETE NOPROMPT BACKUP;

List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
------- ------- --- --- ----------- ----------- ----------
9957    9954    1   1   AVAILABLE   SBT_TAPE    8oic41ad_1_1
9974    9972    1   1   AVAILABLE   SBT_TAPE    c-28014364-20070308-17
10024   10021   1   1   AVAILABLE   SBT_TAPE    8qic41c3_1_1
10045   10042   1   1   AVAILABLE   SBT_TAPE    c-28014364-20070308-18
10446   10443   1   1   AVAILABLE   SBT_TAPE    8uic47fg_1_1
10487   10482   1   1   AVAILABLE   SBT_TAPE    90ic47ih_1_1
10488   10483   1   1   AVAILABLE   SBT_TAPE    91ic47j1_1_1
10524   10514   1   1   AVAILABLE   SBT_TAPE    92ic47q4_1_1
10540   10538   1   1   AVAILABLE   SBT_TAPE    c-28014364-20070308-1a
deleted backup piece
backup piece handle=8oic41ad_1_1 RECID=198 STAMP=616695118
deleted backup piece
backup piece handle=c-28014364-20070308-17 RECID=199 STAMP=616695145
deleted backup piece
backup piece handle=8qic41c3_1_1 RECID=200 STAMP=616695171
deleted backup piece
backup piece handle=c-28014364-20070308-18 RECID=201 STAMP=616695188
deleted backup piece
backup piece handle=8uic47fg_1_1 RECID=204 STAMP=616701424
deleted backup piece
backup piece handle=90ic47ih_1_1 RECID=205 STAMP=616701521
deleted backup piece
backup piece handle=91ic47j1_1_1 RECID=206 STAMP=616701538
deleted backup piece
backup piece handle=92ic47q4_1_1 RECID=207 STAMP=616701764
deleted backup piece
```

```
backup piece handle=c-28014364-20070308-1a RECID=208 STAMP=616701783
Deleted 11 objects

RMAN> RELEASE CHANNEL;

released channel: ORA_MAINT_SBT_TAPE_1
```

Example 2–11 Crosschecking Backups on Multiple Devices

Assume that you want to crosscheck backups of archived redo log files on disk and tape. Assume also that you have the default device type configured to disk, and also have an SBT channel configured, but you want to use different channel settings for both disk and tape. In this case, you can manually allocate maintenance channels with the desired settings.

```
RMAN> SHOW DEFAULT DEVICE TYPE;

RMAN configuration parameters for database with db_unique_name PROD are:
CONFIGURE DEFAULT DEVICE TYPE TO DISK;

RMAN> SHOW CHANNEL;

RMAN configuration parameters for database with db_unique_name PROD are:
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/usr/local/oracle/backup/lib/libobk.so, ENV=(OB_DEVICE_1=stape1)';

RMAN> ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt PARMS
'SBT_LIBRARY=/usr/local/oracle/backup/lib/libobk.so, ENV=(OB_DEVICE_1=stape2)';

allocated channel: ORA_MAINT_SBT_TAPE_1
channel ORA_MAINT_SBT_TAPE_1: SID=135 device type=SBT_TAPE
channel ORA_MAINT_SBT_TAPE_1: Oracle Secure Backup

RMAN> ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK FORMAT "/disk2/%U";

allocated channel: ORA_MAINT_DISK_2
channel ORA_MAINT_DISK_2: SID=101 device type=DISK

Finished Control File and SPFILE Autobackup at 09-MAR-07

RMAN> CROSSCHECK BACKUP OF ARCHIVELOG ALL;

crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/disk2/95ic69jc_1_1 RECID=210 STAMP=616769132
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=/disk2/96ic69jf_1_1 RECID=211 STAMP=616769135
Crosschecked 2 objects

crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/disk2/96ic69jf_1_1 RECID=211 STAMP=616769135
Crosschecked 1 objects


RMAN> RELEASE CHANNEL;

released channel: ORA_MAINT_SBT_TAPE_1
released channel: ORA_MAINT_DISK_2
```

Example 2–12 Crosschecking in an Oracle Real Application Clusters (Oracle RAC) Configuration

All nodes in an Oracle RAC configuration should have the same access to all backups on all storage devices, but this is not a requirement. Assume that you want to crosscheck backups on two nodes of an Oracle RAC configuration, where each node has access to a subset of disk backups. It is assumed that all backups are accessible by at least one of the two nodes used in the crosscheck. Any backups not accessible from at least one of the nodes are marked `EXPIRED` after the crosscheck.

The following example illustrates channel connections to Oracle RAC instances `inst1` and `inst2`. For both channel connections, RMAN uses the same user name and password that were entered for the target database connection.

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK
  CONNECT '@inst1';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK
  CONNECT '@inst2';
CROSSCHECK BACKUP;
```

ALTER DATABASE

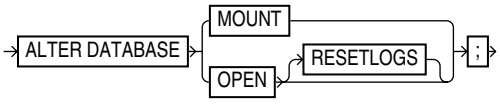
Purpose

Use the `ALTER DATABASE` command to mount or open a database.

> **See Also:** *Oracle Database SQL Language Reference* for `ALTER DATABASE` syntax and semantics

Syntax

***alterDatabase*::=**



Prerequisites

Execute this command either within the braces of a `RUN` command or at the RMAN prompt. The target instance must be started.

Semantics

Syntax Element	Description
`MOUNT`	Mounts the database without opening it. Issuing the command with this option is equivalent to the SQL statement `ALTER DATABASE MOUNT`.
`OPEN`	Opens the database (see Example 2–13). When you open the database after `RECOVER DATABASE`, RMAN re-creates any locally managed tempfiles recorded in the RMAN repository if necessary. However, if you perform recovery with a backup control file and no recovery catalog, then RMAN does not record tempfiles created after the control file backup in the RMAN repository. Also, RMAN does not re-create the tempfiles automatically.
`RESETLOGS`	Archives the current online redo log files (or up to the last redo record before redo corruption if corruption is found), clears the contents of the online redo log files, and resets the online redo log to log sequence 1. The RMAN command `ALTER DATABASE OPEN RESETLOGS` is equivalent to the SQL statement `ALTER DATABASE OPEN RESETLOGS`. If you use a recovery catalog, then RMAN issues an implicit `RESET DATABASE` after the database is opened to make this new incarnation the current one in the catalog. If you execute the SQL statement `ALTER DATABASE OPEN RESETLOGS` rather than the RMAN command of the same name, then you must manually run the `RESET DATABASE` command.

Examples

Example 2–13 Making a Consistent Database Backup

Assume that the database is open and you want to make a consistent backup of the whole database. This example shuts down the database consistently, mounts the database, makes a consistent whole database backup, and then opens the database.

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
BACKUP DATABASE PLUS ARCHIVELOG;
```

```
# Now that the backup is complete, open the database.
ALTER DATABASE OPEN;
```

Example 2–14 Mounting the Database After Restoring the Control File

This example restores the control file, mounts it, and performs recovery. Finally, the example resets the online redo log.

```
STARTUP FORCE NOMOUNT;
RESTORE CONTROLFILE FROM AUTOBACKUP;
ALTER DATABASE MOUNT;
# You must run the RECOVER command after restoring a control file even if no
# datafiles require recovery.
RECOVER DEVICE TYPE DISK DATABASE;
ALTER DATABASE OPEN RESETLOGS;
```

BACKUP

Purpose

Use the BACKUP command to back up a primary or standby database, tablespace, datafile (current or copy), control file (current or copy), server parameter file, archived redo log file, or backup set.

Additional Topics

- Prerequisites
- Usage Notes
- Syntax
- Semantics
- Examples

Prerequisites

RMAN must be connected to a target database. See the CONNECT and RMAN commands to learn how to connect to a database as TARGET.

Database Archiving Modes

If the target database is in ARCHIVELOG mode, then the database must be mounted or open with a current control file. Backups made while the database is open are inconsistent. You must apply redo log files after restoring an inconsistent backup to make the database consistent.

If the target database is in NOARCHIVELOG mode, then the database must be mounted after a consistent shutdown when you make the backup. The shutdown is only consistent if you successfully execute the SHUTDOWN command with the NORMAL, IMMEDIATE, or TRANSACTIONAL options. You cannot use RMAN to back up a NOARCHIVELOG database after an instance failure or SHUTDOWN ABORT.

Backup Media

RMAN can only back up files onto valid media. If you specify DEVICE TYPE DISK, then RMAN makes backups to random access disks. You can make a backup on any device that can store a datafile. If the statement CREATE TABLESPACE *tablespace_name* DATAFILE *'filename'* works, then *'filename'* is a valid backup path name. If you specify DEVICE TYPE sbt, then you can back up files to any media supported by the media manager.

When backing up Oracle Database files to disk, the logical block size of the files must be an even multiple of the physical block size of the destination device. For example, a disk device with a block size of 2 KB can only be used as a destination for backups of Oracle files with logical block sizes of 2 KB, 4 KB, 6 KB and so on. In practice, most disk drives have physical block sizes of 512 bytes, so this limitation rarely affects backup. However, you can encounter this limitation when using BACKUP . . . DEVICE TYPE DISK to back your database up to a writeable CD or DVD, or some other device that has a larger physical block size.

Channels

If no automatic channel is configured for the specified device type, then you must manually allocate a channel for each BACKUP execution. If no manual channel is

allocated, then RMAN uses the default channels set with the CONFIGURE command. RMAN has a DISK channel preconfigured but no preconfigured sbt channels.

> **Note:** Backups that use the disk test API are not supported for production backups. Instead, use the preconfigured DISK channel or manually allocate a DISK channel.

Usage Notes

RMAN can only back up datafiles, control files, server parameter files, archived redo log files, and RMAN backups of these files. RMAN cannot make backups of other database-related files such as network configuration files, password files, the block change tracking file, and the contents of the Oracle home directory. Likewise, some features of Oracle Database, such as external tables or the BFILE data type, store data in files other than those in the preceding list. RMAN cannot back up these files.

RMAN decomposes a BACKUP command into multiple independent backup steps. RMAN can execute each independent step on any channel allocated for a specific device. If multiple channels are allocated, and if one channel fails or encounters a problem during a backup step, then RMAN attempts to complete the work on another channel. RMAN reports a message in V$RMAN_OUTPUT and in the output to the interactive session or log file when channel failover occurs.

RMAN backups made on one platform are not transportable to a different platform.

RMAN backups made in a previous release of Oracle Database are usable after a database migration or upgrade.

If you change the DB_NAME for a database, but not its DBID, then RMAN considers backups made of the database with the previous DB_NAME as eligible to be restored.

Incremental Backups

An INCREMENTAL backup at level 0 backs up all data blocks in datafiles being backed up. An incremental backup at level 0 is identical in content to a FULL backup, but unlike a full backup the level 0 backup is a part of the incremental backup strategy.

A level 1 backup copies only changed blocks. A level 1 incremental backup is either differential or CUMULATIVE. If cumulative, RMAN backs up all blocks changed since the most recent level 0 backup. If differential, RMAN backs up blocks updated since the most recent level 0 or level 1 incremental backup. You can apply a level 1 backup of a standby database to a level 0 backup of a primary database, and also apply a level 1 backup of a primary database to a level 0 backup of a standby database.

Incremental backups at level 0 can be either backup sets or image copies, but incremental backups at level 1 can only be backup sets.

The database performs checks when attempting to create a level 1 incremental backup to ensure that the incremental backup is usable by a subsequent RECOVER command. Among the checks performed are:

- A level 0 backup must exist for each datafile in the BACKUP command as the base backup for an incremental strategy. Level 0 backups must not have status UNAVAILABLE. If no level 0 backup exists, then RMAN makes a level 0 backup automatically.
- Sufficient incremental backups taken since level 0 must exist and be available such that the incremental backup to be created is usable.

> **Note:** When creating an incremental backup, RMAN considers backups from parent incarnations as valid. For example, assume you make a level 0 backup and then `OPEN RESETLOGS`. If you make a level 1 incremental backup, then RMAN backs up all blocks changed since the pre-`RESETLOGS` level 0 backup. When making a level 1 backup, RMAN only makes a new level 0 backup if no level 0 is available in either the current or parent database incarnation.

You can improve incremental backup performance by enabling **block change tracking** on a primary or standby database. In this case, RMAN keeps a record of which blocks have changed in the block change tracking file.

The change tracking file maintains bitmaps that mark changes in the datafiles between backups. The database performs a bitmap switch before each backup. Oracle Database automatically manages space in the change tracking file to retain block change data that covers the 8 most recent backups. After the maximum of 8 bitmaps is reached, the most recent bitmap is overwritten by the bitmap that tracks the current changes.

The first level 0 incremental backup scans the entire datafile. Subsequent incremental backups use the block change tracking file to scan only the blocks that have been marked as changed since the last backup. An incremental backup can be optimized only when it is based on a parent backup that was made after the start of the oldest bitmap in the block change tracking file.

Consider the 8-bitmap limit when developing your incremental backup strategy. For example, if you make a level 0 database backup followed by 7 differential incremental backups, then the block change tracking file now includes 8 bitmaps. If you then make a cumulative level 1 incremental backup, RMAN cannot optimize the backup because the bitmap corresponding to the parent level 0 backup is overwritten with the bitmap that tracks the current changes.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* for details about block change tracking

Encryption of Backup Sets

RMAN can transparently encrypt data written to backup sets and decrypt those backup sets when they are needed in a `RESTORE` operation. To create encrypted backups on disk, the database must use the Advanced Security Option. To create encrypted backups directly on tape, RMAN must use the Oracle Secure Backup SBT interface, but does not require the Advanced Security Option. RMAN issues an `ORA-19916` error if you attempt to create encrypted RMAN backups using an SBT library other than Oracle Secure Backup.

RMAN can encrypt backups by using several different encryption algorithms, which are listed in `V$RMAN_ENCRYPTION_ALGORITHMS`. RMAN supports three modes of encryption for backups:

- Transparent encryption, in which RMAN can create and restore encrypted backups with no special DBA intervention, as long as the required Oracle public key infrastructure (PKI) is available
- Password-based encryption, where a password is specified during the backup, and the same password must be supplied to restore the backup
- Dual-mode encryption, where backups can be created using either as with transparent encryption or password-based encryption, and where decryption can

be performed based upon either the Oracle Wallet, or a password the DBA supplies at decryption time

> **Note:** Wallet-based encryption is more secure than password-based encryption because no passwords are involved. You should use password-based encryption only when absolutely necessary because your backups must be transportable.

The `CONFIGURE` and `SET` commands manage the encryption settings for database backups. See the reference entries for those commands for more details. Backup sets containing archived redo log files are encrypted if any of the following is true:

- `SET ENCRYPTION ON` is in effect at the time that the backup is being created.
- Encryption is configured for the whole database or at least one tablespace.

See Also:

- *Oracle Database Backup and Recovery User's Guide*for an overview of the backup encryption, a guide to its use, and information on choosing among different modes of encryption
- *Oracle Database Advanced Security Administrator's Guide* to learn about PKI and the Oracle Wallet

RMAN Backups in a Data Guard Environment

A recovery catalog is required when you are performing RMAN operations in a Data Guard environment. The catalog enables all RMAN operations to be transparently executable at any primary or standby database. You can offload primary database backups onto any standby database in the environment; the RMAN backups are interchangeable. If you use RMAN in `NOCATALOG` mode, then RMAN uses only the metadata in the mounted control file.

In a Data Guard environment, the database that creates a backup or copy is associated with the file. For example, if RMAN connects as `TARGET` to database `prod` and backs it up, then this database backup is associated with `prod`. A backup remains associated with the database that created in unless you use the `CHANGE ... RESET DB_UNIQUE_NAME` to associate the backup with a different database.

The association of a backup is different from its accessibility. The recovery catalog considers disk backups as accessible only to the database in the Data Guard environment on which it was created, whereas tape backups created on one database are considered accessible to all databases. If a backup file is not associated with any database, then the row describing it in the recovery catalog view shows `null` for the `SITE_KEY` column. By default, RMAN associates files whose `SITE_KEY` is `null` with the database to which RMAN is connected as `TARGET`.

In a Data Guard environment, RMAN commands can operate on any backups that are accessible. For example, assume that databases `prod` and `standby1` reside on different hosts. RMAN backs up datafile 1 on `prod` to `/prodhst/disk1/df1.dbf` on the production host and also to tape. RMAN backs up datafile 1 on `standby1` to `/sby1hst/disk2/df1.dbf` on the standby host and also to tape. If RMAN is connected to database `prod` as `TARGET`, then you cannot use RMAN to perform operations with the `/sby1hst/disk2/df1.dbf` backup located on the standby host. However, RMAN considers the tape backup made on `standby1` as eligible to be restored.

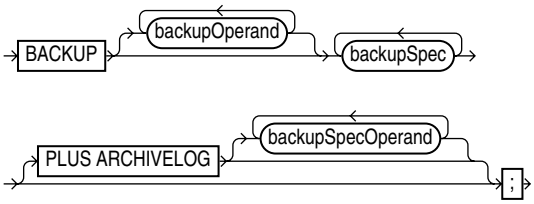
> **Note:** You can FTP a backup from a standby host to a primary host or vice versa and then CATALOG it. After a file is cataloged by the target database, the file is associated with the target database.

As long as backups are accessible to RMAN, you can use RMAN maintenance commands such as CHANGE, CROSSCHECK, and DELETE for backups when connected to any primary or standby database.

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

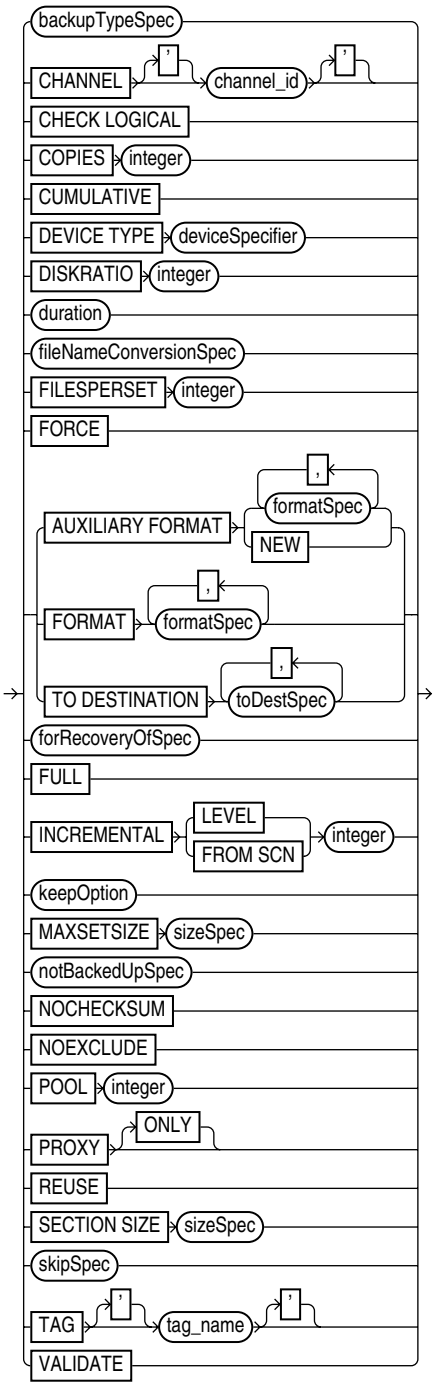
Syntax

***backup*::=**



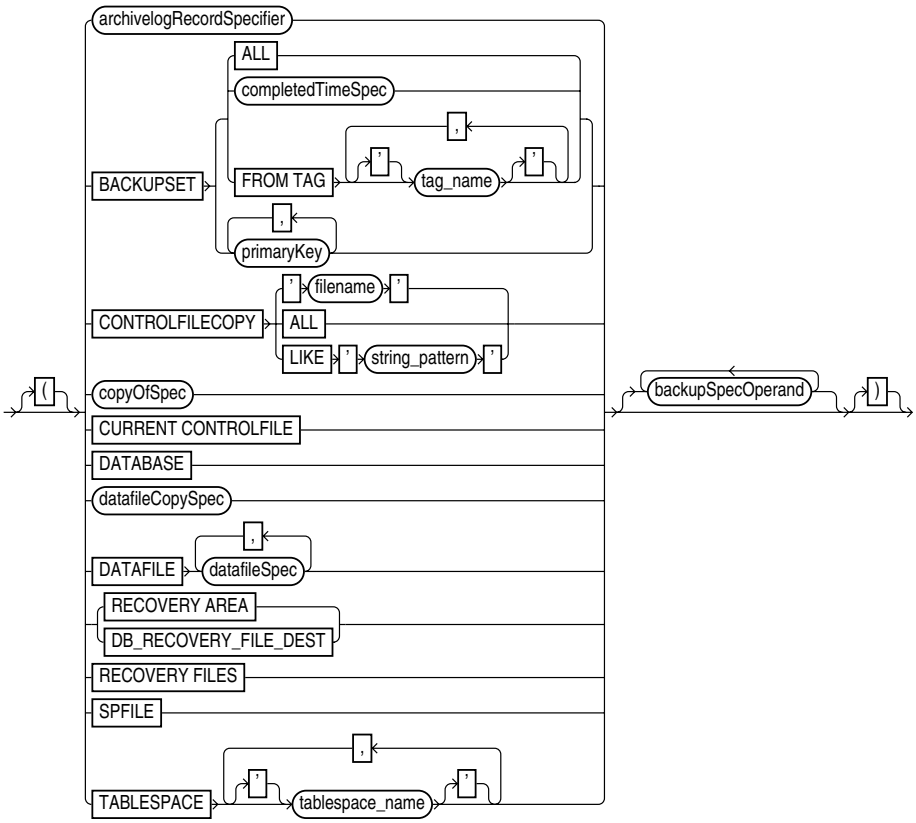
(*backupOperand::=* on page 2-24, *backupSpec::=* on page 2-25, *backupSpecOperand::=* on page 2-26)

***backupOperand*::=**



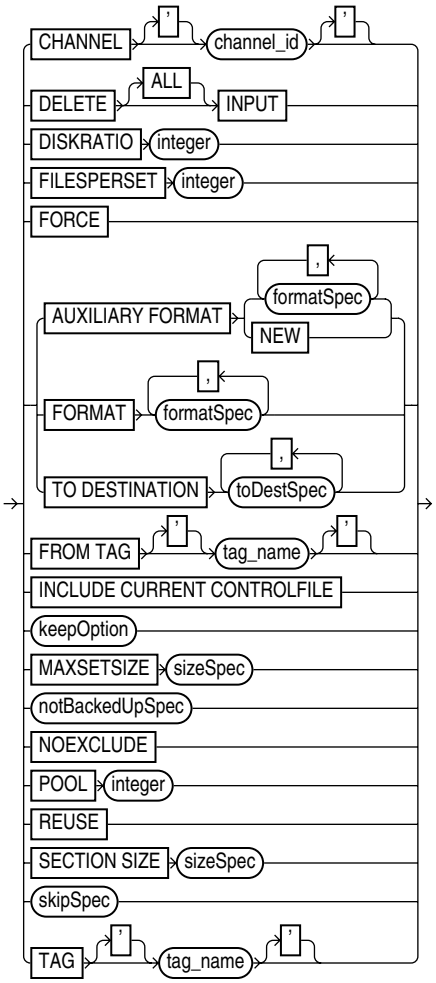
(*deviceSpecifier::=* on page 4-16, *fileNameConversionSpec::=* on page 4-18, *formatSpec::=* on page 4-23, *toDestSpec::=* on page 4-40, *forRecoveryOfSpec::=* on page 2-27, *keepOption::=* on page 4-26, *notBackedUpSpec::=* on page 2-27, *sizeSpec::=* on page 2-27, *skipSpec::=* on page 2-27)

backupSpec::=



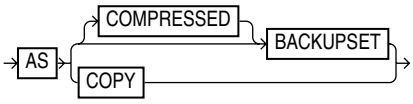
(*archivelogRecordSpecifier*::= on page 4-6, *completedTimeSpec*::= on page 4-11, *copyOfSpec*::= on page 2-26, *datafileCopySpec*::= on page 2-27, *datafileSpec*::= on page 4-15, *backupSpecOperand*::= on page 2-26)

backupSpecOperand::=

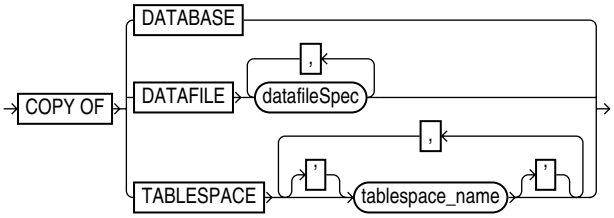


(*formatSpec::=* on page 4-23, *toDestSpec::=* on page 4-40 *keepOption::=* on page 4-26, *notBackedUpSpec::=* on page 2-27, *sizeSpec::=* on page 2-27, *skipSpec::=* on page 2-27)

backupTypeSpec::=

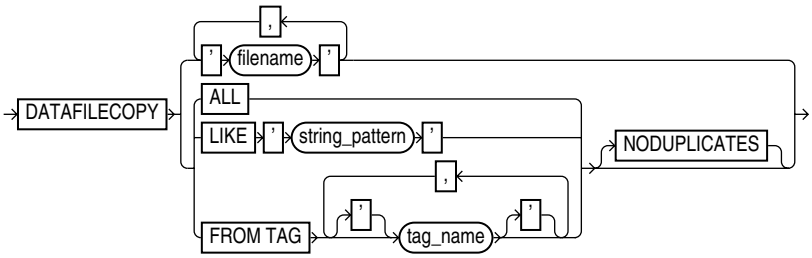


copyOfSpec::=

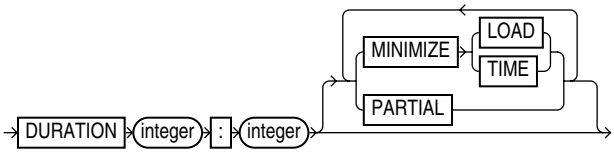


(*datafileSpec::=* on page 4-15)

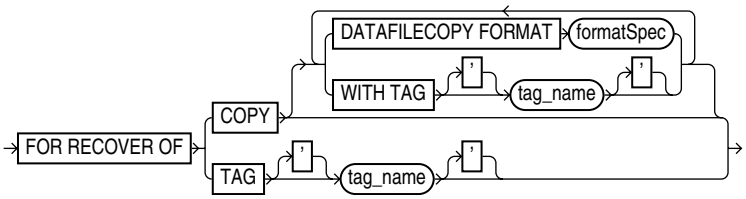
***datafileCopySpec*::=**



***duration*::=**

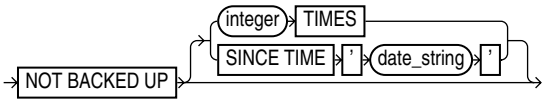


***forRecoveryOfSpec*::=**

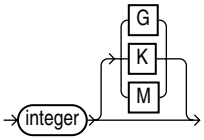


(*formatSpec::=* on page 4-23)

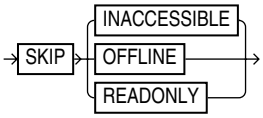
***notBackedUpSpec*::=**



***sizeSpec*::=**



***skipSpec*::=**



Semantics

backup

This clause specifies the objects to be backed up as well as the options to control the backup. Refer to *backup::=* on page 2-23 for the syntax diagram.

Syntax Element	Description
backupOperand	Specifies various options for the BACKUP command.
backupSpec	Specifies one or more objects to be backed up. Each *backupSpec* clause generates one or more backup sets (AS BACKUPSET) or image copies (AS COPY). For AS BACKUPSET, the *backupSpec* clause generates multiple backup sets if the number of datafiles specified in or implied by its list of objects exceeds the FILESPERSET limit.
PLUS ARCHIVELOG	Includes archived redo log files in the backup (see Example 2–15 on page 2-47). Causes RMAN to perform the following steps: **1.** Run an ALTER SYSTEM ARCHIVE LOG CURRENT statement. **2.** Run the BACKUP ARCHIVELOG ALL command. If backup optimization is enabled, then RMAN only backs up logs that have not yet been backed up. **3.** Back up the files specified in the BACKUP command. **4.** Run an ALTER SYSTEM ARCHIVE LOG CURRENT statement. **5.** Back up any remaining archived redo log files. If backup optimization is not enabled, then RMAN backs up the logs generated in step 1 plus all the logs generated during the backup. **Note:** You cannot specify PLUS ARCHIVELOG on the BACKUP ARCHIVELOG command or BACKUP AS COPY INCREMENTAL command (or BACKUP INCREMENTAL command when the default backup type is COPY). You cannot specify PLUS ARCHIVELOG when also specifying INCREMENTAL FROM SCN. **Note:** Unless the online redo log is archived at the end of the backup, DUPLICATE will not be possible with this backup.
backupSpecOperand	Specifies a variety of options and parameters that affect the *backupSpec* clause.

backupOperand

This subclause specifies options such as the device type and output format. Refer to *backupOperand::=* on page 2-24 for the syntax diagram.

Syntax Element	Description
backupTypeSpec	Specifies the type of backup being created, either backup sets (AS BACKUPSET) or image copies (AS COPY). **See Also:** *backupTypeSpec* on page 2-40 for details
CHANNEL *channel_id*	Specifies the case-sensitive name of a channel to use when creating backups. Use any meaningful name, for example ch1 or dev1. The database uses the channel ID to report I/O errors. If you do not set this parameter, then RMAN dynamically assigns the backup sets to any available channels during execution. As shown in Example 2–23 on page 2-49, you can use CHANNEL to specify which channels back up which files. **Note:** You can also specify this parameter in the *backupSpec* clause.

Syntax Element	Description
CHECK LOGICAL	Tests data and index blocks that pass physical corruption checks for logical corruption (see Example 2–25 on page 2-50). This option typically adds 1-3% overhead. Examples of logical corruption are 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. The SET MAXCORRUPT command specifies the total number of physical and logical corruptions permitted in a datafile. By default, the BACKUP command computes a checksum for each block and stores it in the backup. If you specify the NOCHECKSUM option, then RMAN does not perform a checksum of the blocks when writing the backup. If SET MAXCORRUPT and NOCHECKSUM are not set, then CHECK LOGICAL detects all types of corruption that are possible to detect during a backup.
COPIES *integer*	Sets the number of identical backups (1 - 4) that RMAN creates. The default value is 1. You can use multiple format strings to specify different names and locations for the copies. Example 2–22 on page 2-49 illustrates a duplexed backup to different locations on disk. RMAN can duplex backups to either disk or tape, but cannot duplex backups to tape and disk simultaneously. When backing up to tape, ensure that the number of copies does not exceed the number of available tape devices. Also, if COPIES is greater than 1, then the BACKUP_TAPE_IO_SLAVES initialization parameter must be enabled on the target database. You can specify duplexing on more than one command. The order of precedence is as follows, with settings higher on the list overriding lower settings: **1.** BACKUP COPIES **2.** SET BACKUP COPIES **3.** CONFIGURE . . . BACKUP COPIES **Note:** This option does not apply with AS COPY and results in an error message. **Note:** Duplexing cannot be used when creating files in the fast recovery area.
CUMULATIVE	Copies the data blocks used since the most recent level 0 backup (see Example 2–16 on page 2-47). **Note:** This option does not apply with AS COPY and results in an error message.
DEVICE TYPE *deviceSpecifier*	Allocates automatic channels for the specified device type only. For example, if you configure disk and tape channels, then configure sbt as the default device type, then the following command allocates disk channels only: BACKUP DEVICE TYPE DISK DATABASE; The DEVICE TYPE option is valid only for automatic channels and is not valid for manually allocated channels. You cannot use the DEVICE TYPE option for a device other than DISK if you have not already run CONFIGURE DEVICE TYPE for this device. **Note:** To specify a disk channel for the BACKUP RECOVERY AREA command, you must use the TO DESTINATION of the subclause toDestSpec. **See Also:** *deviceSpecifier* on page 4-16

Syntax Element	Description
DISKRATIO *integer*	Directs RMAN to populate each backup set with datafiles from at least *integer* disks. This parameter is only enabled when you are backing up datafiles or control files, and when the operating system can give RMAN disk contention and node affinity data. To manually disable this feature, set DISKRATIO to 0. For example, assume that datafiles are distributed across 10 disks. If the disks supply data at 10 bytes/second, and if the tape drive requires 50 bytes/second to keep streaming, then set DISKRATIO to 5 to direct RMAN to include datafiles from at least 5 disks in each backup set. If you set FILESPERSET but not DISKRATIO, then DISKRATIO defaults to the same value as FILESPERSET. If you specify neither parameter, then DISKRATIO defaults to 4. RMAN compares the DISKRATIO value to the actual number of devices involved in the backup and uses the lowest value. For example, if DISKRATIO is 4 and the datafiles are located on three disks, then RMAN attempts to include datafiles from three disks in each backup set. The DISKRATIO parameter is easier for datafile backups when the 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 need to spread the I/O load across several disk spindles to leave bandwidth for online operations. **Note:** Do not spread I/O over more than the minimum number of disks required to keep the tape streaming. Otherwise, you increase restore time for a file without increasing performance.
duration	Specifies options related to the maximum time for a backup command to run. **See Also:** *duration* on page 2-43
fileNameConversionSpec	This option is valid only when BACKUP is creating image copies. Files being copied are renamed according to the specified patterns. If a file being backed up has a name that does not match any of the specified rename patterns, then RMAN uses FORMAT to name the output image copies. If no FORMAT is specified, then RMAN uses the default format %U. **See Also:** *fileNameConversionSpec* on page 4-17 for file renaming patterns

FILESPERSET *integer*	Specifies the maximum number of input files to include in each output backup set. This parameter is only relevant when BACKUP generates backup sets.
	
	RMAN backs up the files in each *backupSpec* as one or more backup sets. When the number of files in each *backupSpec* exceeds the FILESPERSET setting, then RMAN splits the files into multiple backup sets accordingly. The default value for FILESPERSET is 64.
	
	The RMAN behavior is illustrated by the following BACKUP commands:
	
	`BACKUP AS BACKUPSET (DATAFILE 3, 4, 5, 6, 7) (DATAFILE 8, 9);`
	`BACKUP AS BACKUPSET DATAFILE 3, 4, 5, 6, 7, 8, 9;`
	`BACKUP AS BACKUPSET DATAFILE 3, ... 72;`
	
	In the first command, RMAN places datafiles 3, 4, 5, 6, and 7 into one backup set and datafiles 8 and 9 into another backup set. In the second command, RMAN places all datafiles into one backup set. In the third command, the ellipses indicate datafiles 3 through 72. Because in this case RMAN is backing up 70 datafiles, RMAN places 64 files in one backup set and 6 in another.
	
	By default, RMAN divides files among backup sets in order to make optimal use of channel resources. The number of files to be backed up is divided by the number of channels. If the result is less than 64, then this number is the FILESPERSET value. Otherwise, FILESPERSET defaults to 64.
	
	Note: You cannot specify the number of backup pieces that should be in a backup set.

Syntax Element	Description
FORCE	Forces RMAN to ignore backup optimization. That is, even if CONFIGURE BACKUP OPTIMIZATION is set to ON, RMAN backups up all specified files. **Note:** You can also specify this option in the *backupSpecOperand* clause.
AUXILIARY FORMAT	Copies the files on the target database to the specified location on the auxiliary instance. RMAN can only generate image copies when AUXILIARY FORMAT is specified. RMAN must be connected to both TARGET and AUXILIARY instances and have access to auxiliary channels. You can use the BACKUP AUXILIARY FORMAT command to copy datafiles, archived logs, controlfile, and server parameter file (spfile) over the network between primary and standby databases. For example, if a datafile on a primary database was lost, you could CONNECT to the standby database as TARGET and the primary database as AUXILIARY, and copy an intact datafile from the standby host to the primary host. **See Also:**Example 2–30, "Copying Archived Log From Operating System File to ASM"
formatSpec	Specifies a pattern for naming the output image copies on an auxiliary instance. The path must be valid on the auxiliary host. **See Also:** *formatSpec* on page 4-23 for valid substitution variables
NEW	Creates an image copy in the directory specified in the DB_CREATE_FILE_DEST initialization parameter of the auxiliary instance. The image copy is an Oracle-managed file.

Syntax Element	Description
FORMAT *formatSpec*	Specifies a pattern for naming the output backup pieces or image copies (see Example 2–17 on page 2-47). For AS COPY, if one or more of the directories mentioned in the specified format does not exist, then RMAN signals an error. The default location for disk backups depends on whether a fast recovery area is enabled and whether FORMAT is specified: ■ If a fast recovery area is enabled, and if you *do* specify FORMAT, then RMAN names the output files according to the FORMAT setting. If no location is specified in FORMAT, then RMAN creates the backup in a platform-specific location—not in the recovery area. ■ If a fast recovery area is enabled, and if you do *not* specify FORMAT, then RMAN creates the backup in the recovery area and uses the substitution variable %U to name the backup. ■ If a fast recovery area is *not* enabled, and if you do *not* specify FORMAT, then RMAN creates the backup in a platform-specific location and uses %U to name the backup. To create RMAN backups in the fast recovery area with names in Oracle Managed Files format, do not specify the FORMAT clause on the BACKUP command or channel. **Note:** You cannot specify an Oracle Managed Files filename as the format for a backup. For example, if +DISK1/datafile/system.732.609791431 is an OMF filename, then you cannot specify this filename in the FORMAT parameter. Backup pieces must have unique names. The maximum length of a backup piece filename is platform-specific. For backups to a media manager, the length is also limited by the limit in the supported version of the media management API. Vendors supporting SBT 1.1 must support filenames up to 14 characters. Some SBT 1.1 vendors may support longer filenames. Vendors supporting SBT 2.0 must support filenames up to 512 characters. Some SBT 2.0 vendors may support longer filenames. You cannot specify multiple, identical FORMAT strings within a single *backupSpec* (for example, BACKUP DATAFILE 3 TO '/tmp/df3.f', DATAFILE 4 TO '/tmp/df4.f'). However, RMAN permits a single FORMAT string to exist in multiple *backupSpec* clauses. **Note:** If you are making an archival backup with the KEEP option (see Example 2–26 on page 2-50), then the format string must contain %U. The autobackup also uses this format string. **See Also:** *formatSpec* on page 4-23 for valid substitution variables
TO DESTINATION *toDestSpec*	Specifies the directory where the backup is created. This parameter is valid for disk and not SBT channels. The backup files are created in a Oracle Managed Files (OMF) directory. Backup skips the files only when backups do not exist in the specified TO DESTINATION.
forRecoveryOfSpec	Identifies the backup being created as an incremental backup to be used in rolling forward an image copy. **See Also:** *forRecoveryOfSpec* on page 2-44
FULL	Creates a backup of all blocks of datafiles included in the backup. FULL is the opposite of INCREMENTAL. RMAN makes full backups by default if neither FULL nor INCREMENTAL is specified. A full backup has no effect on subsequent incremental backups and is not considered a part of any incremental backup strategy. A full image copy backup can be incrementally updated, however, by applying incremental backups with the RECOVER command. **Note:** Unused block compression, which is described in the entry for BACKUP AS BACKUPSET, causes some datafile blocks to be skipped during full backups.

Syntax Element	Description
INCREMENTAL LEVEL *integer*	Copies only those data blocks that have changed since the last incremental *integer* backup, where *integer* is 0 or 1 (see Example 2–16 on page 2-47). See "Incremental Backups" on page 2-20 for an explanation of incremental backups.
INCREMENTAL FROM SCN *integer*	Creates an incremental backup of all specified datafiles that includes all datafile blocks changed at SCNs greater than or equal to the specified SCN. One use of this option is to refresh a standby database with changes from the primary database (see Example 2–24 on page 2-49, and the chapter on RMAN backups in *Oracle Data Guard Concepts and Administration*). This backup contains all changed blocks since the standby database was created or last synchronized. At the standby database, you can use RECOVER with NOREDO to apply the incremental backup. All changed blocks captured in the incremental backup are applied at the standby database, bringing it current with the primary database. If you are *not* making incremental backups based on Volume Shadow Copy Service (VSS) snapshots, then you should specify *formatSpec* when you specify INCREMENTAL FROM SCN. The FORMAT string should include substitution variables such as %U because RMAN generates a control file backup. If you specify FROM SCN with the NOKEEP option, and if you do *not* specify *formatSpec* when you specify INCREMENTAL FROM SCN, then RMAN creates incremental backups in the fast recovery area so that you can create incremental backups based on VSS snapshots in a Windows environment. In this way, you can use incremental backup sets and VSS shadow copies in conjunction. The checkpoint SCN value specified in the FROM SCN parameter should be same as the BACKUP_CHECKPOINT value in the VSS backup metadata document. If block change tracking is enabled, then the backups will use the change tracking mechanism, which significantly reduces the time taken to create incremental backups. RMAN can apply incremental backups from the fast recovery area during recovery transparently. **Note:** You cannot use PLUS ARCHIVELOG when also specifying INCREMENTAL FROM SCN. **See Also:** *Oracle Database Platform Guide for Microsoft Windows* to learn about making backups with VSS
keepOption	Overrides any configured retention policy for this backup so that the backup is not considered obsolete, as shown in Example 2–26 on page 2-50. You can use the KEEP syntax to generate archival database backups that satisfy business or legal requirements. The KEEP setting is an attribute of the backup set (not individual backup piece) or image copy. **Note:** You cannot use KEEP with BACKUP BACKUPSET. With the KEEP syntax, you can keep the backups so that they are considered obsolete after a specified time (KEEP UNTIL), or make them never obsolete (KEEP FOREVER). As shown in Example 2–27 on page 2-50, you must be connected to a recovery catalog when you specify KEEP FOREVER. **Note:** You can use CHANGE to alter the status of a backup generated with KEEP. **See Also:** *keepOption* on page 4-26 for more information about backups made with the KEEP option

Syntax Element	Description
MAXSETSIZE *sizeSpec*	Specifies a maximum size for a backup set (as shown in Example 2–17 on page 2-47). RMAN limits all backup sets to this size. It is possible for a backup set to span multiple tapes, so blocks from each datafile are written to multiple tapes. If one tape of a multivolume backup set fails, then you lose the data on all the tapes rather than just one. Because a backup set always include a whole file rather than part of a file, you can use MAXSETSIZE to specify that each backup set should fit on one tape. Specify size in bytes (default), kilobytes (K), megabytes (M), or gigabytes (G). For example, to limit a backup set to 3 MB, specify MAXSETSIZE 3M. The default size is in bytes, rounded down from kilobytes. For example, MAXSETSIZE 3000 is rounded down to 2 KB (2048 bytes). The minimum value must be greater than or equal to the database block size. The default number of files in each backup set is determined by FILESPERSET, which defaults to 64. When you specify MAXSETSIZE, RMAN attempts to limit the size in bytes of the backup sets according to the MAXSETSIZE parameter. The limit on the number of files in a backup set will apply even if the total size of the resulting backup set is less than MAXSETSIZE. **Note:** This option results in an error message if used with BACKUP AS COPY. If you run BACKUP AS COPY on a channel that has MAXSETSIZE set, then MAXSETSIZE is silently ignored.
notBackedUpSpec	Limits the set of archived redo log files to be backed up according to whether a specified number of backups are already present (and not obsolete), or whether the logs have been backed up since a specified date. **See Also:** *notBackedUpSpec* on page 2-45
NOCHECKSUM	Suppresses block checksums during the backup. A checksum is a number that is computed from the contents of a data block. DB_BLOCK_CHECKSUM is a database initialization parameter that controls the writing of checksums for the blocks in datafiles in the database (not backups). If DB_BLOCK_CHECKSUM is typical, then the database computes a checksum for each block during normal operations and stores it in the block before writing it to disk. When the database reads the block from disk later, it recomputes the checksum and compares it to the stored value. If they do not match, then the block is damaged. **Note:** You cannot disable checksums for datafiles in the SYSTEM tablespace even if DB_BLOCK_CHECKSUM=false. By default, the BACKUP command computes a checksum for each block and stores it in the backup. The BACKUP command ignores the values of DB_BLOCK_CHECKSUM because this initialization parameter applies to datafiles in the database, not backups. If you specify the NOCHECKSUM option, then RMAN does not perform a checksum of the blocks when writing the backup. When restoring a backup datafile, RMAN honors the DB_BLOCK_CHECKSUM initialization parameter setting. RMAN clears the checksum if DB_BLOCK_CHECKSUM is set to false. If set to typical, then RMAN verifies the checksum when restoring from the backup and writing to the datafile. **Note:** You can turn off checksum checking by specifying NOCHECKSUM, but other physical consistency checks, such as checks of the block headers and footers, cannot be disabled. **See Also:** *Oracle Database Reference* for more information about the DB_BLOCK_CHECKSUM initialization parameter
NOEXCLUDE	When specified on a BACKUP DATABASE or BACKUP COPY OF DATABASE command, RMAN backs up all tablespaces, including any for which a CONFIGURE EXCLUDE command has been entered. This option does not override SKIP OFFLINE or SKIP READONLY.

Syntax Element	Description
POOL *integer*	Specifies the media pool in which the backup should be stored. Consult your media management documentation to see whether POOL is supported. **Note:** This option does not work with AS COPY and results in an error.
PROXY	Backs up the specified files by means of the proxy copy functionality, which gives the media management software control over the data transfer between storage devices and the datafiles on disk. The media manager—not RMAN—decides how and when to move data. When you run BACKUP with the PROXY option, RMAN performs these steps: **1.** Searches for a channel of the specified device type that is proxy-capable. If no such channel is found, then RMAN issues a warning and attempts a a conventional (that is, non-proxy) backup of the specified files. **2.** If RMAN locates a proxy-capable channel, then it calls the media manager to check if it can proxy copy the files. If the media manager cannot proxy copy, then RMAN uses conventional backup sets to back up the files. **Note:** If you specify PROXY, then the %p variable must be included in the FORMAT string either explicitly or implicitly within %U. **Note:** This option does not work with AS COPY and results in an error.
ONLY	Causes the database to issue an error message when it cannot proxy copy rather than creating conventional backup sets. If you do not want RMAN to try a conventional copy when a proxy copy fails, use the ONLY option
REUSE	Enables RMAN to overwrite an already existing backup or copy with the same filename as the file that BACKUP is currently creating.
SECTION SIZE *sizeSpec*	Specifies the size of each backup section produced during a datafile or datafile copy backup. By setting this parameter, RMAN can create a **multisection backup**. In a multisection backup, RMAN creates a backup piece that contains one **file section**, which is a contiguous range of blocks in a file. All sections of a multisection backup are the same size. File sections enable RMAN to create multiple steps for the backup of a single large datafile. RMAN channels can process each step independently and in parallel, with each channel producing one section of a multisection backup set. If you specify a section size that is larger than the size of the file, then RMAN does not use multisection backup for the file. If you specify a small section size that would produce more than 256 sections, then RMAN increases the section size to a value that results in exactly 256 sections. **Note:** Depending on where you specify this parameter in the RMAN syntax, you can specify different section sizes for different files in the same backup job. **Note:** You cannot use SECTION SIZE in conjunction with MAXPIECESIZE.
skipSpec	Excludes datafiles or archived redo log files from the backup if they are inaccessible, offline, or read-only. **See Also:** *skipSpec* on page 2-46 for details.

Syntax Element	Description
TAG *tag_name*	Specifies a user-specified tag name for a backup set, proxy copy, datafile copy, or control file copy. The tag is applied to the output files generated by the BACKUP command. The tag name is not case-sensitive. The name must be 30 characters or less. The characters are limited to the characters that are valid in filenames on the target file system. For example, ASM does not support the use of the hyphen (-) character in the filenames it uses internally, so weekly-incremental is not a valid tag name for backups in ASM disk groups. Environment variables are not valid in the TAG parameter. Typically, a tag name is a meaningful name such as MON_PM_BKUP or WEEKLY_FULL_BKUP. Tags are reusable, so that backup set 100 can have the tag MON_PM_BKUP one week while backup set 105 has the same tag the next week. If you do not specify a tag name, then by default RMAN creates a tag for backups (except for control file autobackups). The default tag uses the format TAG*YYYYMMDDTHHMMSS*, where *YYYY* is the year, *MM* is the month, *DD* is the day, *HH* is the hour (in 24-hour format), *MM* is the minutes, and *SS* is the seconds. For example, a backup of datafile 1 might receive the tag TAG20070208T133437. The date and time refer to when RMAN started the backup. If multiple backup sets are created by one BACKUP AS BACKUPSET command, then each backup piece is assigned the same default tag. 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 the command have the 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:** A tag is an attribute of each backup piece in a given copy of a backup set (for AS BACKUPSET) or each image copy (for AS COPY). For example, if you run BACKUP AS BACKUPSET COPIES 1 DATABASE TAG TUE_PM, then only one copy of the backup set exists and each backup piece has tag TUE_PM. Assume that this backup set has primary key 1234. If you then run BACKUP BACKUPSET 1234 TAG WED_PM, then the first copy of the backup set has tag TUE_PM and the second copy of the backup set has tag WED_PM.
VALIDATE	Scans the specified files and verifies their contents, testing whether this file can be backed up and whether the data blocks are corrupt. RMAN creates no output files. This option is equivalent to using the VALIDATE command on the database files specified in the backup. If you do not specify CHECK LOGICAL, then BACKUP VALIDATE checks for physical corruption only. If you specify CHECK LOGICAL, then BACKUP VALIDATE checks for both physical and logical corruption. RMAN populates the V$DATABASE_BLOCK_CORRUPTION view with any corruptions that it finds. You can use the SET MAXCORRUPT command to set a limit for the number of corrupt blocks tolerated during the backup validation. The default is zero. If you execute BACKUP INCREMENTAL with VALIDATE, then the behavior depends on whether block change tracking is enabled. If change tracking is enabled, then RMAN validates only changed blocks; otherwise, RMAN validates all blocks in the files included in the backup. **Note:** You cannot validate backups of backup sets.

backupSpec

This subclause specifies a list of one or more objects to be backed up. Each *backupSpec* clause generates one or more backup sets (AS BACKUPSET) or image copies (AS COPY). For AS BACKUPSET, the *backupSpec* clause generates multiple backup sets if the number of datafiles specified in or implied by its list of objects exceeds the

default limit of 4 datafiles or 16 archived redo log files in each backup set. Refer to *backupSpec::=* on page 2-25 for the syntax diagram.

Syntax Element	Description
archivelogRecordSpecifier	Specifies a range of archived redo log files to be backed up. When backing up archived redo log files, RMAN can perform archived log failover automatically. RMAN backs up the log when at least one archived log corresponding to a given log sequence number and thread is available. Also, if the copy that RMAN is backing up contains corrupt blocks, then it searches for good copies of these blocks in other copies of the same archived redo log files. RMAN does not signal an error if the command finds no logs to back up, since this situation probably exists because no new logs were generated after the previous `BACKUP ARCHIVELOG ALL DELETE INPUT` command. The only exception to this behavior is when the `SEQUENCE` number clause is specified. In this scenario, RMAN signals the `RMAN-06004` error if an archived redo log file of the specified sequence is not found. If you specify `BACKUP ARCHIVELOG ALL`, then RMAN backs up exactly one copy of each distinct log sequence number. For example, if you archive to multiple destinations, RMAN backs up *one* copy of each log sequence number—not each copy of each log sequence number. For other commands, such as `DELETE ALL` does refer to every log, even duplicate log sequences. If the database is open when you run `BACKUP ARCHIVELOG`, and if the `UNTIL` clause or `SEQUENCE` parameter is not specified, then RMAN runs `ALTER SYSTEM ARCHIVE LOG CURRENT`. **Note:** If you run `BACKUP ARCHIVELOG ALL`, or if the specified log range includes logs from prior incarnations, then RMAN backs up logs from prior incarnations to ensure availability of all logs that may be required for recovery through an `OPEN RESETLOGS`. **See Also:** *archivelogRecordSpecifier* on page 4-6 for syntax, and *Oracle Database Backup and Recovery User's Guide* explanations of backup failover for logs and automatic log switching
`BACKUPSET`	Specifies a backup of backup sets. Use this parameter in conjunction with the `DEVICE TYPE sbt` clause to offload backups on disk to tape (as shown in Example 2–21). You cannot back up from tape to tape or from tape to disk: only from disk to disk or disk to tape. If you specify the `DELETE INPUT` option on the `BACKUP BACKUPSET` command, then RMAN deletes all copies of the backup set that exist on disk. For example, if you duplexed a backup to 4 locations, then RMAN deletes all 4 backup sets. The `ALL` option does not add any functionality. RMAN performs **backup set failover** when backing up backup sets. RMAN searches for all available backup copies when the copy that it is trying to back up is corrupted or missing. This behavior is similar to RMAN's behavior when backing up archived redo log files that exist in multiple archiving destinations. If **backup optimization** is enabled when you back up a backup set, and if the identical backup set has already been backed up to the same device type, then RMAN skips the backup of this backup set. **Note:** You can duplex backups of backup sets with `BACKUP COPIES` and `SET BACKUP COPIES`. **Note:** When you use `BACKUP BACKUPSET` command with encrypted backup sets, the backup sets are backed up in their encrypted form. Because `BACKUP BACKUPSET` just copies an already-encrypted backup set to disk or tape, no decryption key is needed during a `BACKUP BACKUPSET` operation. The data is never decrypted during any part of the operation. The `BACKUP BACKUPSET` command can neither encrypt nor decrypt backup sets.
`ALL`	Specifies all backup sets.

Syntax Element	Description
completedTimeSpec	Identifies backup sets according to completion time. **See Also:** *completedTimeSpec* on page 4-11
integer	Specifies backup sets according to primary key. You can obtain the primary keys for backup sets from the output of the LIST BACKUP command.
FROM TAG *tag_name*	Identifies one or more backup sets by tag name. If multiple backup sets have the same tag name, then they are all are backed up. Note that *tag_name* is not case-sensitive.
CONTROLFILECOPY	Specifies one or more control file copies for backups. A control file copy can be created with the BACKUP AS COPY CURRENT CONTROLFILE command or the SQL statement ALTER DATABASE BACKUP CONTROLFILE TO '...'. **Note:** A control file autobackup is not a control file copy.
'*filename*'	Specifies a control file copy by filename.
ALL	Specifies all control file copies.
LIKE '*string_pattern*'	Specifies a control file copy by a filename pattern. The percent sign (%) as a wildcard meaning 0 o r more characters; an underscore (_) is a wildcard meaning 1 character.
copyOfSpec	Makes a backup of previous image copies of datafiles and possibly control files. **See Also:** *copyOfSpec* on page 2-42
CURRENT CONTROLFILE	Specifies the current control file.
DATABASE	Creates a backup of all datafiles in the database. If generating a backup set, then RMAN can include only datafiles and control files: it cannot include archived redo log files. If the *backupSpec* includes datafile 1, and if CONFIGURE CONTROLFILE AUTOBACKUP is OFF, then RMAN automatically includes the control file in the backup. If the instance is started with a server parameter file, then RMAN also includes this parameter file in the backup. If the *backupSpec* includes datafile 1, and if CONFIGURE CONTROLFILE AUTOBACKUP is ON, then RMAN does *not* automatically include the control file in the output. Instead, RMAN generates a separate control file autobackup piece. If the instance is started with a server parameter file, then RMAN includes this parameter file in the autobackup piece. Full database backups should usually be either image copies or compressed backup sets. Image copies are more flexible than backup sets for some purposes (such as use in an incrementally updated backups strategy), and compressed backup sets make more efficient use of storage space, if the CPU overhead involved in creating them is tolerable. **Note:** To force RMAN to include the current control file in the backup when CONTROLFILE AUTOBACKUP is ON, specify the INCLUDE CURRENT CONTROLFILE clause. **See Also:** The TABLESPACE description to learn about backup behavior when the database includes bigfile tablespaces
datafileCopySpec	Specifies the filenames of one or more datafile image copies. **See Also:** *datafileCopySpec* on page 2-43 for details
DATAFILE *datafileSpec*	Specifies a list of one or more datafiles. Refer to description of BACKUP DATABASE for RMAN behavior when datafile 1 is backed up. **See Also:** *datafileSpec* on page 4-15

Syntax Element	Description
RECOVERY AREA	Backups up recovery files created in the current and all previous fast recovery area destinations. Backups can go to SBT and disk. To backup to disk, you must use the TO DESTINATION syntax outlined in *toDestSpec*. Recovery files are full and incremental backup sets, control file autobackups, datafile copies, and archived redo log files. If an archived redo log file is missing or corrupted, then RMAN looks outside of the recovery area for a good copy of the log that it can use for the backup. Flashback logs, the current control file, and online redo log files are *not* backed up. By default, backup optimization is enabled for this command even if the CONFIGURE BACKUP OPTIMIZATION setting is OFF. You can disable backup optimization for BACKUP RECOVERY AREA by specifying FORCE. **Note:** If the fast recovery area is not enabled but has been enabled in the past, then files created in the previous fast recovery area location are backed up.
DB_RECOVERY_FILE_DEST	RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms.
RECOVERY FILES	Backs up *all* recovery files on disk, whether they are stored in the fast recovery area or other locations on disk. The backups can go to SBT or disk. To backup to disk, you must use the TO DESTINATION syntax outlined in *toDestSpec*. Recovery files include full and incremental backup sets, control file autobackups, archived redo log files, and datafile copies. By default, backup optimization is enabled for this command even if the CONFIGURE BACKUP OPTIMIZATION setting is OFF. You can disable backup optimization for RECOVERY FILES by specifying FORCE.
SPFILE	Specifies that the server parameter file should be included in a backup set. The AS COPY option is not supported for server parameter file backups. RMAN backups up the server parameter file currently in use by the target database. RMAN cannot back up the server parameter file when the instance was started with an initialization parameter file. RMAN cannot make incremental backups of the SPFILE.
TABLESPACE *tablespace_name*	Specifies the names of one or more tablespaces. RMAN translates tablespace names internally into a list of datafiles, then backs up all datafiles that are currently part of the tablespaces. If the SYSTEM tablespace (and thus datafile 1) is included in the backup, and if CONTROLFILE AUTOBACKUP is not configured, then RMAN also creates a copy of the control file. You cannot back up locally-managed temporary tablespaces, although you can back up dictionary-managed tablespaces. If the following conditions are met, then RMAN can back up transportable tablespaces that have *not* been made read/write after being transported: ■ The COMPATIBLE initialization parameter is set to 11.0.0 or higher. ■ You are using an Oracle Database 11*g* RMAN client. If any of the preceding conditions is not met, then RMAN automatically skips transportable tablespaces that have not yet been made read/write. Note that if you specify a transportable tablespace explicitly when any of the conditions is not met, then RMAN issues an error saying that the tablespace does not exist. **Note:** If you rename a tablespace, then RMAN detects that the tablespace has changed its name and updates the recovery catalog on the next resynchronization.
backupSpecOperand	The *backupSpecOperand* that follows a *backupSpec* specifies options that apply to the *backupSpec*.

backupSpecOperand

This subclause specifies a variety of options and parameters that affect the *backupSpec* clause. Many subclauses are also used with *backupOperand*. Options

that are not shared in common with *backupOperand* are listed here. Refer to *backupSpecOperand::=* on page 2-25 for the syntax diagram.

Syntax Element	Description
DELETE [ALL] INPUT	Deletes the input files after successfully backing them up. Specify this option only when backing up archived redo log files, datafile copies (COPY OF or DATAFILECOPY), or backup sets. The BACKUP ARCHIVELOG command only backs up one copy of each distinct log sequence number, so if the DELETE INPUT option is used *without* the ALL keyword, RMAN only deletes the copy of the file that it backs up. Specifying the DELETE INPUT option is equivalent to issuing the DELETE command for the input files. When backing up archived redo log files, RMAN uses the configured settings to determine whether an archived redo log can be deleted (CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP). The ALL option applies only to archived redo log files. If you run DELETE ALL INPUT, then the command deletes all copies of corresponding archived redo log files or datafile copies that match the selection criteria in the BACKUP command (as shown in Example 2–19 on page 2-47). For example, if you specify the SEQUENCE *n* clause, then RMAN deletes all archived redo log files with same sequence number *n*. **Note:** The database retains archived redo log files in the fast recovery area as long as possible and deletes them automatically when disk space is required. You can use the BACKUP DELETE INPUT, DELETE ARCHIVELOG, and DELETE OBSOLETE commands to delete log files manually from inside or outside the recovery area. You do not need to specify BACKUP DELETE INPUT when backing up the recovery area because the database automatically deletes log files based on the archived redo log deletion policy and other fast recovery area rules.
FROM TAG *tag_name*	Identifies files by tag name (see Example 2–18 on page 2-47). Only the files that match the tag_name are backed up. Defined in context with several other commands.
INCLUDE CURRENT CONTROLFILE	Creates a snapshot of the current control file and places it into one of backup sets produced by the BACKUP command. **Note:** This option does not apply with AS COPY and results in an error message.

backupTypeSpec

This subclause specifies the form of the BACKUP command output: backup set or image copy. Refer to *backupTypeSpec::=* on page 2-26 for the syntax diagram.

Syntax Element	Description
`AS BACKUPSET`	Creates backup sets on the specified device. This is the default backup type. `AS BACKUPSET` is the only possibility when backing up to tape, and for creating level 1 incremental backups to any destination. Backup sets are RMAN-specific logical structures. The backup set is the smallest unit of a backup. The `FILESPERSET` parameter of the `BACKUP` command determines the maximum number of files in each backup set. Archived redo log files and datafiles are never combined into a single backup set. When using encrypted backups (see "Encryption of Backup Sets" on page 2-21), datafiles from tablespaces with different encryption settings are never written into the same backup set. RMAN cannot back up files with different block sizes into the same backup set. RMAN can back up tablespaces with different block sizes, but puts each differently sized datafile into its own backup set. When **unused block compression** is applied, RMAN reads only the blocks that are currently allocated to a table. RMAN still checks each of the blocks to see whether the header has marked the block as unused. If a block has been unused, it is not written to the backup. Unused block compression is turned on automatically when all of the following five conditions are true: **1.** The `COMPATIBLE` initialization parameter is set to 10.2 or higher. **Note:** If `COMPATIBLE` is set to 10.2, then only tablespaces created with 10.2 compatibility will be optimized to exclude blocks that do not currently contain data. If `COMPATIBLE` is set to 11.0.0 or higher, however, then the first backup that produces backup sets after COMPATIBLE is set to 11.0.0 or higher will update the headers of all locally managed datafiles so that all locally managed datafiles can be optimized. **2.** There are currently no guaranteed restore points defined for the database. **3.** The datafile is locally managed **4.** The datafile is being backed up to a backup set as part of a full backup or a level 0 incremental backup **5.** The backup set is created on disk or Oracle Secure Backup is the media manager. **Note:** When backing up to a media manager that is not Oracle Secure Backup, RMAN copies all the blocks regardless of whether they contain data or not. **Note:** A corrupt unused block is not harmful. This is because when a block is corrupt and RMAN does not read it because of unused block compression, RMAN does not detect the corruption. **See Also:** *Oracle Backup and Recovery User's Guide* for a discussion of null block compression and unused block compression. Each backup set contains at least one **backup piece**, which is an RMAN-specific physical file containing the backed up data. You can also use the `BACKUP` command to generate a **proxy copy**, which is a backup in which the entire data transfer is conducted by a media manager. RMAN only records complete backup sets in the RMAN repository. There are no partial backup sets. When a `BACKUP` command creates backup pieces but does not produce a complete backup set, the backup pieces are discarded. **Note:** You cannot stripe a single backup set across multiple channels. You also cannot stripe a single input file across multiple backup sets. **See Also:** *Oracle Secure Backup Administrator's Guide* to learn how to use Oracle Secure Backup with RMAN

Syntax Element	Description
COMPRESSED	Enables binary compression. RMAN compresses the data written into the backup set to reduce the overall size of the backup set. All backups that create backup sets can create compressed backup sets. Restoring compressed backup sets is no different from restoring uncompressed backup sets. RMAN applies a binary compression algorithm as it writes data to backup sets. This compression is similar to the compression provided by many media manager vendors. When backing up to a *locally attached* tape device, compression provided by the media management vendor is usually preferable to the binary compression provided by `BACKUP AS COMPRESSED BACKUPSET`. Therefore, use uncompressed backup sets and turn on the compression provided by the media management vendor when backing up to locally attached tape devices. You should not use RMAN binary compression and media manager compression together. Some CPU overhead is associated with compressing backup sets. If the target database is running at or near its maximum load, then you may find the overhead unacceptable. In most other circumstances, compressing backup sets saves enough disk space to be worth the CPU overhead.
AS COPY	Creates image copies (rather than backup sets). An image copy is a byte-for-byte identical copy of the original file. You can create image copy backups of datafiles, datafile copies, and archived redo log files. Image copy files can only exist on disk. When using incrementally updated backups, the level 0 incremental must be an image copy backup. By default, `BACKUP` generates backup sets. You can change the default backup type for disk backups to image copies using the `CONFIGURE DEVICE TYPE ... BACKUP TYPE TO COPY` command. RMAN chooses a location for the copy according to the following rules, listed in order of precedence: **1.** `FORMAT` specified on `BACKUP` command for the object being backed up **2.** `FORMAT` specified for the `BACKUP` command **3.** *`fileNameConversionSpec`* setting for `BACKUP` command **4.** `CONFIGURE CHANNEL` *`integer`* `... FORMAT` **5.** `CONFIGURE CHANNEL DEVICE TYPE ... FORMAT` **6.** Platform-specific default `FORMAT` (which includes a `%U` for generating a unique filename) You can create and restore image copy backups with RMAN or use a native operating system command for copying files. When you use RMAN, copies are recorded in the RMAN repository and are more easily available for use in restore and recovery. Otherwise, you must use the `CATALOG` command to add the user-managed copies to the RMAN repository so that RMAN can use them. You cannot make a copy of a backup set, although you can make an image copy of an image copy. To back up a backup set, use `BACKUP BACKUPSET`.

copyOfSpec

This subclause specifies the form of the `BACKUP` command output: backup set or image copy. Refer to *copyOfSpec::=* on page 2-26 for the syntax diagram.

Syntax Element	Description
COPY OF DATABASE	Makes a backup of previous image copies of all datafiles and control files in the database. All datafiles that would normally be included by BACKUP DATABASE are expected to have copies: if not, RMAN signals an error. It is not necessary for all copies to have been produced by a single BACKUP command. If multiple copies exist of datafile, then RMAN backs up the latest. Optionally, specify the copies by tag name (for example, FULL_COLD_COPY). **Note**: The output of this command can be image copies or backup sets.
COPY OF DATAFILE *datafileSpec*	Makes a backup of a previous image copy of one or more datafiles. Specify the datafile by file number (DATAFILE 3) or filename (DATAFILE '?/oradata/trgt/users01.dbf'). You specify the datafile filename and *not* the filename of the copy of the datafile. If more than one copy exists of a given datafile, then RMAN backs up the most recent copy. **Note:** It is not necessary for the image copies that you are backing up to have been created by a single BACKUP command. **Note**: The output of this command can be image copies or backup sets. **See Also:** *datafileSpec* on page 4-15
COPY OF TABLESPACE *tablespace_name*	Makes a backup of previous image copies of the datafiles in one or more specified tablespaces. All datafiles that would normally be included by BACKUP TABLESPACE should have copies: if not, then RMAN signals an error. It is not necessary for all copies to have been produced by a single BACKUP command. If multiple copies exist of datafile, then RMAN backs up the latest. Specify the tablespaces in the list by tablespace name (for example, users) or specify a particular copy by tag name (for example, 0403_CPY_OF_USERS). If you do not specify TAG, then RMAN backs up the most recent datafile copy for each datafile in the tablespace. **Note**: The output of this command can be image copies or backup sets.

datafileCopySpec

This subclause specifies datafile copies. Refer to *datafileCopySpec::=* on page 2-27 for the syntax diagram.

Syntax Element	Description
'*filename*'	Specifies the filenames of one or more datafile image copies.
ALL	Specifies that all datafile image copies should be backed up.
LIKE '*string_pattern*'	Specifies a filename pattern. The percent sign (%) is a wildcard that means zero or more characters; an underscore (_) is a wildcard that means one character.
FROM TAG *tag_name*	Specifies a list of one or more datafile copies, identified by the tag name. If multiple datafile copies with this tag exist, then RMAN backs up only the most current datafile copy of any particular datafile. Tags are not case sensitive.
NODUPLICATES	Prevents the inclusion of identical datafile copies in a backup operation (Example 2–29 on page 2-51). For each set of duplicate datafile copies, the file with the most recent timestamp will be selected.

duration

This subclause specifies datafile copies. Refer to *duration::=* on page 2-27 for the syntax diagram.

Syntax Element	Description
DURATION *hh:mm*	Specifies a maximum time for a backup command to run. If a backup command does not complete in the specified duration, then the backup stops. Without the PARTIAL option, the backup command is considered to have failed if it does not complete in the specified duration, and RMAN reports an error. If the backup command is part of a RUN block, then subsequent commands in the RUN block do not execute.
MINIMIZE {LOAD | TIME}	With disk backups, you can use MINIMIZE TIME run the backup at maximum speed (default), or MINIMIZE LOAD to slow the rate of backup to lessen the load on the system. With MINIMIZE LOAD the backup will take the full specified duration. If you specify TIME, then file most recently backed up is given the lowest priority to back up. This scheduling mechanism provides for the eventual complete backup of the database during successive backup windows, as different datafiles are backed up in round-robin fashion.
PARTIAL	With the PARTIAL option, the command is considered to have completed successfully and no error is reported by RMAN even if the whole backup is not completed in the specified duration. Without the PARTIAL option, the backup command is considered to have failed if it does not complete in the specified duration, and RMAN reports an error. If the backup command is part of a RUN block, then subsequent commands in the RUN block do not execute. Whether PARTIAL is used or not, all backup sets completed before the backup is interrupted are retained and can be used in RESTORE and RECOVER operations.

forRecoveryOfSpec

This subclause specifies that a backup should be used in an incrementally updated backup strategy. You must specify INCREMENTAL LEVEL 1 when specifying FOR RECOVER OF. Refer to *forRecoveryOfSpec::=* on page 2-27 for the syntax diagram.

Syntax Element	Description
FOR RECOVER OF COPY	Specifies that this incremental backup should contain all changes since a previous datafile copy or incremental backup. By default, RMAN creates a differential incremental backup. You must specify CUMULATIVE to force RMAN to create cumulative backups. You should use the WITH TAG clause to separate this incremental backup strategy from the rest of your backup strategies. If you do *not* specify WITH TAG, then RMAN uses the most recent datafile copy as the basis for the incremental backup. A BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF COPY command can create image copies or backup sets. RMAN creates level 0 image copies if no such copies already exist, but otherwise creates backup sets. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to make incrementally updated backups

Syntax Element	Description
WITH TAG *tag_name*	Specifies the tag of the level 0 incremental backup serving as the basis of the backup strategy (see Example 2–20 on page 2-48). By using the BACKUP INCREMENTAL ... WITH TAG syntax, you can create level 1 incremental backups suitable for rolling forward a level 0 image copy. You can then use RECOVER COPY OF ... WITH TAG to apply level 1 backups to this copy. In this way, the strategy continues to roll forward the datafile copy so that media recovery does not need to apply as many changes. This technique is the basis for the Enterprise Manager strategy for disk backups. When WITH TAG is specified, RMAN automatically assigns all new incremental backups in the strategy the same tag as the level 0 datafile copy. RMAN decides which blocks to include in the level 1 backup based on the available incremental backups that have this tag. **Note:** If the BACKUP command has both the TAG and WITH TAG options, then RMAN issues a warning stating that the TAG option will be ignored and that incremental backups will receive the tag specified by WITH TAG. If no level 0 backup with the tag specified in the WITH TAG parameter exists in either the current or parent database incarnation, then FOR RECOVER OF COPY option creates a level 0 datafile copy tagged with the value specified in the WITH TAG parameter.
DATAFILECOPY FORMAT *formatSpec*	Specifies a pattern for naming the output image copies. If you add a datafile to the database, then you do not need to change the backup script because RMAN automatically creates the level 0 datafile copy required by the incrementally updated backup strategy for the newly created file.
FOR RECOVER OF TAG *tag_name*	Backs up the archived redo log files or incremental backups that are intended to recover the level 0 incremental backup specified by *tag_name*. For example, the BACKUP INCREMENTAL LEVEL 1 FOR RECOVER OF TAG wholedb ARCHIVELOG ALL command backs up all archived redo log files needed to recover the level 0 incremental backup tagged wholedb.

notBackedUpSpec

This subclause specifies that RMAN should only back up files that have not yet been backed up for the same device type. Refer to *notBackedUpSpec::=* on page 2-27 for the syntax diagram.

Syntax Element	Description
NOT BACKED UP	Backs up only those files—of the files specified on the BACKUP command—that RMAN has never backed up for the same device type (see Example 2–28 on page 2-50). This subclause is a convenient way to back up new datafiles after adding them to the database. Note that RMAN does not examine datafile checkpoints, but backs up any datafile that is not already backed up. You can also specify NOT BACKED UP when backing up backup sets. Using BACKUP with this clause does manually what an archived redo log deletion policy can do automatically. If you specify the CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP *integer* TIMES command, then a BACKUP ARCHIVELOG ALL command copies all logs unless *integer* backups already exist on the specified device type. If *integer* backups of the logs exist, then the BACKUP command skips the logs. In this way, the archived log deletion policy functions as a default NOT BACKED UP *integer* TIMES clause on the BACKUP command. **Note:** This clause overrides backup optimization (CONFIGURE BACKUP OPTIMIZATION) and archived redo log file deletion policies (CONFIGURE ARCHIVELOG DELETION POLICY).

Syntax Element	Description
integer TIMES	Backs up only those archived redo log files that have not been backed up at least *integer* times. **Note:** You can only specify NOT BACKED UP *integer* TIMES when backing up archived redo log files into backup sets. To determine the number of backups for a file, RMAN only considers backups created on the same device type as the current backup. This option is a convenient way to back up archived redo log files on a specified media. For example, you want to keep at least three copies of each log on tape.
SINCE TIME '*date_string*'	Specifies the date after which RMAN should back up files that have no backups. The *date_string* is either a date in the current NLS_DATE_FORMAT or a SQL date expression such as 'SYSDATE-1'. When calculating the number of backups for a file, RMAN only considers backups created on the same device type as the current backup. This option is a convenient way to back up datafiles that were not backed up during a previous failed backup. For example, you back up the database, but the instance fails halfway through. You can restart the backup with the NOT BACKED UP SINCE TIME clause and avoid backing up those files that you already backed up. If AS BACKUPSET is specified, then this feature is only useful if RMAN generates multiple backup sets during the backup. When determining whether a file has been backed up, the SINCE date is compared with the completion time of the most recent backup. For BACKUP AS BACKUPSET, the completion time for a file in a backup set is the completion time of the entire backup set. In other words, all files in the same backup set have the same completion time.

sizeSpec

This subclause specifies the size of data. Refer to *sizeSpec::=* on page 2-27 for the syntax diagram.

Syntax Element	Description
integer [G | K | M]	Specifies the size of data in gigabytes (G), kilobytes (K), or megabytes (M).

skipSpec

This subclause specifies which files to skip. Refer to *skipSpec::=* on page 2-27 for the syntax diagram.

Syntax Element	Description
SKIP	Excludes datafiles or archived redo log files from the backup according to the criteria specified by the following keywords. **Note:** You can also specify this option in the *backupSpec* clause.
INACCESSIBLE	Specifies that datafiles or archived redo log files that cannot be read due to I/O errors should be excluded from the backup. 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.
OFFLINE	Specifies that offline datafiles should be excluded from the backup.
READONLY	Specifies that read-only datafiles should be excluded from the backup.

Examples

Example 2–15 Backing Up a Database

This example starts the RMAN client from the operating system command line and then connects to a target database using operating system authentication. The BACKUP command backs up all datafiles, the current control file, the server parameter file, and archived redo log files to the default storage device:

```
% rman
RMAN> CONNECT TARGET /
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

Example 2–16 Performing a Cumulative Incremental Backup

This example backs up all blocks changed in the database since the most recent level 0 incremental backup. If no level 0 backup exists when you run a level 1 backup, then RMAN makes a level 0 backup automatically. Any inaccessible files are skipped.

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

Example 2–17 Distributing a Backup Across Multiple Disks

This example backs up tablespaces to two different disks and lets RMAN perform automatic parallelization of the backup. The %U in the FORMAT string is a substitution variable that generates a unique filename for each output image copy.

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK FORMAT '/disk1/%U';
  ALLOCATE CHANNEL dev2 DEVICE TYPE DISK FORMAT '/disk2/%U';
  BACKUP AS COPY
    TABLESPACE SYSTEM, tools, users, undotbs;
}
```

Example 2–18 Identifying Datafile Copies by Tag

In this example, you back up datafile image copies to tape. The BACKUP command locates all datafile copies with the tag LATESTCOPY, backs them up to tape, and names the backups by means of substitution variables. The variable %f specifies the absolute file number, whereas %d specifies the name of the database. After the datafile copies are on tape, the example deletes all image copies with the tag LATESTCOPY.

```
BACKUP
  DEVICE TYPE sbt
  DATAFILECOPY
    FROM TAG 'LATESTCOPY'
  FORMAT 'Datafile%f_Database%d';
DELETE COPY TAG 'LATESTCOPY';
```

Example 2–19 Backing Up and Deleting Archived Redo Log Files

This example assumes that you have two archiving destinations set: /disk2/PROD/archivelog/ and /disk1/arch/. The command backs up one archived redo log for each unique sequence number. For example, if archived redo log 1000 is in both directories, then RMAN only backs up one copy this log. The DELETE INPUT clause with the ALL keyword specifies that RMAN should delete all archived redo log files from both archiving directories after the backup.

```
BACKUP DEVICE TYPE sbt
```

```
  ARCHIVELOG LIKE '/disk%arc%'
  DELETE ALL INPUT;
```

Sample output for the preceding command appears as follows:

```
Starting backup at 12-MAR-07
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=150 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
channel ORA_SBT_TAPE_1: starting archived log backup set
channel ORA_SBT_TAPE_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=4 RECID=4 STAMP=616789551
input archived log thread=1 sequence=5 RECID=5 STAMP=616789551
input archived log thread=1 sequence=6 RECID=6 STAMP=616789554
input archived log thread=1 sequence=7 RECID=7 STAMP=616789731
input archived log thread=1 sequence=8 RECID=8 STAMP=616789825
input archived log thread=1 sequence=9 RECID=10 STAMP=616789901
input archived log thread=1 sequence=10 RECID=12 STAMP=616789985
channel ORA_SBT_TAPE_1: starting piece 1 at 12-MAR-07
channel ORA_SBT_TAPE_1: finished piece 1 at 12-MAR-07
piece handle=0vice0g7_1_1 tag=TAG20070312T105917 comment=API Version 2.0,MMS Version 10.1.0.3
channel ORA_SBT_TAPE_1: backup set complete, elapsed time: 00:00:25
channel ORA_SBT_TAPE_1: deleting archived log(s)
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_4_2z45sgrc_.arc RECID=4
STAMP=616789551
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_5_2z45sgrc_.arc RECID=5
STAMP=616789551
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_6_2z45sl3g_.arc RECID=6
STAMP=616789554
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_7_2z45z2kt_.arc RECID=7
STAMP=616789731
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_8_2z4620sk_.arc RECID=8
STAMP=616789825
archived log file name=/disk1/arch/archiver_1_8_616789153.arc RECID=9 STAMP=616789825
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_9_2z464dhk_.arc RECID=10
STAMP=616789901
archived log file name=/disk1/arch/archiver_1_9_616789153.arc RECID=11 STAMP=616789901
archived log file name=/disk2/PROD/archivelog/2007_03_09/o1_mf_1_10_2z4670gr_.arc RECID=12
STAMP=616789985
archived log file name=/disk1/arch/archiver_1_10_616789153.arc RECID=13 STAMP=616789985
Finished backup at 12-MAR-07

Starting Control File and SPFILE Autobackup at 12-MAR-07
piece handle=c-28643857-20070312-02 comment=API Version 2.0,MMS Version 10.1.0.3
Finished Control File and SPFILE Autobackup at 12-MAR-07
```

Example 2–20 Scripting Incrementally Updated Backups

By incrementally updating backups, you can avoid the overhead of making full image copy backups of datafiles, while also minimizing media recovery time. For example, if you run a daily backup script, then you never have more than one day of redo to apply for media recovery.

Assume you run the following script daily. On first execution, the script creates an image copy backup of the database on disk with the specified tag. On second execution, the script creates a level 1 differential incremental backup of the database. On every subsequent execution, RMAN applies the level 1 incremental backup to the datafile copy and then makes a new level 1 backup.

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

```
    FOR RECOVER OF COPY WITH TAG 'incr_update'
    DATABASE;
}
```

Example 2–21 Backing Up Disk-Based Backup Sets to Tape

Assume your goal is to keep recent backup sets on disk and older backup sets on tape. Also, you want to avoid keeping copies of the same backup set on disk and tape simultaneously. This example backs up backup sets created more than two weeks ago to tape and then deletes the backup pieces from disk.

```
BACKUP
  DEVICE TYPE sbt
  BACKUPSET
    COMPLETED BEFORE 'SYSDATE-14'
  DELETE INPUT;
```

Example 2–22 Duplexing a Database Backup

This example uses the `COPIES` parameter to create two compressed backups of the database, with each backup on a separate disk. The output locations are specified in the `FORMAT` parameter.

```
BACKUP AS COMPRESSED BACKUPSET
  DEVICE TYPE DISK
  COPIES 2
  DATABASE
  FORMAT '/disk1/db_%U', '/disk2/db_%U';
```

Example 2–23 Specifying How Channels Divide Workload

This example explicitly parallelizes a backup by using the `CHANNEL` parameter to specify which channels should back up which files and to which locations.

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt
    PARMS 'ENV=(OB_DEVICE_1=stape1)';
  ALLOCATE CHANNEL ch2 DEVICE TYPE sbt
    PARMS 'ENV=(OB_DEVICE_1=stape2)';
  BACKUP
    (DATABASE        # ch1 backs up database to tape drive stape1
      CHANNEL ch1)
    (ARCHIVELOG ALL
      CHANNEL ch2);  # ch2 backs up archived redo log files to tape drive stape2
}
```

Example 2–24 Creating an Incremental Backup for Refresh of a Standby Database

In this example, your goal is make an incremental backup of the primary database and use it to update an associated standby database. You start the RMAN client, `CONNECT` to the primary database as `TARGET`, and then connect to the recovery catalog. The `BACKUP` command creates an incremental backup of the primary database that can be applied at a standby database to update it with changes since the specified SCN.

```
RMAN> CONNECT TARGET /

connected to target database: PROD (DBID=39525561)

RMAN> CONNECT CATALOG rman@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> BACKUP DEVICE TYPE DISK
```

```
2> INCREMENTAL FROM SCN 404128 DATABASE
3>  FORMAT '/disk1/incr_standby_%U';
```

Example 2–25 Specifying Corruption Tolerance for Datafile Backups

This example assumes a database that contains 5 datafiles. It uses the `SET MAXCORRUPT` command to indicate than no more than 1 corruption should be tolerated in each datafile. Because the `CHECK LOGICAL` option is specified on the `BACKUP` command, RMAN checks for both physical and logical corruption.

```
RUN
{
  SET MAXCORRUPT FOR DATAFILE 1,2,3,4,5 TO 1;
  BACKUP CHECK LOGICAL
    DATABASE;
}
```

Example 2–26 Creating a Consistent Database Backup for Archival Purposes

This example uses a *`keepOption`* to create an archival backup set that cannot be considered obsolete for one year. The example backs up the database, archives the redo in the current online logs to ensure that this new backup is consistent, and backs up only those archived redo log files needed to restore the datafile backup to a consistent state.

The `BACKUP` command also creates a restore point to match the SCN at which this backup will be consistent. Note that the `FORMAT` parameter must be capable of creating multiple backup pieces in multiple backup sets.

```
BACKUP DATABASE
  FORMAT '/disk1/archival_backups/db_%U.bck'
  TAG quarterly
  KEEP UNTIL TIME 'SYSDATE + 365'
  RESTORE POINT Q1FY06;
```

Example 2–27 Exempting Copies from the Retention Policy

The following example copies two datafiles and exempts them from the retention policy forever. (Note that `KEEP FOREVER` requires a recovery catalog.) The control file and server parameter file will also be backed up, even with autobackup off.

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
BACKUP
  KEEP FOREVER
  FORMAT '?/dbs/%U_longterm.cpy'
  TAG LONGTERM_BCK
  DATAFILE 1 DATAFILE 2;
ALTER DATABASE OPEN;
```

Example 2–28 Backing Up Files That Need Backups

Assume that you back up the database and archived redo log files every night to tape by running the following command.

```
BACKUP
  MAXSETSIZE 500M
  DATABASE PLUS ARCHIVELOG;
```

The preceding command sets an upper limit to the size of each backup set so that RMAN produces multiple backup sets. Assume that the media management device fails halfway through the backup and is then restarted. The next day you discover that only half the backup sets completed. In this case, you can run the following command in the evening:

```
BACKUP
  NOT BACKED UP SINCE TIME 'SYSDATE-1'
  MAXSETSIZE 500M
  DATABASE PLUS ARCHIVELOG;
```

With the preceding command, RMAN backs up only files not backed up during in the previous 24 hours. When RMAN determines that a backup from the specified time window is available, it displays output like the following:

```
skipping datafile 1; already backed up on 18-JAN-07
skipping datafile 2; already backed up on 18-JAN-07
skipping datafile 3; already backed up on 18-JAN-07
```

If you place the `NOT BACKED UP SINCE` clause immediately after the `BACKUP` command, then it affects all objects to be backed up. You can also place it after individual *`backupSpec`* clauses to cause only backups for those objects described by the *`backupSpec`* to be subject to the limitation.

Example 2–29 Using NODUPLICATES To Back Up Datafile Copies

This example creates a datafile copy of datafile 2 named `/disk2/df2.cpy`. The example then backs up this datafile copy to the `/disk1` and `/disk3` directories. The `NODUPLICATES` option on the final `BACKUP` command indicates that only one copy of datafile 2 should be backed up.

```
BACKUP AS COPY
  DATAFILE 2
  FORMAT '/disk2/df2.cpy' TAG my_tag;
BACKUP AS COPY
  DATAFILECOPY '/disk2/df2.cpy'
  FORMAT '/disk1/df2.cpy';
BACKUP AS COPY
  DATAFILECOPY '/disk1/df2.cpy'
  FORMAT '/disk3/df2.cpy';
BACKUP
  DEVICE TYPE sbt
  DATAFILECOPY ALL NODUPLICATES; # backs up only copy of datafile 2
```

Example 2–30 Copying Archived Log From Operating System File to ASM

```
BACKUP AS COPY REUSE
ARCHIVELOG LIKE "/ade/b/369893928/oracle/work/arc_dest/arcr_1_11_686060575.arc"
AUXILIARY FORMAT "+RCVAREA";
```

CATALOG

Purpose

Use the CATALOG command to do the following:

- Add backup pieces and image copies on disk to the RMAN repository
- Record a datafile copy as a level 0 incremental backup in the RMAN repository, which enables you to use it as part of an incremental backup strategy

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to manage target database records stored in the catalog

Prerequisites

You must be connected to the target database, which must be mounted or open. If RMAN is connected to a recovery catalog, then the catalog database must be open.

The file that you are cataloging must meet the following conditions:

- It must not exist on an SBT device.
- If it is a user-managed copy, then it must be a datafile copy, control file copy, archived redo log, or backup piece.

Usage Notes

RMAN considers all user-managed backups as image copies. Note that during cataloging, RMAN does not check whether the file was correctly copied by the operating system utility: it just checks the header.

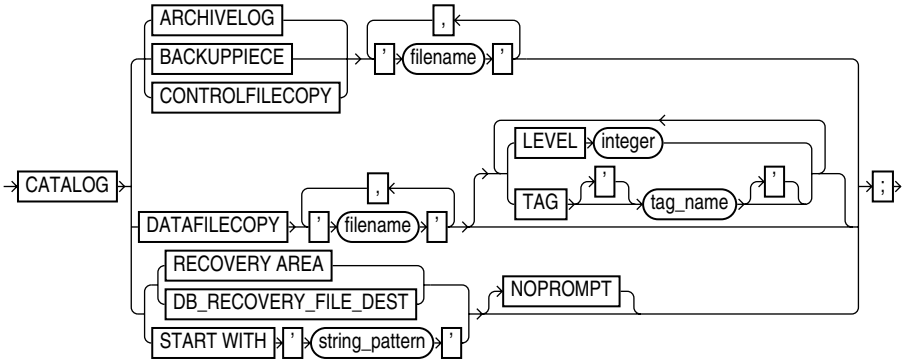
A recovery catalog is required when using RMAN in a Data Guard environment. The recovery catalog supports a unified file namespace for all primary and standby databases with the same DBID but different DB_UNIQUE_NAME values. Thus, the recovery catalog keeps track of database file names for all primary and standby databases, as well as where online redo logs, standby redo logs, tempfiles, archived redo log files, backup sets, and image copies were created.

"RMAN Backups in a Data Guard Environment" on page 2-22 explains how RMAN handles backups made on a different primary and standby databases. In general, tape backups made on one database are accessible to any database in the environment, whereas disk backups are accessible only to the database that created them.

As long as backups are accessible to the connected target database, RMAN commands such as RESTORE and RECOVER behave transparently across different databases. You can manually transfer a disk backup from one host in the environment to another host and then catalog the backup. If a backup is on shared disk, then you can use CHANGE RESET DB_UNIQUE_NAME to associate the backup with a new database.

Syntax

***catalog*::=**



Semantics

Syntax Element	Description
`ARCHIVELOG 'filename'`	Specifies the filename of an archived redo log to be added to the RMAN repository. **Note:** This command does not catalog foreign archived redo log files, which are redo logs received by a logical standby database for a LogMiner session. Unlike normal archived redo log files, foreign archived redo log files have a different DBID.

Syntax Element	Description
BACKUPPIECE '*filename*'	Specifies the name of a backup piece to be added to the RMAN repository (see Example 2–31 on page 2-55). The backup piece must be on disk. RMAN verifies the backup piece header before cataloging it. RMAN can catalog a backup piece from a previous database incarnation. You may choose to catalog backup pieces in the following situations: ▪ You copy or move a backup piece with an operating system utility and want it to be usable by RMAN. ▪ The RMAN metadata for the backup piece was removed, but the backup piece still exists. This situation can occur if you ran the DELETE command on a backup piece that was only temporarily unavailable. ▪ You make a NOCATALOG backup on one database host in a Data Guard environment and move the backup piece to the same location on a different database host. In this case, the recovery catalog has no record of the original backup piece. ▪ You do not use a recovery catalog and must re-create the control file, thereby losing all RMAN repository data. Cataloging your backups makes them available again. ▪ When control file autobackup is disabled, you back up the control file and then back up the archived redo log files. You can restore and mount the control file, but must catalog the backup pieces containing the archived redo log files backed up after the control file. If you specify a list of backup pieces, then RMAN attempts to catalog all pieces in the given list even if some of them fail. Cataloging a backup piece creates a new row in V$BACKUP_PIECE. A backup set is only usable when *all* backup pieces are cataloged because otherwise it is only in a partially available state. **Note:** If RMAN creates a server parameter file backup when the COMPATIBLE parameter of the database is set to 11.0.0 or higher, then the backup is associated with this database. In this case, even if you connect RMAN to a different database and explicitly catalog the backup piece, the DB_UNIQUE_NAME associated with this backup does not change. For example, if RMAN backs up the server parameter file of the database with DB_UNIQUE_NAME 'NEWYORK' when COMPATIBLE is 11.0.0, then RMAN cannot use the server parameter file backup created at database NEWYORK to restore the server parameter file on database BOSTON.
CONTROLFILECOPY '*filename*'	Specifies the filename of a control file copy to be added to the RMAN repository. The control file copy can be a normal or standby control file copy created by one of the following commands ▪ The RMAN command BACKUP AS COPY CURRENT CONTROLFILE ▪ The SQL statement ALTER DATABASE BACKUP CONTROLFILE ▪ The SQL statement ALTER DATABASE CREATE STANDBY CONTROLFILE **Note:** RMAN can automatically convert a primary database control file backup to a standby control file during a restore operation.
DATAFILECOPY '*filename*'	Specifies the filename of a datafile copy to be added to the RMAN repository (see Example 2–31 on page 2-55). You can create datafile copies with the RMAN BACKUP AS COPY command or with operating system utilities used in conjunction with ALTER TABLESPACE BEGIN/END BACKUP.
LEVEL *integer*	Specifies that the datafile copy should be recorded as a level 0 incremental backup (0 is the only valid value of LEVEL). You can perform incremental backups by using this datafile copy as the base level 0 backup.
TAG *tagname*	Specifies a tag for the datafile copy.

Syntax Element	Description
RECOVERY AREA	Catalogs all valid backup sets, datafile copies, and archived redo log files in the fast recovery area (see Example 2–33 on page 2-56). RMAN must be connected to a database as TARGET. The target database must be mounted or open. The keywords RECOVERY AREA and DB_RECOVERY_FILE_DEST are exact synonyms. **Note:** This command also catalogs foreign archived redo log files, which are archived redo log files received by logical standby for a LogMiner session, if they exist in the fast recovery area.
DB_RECOVERY_FILE_DEST	The keywords RECOVERY AREA and DB_RECOVERY_FILE_DEST are exact synonyms.
START WITH '*string_pattern*'	Catalogs all valid backup sets, datafile and control file copies, and archived redo log files whose name start with *string_pattern*. The string pattern can be an ASM disk group, Oracle-managed files directory, or part of a file name (see Example 2–32 on page 2-55). RMAN reports any files in the disk location that it cannot catalog. RMAN must be connected to a mounted target database. If the string pattern specifies a filename, then it matches the left part of the filename pattern. For example, /tmp/arc matches everything in directory /tmp/arc_dest and /tmp/archive/january as well as file /tmp/arc.cpy. **Note:** You cannot use wildcard characters in the string pattern, only a strict prefix.
NOPROMPT	Suppresses the confirmation prompt. By default, RMAN prompts after every match.

Examples

Example 2–31 Cataloging a Datafile Copy as an Incremental Backup

Assume that you used a Linux utility to back up the users01.dbf datafile to /disk2/backup/users01.bak. This example catalogs the datafile copy as an incremental level 0 backup and then lists all copies.

```
CATALOG DATAFILECOPY '/disk2/backup/users01.bak' LEVEL 0;
LIST COPY;
```

Example 2–32 Cataloging Multiple Copies in a Directory

This example catalogs a directory full of archived redo log files that were copied into the /disk2/archlog directory with an operating system utility. The example includes sample output.

```
CATALOG START WITH '/disk2/archlog' NOPROMPT;

searching for all files that match the pattern /disk2/archlog

List of Files Unknown to the Database
=====================================
File Name: /disk2/archlog/o1_mf_1_10_24trtc7s_.arc
File Name: /disk2/archlog/o1_mf_1_11_24trtg7s_.arc
File Name: /disk2/archlog/o1_mf_1_12_24trtk84_.arc
File Name: /disk2/archlog/o1_mf_1_13_24trtn85_.arc
File Name: /disk2/archlog/o1_mf_1_14_24trtq84_.arc
File Name: /disk2/archlog/o1_mf_1_15_24trtt84_.arc
File Name: /disk2/archlog/o1_mf_1_16_24trtx84_.arc
File Name: /disk2/archlog/o1_mf_1_17_24trv085_.arc
```

```
File Name: /disk2/archlog/o1_mf_1_18_24trv385_.arc
File Name: /disk2/archlog/o1_mf_1_19_24trv685_.arc
cataloging files...
cataloging done

List of Cataloged Files
=======================
File Name: /disk2/archlog/o1_mf_1_10_24trtc7s_.arc
File Name: /disk2/archlog/o1_mf_1_11_24trtg7s_.arc
File Name: /disk2/archlog/o1_mf_1_12_24trtk84_.arc
File Name: /disk2/archlog/o1_mf_1_13_24trtn85_.arc
File Name: /disk2/archlog/o1_mf_1_14_24trtq84_.arc
File Name: /disk2/archlog/o1_mf_1_15_24trtt84_.arc
File Name: /disk2/archlog/o1_mf_1_16_24trtx84_.arc
File Name: /disk2/archlog/o1_mf_1_17_24trv085_.arc
File Name: /disk2/archlog/o1_mf_1_18_24trv385_.arc
File Name: /disk2/archlog/o1_mf_1_19_24trv685_.arc
```

Example 2–33 Cataloging Files in the Fast Recovery Area

This example catalogs all files in the currently enabled fast recovery area without prompting the user for each one. As shown in the sample output, RMAN displays a message if it finds no files to catalog.

```
CATALOG RECOVERY AREA;

searching for all files in the recovery area
no files found to be unknown to the database
```

Example 2–34 Cataloging a Backup Piece

Assume that you use an operating system utility to copy a backup piece from one location to another. This example catalogs the backup piece in the new location (sample output included):

```
CATALOG BACKUPPIECE '/disk1/c-874220581-20061128-01';

using target database control file instead of recovery catalog
cataloged backup piece
backup piece handle=/disk1/c-874220581-20061128-01 RECID=12 STAMP=607695990
```

CHANGE

Purpose

Use the CHANGE command to perform the following tasks:

- Update the availability status of backups and copies recorded in the RMAN repository
- Change the priority of or close failures recorded in the automatic diagnostic repository
- Update the DB_UNIQUE_NAME recorded in the recovery catalog for the target database
- Associate the backup of a database in a Data Guard environment with a different database in the environment

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to change the availability status of a backup or copy

Prerequisites

RMAN must be connected as TARGET to a database instance, which must be started.

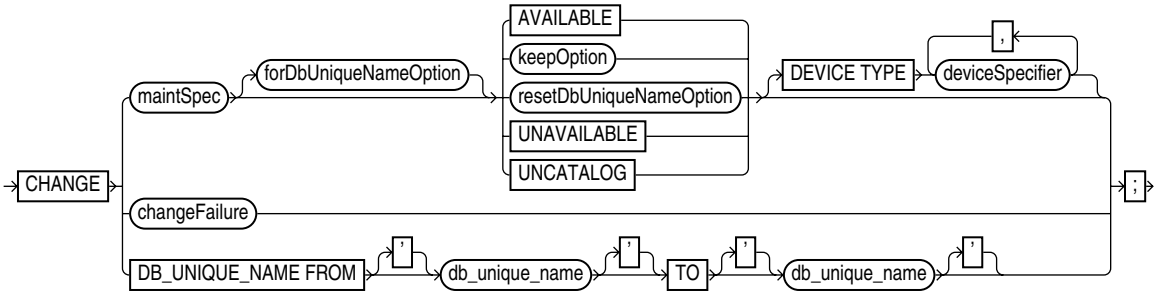
Usage Notes

"RMAN Backups in a Data Guard Environment" on page 2-22 explains the difference between the association and accessibility of a backup. In a Data Guard environment, the database that creates a backup or copy is associated with the file. You can use maintenance commands such as CHANGE, DELETE, and CROSSCHECK for backups when connected to any database in the Data Guard environment as long as the backups are accessible. In general, RMAN considers tape backups created on any database as accessible to all databases in the environment, whereas disk backups are accessible only to the database that created them.

For example, suppose that you connect RMAN as TARGET to standby database standby1 and back it up to tape and disk. If the tape drive becomes unavailable, then you can connect RMAN as TARGET to any primary or standby database in the Data Guard environment to change the status of the tape backup to UNAVAILABLE. After the tape drive is repaired, you can connect RMAN as TARGET to any database to change the status of the tape backup back to AVAILABLE. However, if the disk backup is accidentally removed by an operating system utility, then RMAN can only change the status of the disk backup when connected as TARGET to standby1.

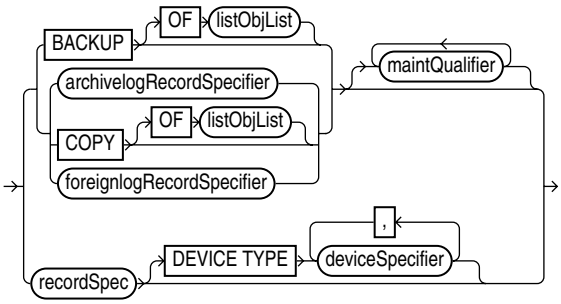
Syntax

change::=



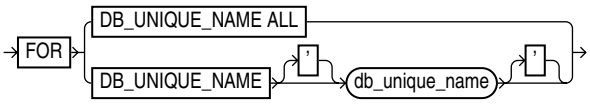
(*maintSpec*::= on page 2-58, *forDbUniqueNameOption*::= on page 4-20, *keepOption*::= on page 4-26, *deviceSpecifier*::= on page 4-16)

maintSpec::=

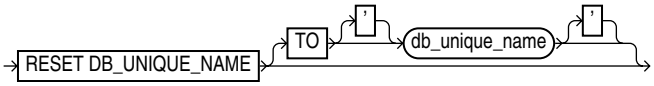


(*listObjList*::= on page 4-29, *archivelogRecordSpecifier*::= on page 4-6, *maintQualifier*::= on page 4-32, *recordSpec*::= on page 4-37, *deviceSpecifier*::= on page 4-16)

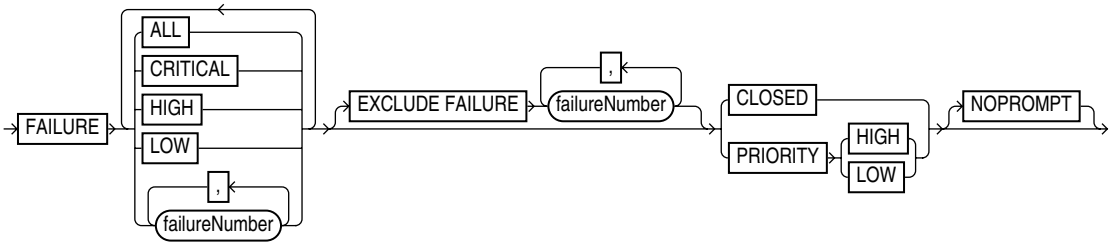
forDbUniqueNameOption::=



resetDbUniqueNameOption::=



changeFailure::=



Semantics

change

This clause enables you to change the status of RMAN repository records. To obtain the primary keys of RMAN repository records whose status you want to change, run a LIST command or query the recovery catalog views.

Syntax Element	Description
maintSpec	Specifies which files you want to CHANGE. **See Also:** *maintSpec* on page 4-34 for descriptions of the options in this clause.
forDbUniqueNameOption	Changes the metadata for objects that are exclusively associated with the specified DB_UNIQUE_NAME in a Data Guard environment. **See Also:** *forDbUniqueNameOption* on page 4-20 for descriptions of the options in this clause.
AVAILABLE	Changes the status of a backup or copy to AVAILABLE in the repository. RMAN searches for the file and verifies that it exists. This feature is useful when a previously unavailable file is made available again. You can also use this option to alter the repository status of backups and copies from prior incarnations. This is the only CHANGE option that requires either a manual or automatic maintenance channel. A maintenance channel is not required, however, when CHANGE ... AVAILABLE is used with a file that is disk only (that is, an ARCHIVELOG, DATAFILECOPY, or CONTROLFILECOPY). If you use CHANGE ... AVAILABLE on files that are not disk-only, and have objects created on device types that are not configured for automatic channels, then issue manual maintenance commands on these channels. For example, if you created a backup on an sbt channel, but have only a DISK channel automatically configured, then you must manually allocate an sbt channel before CHANGE ... AVAILABLE can operate on the backup. If you execute CHANGE ... AVAILABLE for a file in a Data Guard environment, then RMAN attempts to crosscheck the file before updating its status to AVAILABLE. If the file is inaccessible, then RMAN prompts you to perform the same operation when connected as TARGET to the database associated with the file. If the file is accessible, then RMAN updates the status as requested. **Note:** You can view the status of backups in the LIST output or recovery catalog views. **Note:** CHANGE ... AVAILABLE is not valid for foreign archived redo log files, which are received by a logical standby database for a LogMiner session. Unlike normal archived redo log files, foreign archived redo log files have a different DBID.
keepOption	Changes the exemption status of a backup or copy in relation to the configured retention policy. For example, specify CHANGE ... NOKEEP to remove the KEEP attributes for a backup, making it subject to the backup retention policy. The KEEP FOREVER clause requires use of a recovery catalog (see Example 2–37 on page 2-63). The RESTORE POINT option is not valid with CHANGE. You cannot use CHANGE ... UNAVAILABLE or KEEP attributes for files stored in the fast recovery area. **See Also:** *keepOption* on page 4-26
resetDbUniqueNameOption	Associates the files in *maintSpec* with a different database in a Data Guard environment. **See Also:** *resetDbUniqueNameOption* on page 2-61

Syntax Element	Description
UNAVAILABLE	Changes the status of a backup or copy to UNAVAILABLE in the repository (see Example 2–35 on page 2-63). View the status in the LIST output or recovery catalog views. This option is useful when a file cannot be found or has migrated offsite. RMAN does not use a file that is marked UNAVAILABLE in a RESTORE or RECOVER command. If the file is later found or returns to the main site, then use the AVAILABLE option to update its status. The UNAVAILABLE option is also useful when you do not want a specific backup or copy to be eligible to be restored but also do not want to delete it, or when you want to alter the repository status of backups and copies from prior incarnations. CHANGE ... UNAVAILABLE is not valid for files in the fast recovery area. This command is also not valid for foreign archived redo log files, which are received by a logical standby database for a LogMiner session. Unlike normal archived redo log files, foreign archived redo log files have a different DBID. **Note:** If you execute CHANGE ... UNAVAILABLE for a file in a Data Guard environment, then RMAN does not attempt to crosscheck the file before updating its status to UNAVAILABLE. RMAN updates the status as requested regardless of whether the file physically exists.
UNCATALOG	Removes references to a datafile copy, backup piece, or archived redo log from the recovery catalog, and updates records in the target control file to status DELETED (see Example 2–36 on page 2-63). Note that the CHANGE ... UNCATALOG command does not touch physical backups and copies. Use this command to notify RMAN when a file is deleted by some means other than a DELETE command. If you execute CHANGE ... UNCATALOG for a file in a Data Guard environment, then RMAN does not attempt to crosscheck the file before removing its metadata from the recovery catalog. RMAN removes the metadata as requested regardless of whether the file physically exists. **Caution:** If you resynchronize from a backup control file, or upgrade the recovery catalog, then records previously removed from the RMAN repository with CHANGE ... UNCATALOG may reappear in the recovery catalog.
DEVICE TYPE *deviceSpecifier*	Executes the CHANGE for the specified device type only (see *deviceSpecifier* on page 4-16). This option is valid only if you have configured automatic channels and have not manually allocated channels. For example, if you run CHANGE UNCATALOG ... DEVICE TYPE DISK, then RMAN only uncatalogs files on disk.
changeFailure	Specifies changes for failures recorded by the Data Recovery Advisor.

Syntax Element	Description
DB_UNIQUE_NAME FROM *db_unique_name* TO *db_unique_name*	Updates the metadata in the recovery catalog to reflect a new DB_UNIQUE_NAME for a database in a Data Guard environment. The first value specifies the old DB_UNIQUE_NAME for the database currently recorded in the recovery catalog, whereas the second specifies the new DB_UNIQUE_NAME. RMAN must be connected to a recovery catalog and a mounted target database. The target database should not have the DB_UNIQUE_NAME specified in the FROM *db_unique_name* parameter; otherwise, RMAN signals an error. Typically, you use this command after you have already changed the DB_UNIQUE_NAME initialization parameter of a database and must update its metadata in the recovery catalog. In general, you should run this command before performing any other RMAN operations on a renamed database. The recommended practice is to execute LIST DB_UNIQUE_NAME before CHANGE DB_UNIQUE_NAME. Assume that you have changed the DB_UNIQUE_NAME initialization parameter for a standby database from standby_old to standby_new. Typically, you execute CHANGE DB_UNIQUE_NAME in the following scenarios: ■ A LIST DB_UNIQUE_NAME command shows the old DB_UNIQUE_NAME value but does *not* show the new one (see Example 2–40 on page 2-64). ■ A LIST DB_UNIQUE_NAME command shows both the old *and* new DB_UNIQUE_NAME values. When RMAN connects as TARGET to a database with an unrecognized DB_UNIQUE_NAME, RMAN implicitly registers the instance as a new database. For this reason, LIST DB_UNIQUE_NAME command can show both the old and new names (in this example, standby_old and standby_new) for a database whose DB_UNIQUE_NAME initialization parameter has been changed. In the scenario in which only the old name is listed, execute CHANGE DB_UNIQUE_NAME FROM standby_old TO standby_new so that RMAN changes the DB_UNIQUE_NAME for standby_old to standby_new in the recovery catalog. In the scenario in which both the old and new names are listed, RMAN automatically executes the following commands when you run CHANGE DB_UNIQUE_NAME FROM standby_old TO standby_new: **1.** CHANGE ARCHIVELOG ALL FOR DB_UNIQUE_NAME standby_old RESET DB_UNIQUE_NAME **2.** CHANGE BACKUP FOR DB_UNIQUE_NAME standby_old RESET DB_UNIQUE_NAME **3.** UNREGISTER DB_UNIQUE_NAME standby_old Thus, RMAN changes the association of all backups for the DB_UNIQUE_NAME specified in the FROM clause to the DB_UNIQUE_NAME specified in the TO clause.

resetDbUniqueNameOption

This clause enables you to associate backups made on one database in a Data Guard environment with a different database in the environment. The following table explains the RMAN behavior when different options are specified with RESET DB_UNIQUE_NAME.

Table 2–2 RESET DB_UNIQUE_NAME Options

TO *db_unique_name*	FOR DB_UNIQUE_NAME	RMAN Behavior
No	No	RMAN associates the *maintSpec* files with the target database. RMAN also changes the association of all backups that are not associated with any database. Typically, you would execute CHANGE with these options after upgrading to an Oracle Database 11*g* recovery catalog schema, so that you can associate the backups with the target database.
Yes	No	RMAN associates the *maintSpec* files with the database indicated by the TO *db_unique_name*. RMAN also changes the association of all backups that are not associated with any database.
No	Yes	RMAN restricts its operations to *maintSpec* files that are associated with the database in the FOR DB_UNIQUE_NAME clause, and then associates these files with the target database.
Yes	Yes	RMAN restricts its operations to *maintSpec* files that are associated with the database in the FOR DB_UNIQUE_NAME clause, and then associates these files with the database specified by TO *db_unique_name*.

Syntax Element	Description
RESET DB_UNIQUE_NAME	Associates the files in *maintSpec* with the target database (see Example 2–39 on page 2-64). Table 2–2 explains the RMAN behavior when different options are specified. When changing the association of the files from one database to another database, RMAN deletes the duplicate names from the recovery catalog. For example, if you change the association of datafile copy /d1/df1.bak from database standby1 to database prod, then the recovery catalog has only one record for this file rather than two. Use caution when specifying the RESET DB_UNIQUE_NAME option because you cannot undo the effect of this command. For example, after you have change the association of the files associated with database standby1 to database prod, the recovery catalog does not retain historical metadata about the database with which these files were previously associated. However, you can unregister database standby1 and connect RMAN again to standby1, but in this case the recovery catalog will be updated with all metadata from the standby1 control file.
TO *db_unique_name*	Associates the files in *maintSpec* with the specified database in a Data Guard environment.

changeFailure

This clause enables you to change the status of failures. Use the LIST FAILURE command to show the list of failures.

Syntax Element	Description
FAILURE	Enables you to change priority or close failures recorded in the Automatic Diagnostic Repository. By default RMAN prompts for confirmation before performing the requested change. The target database to which RMAN is connected must be a single-instance database and must not be a physical standby database.
ALL	Changes only open failures.
CRITICAL	Changes only critical failures.
HIGH	Changes only failures with HIGH priority.
LOW	Changes only failures with LOW priority.
failnum	Changes only the specified failure.
EXCLUDE FAILURE *failnum*	Excludes the specified failures from the change.

Examples

Example 2–35 Updating Backups to Status UNAVAILABLE

Assume that you have temporarily moved backup set 4 to a different location because of a space issue on disk. The backup, which has the key 4, is still listed as available:

```
RMAN> LIST BACKUP SUMMARY;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------- ---
1       B  A  A DISK        24-FEB-07       1       1       NO         TAG20070427T115348
3       B  A  A DISK        24-MAR-07       1       1       NO         TAG20070427T115452
4       B  F  A DISK        24-APR-07       1       1       NO         TAG20070427T115456
```

You do not want to uncatalog the backup because you plan to move it back to its original location when more disk space is available. Thus, you make the backup unavailable as follows (sample output included):

```
RMAN> CHANGE BACKUPSET 4 UNAVAILABLE;

changed backup piece unavailable
backup piece handle=/disk2/backup/c-3257893776-20070424-00 RECID=4 STAMP=588858897
Changed 1 objects to UNAVAILABLE status
```

Example 2–36 Uncataloging and Recataloging Archived Redo Log Files

In this example, you move all archived redo log files to a new directory, uncatalog them, and then recatalog them in the new location:

```
RMAN> HOST '/bin/mv $ORACLE_HOME/dbs/*.arc /disk2/archlog/';
RMAN> CHANGE ARCHIVELOG ALL UNCATALOG;
RMAN> CATALOG START WITH '/disk2/archlog' NOPROMPT;
```

Example 2–37 Changing a Database Backup into an Archival Backup

Assume your goal to is to change a database backup into an archival backup, which you plan to store offsite. Because the backup is consistent and requires no recovery, you do not need to store archived redo log files with the backup. The example uses the `CHANGE ... KEEP FOREVER` command to specify that the backup should never become obsolete.

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

recovery catalog database Password: password

RMAN> CHANGE BACKUP TAG 'consistent_db_bkup' KEEP FOREVER;
```

Example 2–38 Changing the Status of a Failure

In the following example, the `LIST FAILURE` command shows that a datafile has corrupt blocks. The failure number is 5 and has a priority of `HIGH`. This datafile contains nonessential data, so you decide to change the priority of this failure to low.

```
RMAN> LIST FAILURE;

List of Database Failures
Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
5          HIGH     OPEN      11-DEC-06     datafile 8 contains corrupt blocks

RMAN> CHANGE FAILURE 5 PRIORITY LOW;

List of Database Failures
Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
5          HIGH     OPEN      11-DEC-06     datafile 8 contains corrupt blocks

Do you really want to change the above failures (enter YES or NO)? YES
changed 1 failures to LOW priority
```

Example 2–39 Associating Backups with a New Database in a Data Guard Environment

Assume that `standby1`, `standby2`, and `standby3` are standby databases associated with primary database is `prod`. This example assumes that RMAN is connected to target database `prod` and a recovery catalog.

You are planning to remove `standby1` from your environment, so you want to associate the `standby1` backups with your primary database. You are also planning to remove `standby3` from your environment, so you want to associate the `standby3` backups with `standby2`. You execute the following commands:

```
CHANGE BACKUP FOR DB_UNIQUE_NAME standby1 RESET DB_UNIQUE_NAME;
CHANGE BACKUP FOR DB_UNIQUE_NAME standby3 RESET DB_UNIQUE_NAME TO standby2;
```

Example 2–40 Updating a DB_UNIQUE_NAME in the Recovery Catalog

Assume that a standby database has the `DB_UNIQUE_NAME` initialization parameter setting of `dgrdbms4`, which you decide to change to `sfrdbms4`. You shut down the standby instance, change the `DB_UNIQUE_NAME` initialization parameter to `sfrdbms4`, and restart the standby instance.

Later, to update the recovery catalog to reflect the changed unique name of the standby database, you connect RMAN to the primary database and recovery catalog, and then execute the `CHANGE` command as follows:

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

recovery catalog database Password: password

RMAN> CHANGE DB_UNIQUE_NAME FROM dgrdbms4 TO sfrdbms4;
```

CONFIGURE

Purpose

Use the CONFIGURE command to create or change a persistent configuration affecting RMAN backup, restore, duplication, and maintenance jobs on a particular database. A configuration is in effect for any RMAN session on this database until the configuration is explicitly cleared or changed. You can use the SHOW command to display the configurations for one or more databases.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to configure the RMAN environment

Additional Topics

- Prerequisites
- Usage Notes
- Syntax
- Semantics
- Examples

Prerequisites

Execute this command only at the RMAN prompt.

Unless you specify the FOR DB_UNIQUE_NAME clause, an RMAN connection to a target database is required. The target database must be mounted or open.

Usage Notes

The CONFIGURE command always stores a configuration for a target database in the target database control file. If you use RMAN with a recovery catalog, then RMAN also stores persistent configuration settings for each registered database in the catalog.

Default RMAN Configuration Settings

The RMAN CONFIGURE settings have defaults. You can return to the default for any CONFIGURE command by rerunning the command with the CLEAR option, but you cannot clear individual parameters in this way. For example, the following command is valid:

```
CONFIGURE CHANNEL DEVICE TYPE sbt CLEAR
```

However, the following command is invalid:

```
CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 5M CLEAR
```

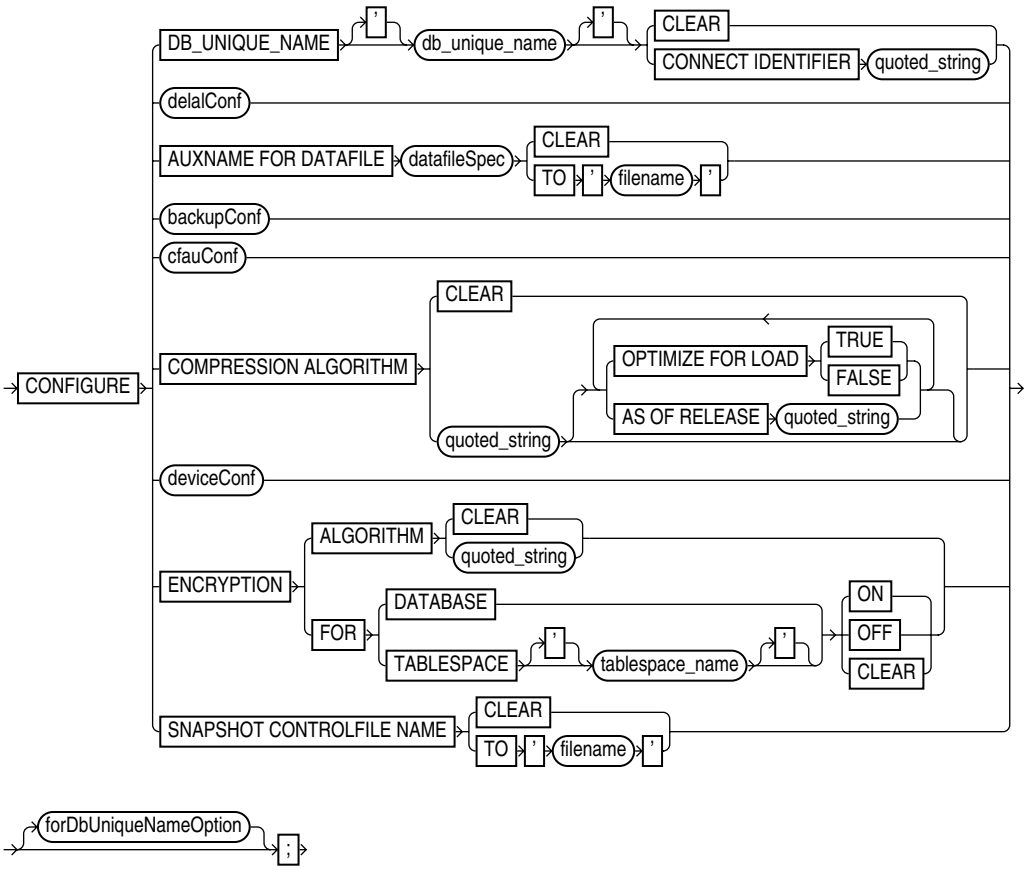
RMAN Configuration in a Data Guard Environment

In a Data Guard environment, Oracle recommends that you always use RMAN with a recovery catalog. You can use the CONFIGURE command to create persistent RMAN configurations for any individual primary or standby database in the Data Guard environment, with the exception of settings for backup retention policy, tablespace exclusion, and auxiliary names. Thus, the primary and standby databases can have different channel configurations, control file autobackup locations, and so on.

You can use the `FOR DB_UNIQUE_NAME` clause to configure a database to which RMAN is not connected as `TARGET`. You can use `CONFIGURE DB_UNIQUE_NAME` to make a new physical standby database known to the recovery catalog and implicitly register it.

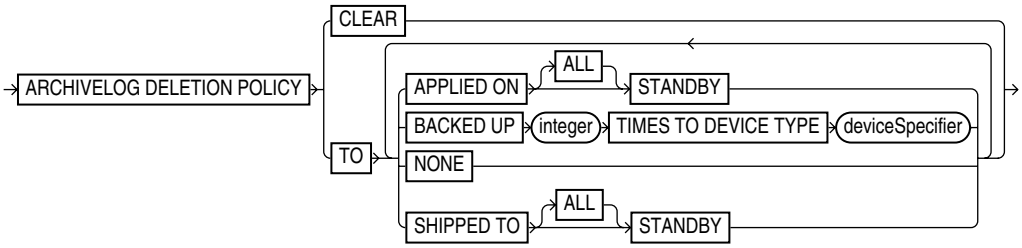
Syntax

configure::=



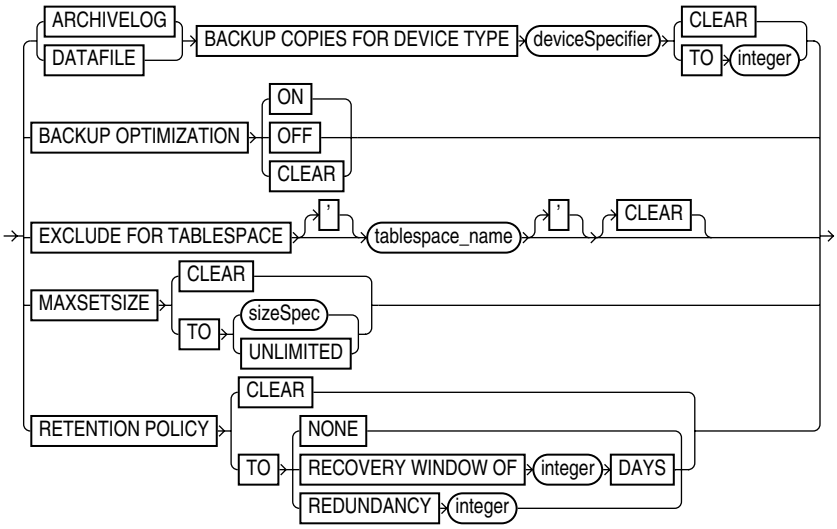
(*datafileSpec::=* on page 4-15, *backupConf::=* on page 2-67, *cfauConf::=* on page 2-67, *deviceConf::=* on page 2-67, *forDbUniqueNameOption::=* on page 4-20)

delalConf::=



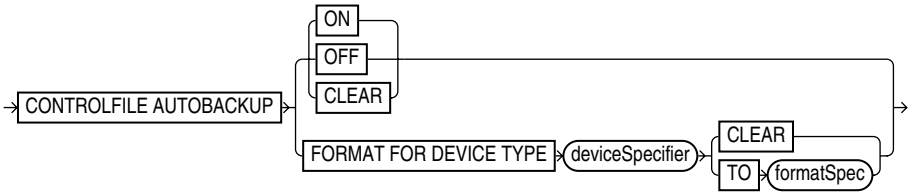
(*deviceSpecifier::=* on page 4-16)

backupConf::=



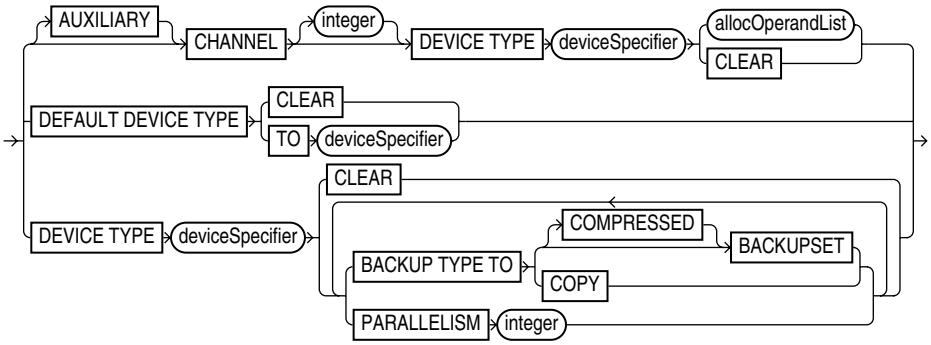
(*deviceSpecifier::=* on page 4-16, *sizeSpec::=* on page 2-67)

cfauConf::=



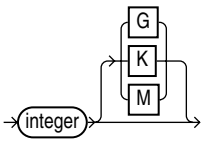
(*deviceSpecifier::=* on page 4-16, *formatSpec::=* on page 4-23)

deviceConf::=

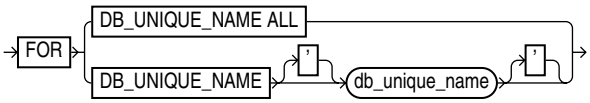


(*deviceSpecifier::=* on page 4-16, *allocOperandList::=* on page 4-2)

sizeSpec::=



forDbUniqueNameOption::=



Semantics

configure

Syntax Element	Description
DB_UNIQUE_NAME *db_unique_name* {CLEAR | CONNECT IDENTIFIER '*connect_string*'}	Specifies the net service name for the physical standby database specified by DB_UNIQUE_NAME. The CONNECT IDENTIFIER string must not include the database user name and password. RMAN must also be connected to the primary database as TARGET. RMAN must be connected to a recovery catalog. When you run the RESYNC CATALOG FROM DB_UNIQUE_NAME command, databases in a Data Guard environment use the net service name to connect with the *db_unique_name* database. For example, assume that a standby database has the unique name standby1 and the net service name sby1. You connect RMAN as TARGET to the primary database and execute CONFIGURE DB_UNIQUE_NAME 'standby1' CONNECT IDENTIFIER 'sby1'. Every primary and standby database in the environment will use the net service name sby1 when it needs to make an Oracle Net connection to standby1. **Note:** When the target database needs to connect to other standby or primary databases, it connects as the SYS user by using the existing Data Guard authentication mechanisms. Suppose that you recently connected RMAN as TARGET to the primary database and used CONFIGURE ... FOR DB_UNIQUE_NAME standby_new to configure backup settings for standby database standby_new. However, you have not yet connected RMAN as TARGET to standby_new. In this case, you can execute RESYNC CATALOG FROM DB_UNIQUE_NAME standby_new. The primary database uses the connect identifier to make an Oracle Net connection to the standby database. When you later connect RMAN to the standby database, RMAN pushes the configuration from the recovery catalog to the mounted control file. **Note:** If the database specified by CONFIGURE DB_UNIQUE_NAME is not registered in the recovery catalog, then RMAN implicitly registers it.
delalConf	Configures an archived redo log deletion policy.

Syntax Element	Description
AUXNAME FOR DATAFILE *datafileSpec* TO '*filename*'	Configures the auxiliary filename for the specified target datafile to '*filename*' (see Example 2–45 on page 2-83). Specify CLEAR to unspecify the auxiliary filename. If you are performing TSPITR or using DUPLICATE, then you can set AUXNAME to preconfigure the filenames for use on the auxiliary database without manually specifying the auxiliary filenames during the procedure. You cannot use CONFIGURE AUXNAME for recovery set, you must use SET NEWNAME. For example, use this command during TSPITR if the datafiles are on raw disk and you must restore auxiliary datafiles to raw disk for performance reasons. Typically, you set the AUXNAME parameter in TSPITR for the datafiles of the SYSTEM and SYSAUX tablespaces and the tablespaces containing rollback or undo segments. Do not overlay files which are in use by the production database and 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, CONFIGURE AUXNAME is an alternative to SET NEWNAME. The difference is that after you set the AUXNAME the first time, you do not need to reset the filename when you issue another DUPLICATE command: the AUXNAME setting remains in effect until you issue CONFIGURE AUXNAME ... CLEAR. In contrast, you must reissue the SET NEWNAME command every time you execute the DUPLICATE command. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to perform RMAN TSPITR, and *Oracle Database Backup and Recovery User's Guide* to learn how to duplicate a database with RMAN
backupConf	Configures default backup options such as duplexing, optimization, excluding tablespaces, backup set sizes, and retention policies.
cfauConf	Configures control file autobackup settings.

Syntax Element	Description
COMPRESSION ALGORITHM *'algorithm_name'*	Specifies the algorithm that RMAN uses to create compressed backup sets. The default compression algorithm setting is BASIC and does not require the Advanced Compression Option. If, however, you have enabled the Oracle Database 11*g* Release 2 Advanced Compression Option, you can choose from the following compression levels: ■ HIGH – Best suited for backups over slower networks where the limiting factor is network speed ■ MEDIUM –Recommended for most environments. Good combination of compression ratios and speed ■ LOW - Least impact on backup throughput and suited for environments where CPU resources are the limiting factor. **Note:** The compression ratio generally increases from LOW to HIGH, with a trade-off of potentially consuming more CPU resources. Since the performance of the various compression levels depends on the nature of the data in the database, network configuration, system resources and the type of computer system and its capabilities, Oracle cannot document universally applicable performance statistics. The decision as to which level is best must factor in on how balanced your system is regarding bandwidth into the CPU as well as the actual speed of the CPU. It is highly recommended that you run tests with the different compression levels on the data in your environment. Choosing a compression level based on your environment, network traffic characteristics (workload) and dataset is the only way to ensure that the backup set compression level can satisfy your organization's performance requirements and any applicable service level agreements. **Note:** The V$RMAN_COMPRESSION_ALGORITHM view describes the supported algorithms. **See Also**: *Oracle Database Reference* entry for V$RMAN_COMPRESSION_ALGORITHM.
OPTIMIZE FOR LOAD {TRUE|FALSE}	Specifies whether Oracle performs pre-compression block processing when compressed backups have been requested. TRUE is the default and FALSE enables pre-compression processing. The default behavior is not to perform pre-compression block processing. Such processing can consume extra CPU resources, and is not needed for blocks that contain all originally loaded data, and have never been the subject of single-row inserts and deletes. Specifying FALSE uses additional CPU resources to perform pre-compression block processing which consists of internal block cleanups and defragmentation that can result in improved levels of binary compression. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn more about this option.
AS OF RELEASE *'version'*	Specifies the release version. The version number uses the release number format and may use as many as 5 numbers to fully qualify the release. For example, 10.2.0.3.0 and 11.2 are acceptable values. This option ensures compression algorithm stability across future releases.
deviceConf	Configures default backup settings for devices, such as the default backup device, channel configurations for devices, default backup types for each device, and parallelism.
ENCRYPTION	Specifies transparent-mode encryption settings for the database or tablespaces. This configuration applies unless overridden with the SET ENCRYPTION command. Options specified for an individual tablespace take precedence over options specified for the whole database. **See Also:** "Encryption of Backup Sets" on page 2-21 to learn about the different modes of backup encryption, and *Oracle Database Advanced Security Administrator's Guide* to learn about transparent data encryption

Syntax Element	Description
ALGORITHM '*algorithm_name*'	Specifies the default algorithm to use for encryption when writing backup sets. Possible values are listed in V$RMAN_ENCRYPTION_ALGORITHMS. The CLEAR option resets the database to the default value.
FOR DATABASE [ON | OFF | CLEAR]	Specifies whether to enable transparent encryption for the entire database. The options are as follows: ■ ON enables encryption for all database files. ■ OFF turns off encryption for all database files. ■ CLEAR restores the default setting of OFF. **Note:** You must use the SET ENCRYPTION IDENTIFIED BY command to enable password encryption.
FOR TABLESPACE *tablespace_name* [ON | OFF | CLEAR]	Specifies whether to enable transparent encryption for one or more tablespaces. Configured settings for a tablespace always override configuration set at the database level. The options are as follows: ■ ON enables encryption for the specified tablespace unless SET ENCRYPTION OFF FOR ALL TABLESPACES is used. ■ OFF disables encryption for the specified tablespace unless SET ENCRYPTION ON FOR ALL TABLESPACES is used. ■ CLEAR means that encryption for the specified tablespace is determined by the current default for the whole database. **Note:** You must use the SET ENCRYPTION IDENTIFIED BY command to enable password encryption.
SNAPSHOT CONTROLFILE NAME TO '*filename*'	Configures the snapshot control file name and location to '*filename*'. If you run CONFIGURE SNAPSHOT CONTROLFILE NAME CLEAR, then RMAN sets the snapshot control file name to its default. The default value for the snapshot control file name is platform-specific and dependent on Oracle home. For example, the default on some UNIX systems is ?/dbs/snapcf_@.f. If you clear the control file name, and if you change Oracle home, then the default location of the snapshot control file changes as well. The snapshot control file name is valid for this database only. Assume that you configure the snapshot control file name to a nondefault value on the primary database. If you use DUPLICATE to create a standby database, then the snapshot control file location on the standby database will be set to the default value. If desired, you can then configure the snapshot location on the standby database to a nondefault value. **See Also:** *Oracle Database Backup and Recovery User's Guide* for more information about snapshot control files

Syntax Element	Description
forDbUniqueNameOption	Creates an RMAN configuration in the recovery catalog for the database in a Data Guard environment specified by DB_UNIQUE_NAME. You can specify a single database with *db_unique_name* or use ALL for all databases in the recovery catalog that share the DBID of the target database (or DBID specified by the SET DBID command). A recovery catalog is required when performing operations in a Data Guard environment. RMAN must be connected as TARGET to a mounted or open database (which can be a primary or standby database), *or* you must identify the target database with the SET DBID command. Thus, you can use this clause to create a persistent configuration for a standby database *without* connecting as TARGET to the standby or primary database. For example, you can create a configuration for a standby database before its creation so that the configuration applies after the database is created (see Example 2–47 on page 2-84). When you specify FOR DB_UNIQUE_NAME, RMAN directly updates the configuration metadata in the recovery catalog. When RMAN connects as TARGET to a database whose configurations were changed with FOR DB_UNIQUE_NAME, RMAN updates the mounted control file with the configuration metadata from the recovery catalog. **Note:** It is possible to run CONFIGURE locally on a standby database and then run CONFIGURE FOR DB_UNIQUE_NAME for the same database while RMAN is not connected to this database as TARGET. In this case, the configuration in the recovery catalog overrides the configuration in the control file for the specific database.

delalConf

This subclause manages persistent configurations for archived redo log deletion policy.

Syntax Element	Description
ARCHIVELOG DELETION POLICY	Determines when archived redo log files are eligible for deletion. The archived log deletion policy applies to all log archiving destinations, including the fast recovery area. Note that the policy does not apply to archived redo log files in backup sets. Only archived redo log files in the fast recovery area are automatically deleted by the database. You can execute the BACKUP . . . DELETE INPUT, DELETE ARCHIVELOG, or DELETE OBSOLETE commands to delete logs manually from log archiving destinations, including the recovery area. If FORCE is not specified on the deletion commands, then these deletion commands obey the archived log deletion policy. If FORCE is specified, then the deletion commands ignore the archived log deletion policy. In the recovery area, the database retains logs eligible for deletion as long as possible. The database deletes the oldest logs first when disk space is required. When the recovery area is under disk pressure, the database may delete archived redo log files needed by Oracle Streams. **Note:** The deletion policy does not apply to foreign archived redo log files, which are logs received by a logical standby database for a LogMiner session. These logs are transferred from a primary database, but unlike ordinary archived redo log files they have a different DBID. Foreign archived redo log files cannot be backed up or restored on a logical standby database.

Syntax Element	Description
`TO APPLIED ON [ALL] STANDBY`	Specifies that archived redo log files are eligible for deletion if *both* of the following conditions are met: ▪ The archived redo log files have been applied to the required standby databases. ▪ The logs are not needed by the `BACKED UP ... TIMES TO DEVICE TYPE` deletion policy. If the `BACKED UP` policy is not set, then this condition is always met. Which remote destinations are considered depends on the following criteria: ▪ If you do not specify `ALL`, then archived redo log files are eligible for deletion after being applied to all mandatory remote destinations. ▪ If you specify `ALL`, then archived redo log files are eligible after being applied or consumed on all remote destinations, whether mandatory or not. For example, standby database `sby1` may be the only remote destination receiving logs, but other remote destinations may apply logs by referring to the same location on `sby1`. With `ALL`, `sby1` marks the log on the primary database as consumed as soon as it is not required at `sby1`, but does not permit deletion of this log until it is applied or consumed by all other dependent remote destinations referring to the same location. **Note:** It is invalid to specify the `TO APPLIED` clause in combination with either `NONE` or the `TO SHIPPED` clause. **See Also:** *Oracle Data Guard Concepts and Administration* for details
`BACKED UP` *`integer`* `TIMES TO DEVICE TYPE` *`deviceSpecifier`*	Specifies that archived redo log files are eligible for deletion if *both* of the following conditions are met: ▪ The specified number of archived log backups exist on the specified device type. ▪ The logs are not needed by the `TO SHIPPED TO ... STANDBY` or `TO APPLIED ON ... STANDBY` deletion policy. If neither standby deletion policy is set, then this condition is always met. If you configure the deletion policy with this clause, then a `BACKUP ARCHIVELOG` command copies the logs unless *`integer`* backups already exist on the specified device type. If *`integer`* backups of the logs exist, then the `BACKUP ARCHIVELOG` command skips the logs. In this way, the archived log deletion policy functions as a default `NOT BACKED UP` *`integer`* `TIMES` clause on the `BACKUP ARCHIVELOG` command. You can override this deletion policy by specifying `FORCE` option on the `BACKUP` command. **See Also:** *`deviceSpecifier`* on page 4-16

Syntax Element	Description
`TO NONE`	Disables the archived log deletion policy. This is the default setting. Archived redo log files can be located inside or outside of the fast recovery area. Logs in any location can be deleted by manual commands. Only logs in the fast recovery area can be deleted automatically by the database. When the deletion policy is set to `NONE`, RMAN considers archived redo log files as eligible for deletion if they meet *both* of the following conditions: ▪ The archived redo log files, whether in the fast recovery area or outside of it, have been transferred to the required remote destinations specified by `LOG_ARCHIVE_DEST_n`. ▪ The archived redo log files in the fast recovery area have been backed up at least once to disk or SBT *or* the logs are obsolete according to the backup retention policy. The backup retention policy considers logs obsolete *only if* the logs are not needed by a guaranteed restore point *and* the logs are not needed by Flashback Database. Archived redo log files are needed by Flashback Database if the logs were created later than `SYSDATE-'DB_FLASHBACK_RETENTION_TARGET'`. For example, suppose that archived redo log files have been transferred to required remote destinations. The logs are obsolete according to the recovery window retention policy, but have not been backed up. In this case, the logs are eligible for deletion. Alternatively, suppose that the logs are obsolete and have been backed up to SBT, but have *not* been transferred to required remote destinations. In this case, the logs are not eligible for deletion. If the deletion policy is set to `NONE`, and if you execute a deletion command for archived redo log files outside the fast recovery area, then RMAN obeys only the conditions specified on the deletion command.
`TO SHIPPED TO [ALL] STANDBY`	Specifies that archived redo log files are eligible for deletion if *both* of the following conditions are met: ▪ The archived redo log files have been transferred to the required remote destinations. ▪ The logs are not needed by the `BACKED UP ... TIMES TO DEVICE TYPE` deletion policy. If the `BACKED UP` deletion policy is not set, then this condition is always met. Which remote destinations are considered depends on the following criteria: ▪ If you do not specify `ALL`, then the archived redo log files are eligible for deletion after transfer to mandatory remote destinations only. ▪ If you specify `ALL`, then the logs are eligible for deletion after transfer to all remote destinations, whether mandatory or not. **Note:** It is invalid to specify the `TO SHIPPED` clause in combination with `NONE` or the `TO APPLIED` clause. **See Also:** *Oracle Data Guard Concepts and Administration* for details

backupConf

This subclause manages persistent configurations relating to the `BACKUP` command. One configuration is backup optimization. If you enable backup optimization, then RMAN does not back up a file to a device type if the identical file is already backed up on the device type.

Table 2–3 explains the criteria used by backup optimization to determine whether a file is identical and can potentially be skipped. The table also explains the algorithm that RMAN uses when backup optimization is enabled and it needs to determine whether to skip the backup of an identical file. If RMAN does not skip a backup, then it makes the backup exactly as specified.

Table 2–3 Backup Optimization Algorithm

File Type	Criteria for an Identical File	Backup Algorithm When Backup Optimization Is Enabled
Datafile	The datafile must have the same DBID, checkpoint SCN, creation SCN, and RESETLOGS SCN and time as a datafile already in a backup. The datafile must be offline-normal, read-only, or closed normally.	If a recovery window-based retention policy is enabled, then whether RMAN skips a datafile depends on the backup media. For backups to tape, if the most recent backup is older than the recovery window, then RMAN takes another backup of a datafile even if a backup of an identical datafile exists. In this way, tapes can be recycled after they expire. For backups to disk, RMAN skips the backup if an identical datafile is available on disk, even if that backup is older than the beginning of the recovery window. The window-based retention policy causes RMAN to retain the old backup for as long as it is needed. If a retention policy is enabled with CONFIGURE RETENTION POLICY TO REDUNDANCY *r*, then RMAN skips backups only if at least *n* backups of an identical file exist on the specified device, where *n*=*r*+1. If no retention policy is enabled, then RMAN skips a backup only if at least *n* backups of an identical file exist on the specified device. RMAN searches for values of *n* in this order of precedence (that is, values higher on the list override values lower on the list): **1.** BACKUP ... COPIES *n* **2.** SET BACKUP COPIES *n* **3.** CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE ... TO *n* **4.** *n*=1
Archived redo log	The archived redo log must have the same thread, sequence number, and RESETLOGS SCN and time as an archived log already in a backup.	RMAN skips a backup only if at least *n* backups of an identical file exist on the specified device. RMAN searches for values of *n* in this order of precedence (that is, values higher on the list override values lower on the list): **1.** BACKUP ... COPIES *n* **2.** SET BACKUP COPIES *n* **3.** CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE ... TO *n* **4.** *n*=1
Backup set	The backup set must have the same record ID and stamp as an existing backup set.	RMAN skips a backup only if at least *n* backups of an identical file exist on the specified device. By default, *n*=1. RMAN searches for other values of *n* in this order of precedence (that is, values higher on the list override values lower on the list): **1.** BACKUP ... COPIES *n* **2.** SET BACKUP COPIES *n* **3.** *n*=1

Syntax Element	Description
{ARCHIVELOG | DATAFILE} BACKUP COPIES FOR DEVICE TYPE *deviceSpecifier* TO *integer*	Specifies the number of copies of each backup set for DATAFILE (both datafiles and control files) or ARCHIVELOG files on the specified device type (see Example 2–42 on page 2-82). You can create from 1 (default) to 4 copies. RMAN can duplex backups to either disk or tape, but cannot duplex backups to tape and disk simultaneously. When backing up to tape, ensure that the number of copies does not exceed the number of available tape devices. Also, if COPIES is greater than 1, then the BACKUP_TAPE_IO_SLAVES initialization parameter must be enabled on the target database. Control file autobackups are never duplexed. Also, duplexing is not permitted in the fast recovery area. If duplexing is specified in the BACKUP command or in a SET BACKUP COPIES command, then the CONFIGURE setting is overridden.

Syntax Element	Description
BACKUP OPTIMIZATION [ON | OFF | CLEAR]	Toggles backup optimization ON or OFF (default). Specify CLEAR to return optimization to its default value of OFF. Backup optimization is enabled when all of the following conditions are met: ■ The CONFIGURE BACKUP OPTIMIZATION ON command has been run. ■ You run BACKUP DATABASE, BACKUP ARCHIVELOG with the ALL or LIKE options, BACKUP BACKUPSET ALL, BACKUP RECOVERY AREA, BACKUP RECOVERY FILES, or BACKUP DATAFILECOPY. ■ The RMAN job uses a channel of only one device type. Optimization prevents RMAN from backing up a file to a device type if the identical file is already backed up on the device type. RMAN does not signal an error if backup optimization causes all files to be skipped during a backup. Note that the backup retention policy has an effect on which files backup optimization skips. For two files to be identical, their content must be exactly the same, as described in Table 2–3 on page 2-75. Note that when you create backup pieces on disk or on media managed by Oracle Secure Backup, optimization excludes undo data from the backup when the data does not belong to an active transaction. **Note:** BACKUP ... DELETE INPUT deletes all specified archived redo log whether or not optimization would skip these files during a backup. **Note:** You can override backup optimization with the FORCE option of the BACKUP command. **See Also:** *Oracle Database Backup and Recovery User's Guide* for a description of how RMAN determines that it can skip the backup of a file
EXCLUDE FOR TABLESPACE *tablespace_name* [CLEAR]	Excludes the specified tablespace from BACKUP DATABASE and RESTORE DATABASE commands (see Example 2–44 on page 2-83). Note that you cannot exclude the SYSTEM tablespace. The BACKUP command default does not exclude tablespaces. You must declare the option in order to use it. The exclusion is stored as an attribute of the tablespace and not to the individual datafiles. In this manner, the exclusion applies not only to the present set of datafiles but also to any files that are added to this tablespace in the future. If you run CONFIGURE ... CLEAR on a tablespace after excluding it, then it returns to the default configuration of nonexclusion. You can still back up an excluded tablespace by explicitly specifying it in a BACKUP command or by specifying the NOEXCLUDE option on a BACKUP DATABASE command. Similarly, you can restore an excluded tablespace by specifying it in the RESTORE TABLESPACE command.
MAXSETSIZE	Specifies the maximum size of each backup set created on a channel. Use the CLEAR option to return MAXSETSIZE to the default value of UNLIMITED. **Note:** This option is ignored by BACKUP AS COPY.
TO *sizeSpec*	Specifies the maximum size of each backup set as *integer* gigabytes, kilobytes, or megabytes.
TO UNLIMITED	Specifies no size limit for backup sets.

Syntax Element	Description
`RETENTION POLICY`	Specifies a persistent, ongoing policy for backup sets and copies that RMAN marks as obsolete, that is, not needed and eligible for deletion. As time passes, RMAN marks backup sets and copies as obsolete according to the criteria specified in the retention policy. RMAN automatically deletes obsolete backup sets and copies in the fast recovery area when space is needed. RMAN does not automatically delete obsolete files outside the fast recovery area: you must manually execute `DELETE OBSOLETE` to remove them. For backups, the basic unit of the retention policy is a backup set (not a backup piece) or image copy. For example, `BACKUP AS BACKUPSET COPIES 4 TABLESPACE users` creates a single backup set that is duplexed into four identical backup pieces. The retention policy considers this as *one* backup, not four separate backups. **Note:** Use the `CLEAR` option to return `RETENTION POLICY` to its default of `REDUNDANCY 1`.
`TO NONE`	Disables the retention policy feature. RMAN does not consider any backup sets or copies as obsolete.
`TO RECOVERY WINDOW OF` *`integer`* `DAYS`	Specifies a time window in which RMAN should be able to recover the database. The window stretches from the current time (`SYSDATE`) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is `SYSDATE` - *`integer`* days in the past. **Note:** The `REDUNDANCY` and `RECOVERY WINDOW` options are mutually exclusive. Only one type of retention policy can be in effect at any time.
`TO REDUNDANCY` *`integer`*	Specifies that RMAN should retain *`integer`* full or level 0 backups of each datafile and control file. The default retention policy setting is `REDUNDANCY 1`. If more than *`integer`* full or level 0 backups of a datafile or control file exist, then RMAN marks these extra files as obsolete. RMAN then determines the oldest of the retained backups and marks all archived redo log files and log backups older than this backup as obsolete. The `DELETE OBSOLETE` command removes obsolete datafile backups (full or incremental), control file backups, and archived log backups or image copies. The following scenario illustrates how redundancy works in an incremental backup strategy. Assume that the redundancy level is 1. You run a level 0 database backup at noon Monday, a level 1 cumulative backup at noon on Tuesday and Wednesday, and a level 0 backup at noon on Thursday. Immediately after each daily backup you run a `DELETE OBSOLETE`. The Wednesday `DELETE` command does not remove the Tuesday level 1 backup because this backup is not redundant: the Tuesday level 1 backup could be used to recover the Monday level 0 backup to a time between noon on Tuesday and noon on Wednesday. However, the `DELETE` command on Thursday removes the previous level 0 and level 1 backups. **Note:** The `REDUNDANCY` and `RECOVERY WINDOW` options are mutually exclusive. Only one type of retention policy can be in effect at any time.

cfauConf

This subclause creates persistent configurations relating to control file autobackups.

Syntax Element	Description
`CONTROLFILE AUTOBACKUP`	Controls the control file autobackup feature. **Note:** You should enable control file autobackup feature when using RMAN without a recovery catalog.

Syntax Element	Description
ON	Specifies that RMAN should perform a control file autobackup in the following circumstances: ■ After every BACKUP or CREATE CATALOG command issued at the RMAN prompt. ■ Whenever a BACKUP command within a RUN block is followed by a command that is not BACKUP. ■ At the end of every RUN block if the last command in the block was BACKUP. ■ After structural changes for databases in ARCHIVELOG mode. The autobackup after structural changes does not occur for databases in NOARCHIVELOG mode. Structural changes include adding tablespaces, altering the state of a tablespace or datafile (for example, bringing it online), adding a new online redo log, renaming a file, adding a new redo thread, enabling or disabling Flashback Database, and so on. This type of autobackup, unlike autobackups that occur in the preceding circumstances, is only to disk. You can run CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK to set a nondefault disk location. Starting with Oracle 11*g* Release 2, RMAN creates a single autobackup file encompassing all of the structural changes that have occurred within a few minutes of each other rather than creating a new backup of the controlfile on each structural change to the database. The first channel allocated during the backup or copy job creates the autobackup and places it into its own backup set; for post-structural autobackups, the default disk channel makes the backup. RMAN writes the control file and server parameter file to the same backup piece. After the control file autobackup completes, the database writes a message containing the complete path of the backup piece and the device type to the alert log. The default location for the autobackup on disk is the fast recovery area (if configured) or a platform-specific location (if not configured). RMAN automatically backs up the current control file using the default format of %F. You can change the location and filename format with the CONFIGURE CONTROLFILE AUTOBACKUP FORMAT and SET CONTROLFILE AUTOBACKUP FORMAT commands. You cannot configure RMAN to write the autobackup to multiple locations. To create multiple control file backups, you can make the last command in your backup job a BACKUP CURRENT CONTROLFILE FORMAT command, which will back up the control file to the specified FORMAT location and then execute an autobackup. **Note:** The SET CONTROLFILE AUTOBACKUP FORMAT command, which you can specify either within a RUN block or at the RMAN prompt, overrides the configured autobackup format in the session only. The order of precedence is: **1.** SET within a RUN block **2.** SET at RMAN prompt **3.** CONFIGURE CONTROLFILE AUTOBACKUP FORMAT You can configure the autobackup format even when CONFIGURE CONTROLFILE AUTOBACKUP is set to OFF, but RMAN does not generate autobackups in this case. For RMAN to make autobackups, you must set CONFIGURE CONTROLFILE AUTOBACKUP to ON.
OFF	Disables the autobackup feature (default). Any BACKUP command that includes datafile 1 automatically includes the current control file and server parameter file in the backup set. Otherwise, RMAN does not include these files.
CLEAR	Returns this configuration to its default setting of OFF.

Syntax Element	Description
FORMAT FOR DEVICE TYPE *deviceSpecifier* TO *formatSpec*	Configures the default location and filename format for the control file autobackup on the specified device type (see Example 2–46 on page 2-84). By default, the format is %F for all devices. Any default format string specified with CONFIGURE must include the %F substitution variable. Use of any other substitution variable is an error. Specify CLEAR to return the format to the default %F. If a fast recovery area is enabled, and if the format is the default '%F', then RMAN creates the autobackup in the recovery area in a directory named autobackup. Otherwise, the default autobackup location is an operating system-specific location (?/dbs on UNIX, Linux, and Windows). The string # default in the output of the SHOW command indicates when RMAN is using the default format. If you manually configure the disk format to '%F', then RMAN creates the autobackups in the operating system-specific default location even though the recovery area is enabled. To change the format back to its default so that RMAN creates the autobackups in the recovery area, run CONFIGURE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR. The *formatSpec* can specify an Automatic Storage Management disk group. The following example configures a channel for an ASM disk group: CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK TO '+dgroup1'; **See Also:** *formatSpec* on page 4-23 for the semantics of the %F substitution variable

deviceConf

This subclause creates persistent configurations relating to channels and devices.

Names for Configured Channels

RMAN determines the names for configured channels. RMAN uses the following convention: ORA_*devicetype*_*n*, where *devicetype* refers to the user device type (such as DISK or sbt_tape) and *n* refers to the channel number. Channel names beginning with the ORA_ prefix are reserved by RMAN for its own use. You cannot manually allocate a channel with a name that begins with ORA_.

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

RMAN names the first DISK channel ORA_DISK_1, the second ORA_DISK_2, and so on. RMAN names the first sbt channel ORA_SBT_TAPE_1, the second ORA_SBT_TAPE_2, and so on. When you parallelize channels, RMAN always allocates channels in numeric order, starting with 1 and ending with the parallelism setting (CONFIGURE DEVICE TYPE ... PARALLELISM *n*).

To run BACKUP or jobs on specific configured channels, use the system-generated channel names. If you specify channel numbers in the CONFIGURE CHANNEL command (see the *deviceConf* clause), then RMAN uses the same numbers in the system-generated channel names.

Automatic channel allocation also applies to maintenance commands. If RMAN allocates an automatic maintenance channel, then it uses the same naming convention as any other automatically allocated channel.

Configured Channels in an Oracle RAC Environment

As long as the SYSDBA password for all Oracle RAC instances is the same, then you never need to put passwords in the CONNECT option of the ALLOCATE CHANNEL or CONFIGURE command. If you use a connect string of the form *user@database*, then RMAN automatically uses the same password that was used for the TARGET connection when the RMAN session was started (see Example 2–44 on page 2-83).

Syntax Element	Description

[AUXILIARY] CHANNEL [*integer*] DEVICE TYPE *deviceSpecifier*	Specifies the standard or AUXILIARY channel that you are configuring or clearing, as well as the device type of the channel.
	Note: Channels allocated with ALLOCATE CHANNEL within a RUN command override configured automatic channels.
	Either configure a generic channel or specify a channel number, where *integer* is less than 255. See Example 2–44 on page 2-83 for an illustration of numbered channels.
	If AUXILIARY is specified, then this configuration is used only for channels allocated at the auxiliary instance. Specify configuration information for auxiliary channels if they require different parameters from the channels allocated at the target instance. If no auxiliary device configuration is specified, then RMAN configures any auxiliary channels using the target database device configuration.
	You must specify at least one channel option. For example, you cannot issue a command such as CONFIGURE CHANNEL 2 DEVICE TYPE DISK, but you can issue a command such as CONFIGURE CHANNEL 2 DEVICE TYPE DISK MAXPIECESIZE 2500K.
	For generic channels of a specified device type, a new command erases previous settings for this device type. Assume that you run these commands:
	`CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 1G;`
	`CONFIGURE CHANNEL DEVICE TYPE sbt FORMAT 'bkup_%U';`
	The second command erases the MAXPIECESIZE setting of the first command.
	Note: RMAN does not simultaneously allocate automatic channels for multiple device types in the BACKUP command.
	See Also: *Oracle Database Backup and Recovery User's Guide* to learn how configure automatic channels specified by channel number

allocOperandList	Specifies control options for the configured channel. If you configure channels by using the nondefault CONNECT or PARMS options to create backups or copies, then you must either use the same configured channels or manually allocate channels with the same options to restore or crosscheck these backups. Note that the FORMAT parameter can specify an Automatic Storage Management disk group. The following example configures a channel for an ASM disk group: `CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '+dgroup1';` **See Also:** *allocOperandList* on page 4-2
CLEAR	Clears the specified channel. For example, CONFIGURE CHANNEL 1 DEVICE TYPE DISK CLEAR returns only channel 1 to its default, whereas CONFIGURE CHANNEL DEVICE TYPE DISK CLEAR returns the generic disk channel to its default. Note that you cannot specify any other channel options (for example, PARMS) when you specify CLEAR.

Syntax Element	Description
DEFAULT DEVICE TYPE TO *deviceSpecifier*	Specifies the default device type for automatic channels. By default, DISK is the default device type. CLEAR returns the default device type to DISK. By default, the BACKUP command only allocates channels of the default device type. For example, if you configure automatic channels for DISK and sbt and set the default device type to sbt, then RMAN only allocates tape channels when you run the BACKUP DATABASE command. You can override this behavior either by manually allocating channels in a RUN command, or by specifying DEVICE TYPE on the BACKUP command itself (see Example 2–42 on page 2-82). The RESTORE command allocates automatic channels of all configured device types, regardless of the default device type. The RESTORE command obeys the PARALLELISM setting for each configured device type.

DEVICE TYPE *deviceSpecifier*	Specifies the device type (disk or sbt) to which to apply the settings specified in this CONFIGURE command. The CLEAR option resets backup type and parallelism settings for this device to their defaults.
	
	If you run the CONFIGURE DEVICE TYPE command to configure default settings for a device type and do not run CONFIGURE CHANNEL for this device type, then RMAN allocates all channels without other channel control options. Assume that you configure the sbt device and run a backup as follows:
	
	`CONFIGURE DEVICE TYPE sbt PARALLELISM 1;`
	`BACKUP DEVICE TYPE sbt DATABASE;`
	
	In effect, RMAN does the following when executing this backup:
	
	`RUN`
	`{`
	`  ALLOCATE CHANNEL ORA_SBT_TAPE_1 DEVICE TYPE sbt;`
	`  BACKUP DATABASE;`
	`}`

BACKUP TYPE TO [[COMPRESSED] BACKUPSET | COPY]	Configures the default backup type for disk or tape backups. For SBT devices the COPY option is not supported. The default for disk is BACKUPSET. If BACKUP TYPE is set to BACKUPSET, then the BACKUP command always produces backup sets regardless of which media the backup is created on. With the COMPRESSED option, the backup sets produced use binary compression. The default location for disk backups is the fast recovery area, if one is configured; otherwise, RMAN stores backups in a platform-specific location. The default format for backup filenames is %U.

PARALLELISM *integer*	Configures the number of automatic channels of the specified device type allocated for RMAN jobs. By default, PARALLELISM is set to 1.
	
	Note: The CONFIGURE ... PARALLELISM parameter specifies channel parallelism, that is, the number of channels that RMAN allocates during backup and restore operations. The RECOVERY_PARALLELISM initialization parameter specifies the number of processes used in instance recovery.
	
	Suppose you set PARALLELISM for disk backups to 2 (see Example 2–43 on page 2-82). If you set the default device type as disk, then RMAN allocates two disk channels when you run BACKUP DATABASE at the RMAN prompt. RMAN always allocates the number of channels set by PARALLELISM, although it may use only a subset of these channels.
	
	Note: If you configure *n* manually numbered channels, then the PARALLELISM setting can be greater than or less than *n*. For example, you can manually number 10 automatic channels and configure PARALLELISM to 2 or 12.
	
	To change the parallelism for a device type to *n*, run a new CONFIGURE DEVICE TYPE ... PARALLELISM *n* command. For example, you can change configure PARALLELISM to 3 for sbt and then change it to 2 as follows:
	
	`CONFIGURE DEVICE TYPE sbt PARALLELISM 3;`
	`CONFIGURE DEVICE TYPE sbt PARALLELISM 2;`

Examples

Example 2–41 Configuring Device and Backup Options

This example configures channels of device type `DISK` and `sbt` and sets the default device type as `sbt`. The example also enables backup optimization and configures a recovery windows of two weeks.

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/disk1/backups/%U';
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(OB_DEVICE_1=tape1)';
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 14 DAYS;
```

Example 2–42 Overriding the Default Device Type

This example configures duplexing to `2` for `DISK` backups of datafiles and control files (control file autobackups on disk are a special case and are never duplexed), then configures `sbt` as the default device.

```
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 2;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(OB_DEVICE_1=tape1)';
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

The first `BACKUP` command backs up the archived redo log files on the default `sbt` channel. The second `BACKUP` command backs up the database to disk locations. Because duplexing is enabled for disk backups, two copies of each output backup set are created.

```
BACKUP ARCHIVELOG ALL;
BACKUP DEVICE TYPE DISK
  DATABASE
  FORMAT '/disk1/db_backup_%U','/disk2/db_backup_%U';
```

Example 2–43 Configuring Automatic Channels Across File Systems

This example configures automatic disk channels across two file systems:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT '/disk1/%U';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT '/disk2/%U';
```

Because `PARALLELISM` is set to `2`, the following command divides the output data between two file systems:

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

The following `LIST` command shows how the datafile backup was parallelized:

```
RMAN> LIST BACKUPSET 2031, 2032;

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

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
2031    Full    401.99M    DISK        00:00:57     19-JAN-07
        BP Key: 2038   Status: AVAILABLE  Compressed: NO  Tag: TAG20070119T100532
        Piece Name: /disk1/24i7ssnc_1_1
  List of Datafiles in backup set 2031
  File LV Type Ckp SCN    Ckp Time  Name
```

```
  ---- -- ---- ---------- --------- ----
  1       Full 973497     19-JAN-07 /disk3/oracle/dbs/t_db1.f
  5       Full 973497     19-JAN-07 /disk3/oracle/dbs/tbs_112.f

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
2032    Full    133.29M    DISK        00:00:57     19-JAN-07
        BP Key: 2039   Status: AVAILABLE  Compressed: NO  Tag: TAG20070119T100532
        Piece Name: /disk2/25i7ssnc_1_1
  List of Datafiles in backup set 2032
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  2       Full 973501     19-JAN-07 /disk3/oracle/dbs/t_ax1.f
  3       Full 973501     19-JAN-07 /disk3/oracle/dbs/t_undo1.f
  4       Full 973501     19-JAN-07 /disk3/oracle/dbs/tbs_111.f
```

Example 2–44 Configuring Automatic Channels in an Oracle Real Application Clusters (Oracle RAC) Configuration

This example assumes an Oracle RAC database with two nodes. Oracle Secure Backup is the media manager. Tape drive `tape1` is directly attached to `node1`, while tape drive `tape2` is directly attached to `node2`. The example configures an automatic SBT channel for each cluster node.

This example illustrates channel connections to Oracle RAC instances `node1` and `node2`. For both channel connections, RMAN uses the same user name and password that were entered for the target database connection.

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT '@node1'
  PARMS 'ENV=(OB_DEVICE=tape1)';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT '@node2'
  PARMS 'ENV=(OB_DEVICE=tape2)';
```

Example 2–45 Configuring Auxiliary Filenames

This example uses `CONFIGURE AUXNAME` to specify new filenames for the datafiles. The `DUPLICATE` command duplicates a database to a remote host with a different directory structure.

```
# set auxiliary names for the datafiles
CONFIGURE AUXNAME FOR DATAFILE 1 TO '/oracle/auxfiles/aux_1.f';
CONFIGURE AUXNAME FOR DATAFILE 2 TO '/oracle/auxfiles/aux_2.f';
CONFIGURE AUXNAME FOR DATAFILE 3 TO '/oracle/auxfiles/aux_3.f';
CONFIGURE 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 ('?/dbs/dupdb_log_1_1.f',
             '?/dbs/dupdb_log_1_2.f') SIZE 4M,
    GROUP 2 ('?/dbs/dupdb_log_2_1.f',
             '?/dbs/dupdb_log_2_2.f') SIZE 4M REUSE;
}
# Un-specify the auxiliary names for the datafiles so that they are not
# overwritten by mistake:
CONFIGURE AUXNAME FOR DATAFILE 1 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
```

```
CONFIGURE AUXNAME FOR DATAFILE 3 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 4 CLEAR;
```

Example 2–46 Specifying the Default Format for Control File Autobackup

The following example enables the autobackup feature and configures the default autobackup format for the `DISK` and `sbt` devices:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/disk2/%F';
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'cf_auto_%F';
```

Example 2–47 Creating Configurations for Standby Databases

Assume that primary database `prod` is associated with two standby databases with the `DB_UNIQUE_NAME` names `dgprod3` and `dgprod4`. Assume that you start RMAN and connect to `prod` as `TARGET` and connect to a recovery catalog. The following commands configure the default device type for databases `dgprod3` and `dgprod4`.

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt
  FOR DB_UNIQUE_NAME dgprod3;
CONFIGURE DEVICE TYPE sbt PARALLELISM 2
  FOR DB_UNIQUE_NAME dgprod3;
CONFIGURE DEFAULT DEVICE TYPE TO DISK
  FOR DB_UNIQUE_NAME dgprod4;
```

The control files of the two standby databases are updated with the configuration only after the reverse resynchronization from the recovery catalog to the control file, which occurs the first time that the user connects to `dgprod3` and `dgprod4`.

The following `SHOW` command displays the persistent device type configurations for the database whose unique name is `dgprod3`:

```
RMAN> SHOW DEVICE TYPE FOR DB_UNIQUE_NAME dgprod3;
RMAN configuration parameters for database with db_unique_name DGPROD3 are:

CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 2 BACKUP TYPE TO BACKUPSET;
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
```

The following `SHOW` command displays the persistent configurations for all databases known to the recovery catalog whose DBID is 3257174182 (the value specified by the preceding `SET DBID` command):

```
SHOW ALL FOR DB_UNIQUE_NAME ALL;
```

Example 2–48 Optimizing Backups

This scenario illustrates the backup optimization behavior described in Table 2–3 on page 2-75. Assume that backup optimization is disabled. At 9 a.m., you back up three copies of all existing archived redo log files to tape. Note that the `BACKUP_TAPE_IO_SLAVES` initialization parameter must be `true` when duplexing backups to tape.

```
BACKUP DEVICE TYPE sbt COPIES 3 ARCHIVELOG ALL;
```

At 11 a.m., you enable backup optimization:

```
CONFIGURE BACKUP OPTIMIZATION ON;
```

At noon, you run the following archived redo log backup:

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

```
Starting backup at 19-JAN-07
current log archived
using channel ORA_SBT_TAPE_1
skipping archived log file /d1/db1r_605ab325_1_34_612112605.arc; already backed up 3 time(s)
skipping archived log file /d1/db1r_605ab325_1_35_612112605.arc; already backed up 3 time(s)
skipping archived log file /d1/db1r_605ab325_1_36_612112605.arc; already backed up 3 time(s)
skipping archived log file /d1/db1r_605ab325_1_37_612112605.arc; already backed up 3 time(s)
skipping archived log file /d1/db1r_605ab325_1_38_612112605.arc; already backed up 3 time(s)
skipping archived log file /d1/db1r_605ab325_1_39_612112605.arc; already backed up 3 time(s)
channel ORA_SBT_TAPE_1: starting archived log backup set
channel ORA_SBT_TAPE_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=40 RECID=170 STAMP=612270506
channel ORA_SBT_TAPE_1: starting piece 1 at 19-JAN-07
channel ORA_SBT_TAPE_1: finished piece 1 at 19-JAN-07 with 2 copies and tag
TAG20070119T110827
piece handle=2hi7t0db_1_1 comment=API Version 2.0,MMS Version 10.1.0.0
piece handle=2hi7t0db_1_2 comment=API Version 2.0,MMS Version 10.1.0.0
```

In this case, the `BACKUP ... COPIES` setting overrides the `CONFIGURE ... COPIES` setting, so RMAN sets *n*=2. RMAN skips the backup of a log only if at least two copies of the log exist on the `sbt` device. Because three copies of each log exist on `sbt` of all the logs generated on or before 9 a.m., RMAN skips the backups of these logs. However, RMAN backs up two copies of all logs generated after 9 a.m. because these logs have not yet been backed up to tape.

Example 2–49 Configuring a Default Compression Algorithm

This example assumes that you have a license for the Advanced Compression Option (ACO) of the database.

If you want to make the `MEDIUM` compression algorithm the default compression algorithm for all compressed backups, issue the following:

```
CONFIGURE COMPRESSION ALGORITHM 'MEDIUM';
```

From this point on, you can use the `MEDIUM` compression algorithm by issuing the following command:

```
BACKUP AS COMPRESSED BACKUPSET DATABASE;
```

CONNECT

Purpose

Use the CONNECT command to establish a connection between RMAN and a target, auxiliary, or recovery catalog database.

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

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

See Also: RMAN on page 3-59 for command-line connection options

Prerequisites

You can only run the CONNECT TARGET, CONNECT CATALOG, and CONNECT AUXILIARY commands at the RMAN prompt and only if RMAN is not already connected to the databases specified by these commands. To connect to a different target, catalog, or auxiliary database you must start a new RMAN session.

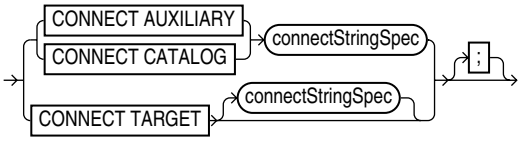
Usage Notes

An RMAN session runs in NOCATALOG mode by default if all of the following conditions are met:

- You did not specify CATALOG or NOCATALOG when you started RMAN.
- You have not yet run CONNECT CATALOG in an RMAN session.
- You run a command such as BACKUP that requires an RMAN repository connection (as shown in Example 2–51 on page 2-87).

Syntax

***connect*::=**



(*connectStringSpec::=* on page 4-13)

Semantics

Syntax Element	Description
CONNECT AUXILIARY	Establishes a connection between RMAN and an auxiliary database instance. Auxiliary instances are used with the TRANSPORT TABLESPACE and DUPLICATE commands, and during RMAN TSPITR.
CONNECT CATALOG	Establishes a connection between RMAN and a recovery catalog database. If the recovery catalog is a virtual private catalog (see CREATE CATALOG), then the RMAN client connecting to this catalog must be at patch level 10.1.0.6 or 10.2.0.3. Oracle9*i* RMAN clients cannot connect to a virtual private catalog. This version restriction does not affect RMAN client connections to an Oracle Database 11*g* base recovery catalog, even if it has some virtual private catalog users. RMAN issues an RMAN-06445 error if you attempt to use the CONNECT CATALOG command in an RMAN session when RMAN is already in the default NOCATALOG mode (see "Usage Notes" on page 2-86). **Note:** You must use RMAN with a recovery catalog in a Data Guard environment.
CONNECT TARGET	Establishes a connection between RMAN and a target database. **Note:** RMAN can connect to physical standby databases as TARGET in a Data Guard environment. If you run CONNECT TARGET for a database that has a DB_UNIQUE_NAME that is unknown to the recovery catalog, but the DBID is the same as a registered database, then RMAN automatically and implicitly registers the database in the recovery catalog.
connectStringSpec	Specifies the connection information for the database.

Examples

Example 2–50 Connecting to a Target Database Without a Recovery Catalog

This example starts RMAN in NOCATALOG mode and then connects to the target database with an Oracle Net service name prod1. RMAN prompts for the SYS password.

```
% rman NOCATALOG
RMAN> CONNECT TARGET SYS@prod1;

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

Example 2–51 Connecting to a Target Database in the Default NOCATALOG Mode

This example starts RMAN without specifying either CATALOG or NOCATALOG and then uses operating system authentication to connect to a target database with operating system authentication. Because no CONNECT CATALOG command has been run, RMAN defaults to NOCATALOG mode when you run the BACKUP command.

```
% rman
RMAN> CONNECT TARGET /
RMAN> BACKUP DATABASE;
```

At this point in the RMAN session, you cannot run CONNECT CATALOG because the session has defaulted to NOCATALOG mode. An attempt to connect to the catalog in this session receives an error:

```
RMAN> CONNECT CATALOG rman@catdb
```

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-06445: cannot connect to recovery catalog after NOCATALOG has been used
```

Example 2–52 Connecting to Target, Recovery Catalog, and Auxiliary Databases

This example connects to a target database by means of operating system authentication and connects to the recovery catalog and auxiliary databases by means of password files. RMAN prompts for the password.

```
% rman
RMAN> CONNECT TARGET;

connected to target database: PROD (DBID=39525561)

RMAN> CONNECT CATALOG rman@catdb;

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CONNECT AUXILIARY SYS@dupdb;

auxiliary database Password: password
connected to auxiliary database: DUPDB (not mounted)
```

CONVERT

Purpose

Use the CONVERT command to convert a tablespace, datafile, or database to the format of a destination platform in preparation for transport across different platforms.

In Oracle Database 10*g* and later releases, CONVERT DATAFILE or CONVERT TABLESPACE is required in the following scenarios:

- Transporting datafiles between platforms for which the value in V$TRANSPORTABLE_PLATFORM.ENDIAN_FORMAT differs.
- Transporting tablespaces with undo segments (typically SYSTEM and UNDO tablespaces, but also tablespaces using rollback segments) between platforms, regardless of whether the ENDIAN_FORMAT is the same or different. Typically, the SYSTEM and UNDO tablespaces are converted only when converting the entire database.
- Other platform specific datafiles, (like when converting to or from the hp Tru64 operating system), require additional datafile conversion .

One use of CONVERT is to transport a tablespace into a database stored in ASM. Native operating system commands such as Linux cp and Windows COPY cannot read from or write to ASM disk groups.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* for a complete discussion of the use of CONVERT DATAFILE, CONVERT TABLESPACE, and CONVERT DATABASE

Prerequisites

The platforms must be supported by the CONVERT command. Query V$TRANSPORTABLE_PLATFORM to determine the supported platforms. Cross-platform tablespace transport is only supported when both the source and destination platforms are contained in this view.

Both source and destination databases must be running with initialization parameter COMPATIBLE set to 10.0.0 or higher. Note the following compatibility prerequisites:

- If COMPATIBLE is less than 11.0.0, then read-only tablespaces or existing transported tablespaces must have been made read/write at least once before they can be transported to a different platform. Note that you can open a tablespace read/write and then immediately make it read-only again.
- If COMPATIBLE is 11.0.0 or higher, then the preceding read/write tablespace restriction does not apply. However, any *existing* transported tablespaces must already have the 10.0 format, that is, they must have been have been made read/write with COMPATIBLE set to 10.0 before they were transported.

CONVERT TABLESPACE Prerequisites

You can only use CONVERT TABLESPACE when connected as TARGET to the *source* database and converting tablespaces on the source platform.

The source database must be mounted or open. The tablespaces to be converted must be read-only at the time of the conversion. The state of the destination database is irrelevant when converting tablespaces on the source database.

CONVERT DATAFILE Prerequisites

You can only use `CONVERT DATAFILE` when connected as `TARGET` to the *destination* database and converting datafile copies on the destination platform.

If you are running a `CONVERT DATAFILE` script generated by `CONVERT DATABASE ON DESTINATION`, then the destination database instance must be started with the `NOMOUNT` option. If you are *not* running a `CONVERT DATAFILE` script generated by `CONVERT DATABASE ON DESTINATION`, then the destination database can be started, mounted, or open.

The state of the source database is irrelevant when converting datafile copies on the destination database. However, if you are running a `CONVERT DATAFILE` script as part of a database conversion on the destination database, and if the script is directly accessing the datafiles on the source database (for example, through an NFS mount), then the source database must be open read-only.

When converting a tablespace on the destination host, you must use `CONVERT DATAFILE` rather than `CONVERT TABLESPACE` because the target database cannot associate the datafiles with tablespaces during the conversion. After you have converted the datafiles required for a tablespace, you can transport them into the destination database.

CONVERT DATABASE Prerequisites

You can only use `CONVERT DATABASE` when connected as `TARGET` to the *source* database, which must be opened read-only. The state of the destination database is irrelevant when executing `CONVERT DATABASE`, even if you run `CONVERT DATABASE ON DESTINATION`.

Because `CONVERT DATABASE` uses the same mechanism as `CONVERT TABLESPACE` and `CONVERT DATAFILE` to convert the datafiles, the usage notes and restrictions for tablespaces and datafiles also apply.

The primary additional prerequisite for `CONVERT DATABASE` is that the source and target platforms must share the same endian format. For example, you can transport a database from Microsoft Windows to Linux for x86 (both little-endian), or from HP-UX to AIX (both big-endian), but not from Solaris to Linux x86. You can create a new database on a target platform manually, however, and transport individual tablespaces from the source database with `CONVERT TABLESPACE` or `CONVERT DATAFILE`.

Even if the endian formats for the source and destination platform are the same, the datafiles for a transportable database must undergo a conversion on either the source or destination host. Unlike transporting tablespaces across platforms, where conversion is not necessary if the endian formats are the same, transporting an entire database requires that certain types of blocks, such as blocks in undo segments, be converted to ensure compatibility with the destination platform.

Usage Notes

Input files are not altered by `CONVERT` because the conversion is not performed in place. Instead, RMAN writes converted files to a specified output destination.

Data Type Restrictions

`CONVERT` does not process user data types that require endian conversions. To transport objects between databases that are built on underlying types that store data in a platform-specific format, use the Data Pump Import and Export utilities.

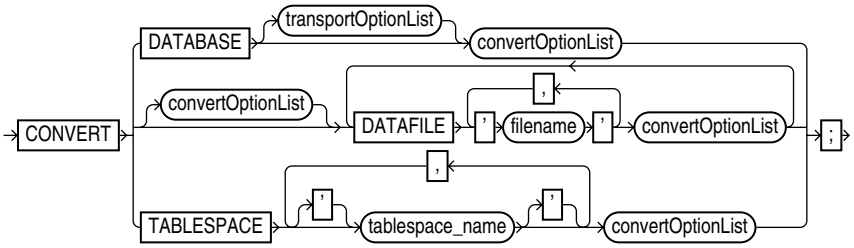
Before Oracle Database 10*g*, CLOBs in a variable-width character set such as `UTF8` were stored in an endian-dependent fixed width format. The `CONVERT` command does

not perform conversions on these CLOBs. Instead, RMAN captures the endian format of each LOB column and propagates it to the target database. Subsequent reads of this data by the SQL layer interpret the data correctly based on either endian format and write it out in an endian-independent way if the tablespace is writeable. CLOBs created in Oracle Database 10*g* and later releases are stored in character set `AL16UTF16`, which is platform-independent.

> **See Also:** *Oracle Database Administrator's Guide* to learn how to transport tablespaces

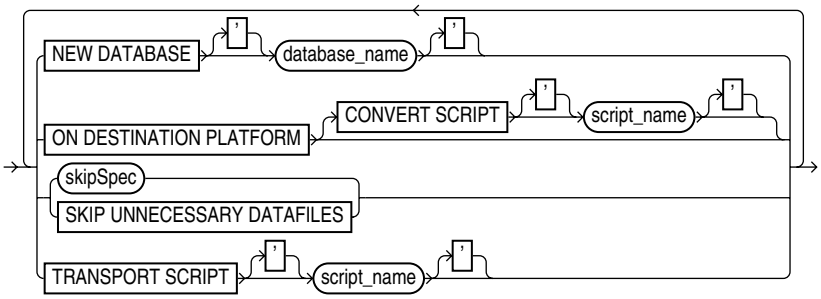
Syntax

convert::=



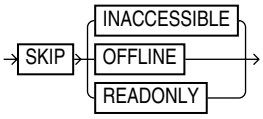
(*transportOptionList::=* on page 2-91, *convertOptionList::=* on page 2-91)

transportOptionList::=

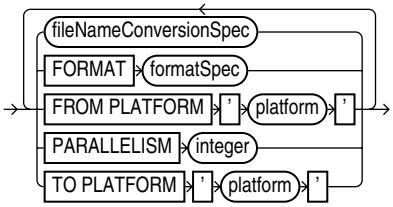


(*skipSpec::=* on page 2-91)

skipSpec::=

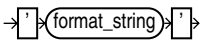


convertOptionList::=



(*fileNameConversionSpec::=* on page 4-18, *formatSpec::=* on page 2-92)

***formatSpec*::=**



Semantics

convert

This clause specifies the objects to be converted: datafiles, tablespaces, or database.

Syntax Element	Description
`DATABASE`	Converts the datafiles to the format of the destination platform and ensures the creation of other required database files. You use `CONVERT DATABASE` to transport an entire database from a source platform to a destination platform. The source and destination platforms must have the same endian format. Depending on the situation, you can use `CONVERT DATABASE` on either the source or destination platform (see Example 2–56 on page 2-99). The following parts of the database are not transported directly: ▪ Redo logs and control files from the source database are not transported. RMAN creates new control files and redo logs for the target database during the transport and performs an `OPEN RESETLOGS` after the new database is created. The control file for the converted database does not contain the RMAN repository from the source database. Backups from the source database are not usable with the converted database. ▪ BFILEs are not transported. The `CONVERT DATABASE` output provides a list of objects that use the BFILE data type, but you must copy the BFILEs manually and fix their locations on the target platform. ▪ Datafiles for locally managed temporary tablespaces are not transported. The temporary tablespaces are re-created at the target platform by running the transport script. ▪ External tables and directories are not transported. The `CONVERT DATABASE` output shows a list of affected objects, but you must redefine these objects on the target platform. See *Oracle Database Administrator's Guide* for more information on managing external tables and directories. ▪ Password files are not transported. If a password file was used with the source database, then the output of `CONVERT DATABASE` includes a list of all user names and their associated privileges. Create a new password file on the target database with this information. See *Oracle Database Administrator's Guide* for more information on managing password files. When using `CONVERT DATABASE`, RMAN detects the following problems and will not proceed until they are fixed: ▪ The database has active or in-doubt transactions. ▪ The database has save undo segments. ▪ The database `COMPATIBILITY` setting is below 10. ▪ Some tablespaces have not been open read/write when the database `COMPATIBILITY` setting is 10 or higher.
`transportOptionList`	Specifies options that control the transport. **See Also:** *`transportOptionList`* on page 2-94

Syntax Element	Description

[*convertOptionList*] DATAFILE '*filename*' *convertOptionList*	Specifies the name of a datafile to be transported into a destination database (see Example 2–54 on page 2-97).
	
	The CONVERT DATAFILE command is only one part of a multiple-step procedure for transporting datafiles across platforms. You can transport datafiles using your live datafiles with the procedure described in *Oracle Database Administrator's Guide* or from backups using the procedure described in *Oracle Database Backup and Recovery User's Guide*. Refer to that document before attempting to transport a tablespace across platforms.
	
	Use FROM PLATFORM in *convertOptionList* to identify the source platform of the datafiles to be converted. If you do not specify FROM PLATFORM, then the value defaults to the platform of the destination database, that is, the database to which RMAN is connected as TARGET. The destination platform is, implicitly, the platform of the destination host.
	
	You can use CONVERT DATAFILE without FROM PLATFORM or TO PLATFORM to move datafiles into and out of ASM (see Example 2–55 on page 2-98). In this case, CONVERT DATAFILE creates datafiles copies that do not belong to the target database. Thus, a LIST DATAFILECOPY command does not display them. The following SQL query shows all converted datafiles that do not belong to the database:
	
	`SELECT NAME`
	`FROM  V$DATAFILE_COPY`
	`WHERE  CONVERTED_FILE='YES';`
	
	The CONVERT DATAFILE syntax supports multiple format names, so that each datafile can have a separate format. The DATAFILE syntax supports *convertOptionList* both immediately following the CONVERT keyword and after each DATAFILE '*filename*' clause. However, RMAN generates an error in the following situations:
	
	■ Any option in *convertOptionList* except FORMAT is specified more than once
	■ Any option in *convertOptionList* except FORMAT is specified in the DATAFILE options list when multiple DATAFILE clauses are specified

Syntax Element	Description
TABLESPACE *tablespace_name* *convertOptionList*	Specifies the name of a tablespace in the source database that you intend to transport into the destination database on a different platform (see Example 2–53 on page 2-96). Specify this option to produce datafiles for the specified tablespaces in the format of a different destination platform. You can then transport the converted files to the destination platform. You can only use CONVERT TABLESPACE when connected as TARGET to the source database and converting on the source platform. The tablespaces to be converted must be read-only at the time of the conversion. You use CONVERT TABLESPACE when the datafiles that you intend to convert are known to the database. Use TO PLATFORM to identify the destination platform of the tablespaces to be converted. If you do not specify TO PLATFORM, then the value defaults to the platform of the database to which RMAN is connected as TARGET. The source platform is, implicitly, the platform of the source host. The CONVERT TABLESPACE command is only one part of a multiple-step process for transporting tablespaces across platforms. You can transport tablespaces using your live datafiles with the procedure described in *Oracle Database Administrator's Guide* or from backups using the procedure described in *Oracle Database Backup and Recovery User's Guide*. Refer to that document before attempting to transport a tablespace across platforms. **Note:** To convert the datafiles of a tablespace on the source host, use CONVERT TABLESPACE ... TO and identify the tablespace to be converted and the destination platform. You should not convert individual datafiles on the source platform with CONVERT DATAFILE because RMAN does not verify that datafiles belong to a read-only tablespace, which means you might convert active datafiles.
convertOptionList	Specifies options that control the conversion. **See Also:** *convertOptionList* on page 2-95

transportOptionList

This clause specifies options for the datafiles, tablespaces, or database to be transported.

Syntax Element	Description
NEW DATABASE *database_name*	Specifies the DB_NAME for the new database produced by the CONVERT DATABASE command.
ON DESTINATION PLATFORM	Generates a convert script of CONVERT DATAFILE commands (see CONVERT SCRIPT parameter) that you can run on the destination host to create the database. **Note:** When this option is specified, CONVERT generates a script but does not generate converted datafile copies. This option is useful for avoiding the overhead of the conversion on the source platform, or in cases in which you do not know the destination platform. For example, you may want to publish a transportable tablespace to be used by recipients with many different target platforms. When you run CONVERT with the ON DESTINATION PLATFORM option, the source database must be open read-only. However, the script generated by CONVERT ON DESTINATION PLATFORM must be run on a database instance that is started NOMOUNT. If the convert script will be reading datafiles from the source database during execution of the CONVERT DATAFILE commands, then the source database must not be open read/write during the execution.

Syntax Element	Description
CONVERT SCRIPT *script_name*	Specifies the location of the file to contain the convert script generated by CONVERT DATABASE ... ON TARGET PLATFORM. If not specified, the convert script is not generated.
skipSpec	Specifies that CONVERT DATABASE should skip inaccessible, offline, or read-only datafiles during the conversion process.
SKIP UNNECESSARY DATAFILES	Specifies that only datafiles with undo segments should be converted by the command. If converting at the destination platform then the generated CONVERT script will only include datafiles with undo segments. Datafiles without undo segments do not need to be converted and can be copied directly from the source database to the destination database. If the command is converting from or to hp Tru64, datafiles with ASMM segment headers must also be converted.
TRANSPORT SCRIPT *script_name*	Specifies the location of the file to contain the transport script generated by CONVERT DATABASE. If omitted, the transport script is not generated.

skipSpec

This subclause specifies which files should be excluded from the conversion.

Syntax Element	Description
SKIP	Excludes datafiles from the conversion according to the criteria specified by the following keywords.
INACCESSIBLE	Specifies that datafiles that cannot be read due to I/O errors should be excluded from the conversion. 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.
OFFLINE	Specifies that offline datafiles should be excluded from the conversion.
READONLY	Specifies that read-only datafiles should be excluded from the conversion.

convertOptionList

This subclause specifies input and output options for the conversion.

You can use either the FORMAT or *fileNameConversionSpec* arguments to control the names of the output files generated by the CONVERT command. If you do not specify either, then the rules governing the location of the output files are the same as those governing the output files from a BACKUP AS COPY operation. These rules are described in the *backupTypeSpec* entry on page 2-40.

Syntax Element	Description
fileNameConversionSpec	A set of string pairs. Whenever any of the input filenames contains one of the first halves of a pair, anywhere in the filename, it will be replaced with the second half of the same pair. You can use as many pairs of replacement strings as required. You can use single or double quotation marks. **See Also:** "Duplication with Oracle Managed Files" on page 2-129 to learn about restrictions related to ASM and Oracle Managed Files

Syntax Element	Description
FORMAT *formatSpec*	Specifies the name template for the output files. See the BACKUP AS COPY command for the format values that are valid here. If the database to which RMAN is connected as TARGET uses a recovery area, then you must specify the FORMAT clause. You can use CONVERT ... FORMAT without specifying FROM PLATFORM or TO PLATFORM. If you do not specify platforms, then running CONVERT TABLESPACE on the source database generates datafile copies that are not cataloged. If you run CONVERT DATAFILE on the destination database, and if the datafile copy already uses the same endianess, then the command generates another datafile copy. As shown in Example 2–55 on page 2-98, you can use CONVERT DATAFILE ... FORMAT to convert a datafile into ASM format. For very large datafiles, copying datafiles between hosts consumes a large amount of space. Consider using NFS or disk sharing. You can create a backup on the source host, mount the disk containing the backups on the destination host, and then convert the datafile into ASM.

FROM PLATFORM '*platform*'	Specifies the name of the source platform. If not specified, the default is the platform of the database to which RMAN is connected as TARGET.
	The specified platform must be one of the platforms listed in the PLATFORM_NAME column of V$TRANSPORTABLE_PLATFORM. You must use the exact name of the source or target platform as a parameter to the CONVERT command. The following SQL statement queries supported Linux platforms:
	`SELECT PLATFORM_NAME, ENDIAN_FORMAT`
	`FROM V$TRANSPORTABLE_PLATFORM`
	`WHERE UPPER(PLATFORM_NAME) LIKE 'LINUX%';`

PARALLELISM *integer*	Specifies the number of channels to be used to perform the operation. If not used, then channels allocated or configured for disk determine the number of channels.

TO PLATFORM '*platform*'	Specifies the name of the destination platform. If not specified, the default is the platform of the database to which RMAN is connected as TARGET.
	The specified platform must be one of the platforms listed in the PLATFORM_NAME column of V$TRANSPORTABLE_PLATFORM. You must use the exact name of the source or target platform as a parameter to the CONVERT command. The following SQL statement queries supported Linux platforms:
	`SELECT PLATFORM_NAME, ENDIAN_FORMAT`
	`FROM V$TRANSPORTABLE_PLATFORM`
	`WHERE UPPER(PLATFORM_NAME) LIKE 'LINUX%';`

Examples

Example 2–53 Converting Tablespaces on the Source Platform

Suppose you must convert tablespaces finance and hr in source database prodlin to the platform format of destination database prodsun. The finance tablespace includes datafiles /disk2/orahome/fin/fin01.dbf and /disk2/orahome/fin/fin02.dbf. The hr tablespace includes datafiles /disk2/orahome/fin/hr01.dbf and /disk2/orahome/fin/hr02.dbf.

The prodlin database runs on Linux host lin01. You query V$DATABASE and discover that platform name is Linux IA (32-bit) and uses a little-endian format. The prodsun database runs on Solaris host sun01. You query V$TRANSPORTABLE_PLATFORM and discover that the PLATFORM_NAME for the Solaris host is Solaris[tm] OE (64-bit), which uses a big-endian format.

You plan to convert the tablespaces on the source host and store the converted datafiles in `/tmp/transport_to_solaris/` on host `lin01`. The example assumes that you have set `COMPATIBLE` is to 10.0 or greater on the source database.

On source host `lin01`, you start the RMAN client and run the following commands:

```
CONNECT TARGET SYS@prodlin

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

SQL 'ALTER TABLESPACE finance READ ONLY';
SQL 'ALTER TABLESPACE hr READ ONLY';
CONVERT TABLESPACE finance, hr
  TO PLATFORM 'Solaris[tm] OE (64-bit)'
  FORMAT '/tmp/transport_to_solaris/%U';
```

The result is a set of converted datafiles in the `/tmp/transport_to_solaris/` directory, with data in the right endian-order for the Solaris 64-bit platform.

From this point, you can follow the rest of the general outline for tablespace transport. Use the Data Pump Export utility to create the file of structural information, if you have not already, move the structural information file and the converted datafiles from `/tmp/transport_to_solaris/` to the desired directories on the destination host, and plug the tablespace into the new database with the Data Pump Import utility.

Example 2–54 Converting Datafiles on the Destination Platform

This example assumes that you want to convert the `finance` and `hr` tablespaces from database `prodsun` on host `sun01` into a format usable by database `prodlin` on destination host `lin01`. You will temporarily store the unconverted datafiles in directory `/tmp/transport_from_solaris/` on destination host `lin01` and perform the conversion with `CONVERT DATAFILE`. When you transport the datafiles into the destination database, they will be stored in `/disk2/orahome/dbs`.

The example assumes that you have carried out the following steps in preparation for the tablespace transport:

- You used the Data Pump Export utility to create the structural information file (named, in our example, `expdat.dmp`).
- You made the `finance` and `hr` tablespaces read-only on the source database.
- You used an operating system utility to copy `expdat.dmp` and the unconverted datafiles to be transported to the destination host `lin01` in the `/tmp/transport_from_solaris` directory. The datafiles are stored as:
  - `/tmp/transport_from_solaris/fin/fin01.dbf`
  - `/tmp/transport_from_solaris/fin/fin02.dbf`
  - `/tmp/transport_from_solaris/hr/hr01.dbf`
  - `/tmp/transport_from_solaris/hr/hr02.dbf`
- You queried the name for the source platform in `V$TRANSPORTABLE_PLATFORM` and discovered that the `PLATFORM_NAME` is `Solaris[tm] OE (64-bit)`.

Note the following considerations when performing the conversion:

- Identify the datafiles by filename, not by tablespace name. Until the datafiles are plugged in, the local instance has no way of knowing the intended tablespace names.

- The FORMAT argument controls the name and location of the converted datafiles.
- When converting on the destination host, you must specify the source platform with the FROM argument. Otherwise, RMAN assumes that the source platform is the same as the platform of the host performing the conversion.

You start the RMAN client and connect to the destination database prodlin as TARGET. The following CONVERT command converts the datafiles to be transported to the destination host format and deposits the results in /disk2/orahome/dbs:

```
CONNECT TARGET SYS@prodlin

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

CONVERT DATAFILE
   '/tmp/transport_from_solaris/fin/fin01.dbf',
   '/tmp/transport_from_solaris/fin/fin02.dbf',
   '/tmp/transport_from_solaris/hr/hr01.dbf',
   '/tmp/transport_from_solaris/hr/hr02.dbf'
   DB_FILE_NAME_CONVERT
        '/tmp/transport_from_solaris/fin','/disk2/orahome/dbs/fin',
        '/tmp/transport_from_solaris/hr','/disk2/orahome/dbs/hr'
   FROM PLATFORM 'Solaris[tm] OE (64-bit)';
```

The result is that the following datafiles have been converted to the Linux format:

- /disk2/orahome/dbs/fin/fin01.dbf
- /disk2/orahome/dbs/fin/fin02.dbf
- /disk2/orahome/dbs/hr/hr01.dbf
- /disk2/orahome/dbs/hr/hr02.dbf

From this point, follow the rest of the general outline for tablespace transport. Use Data Pump Import to plug the converted tablespaces into the new database, and make the tablespaces read/write if applicable.

Example 2–55 Copying Datafiles to and from ASM with CONVERT DATAFILE

This example illustrates copying datafiles into ASM from normal storage. Note that the generated files are not considered datafile copies that belong to the target database, so LIST DATAFILECOPY does not display them.

Use CONVERT DATAFILE without specifying a source or destination platform. Specify ASM disk group +DATAFILE for the output location, as shown here:

```
RMAN> CONVERT DATAFILE '/disk1/oracle/dbs/my_tbs_f1.df',
   '/disk1/oracle/dbs/t_ax1.f'
   FORMAT '+DATAFILE';

Starting conversion at 29-MAY-05
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile conversion
input filename=/disk1/oracle/dbs/t_ax1.f
converted datafile=+DATAFILE/asmv/datafile/sysaux.280.559534477
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:16
channel ORA_DISK_1: starting datafile conversion
input filename=/disk1/oracle/dbs/my_tbs_f1.df
converted datafile=+DATAFILE/asmv/datafile/my_tbs.281.559534493
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:04
Finished conversion at 29-MAY-05
```

The following example illustrates copying the datafiles of a tablespace out of ASM storage to directory `/tmp`, with uniquely generated filenames.

```
RMAN> CONVERT TABLESPACE tbs_2 FORMAT '/tmp/tbs_2_%U.df';

Starting conversion at 03-JUN-05
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=20 devtype=DISK
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00006 name=+DATAFILE/tbs_21.f
converted datafile=/tmp/tbs_2_data_D-L2_I-2786301554_TS-TBS_2_FNO-6_11gm2fq9.df
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00007 name=+DATAFILE/tbs_22.f
converted datafile=/tmp/tbs_2_data_D-L2_I-2786301554_TS-TBS_2_FNO-7_12gm2fqa.df
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00019 name=+DATAFILE/tbs_25.f
converted datafile=/tmp/tbs_2_data_D-L2_I-2786301554_TS-TBS_2_FNO-19_13gm2fqb.df
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00009 name=+DATAFILE/tbs_23.f
converted datafile=/tmp/tbs_2_data_D-L2_I-2786301554_TS-TBS_2_FNO-9_14gm2fqc.df
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile conversion
input datafile fno=00010 name=+DATAFILE/tbs_24.f
converted datafile=/tmp/tbs_2_data_D-L2_I-2786301554_TS-TBS_2_FNO-10_15gm2fqd.df
channel ORA_DISK_1: datafile conversion complete, elapsed time: 00:00:01
Finished conversion at 03-JUN-05
```

Example 2–56 Transporting a Database to a Different Platform

The arguments to `CONVERT DATABASE` vary depending on whether you plan to convert the datafiles on the source or destination platform. For a description of the conversion process on source and destination platforms and extended examples, refer to *Oracle Database Backup and Recovery User's Guide*. Read that discussion in its entirely before attempting a database conversion.

Assume that you want to transport database `prod` on a Linux host to a Windows host. You decide to convert the datafiles on the source host rather than on the destination host. The following example connects RMAN to the `prod` database on the Linux host and uses `CONVERT DATABASE NEW DATABASE` to convert the datafiles and generate the transport script:

```
CONNECT TARGET SYS@lin01

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

CONVERT DATABASE
  NEW DATABASE 'prodwin'
  TRANSPORT SCRIPT '/tmp/convertdb/transportscript'
  TO PLATFORM 'Microsoft Windows IA (32-bit)'
    DB_FILE_NAME_CONVERT '/disk1/oracle/dbs','/tmp/convertdb';
```

In the following variation, you want to transport a database running on a Linux host to a Windows host, but you want to convert the datafiles on the destination host rather

than the source host. The following example connects RMAN to the `prod` database on the Linux host and executes `CONVERT DATABASE ON DESTINATION PLATFORM`:

```
CONNECT TARGET SYS@lin01

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

CONVERT DATABASE
  ON DESTINATION PLATFORM
  CONVERT SCRIPT '/tmp/convertdb/convertscript.rman'
  TRANSPORT SCRIPT '/tmp/convertdb/transportscript.sql'
  NEW DATABASE 'prodwin'
  FORMAT '/tmp/convertdb/%U';
```

The `CONVERT DATABASE ON DESTINATION PLATFORM` command, which is executed on a Linux database, generates a convert script that can be run on the Windows host to convert the datafiles to the Windows format. The `CONVERT DATABASE` command also generates a transport script.

Example 2–57 Transporting a Database to a Different Platform and Storage Type

In this scenario, you have a database `prod` on a Solaris host named `sun01` that you want to move to an AIX host named `aix01`. The Solaris datafiles are stored in a non-ASM file system, but you want to store the datafiles in ASM on the AIX host.

The following example connects to `sun01` and runs `CONVERT DATABASE` to generate the necessary scripts:

```
CONNECT TARGET SYS@sun01

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

CONVERT DATABASE
  ON DESTINATION PLATFORM
  CONVERT SCRIPT '/tmp/convert_newdb.rman'
  TRANSPORT SCRIPT '/tmp/transport_newdb.sql'
  NEW DATABASE 'prodaix'
  DB_FILE_NAME_CONVERT '/u01/oradata/DBUA/datafile','+DATA';
```

The convert script will contain statements of the following form, where *your_source_platform* stands for your source platform:

```
CONVERT DATAFILE '/u01/oradata/DBUA/datafile/o1_mf_system_2lg3905p_.dbf'
  FROM PLATFORM 'your_source_platform'
  FORMAT '+DATA/o1_mf_system_2lg3905p_.dbf';
```

To reduce downtime for the conversion, you can use NFS rather than copying datafiles over the network or restoring a backup. For example, you could mount the Solaris files system on the AIX host as `/net/solaris/oradata`. In this case, you would edit the convert script to reference the NFS-mounted directory as the location of the source datafiles to convert, putting the commands into the following form:

```
CONVERT DATAFILE '/net/solaris/oradata/DBUA/datafile/o1_mf_system_2lg3905p_.dbf'
  FROM PLATFORM 'your_source_platform'
  FORMAT '+DATA/o1_mf_system_2lg3905p_.dbf';
```

You then connect RMAN to the destination database instance, in this case the instance on host `aix01`, and convert the datafiles (during the conversion the database at host

sun01 should remain in open read only mode). Afterward, you connect SQL*Plus to the database instance on `aix01` and run the transport script to create the database.

CREATE CATALOG

Purpose

Use the `CREATE CATALOG` command to create a recovery catalog.

The recovery catalog can be a **base recovery catalog** or a **virtual private catalog**.

- A base recovery catalog is a database schema that contains RMAN metadata for a set of target databases.
- A virtual private catalog is a set of synonyms and views that enable user access to a subset of a base recovery catalog.

See Also:

- *Oracle Database Backup and Recovery User's Guide* to learn how to create the recovery catalog
- Appendix B, "RMAN Compatibility" to learn about the requirements for the compatibility of the recovery catalog and the other components of the RMAN environment

Prerequisites

Execute this command only at the RMAN prompt. RMAN must be connected to the recovery catalog database either through the `CATALOG` command-line option or the `CONNECT CATALOG` command, and the catalog database must be open. A connection to a target database is not required.

The recovery catalog owner, whether the catalog is a base recovery catalog or a virtual private catalog, must be granted the `RECOVERY_CATALOG_OWNER` role. This user must also be granted space privileges in the tablespace where the recovery catalog tables will reside. The recovery catalog is created in the default tablespace of the recovery catalog owner.

If you are creating a virtual private catalog, then the base recovery catalog owner must have used the RMAN `GRANT` command to grant either the `CATALOG` or `REGISTER` privilege (see Example 2–59 on page 2-104).

See the `CONNECT CATALOG` description for restrictions for RMAN client connections to a virtual catalog when the RMAN client is from release Oracle Database 10*g* or earlier.

Usage Notes

Typically, you create the recovery catalog in a database created especially for this purpose. It is not recommended to create the recovery catalog in the `SYS` schema.

The best practice is to create one recovery catalog that serves as the central RMAN repository for many databases. For this reason it is called the **base recovery catalog**.

The owner of the base recovery catalog can `GRANT` or `REVOKE` restricted access to the catalog to other database users. Each restricted user has full read/write access to his own metadata, which is called a **virtual private catalog**. The RMAN metadata is stored in the schema of the virtual private catalog owner. The owner of the base recovery catalog controls what each virtual catalog user can access.

You must take an extra step when intending to use a 10.2 or earlier release of RMAN with a virtual catalog. Before using the virtual private catalog, this user must connect to the recovery catalog database as the virtual catalog owner and execute the following

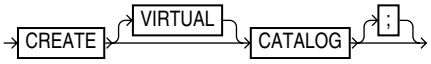
PL/SQL procedure (where *base_catalog_owner* is the database user who owns the base recovery catalog):

```
base_catalog_owner.DBMS_RCVCAT.CREATE_VIRTUAL_CATALOG
```

> **See Also:** *Oracle Database Administrator's Guide* for more information about the RECOVERY_CATALOG_OWNER role

Syntax

***createCatalog*::=**



Semantics

Syntax Element	Description
VIRTUAL	Creates a virtual private catalog in an existing recovery catalog. Run this command after connecting RMAN to the recovery catalog database as the virtual catalog user. **Note:** All of the mechanisms for virtual private catalogs are in the recovery catalog schema itself. The security is provided by the catalog database, not by the RMAN client.

Examples

Example 2–58 Creating a Recovery Catalog and Registering a Database

Assume that you start SQL*Plus and connect to the recovery catalog catdb with administrator privileges. You execute the CREATE USER statement as follows, replacing *password* with a user-specified password (see *Oracle Database Security Guide* for information on creating secure passwords). The SQL statement creates a user catowner in database catdb and grant the catowner user the RECOVERY_CATALOG_OWNER role.

```
SQL> CREATE USER catowner IDENTIFIED BY password
  2  DEFAULT TABLESPACE cattbs
  3  QUOTA UNLIMITED ON cattbs;
SQL> GRANT recovery_catalog_owner TO catowner;
SQL> EXIT
```

You then start RMAN and run the following RMAN commands to connect to the recovery catalog database as catowner and create the recovery catalog:

```
RMAN> CONNECT CATALOG catowner@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CREATE CATALOG;
```

In the same RMAN session, you connect to a target database using operating system authentication and use the REGISTER DATABASE command to register this database in the catalog:

```
RMAN> CONNECT TARGET /
```

```
RMAN> REGISTER DATABASE;
RMAN> EXIT
```

Example 2–59 Creating a Virtual Private Catalog

Assume that you created the recovery catalog and registered a database as shown in Example 2–58. Now you want to create a virtual private catalog for database user `vpc1`. You start SQL*Plus and connect to recovery catalog database `catdb` with administrator privileges. You create the `vpc1` user and grant recovery catalog ownership as follows, replacing *password* with a user-specified password (see *Oracle Database Security Guide* for information on creating secure passwords):

```
SQL> CREATE USER vpc1 IDENTIFIED BY password
  2  DEFAULT TABLESPACE vpcusers
  3  QUOTA UNLIMITED ON vpcusers;
SQL> GRANT recovery_catalog_owner TO vpc1;
SQL> EXIT
```

You then start RMAN and connect to the recovery catalog database as the catalog owner `catowner`. By default, the virtual catalog owner has no access to the base recovery catalog. You use the `GRANT` command to grant virtual private catalog access to `vpc1` for RMAN operations on database `prod1` (but not `prod2`):

```
RMAN> CONNECT CATALOG catowner@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> GRANT CATALOG FOR DATABASE prod1 TO vpc1;
RMAN> EXIT;
```

At this point the backup operator who will use virtual private catalog `vpc1` is ready to create the virtual catalog. In the following example, the backup operator connects to the recovery catalog database as `vpc1` and creates the virtual private catalog for `vpc1`

```
RMAN> CONNECT CATALOG vpc1@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CREATE VIRTUAL CATALOG;
RMAN> EXIT;
```

Because this operator eventually intends to use the virtual catalog with Oracle Database 10*g* target databases, the operator must execute the `CREATE_VIRTUAL_CATALOG` PL/SQL procedure before using the virtual catalog (as explained in "Usage Notes" on page 2-102). In the following example, the backup operator connects to the recovery catalog database as `vpc1` and executes the PL/SQL procedure as follows:

```
SQL> CONNECT vpc1@catdb
Enter password: password
Connected.
SQL> BEGIN
  2  catowner.DBMS_RCVCAT.CREATE_VIRTUAL_CATALOG;
  3  END;
  4  /
```

CREATE SCRIPT

Purpose

Use the `CREATE SCRIPT` command to create a stored script in the recovery catalog. A stored script is a sequence of RMAN commands that is given a name and stored in the recovery catalog for later execution.

See Also:

- *Oracle Database Backup and Recovery User's Guide* to learn how to use stored scripts
- `REPLACE SCRIPT` on page 3-25 to learn how to update a stored script

Prerequisites

Execute `CREATE SCRIPT` only at the RMAN prompt. RMAN must be connected to a target database and a recovery catalog. The recovery catalog database must be open.

If `GLOBAL` is specified, then a global script with this name must not already exist in the recovery catalog. If `GLOBAL` is not specified, then a local script must not already exist with the same name for the same target database. In you do not meet these prerequisites, then RMAN returns error `RMAN-20401`.

Usage Notes

A stored script may be local or global. A local script is created for the current target database only, whereas a global script is available for use with any database registered in the recovery catalog.

It is permissible to create a global script with the same name as a local script, or a local script with the same name as a global script.

Substitution Variables in Stored Scripts

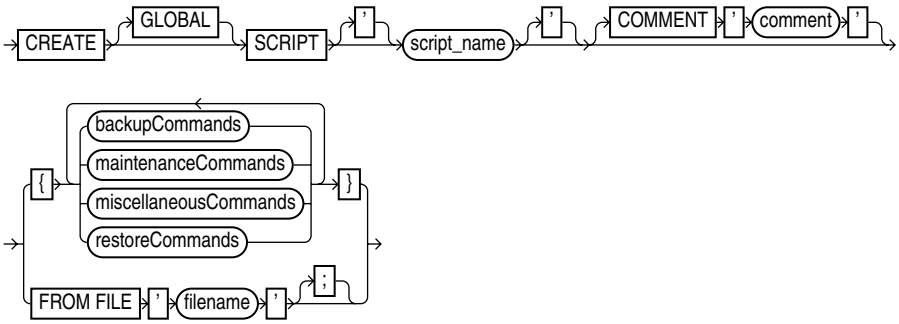
RMAN supports the use of substitution variables in a stored script. `&1` indicates where to place the first value, `&2` indicate where to place the second value, and so on. Special characters must be quoted.

The substitution variable syntax is `&`*`integer`* followed by an optional period, for example, `&1.3`. The optional period is part of the variable and replaced with the value, thus enabling the substitution text to be immediately followed by another integer. For example, if you pass the value `mybackup` to a command file that contains the substitution variable `&1.3`, then the result of the substitution is `mybackup3`. Note that to create the result `mybackup.3`, you would use the syntax `&1..3`.

When you create a stored script with substitution variables, you must provide example values at create time. You can provide these values with the `USING` clause when starting RMAN (see `RMAN` on page 3-59) or enter them when prompted (see Example 2–62).

Syntax

***createScript*::=**



backupCommands::= on page 3-65, *maintenanceCommands::=* on page 3-65, *miscellaneousCommands::=* on page 3-66, *restoreCommands::=* on page 3-66)

Semantics

Syntax Element	Description
`GLOBAL`	Identifies the script as global. **Note:** A virtual private catalog has read-only access to global scripts. Creating or updating global scripts must be done while connected to the base recovery catalog.
`SCRIPT` *`script_name`*	Specifies the name of the script. Quotes must be used around the script name when the name contains either spaces or reserved words.
`COMMENT '`*`comment`*`'`	Associates an explanatory comment with the stored script in the recovery catalog. The comment is used in the output of `LIST SCRIPT NAMES`.
`backupCommands` *`maintenanceCommands`* *`miscellaneousCommands`* *`restoreCommands`*	Specifies commands to include in the stored script. The commands allowable within the brackets of the `CREATE SCRIPT '`*`script_name`*`' {...}` command are the same commands supported within a `RUN` command. Any command that is valid within a `RUN` command is permitted in the stored script. The following commands are not valid within stored scripts: `RUN`, `@ (at sign)`, and `@@ (double at sign)`.
`FROM FILE '`*`filename`*`'`	Reads the sequence of commands to define the script from the specified file. The file should look like the body of a valid stored script. The first line of the file must be a left brace (`{`) and the last line must contain a right brace (`}`). The RMAN commands in the file must be valid in a stored script.

Examples

Example 2–60 Creating a Local Stored Script

Assume that you want to create a local stored script for backing up database `prod`. You start RMAN, connect to `prod` as `TARGET`, and connect to a recovery catalog. You create a stored script called `backup_whole` and then use `EXECUTE SCRIPT` to run it as follows:

```
CREATE SCRIPT backup_whole
COMMENT "backup whole database and archived redo log files"
{
    BACKUP
      INCREMENTAL LEVEL 0 TAG backup_whole
      FORMAT "/disk2/backup/%U"
```

```
      DATABASE PLUS ARCHIVELOG;
}
RUN { EXECUTE SCRIPT backup_whole; }
```

Example 2–61 Creating a Global Stored Script

This example connects RMAN to target database `prod` and recovery catalog database `catdb` as catalog user `rco`. The example creates a global script called `global_backup_db` that backs up the database and archived redo log files:

```
RMAN> CONNECT TARGET SYS@prod

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

RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CREATE GLOBAL SCRIPT global_backup_db { BACKUP DATABASE PLUS ARCHIVELOG; }
RMAN> EXIT;
```

You can now connect RMAN to a different target database such as `prod2` and run the global stored script:

```
RMAN> CONNECT TARGET SYS@prod2

target database Password: password
connected to target database: PROD2 (DBID=36508508)

RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> RUN { EXECUTE SCRIPT global_backup_db; }
```

Example 2–62 Creating a Stored Script That Uses Substitution Variables

The following example connects RMAN to a target database and recovery catalog and uses `CREATE SCRIPT` to create a backup script that includes three substitution variables. RMAN prompts you to enter initial values for the variables (user input is shown in bold).

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

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CREATE SCRIPT backup_df
2> { BACKUP DATAFILE &1 TAG &2.1 FORMAT '/disk1/&3_%U'; }

Enter value for 1: 1

Enter value for 2: df1_backup

Enter value for 3: df1

starting full resync of recovery catalog
```

```
full resync complete
created script backup_df
```

When you run EXECUTE SCRIPT, you can pass different values to the script. The following example passes the values 3, test_backup, and test to the substitution variables in the stored script:

```
RMAN> RUN { EXECUTE SCRIPT backup_df USING 3 test_backup df3; }
```

After the values are substituted, the script executes as follows:

```
BACKUP DATAFILE 3 TAG test_backup1 FORMAT '/disk1/df3_%U';
```

CROSSCHECK

Purpose

Use the CROSSCHECK command to synchronize the physical reality of backups and copies with their logical records in the RMAN repository.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to manage database records in the recovery catalog

Prerequisites

RMAN must be connected to a target database instance, which must be started.

A maintenance channel is not required for a disk crosscheck. If you use a media manager and have not configured automatic channels for it, then you must use run ALLOCATE CHANNEL FOR MAINTENANCE before CROSSCHECK. For example, if you created a backup on an SBT channel, but have not configured automatic SBT channels for this device, then you must manually allocate an SBT channel before CROSSCHECK can check the backup. Furthermore, if you have performed backups with different media manager options (pools, servers, libraries, and so on), then you should allocate maintenance channels for each combination.

Crosscheck validates all specified backups and copies, even if they were created in previous database incarnations.

Usage Notes

RMAN always maintains logical metadata about backups in the control file of every target database on which it performs operations. If you use RMAN with a recovery catalog, then RMAN also maintains the metadata from every registered database in the recovery catalog.

If a backup is on disk, then CROSSCHECK determines whether the header of the file is valid. If a backup is on tape, then RMAN queries the RMAN repository for the names and locations of the backup pieces to be checked. RMAN sends this metadata to the target database server, which queries the media management software about the backups. The media management software then checks its media catalog and reports back to the server with the status of the backups.

EXPIRED and AVAILABLE Status

You can view the status of backup sets and copies recorded in the RMAN repository through LIST, V$ views, or recovery catalog views (if you use RMAN with a catalog). Table 2–4 describes the meaning of each status.

The CROSSCHECK command only processes files created on the same device type as the channels used for the crosscheck. The CROSSCHECK command checks only objects marked AVAILABLE or EXPIRED in the repository by examining the files on disk for DISK channels or by querying the media manager for sbt channels.

Table 2–4 Meaning of Crosscheck Status

Status	Description
EXPIRED	Object is not found either in file system (for DISK) or in the media manager (for sbt). A backup set is EXPIRED if any backup piece in the set is EXPIRED. The CROSSCHECK command does not delete files that it does not find, but updates their repository records to EXPIRED. You can run DELETE EXPIRED to remove the repository records for expired files and any existing physical files whose status is EXPIRED. If backups are EXPIRED, then you can reexecute the crosscheck later and determine whether expired backups are available. This precaution is especially useful when you use RMAN with a media manager. For example, if some backup pieces or copies were erroneously marked as EXPIRED because the PARMS channel settings were incorrect, 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.
AVAILABLE	Object is available for use by RMAN. For a backup set to be AVAILABLE, all backup pieces in the set must have the status AVAILABLE.

Crosschecks in a Data Guard Environment

"RMAN Backups in a Data Guard Environment" on page 2-22 explains the difference between the association and accessibility of a backup. In a Data Guard environment, the database that creates a backup or copy is associated with the file. You can use maintenance commands such as CHANGE, DELETE, and CROSSCHECK for backups when connected to any database in the Data Guard environment as long as the backups are accessible. In general, RMAN considers tape backups created on any database as accessible to all databases in the environment, whereas disk backups are accessible only to the database that created them.

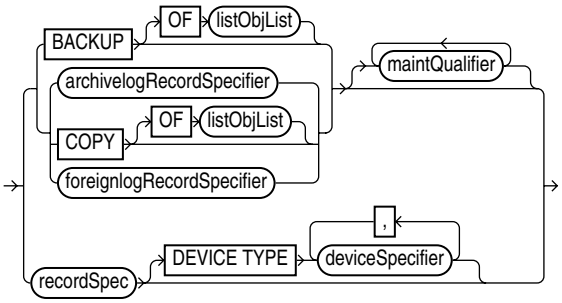
RMAN can only update the status of a backup from AVAILABLE to EXPIRED or DELETED when connected as TARGET to the database associated with the backup. If RMAN cannot delete a backup because it is not associated with the target database, then RMAN prompts you to perform the same CROSSCHECK operation for the file at the database with which it is associated. In this way RMAN protects against unwanted status changes that result from incorrect SBT configurations.

For example, assume that you connect RMAN as TARGET to standby database standby1 and back it up to tape. If the backup is manually removed from the tape, and if you perform a crosscheck of the backup on standby2, then RMAN prompts you to run the crosscheck on standby1. A crosscheck on standby1 updates the status of the tape backup to EXPIRED when the media manager reports that the backup has been deleted.

Syntax

***crosscheck*::=**

***maintSpec*::=**



(*listObjList::=* on page 4-29, *archivelogRecordSpecifier::=* on page 4-6, *foreignlogRecordSpecifier::=* on page 4-21, *maintQualifier::=* on page 4-32, *recordSpec::=* on page 4-37, *deviceSpecifier::=* on page 4-16)

Semantics

Syntax Element	Description
maintSpec	Crosschecks backups. For *maintSpec* options, refer to the parameter descriptions in *maintSpec* on page 4-34.

Examples

Example 2–63 Crosschecking All Backups and Copies

This example, which assumes that the default configured channel is `DEVICE TYPE sbt`, crosschecks all backups and disk (partial output is included). Because RMAN preconfigures a disk channel, you do not need to manually allocate a disk channel.

```
RMAN> CROSSCHECK BACKUP;

allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=84 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=86 device type=DISK
backup piece handle=/disk2/backup/08i9umon_1_1 RECID=7 STAMP=614423319
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=/disk2/backup/09i9umso_1_1 RECID=8 STAMP=614423448
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=/disk1/cfauto/c-26213402-20070213-00 RECID=9 STAMP=614423452
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=0bi9uo81_1_1 RECID=10 STAMP=614424833
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-26213402-20070213-01 RECID=11 STAMP=614424851
crosschecked backup piece: found to be 'AVAILABLE'
.
.
.
```

Example 2–64 Crosschecking Within a Range of Dates

This example queries the media manager for the status of the backup sets in a given six week range. Note that RMAN uses the date format specified in the `NLS_DATE_FORMAT` parameter, which is `'DD-MON-YY'` in this example. The first command crosschecks backups on tape only:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP
  COMPLETED BETWEEN '01-JAN-07' AND '14-FEB-07';
RELEASE CHANNEL;
```

The following command specifies `DEVICE TYPE DISK` to crosscheck only disk:

```
CROSSCHECK BACKUP DEVICE TYPE DISK
  COMPLETED BETWEEN '01-JAN-07' AND '14-FEB-07';
```

If the default channel is SBT, then you can crosscheck both disk and SBT backups by running `CROSSCHECK` with the default channels:

```
CROSSCHECK BACKUP COMPLETED BETWEEN '01-JAN-07' AND '14-FEB-07';
```

DELETE

Purpose

Use the DELETE command to perform the following actions:

- Delete physical backups and copies.
- Update the repository records in the target control file to show that the files are deleted. For example, the record for a backup piece in V$BACKUP_PIECE.STATUS will show the value D.
- Remove the repository records for deleted files from the recovery catalog (if you use a catalog). For example, RC_BACKUP_PIECE will no longer contain a row for a deleted backup piece.

> **See Also:** BACKUP on page 2-19 to learn about the BACKUP . . . DELETE INPUT command

Prerequisites

RMAN must be connected to a target database, which must be mounted or open.

RMAN uses all configured channels to perform the deletion. If you use DELETE for files on devices that are *not* configured for automatic channels, then you must use ALLOCATE CHANNEL FOR MAINTENANCE. For example, if you made a backup with an SBT channel, but only a disk channel is configured, then you must manually allocate an SBT channel for DELETE. An automatic or manually allocated maintenance channel is required when you use DELETE on a disk-only file.

Usage Notes

The best practice is to run CROSSCHECK to update the status of backups and copies in the repository and then run DELETE to remove the desired files. When running RMAN interactively, DELETE displays a list of files and prompts for confirmation before deleting any file in the list. If you confirm, then RMAN shows each item as it is deleted. When reading commands from a command file, RMAN does not prompt for confirmation.

You can view the status of backups and copies recorded in the RMAN repository through LIST, V$ views, or recovery catalog views (if you use a catalog). The repository record for a backup can fail to reflect its physical status. For example, a user deletes a disk backup with the Linux rm command. The backup record cannot be updated by rm, so the RMAN repository shows the file as available although it no longer exists.

Behavior of DELETE Command for Files of Different Status Values

Table 2–5 describes the behavior of DELETE when the FORCE option is not specified.

Table 2–5 Behavior of DELETE Command Without FORCE Option

Repository Status	Physical Status	Behavior of DELETE Command
AVAILABLE	Not found on media	Does not delete the object and reports the list of mismatched objects at the end of the job. RMAN does not update the repository status.

Table 2–5 (Cont.) Behavior of DELETE Command Without FORCE Option

Repository Status	Physical Status	Behavior of DELETE Command
EXPIRED	Found on media	Does not delete the object and reports the list of mismatched objects at the end of the job. RMAN does not update the repository status.
UNAVAILABLE	Any	Removes repository record and deletes object if it exists. All I/O errors are ignored.

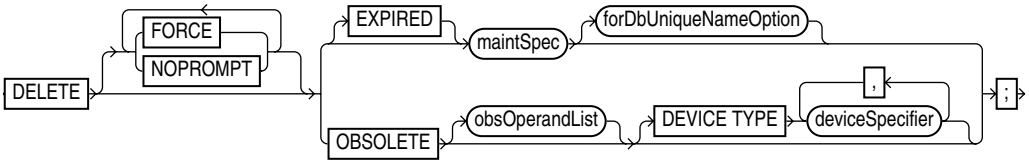
Backup Deletion in a Data Guard Environment

"RMAN Backups in a Data Guard Environment" on page 2-22 explains the difference between the association and accessibility of a backup. In a Data Guard environment, the database that creates a backup or copy is associated with the file. You can use maintenance commands such as CHANGE, DELETE, and CROSSCHECK for backups when connected to any database in the Data Guard environment as long as the backups are accessible. In general, RMAN considers tape backups created on any database as accessible to all databases in the environment, whereas disk backups are accessible only to the database that created them.

If a deletion is successful, then RMAN removes the metadata for the file, even if the file is associated with another database. If a deletion was not successful, and if the file is associated with another database in the Data Guard environment, then RMAN prompts you to perform the same DELETE command while connected as TARGET to the database associated with the file. You must use DELETE . . . FORCE to delete the file metadata.

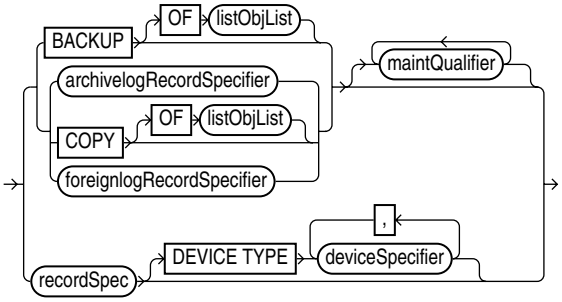
Syntax

***delete*::=**



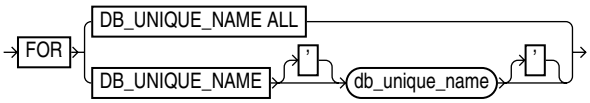
(*maintSpec::=* on page 4-34, *obsOperandList::=* on page 4-36, *deviceSpecifier::=* on page 4-16)

***maintSpec*::=**



(*listObjList::=* on page 4-29, *archivelogRecordSpecifier::=* on page 4-6, *maintQualifier::=* on page 4-32, *deviceSpecifier::=* on page 4-16, *recordSpec::=* on page 4-37)

forDbUniqueNameOption::=



Semantics

Syntax Element	Description
FORCE	Deletes specified files—whether or not they exist on the media—and removes repository records (see Example 2–68 on page 2-117). RMAN ignores any I/O errors for the deleted objects. RMAN also ignores any CONFIGURE ARCHIVELOG DELETION POLICY settings. RMAN displays the number of deleted objects at the end of the job.
NOPROMPT	Deletes specified files without first listing the files or prompting for confirmation. The DELETE NOPROMPT command displays each item as it is deleted.
EXPIRED	Removes only files whose status in the repository is EXPIRED (see Example 2–65 on page 2-116). RMAN marks backups and copies as expired when you run a CROSSCHECK command and the files are absent or inaccessible. To determine which files are expired, run a LIST EXPIRED command. If for some reason a backup marked EXPIRED exists when you run the DELETE EXPIRED command, then RMAN does not delete the physical file.
maintSpec	Deletes backups and copies. You can set rules for the deletion with the *maintQualifier* clause. For example, you can delete archived redo log files that have already been backed up to tape (see Example 2–67 on page 2-116). **Note:** DELETE ARCHIVELOG ALL considers only the archived log deletion policy and does not consider the configured retention policy. **See Also:** *maintSpec* on page 4-34 and *maintQualifier* on page 4-32
forDbUniqueNameOption	Deletes the backups and copies in *maintSpec* that are exclusively associated with the specified DB_UNIQUE_NAME in a Data Guard environment. **Note:** The FOR DB_UNIQUE_NAME option is not allowed with the DELETE OBSOLETE command. If RMAN successfully deletes tape backups associated with the specified DB_UNIQUE_NAME, then RMAN removes the metadata for these files from the recovery catalog. If RMAN could not delete these files because they are associated with a different database in the Data Guard environment, then RMAN prompts you to perform the same DELETE operation for these files at the database that is associated with them. **Note:** You cannot use FORCE to override the default behavior and specify that RMAN should delete files that are associated with a different database. In this way, RMAN protects you from accidental deletions caused by incorrect RMAN configurations for SBT. To remove the metadata for files that RMAN prevents you from deleting, use the CHANGE RESET DB_UNIQUE_NAME command. **See Also:** *forDbUniqueNameOption* on page 4-20 for descriptions of the options in this clause

Syntax Element	Description
OBSOLETE	Deletes datafile backups and copies recorded in the RMAN repository that are obsolete, that is, no longer needed (see Example 2–66 on page 2-116). RMAN also deletes obsolete archived redo log files and log backups. RMAN determines which backups and copies of datafiles are no longer needed, which in turn determines when logs (and backups of logs) are no longer needed. RMAN considers the creation of a datafile as a backup when deciding which logs to keep. RMAN first uses the options specified with *obsOperandList* to determine which files are obsolete. If you do not specify options in *obsOperandList*, then RMAN uses the options specified in CONFIGURE RETENTION POLICY. **Note:** DELETE OBSOLETE considers only the backup retention policy and does not use the configured archived log deletion policy to determine which logs are obsolete. In contrast, DELETE ARCHIVELOG ALL considers only the configured archived log deletion policy. **Note:** If you make a backup with the KEEP UNTIL TIME clause, then this backup becomes obsolete after the specified KEEP time passes and will be removed by DELETE OBSOLETE. RMAN does not consider the backup retention policy for archival backups whose KEEP time has expired.
obsOperandList	Specifies the criteria for determining which backups and copies are obsolete. **See Also:** *obsOperandList* on page 4-36
DEVICE TYPE *deviceSpecifier*	Restricts the deletion to obsolete backups and copies created on the specified device type only. **See Also:** *deviceSpecifier* on page 4-16

Examples

Example 2–65 Deleting Expired Backups

This example uses a configured sbt channel to check the media manager for expired backups of the tablespace users that are more than one month old and removes their recovery catalog records.

```
CROSSCHECK BACKUPSET OF TABLESPACE users
  DEVICE TYPE sbt COMPLETED BEFORE 'SYSDATE-31';
DELETE NOPROMPT EXPIRED BACKUPSET OF TABLESPACE users
  DEVICE TYPE sbt COMPLETED BEFORE 'SYSDATE-31';
```

Example 2–66 Deleting Obsolete Backups

This example deletes backups and copies that are not needed to recover the database to an arbitrary SCN within the last week. RMAN also deletes archived redo log files that are no longer needed.

```
DELETE NOPROMPT OBSOLETE RECOVERY WINDOW OF 7 DAYS;
```

Example 2–67 Deleting Archived Redo Log Files That Have Already Been Backed Up

Assume that you have configured RMAN settings as follows:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE ARCHIVELOG DELETION POLICY TO
  BACKED UP 2 TIMES
  TO DEVICE TYPE sbt;
```

The following DELETE command deletes all archived redo log files on disk if they are not needed to meet the configured deletion policy, which specifies that logs must be backed up twice to tape (sample output included):

```
RMAN> DELETE ARCHIVELOG ALL;

allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=84 device type=DISK

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

Key     Thrd Seq     S Low Time
------- ---- ------- - ---------
107     1    4       A 12-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_12/o1_mf_1_4_2x28bpcm_.arc
108     1    5       A 12-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_12/o1_mf_1_5_2x28g7s9_.arc
109     1    6       A 12-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_6_2x3bbqym_.arc
157     1    7       A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_7_2x3w2cvs_.arc
164     1    8       A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_8_2x3w40vr_.arc
171     1    9       A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_9_2x3w8pf7_.arc
318     1    10      A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_10_2x3zx6d9_.arc
330     1    11      A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_11_2x403wco_.arc
448     1    12      A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_12_2x40wn6x_.arc
455     1    13      A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_13_2x412s3m_.arc
583     1    14      A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_14_2x428p9d_.ar
638     1    15      A 13-FEB-07
        Name: /orcva/PROD/archivelog/2007_02_13/o1_mf_1_15_2x42f0gj_.arc

Do you really want to delete the above objects (enter YES or NO)?
```

Example 2–68 Forcing the Deletion of a Backup Set

The following example attempts to delete the backup set copy with tag weekly_bkup:

```
RMAN> DELETE NOPROMPT BACKUPSET TAG weekly_bkup;
```

RMAN displays a warning because the repository shows the backup set as available, but the object is not actually available on the media:

```
List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
------- ------- --- --- ----------- ----------- ----------
809     806     1   1   AVAILABLE   SBT_TAPE    0ri9uu08_1_1

RMAN-06207: WARNING: 1 objects could not be deleted for SBT_TAPE channel(s) due
RMAN-06208:          to mismatched status.  Use CROSSCHECK command to fix status
RMAN-06210: List of Mismatched objects
RMAN-06211: ==========================
RMAN-06212:   Object Type   Filename/Handle
RMAN-06213: --------------- ---------------------------------------------------
```

```
RMAN-06214: Backup Piece    0ri9uu08_1_1
```

The following command forces RMAN to delete the backup set (sample output included):

```
RMAN> DELETE FORCE NOPROMPT BACKUPSET TAG weekly_bkup;

using channel ORA_SBT_TAPE_1
using channel ORA_DISK_1

List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
------- ------- --- --- ----------- ----------- ----------
809     806     1   1   AVAILABLE   SBT_TAPE    0ri9uu08_1_1
deleted backup piece
backup piece handle=0ri9uu08_1_1 RECID=26 STAMP=614430728
Deleted 1 objects
```

DELETE SCRIPT

Purpose

Use the `DELETE SCRIPT` command to delete a local or global stored script from the recovery catalog.

Prerequisites

Execute `DELETE SCRIPT` only at the RMAN prompt. RMAN must be connected to a recovery catalog and target database. The recovery catalog database must be open.

Usage Notes

A stored script may be local or global. A local script is created for the current target database only, whereas a global script is available for use with any database registered in the recovery catalog.

If `GLOBAL` is specified, then a global script with this name must already exist in the recovery catalog; otherwise, RMAN returns error `RMAN-06710`. If you do not specify `GLOBAL`, then RMAN looks for a local stored script with the specified name defined on the current target database. If no such script is defined on the target database, then RMAN checks for a global stored script with this name and deletes it if it exists.

Syntax

***deleteScript*::=**

Semantics

Syntax Element	Description
`GLOBAL`	Identifies the script as global. If you attempt to delete a global script, then RMAN must not be connected to a virtual private catalog. Virtual catalog users cannot modify global scripts, although they can execute them. **See Also:** "Usage Notes" on page 2-105 for an explanation of the difference between global and local scripts
`SCRIPT` *`script_name`*	Specifies the name of the script to delete. Quotes must be used around the script name when the name contains either spaces or reserved words.

Example

Example 2–69 Deleting a Global Script

This example deletes global script `backup_db` from the recovery catalog (sample output included):

```
RMAN> LIST SCRIPT NAMES;

List of Stored Scripts in Recovery Catalog
```

```
    Scripts of Target Database PROD

       Script Name
       Description
       ---------------------------------------------------------------------
       backup_whole
       backup whole database and archived redo log files


    Global Scripts


       Script Name
       Description
       ---------------------------------------------------------------------
       global_backup_db
       back up any database from the recovery catalog, with logs

RMAN> DELETE GLOBAL SCRIPT global_backup_db;

deleted global script: global_backup_db

RMAN> LIST SCRIPT NAMES;

List of Stored Scripts in Recovery Catalog


    Scripts of Target Database PROD

       Script Name
       Description
       ---------------------------------------------------------------------
       backup_whole
       backup whole database and archived redo log files
```

DROP CATALOG

Purpose

Use the `DROP CATALOG` command to remove the recovery catalog or a virtual private catalog.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to drop the recovery catalog

Prerequisites

Execute this command only at the RMAN prompt.

You must be connected to the recovery catalog schema or virtual private catalog schema with the `CATALOG` command-line option or the `CONNECT CATALOG` command. The recovery catalog database must be open.

You do not have to be connected to a target database.

Usage Notes

After you execute `DROP CATALOG`, RMAN prompts you to enter the command again to confirm that you want to perform the operation.

A base recovery catalog is created with `CREATE CATALOG`, whereas a virtual private catalog is created with `CREATE VIRTUAL CATALOG`. To drop the base recovery catalog, execute `DROP CATALOG` while connected to the recovery catalog database as the recovery catalog owner.

> **Caution:** When you drop the base recovery catalog, all RMAN metadata is removed from the recovery catalog. Any backups recorded in the recovery catalog but not in a target database control are not usable by RMAN.

To drop a virtual private catalog, execute the `DROP CATALOG` command while connected to the virtual private catalog. When connected to a virtual private catalog, the `DROP CATALOG` command does *not* remove the base recovery catalog itself, but only drops the synonyms and views that refer to the base catalog.

Note that you must use a different technique to drop a virtual catalog when using a 10.2 or earlier release of the RMAN client. Before dropping the virtual private catalog, the user must connect to the recovery catalog database as the virtual private catalog owner and execute the following PL/SQL procedure (where *`base_catalog_owner`* is the database user who owns the base recovery catalog):

```
base_catalog_owner.DBMS_RCVCAT.DROP_VIRTUAL_CATALOG
```

If you drop the base recovery catalog but not the virtual private catalog, then the virtual catalog is unusable. However, if a dedicated database user owns the virtual private catalog, then you can execute `DROP USER ... CASCADE` to remove the virtual catalog.

Syntax

***dropCatalog*::=**

Example

Example 2–70 Dropping a Virtual Private Catalog

Assume that you want to remove the virtual private catalog belonging to database user `vpu1`, but do not want to drop the base recovery catalog. This example connects to the recovery catalog database as `vpu1` and drops the virtual private catalog for this user. The base recovery catalog is not affected by the removal of this virtual private catalog.

```
RMAN> CONNECT CATALOG vpu1@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> DROP CATALOG;

recovery catalog owner is VPU1
enter DROP CATALOG command again to confirm catalog removal

RMAN> DROP CATALOG;

virtual catalog dropped
```

DROP DATABASE

Purpose

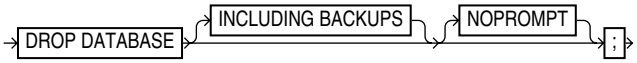
Use the `DROP DATABASE` command to delete the target database and, if RMAN is connected to a recovery catalog, unregister it. RMAN removes all datafiles, online redo logs, and control files belonging to the target database. By default, RMAN prompts for confirmation.

Prerequisites

Execute this command only at the RMAN prompt. You must be connected to a target database. The target database must be mounted exclusive and not open, and started in `RESTRICT` mode.

Syntax

***dropDatabase*::=**



Semantics

Syntax Element	Description
`INCLUDING BACKUPS`	Deletes backup sets, proxy copies, image copies, and archived redo log files associated with the target database from all configured device types. **Note:** If you have been using a recovery catalog but run RMAN in `NOCATALOG` mode when you drop the database, then RMAN will not delete any backups which are known to the recovery catalog but no longer exist in the target database control file.
`NOPROMPT`	Does not prompt for confirmation before deleting the database.

Example

Example 2–71 Deleting a Database

In this example, you want to delete a test database called `test1` that is registered in the recovery catalog. You start the RMAN client, connect to database `test1` as `TARGET`, and connect to the recovery catalog. You then run the following commands to delete the target database files, as well as all backups, copies, and archived redo log files associated with the database:

```
RMAN> CONNECT TARGET SYS@test1

target database Password: password
connected to target database: TEST1 (DBID=39525561)

RMAN> STARTUP FORCE MOUNT
RMAN> SQL 'ALTER SYSTEM ENABLE RESTRICTED SESSION';
RMAN> DROP DATABASE INCLUDING BACKUPS NOPROMPT;
```

DUPLICATE

Purpose

Use the `DUPLICATE` command to create a copy of a **source database**. RMAN can create either of the following types of databases:

- A **duplicate database**, which is a copy of the source database (or a subset of the source database) with a unique DBID. Because a duplicate database has a unique DBID, it is independent of the source database and can be registered in the same recovery catalog. Typically, duplicate databases are used for testing.
- A **standby database**, which is a special copy of the source database (called a **primary database** in a Data Guard environment) that is updated by applying archived redo log files from the primary database. A standby database is not assigned a new DBID.

RMAN can perform the duplication in any of the following supported modes:

- **Active duplication**

  RMAN duplicates the files directly from either an open or mounted database.
- **Backup-based duplication without a target connection**

  RMAN creates duplicate files from pre-existing RMAN backups and copies. The `DUPLICATE` command must have been issued with the `DATABASE` clause. This form requires a connection to an auxiliary instance and a recovery catalog.

  This mode is useful when the target database is not available or a connection to it is not desirable (as mandated by security policy restrictions or a firewall).
- **Backup-based duplication with a target connection**

  RMAN creates duplicate files from pre-existing RMAN backups and copies.
- **Backup-based duplication without connection to target and recovery catalog**

  RMAN creates duplicate files from RMAN backups and copies that were placed in a designated `BACKUP LOCATION`.

  See Also:

  - *Oracle Database Backup and Recovery User's Guide* to learn how to create a duplicate database with the `DUPLICATE` command
  - *Oracle Data Guard Concepts and Administration* to learn how to create, manage, and back up a standby database

Additional Topics

- Prerequisites
- Usage Notes
- Syntax
- Semantics
- Examples

Prerequisites

The prerequisites vary depending on the type of duplication.

Prerequisites Common to All Forms of Duplication

RMAN must be connected as AUXILIARY to the instance of the duplicate database. The instance of the duplicate database is called the **auxiliary instance**. The auxiliary instance must be started with the NOMOUNT option.

The **source host** is the database on which the source database resides. The **destination host** is the host on which you intend to create the duplicate database. If you intend to create the duplicate database on the source host, then set the CONTROL_FILES initialization parameter appropriately so that the DUPLICATE command does not generate an error because the source control file is in use. Also, set all *_DEST initialization parameters appropriately so that the source database files are not overwritten by the duplicate database files.

The source and duplicate databases must be on the same platform. In the context of DUPLICATE, 32-bit and 64-bit versions of the same platform are considered the same platform. For example, Linux IA (32-bit) is considered the same platform as Linux IA (64-bit). However, after duplicating a database between 32-bit and 64-bit platforms, you must run the utlirp.sql script to convert the PL/SQL code to the new format. This script is located in *ORACLE_HOME*/rdbms/admin on Linux and UNIX platforms.

The DUPLICATE command requires one or more auxiliary channels. These channels perform the work of the duplication on the auxiliary database instance. In the following circumstances, RMAN uses the channel configuration from the source database for auxiliary channels:

- You have not used ALLOCATE CHANNEL to manually allocate auxiliary channels.
- You have not used CONFIGURE to configure auxiliary channels.

If you have configured automatic target channels to use CONNECT strings, then RMAN attempts to replicate the channel allocation on the auxiliary instance. However, if you need to control or vary the channel allocation for duplication, you should manually allocate auxiliary channels.

If the COMPATIBLE initialization parameter is set greater than or equal to 11.0.0, then by default RMAN duplicates transportable tablespaces that were not made read/write after being transported. Otherwise, RMAN cannot duplicate transportable tablespaces unless they have been made read/write after being transported.

The following database encryption features both use the wallet: transparent data encryption, which functions at the column level, and tablespace encryption. If you are duplicating an encrypted tablespace, then you must manually copy the wallet to the duplicate database.

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* to learn about transparent data encryption

Prerequisites Specific to Backup-Based Duplication

As shown in Table 2–6, the prerequisites for backup-based duplication depend on whether RMAN is connected as TARGET to the source database.

Table 2–6 Prerequisites for Three Modes of Backup-Based Duplication

Prerequisite	No Target and No Recovery Connection	No Target Connection	Target Connection
RMAN requires a connection to a recovery catalog.	No	Yes	No
All backups and archived redo log files used for creating and recovering the duplicate database must be accessible by the server session on the destination host.	Yes	Yes	Yes
If the destination host is different from the source host, then you must make backups on disk on the source host available to the destination host with the same full path name as in the source database.	No	Yes	Yes
You must provide the name of the source database with the `DATABASE` clause. If the name of the source database is not unique in the recovery catalog, then you must also provide the database ID (DBID) in the `DATABASE` clause.	No if `BACKUP LOCATION` does not have backups from multiple databases.	Yes	No
`UNTIL` clause must be specified in the current incarnation.	Not applicable	No	Yes
`NOREDO` must be specified when RMAN must prevent application of archived redo log files to the backups (see description of `NOREDO` for more information).	Yes	Yes	Yes
If you duplicate a subset of tablespaces, and if the source database is not open, then any duplicated tablespaces with undo segments must be listed in the `UNDO TABLESPACE` clause.	Yes	No	Yes (if no catalog used and target is not open)
RMAN automatically enforces the rule that the set of tablespaces must be self-contained and must not contain database objects owned by `SYS`.	No	No	Yes

Prerequisites Specific to Active Database Duplication

When you execute `DUPLICATE` with `FROM ACTIVE DATABASE`, at least one normal target channel and at least one `AUXILIARY` channel are required.

When you connect RMAN to the source database as `TARGET`, you must specify a password, even if RMAN uses operating system authentication. The source database must be mounted or open. If the source database is open, then archiving must be enabled. If the source database is not open, then it must have been shut down consistently.

When you connect RMAN to the auxiliary instance, you must provide a net service name. This requirement applies even if the auxiliary instance is on the local host.

The source database and auxiliary instances must use the same `SYSDBA` password, which means that both instances must already have password files. You can create the password file with a single password so you can start the auxiliary instance and enable the source database to connect to it.

The `DUPLICATE` behavior for password files varies depending on whether your duplicate database will act as a standby database. If you create a duplicate database that is not a standby database, then RMAN does not copy the password file by default. You can specify the `PASSWORD FILE` option to indicate that RMAN should overwrite

the existing password file on the auxiliary instance. If you create a standby database, then RMAN copies the password file to the standby host by default, overwriting the existing password file. In this case, the `PASSWORD FILE` clause is not necessary.

You cannot use the `UNTIL` clause when performing active database duplication. RMAN chooses a time based on when the online datafiles have been completely copied, so that the datafiles can be recovered to a consistent point in time.

> **See Also:** *Oracle Database Security Guide* to learn about password protection

Usage Notes

Active database duplication uses the auxiliary service name to copy the source database over the network to the auxiliary instance on the destination host, whereas backup-based duplication uses pre-existing RMAN backups and copies. Table 2–7 shows which files from the source database are duplicated.

Table 2–7 Duplicated Files

Source Database Files	Active Database	Backup-Based
Control files	Copied from source database when `FOR STANDBY` specified; otherwise re-created	Restored from backups when `FOR STANDBY` specified or duplication without a connection to the target database; otherwise re-created
Datafiles	Copied from source database (unless excluded with a `SKIP` option)	Restored from backups (unless excluded with a `SKIP` option)
Tempfiles	Re-created (see "Tempfile Re-Creation" on page 2-129)	Re-created (see "Tempfile Re-Creation" on page 2-129)
Online redo log files	Re-created	Re-created
Standby redo log files	Re-created when `FOR STANDBY` specified and defined on primary database	Re-created when `FOR STANDBY` specified and defined on primary database
Archived redo log files	Copied from source database, but only if needed for the duplication	Obtained from backups or cataloged copies, but only if needed for the duplication
Server parameter file	Copied from source database (see `SPFILE` clause in *dupOptionList* on page 2-132)	Restored from backup if `SPFILE` clause is specified (see *dupOptionList* on page 2-132)
Flashback log files	Not re-created	Not re-created
Block change tracking file	Not re-created	Not re-created
Password file	Copied by default for standby databases; for nonstandby databases, copied only if `PASSWORD FILE` option is specified	Not re-created
Backups and other files in fast recovery area	Not copied	Not copied

All datafiles are included in the duplicate database unless they are offline clean or excluded. You can exclude tablespaces by means of the `SKIP` clause, or by including only a subset of tablespaces with `DUPLICATE ... TABLESPACE`.

The fast recovery area is defined on the duplicate or standby database if you explicitly define it. Also, if a flash recovery was defined on the source database, and if the auxiliary instance uses a server parameter file that was copied or restored with the `DUPLICATE` command, then a flash recovery is defined on the duplicate or standby database.

If you use active database duplication, then see the FROM ACTIVE DATABASE description in *dupOptionList* on page 2-132 for usage notes.

Backup-Based Duplication

In backup-based duplication of databases with a connection to the target database and in NOARCHIVELOG mode, media recovery uses the NOREDO option. Thus, if incremental backups exist, RMAN applies only these incremental backups to the restored files during recovery.

For backup-based duplication of databases in ARCHIVELOG mode, RMAN recovers by default up to the last archived redo log generated at the time the command was executed, or until a time specified with a SET UNTIL clause.

For backup-based duplication of databases without a connection to the target database, RMAN cannot determine whether the source database was in NOARCHIVELOG mode. Therefore, you must use the NOREDO option when the source database was in NOARCHIVELOG mode when the backups were taken. You can also use the NOREDO option when you do not want to apply archived redo log files to a consistent backup.

If you are using backup-based duplication, and if the source database and auxiliary instances reside on different hosts, then you must decide how to make the backups of the source database available to the auxiliary instance. For more information on how to do this with BACKUP LOCATION, review the options described in the section *Making Disk Backups Accessible to the Auxiliary Instance* in the *Oracle Database Backup and Recovery User's Guide*.

If the target database does not use a recovery area in ASM storage, then perform one of the following tasks before executing the DUPLICATE command:

- If you are using SBT backups, then make the tapes with the backups accessible to the destination host.
- If you are using disk backups, and if you can use the same backup directory names on the destination host as the source host, then do one of the following:
  - Manually transfer the backups and copies from the source host to the destination host to an identical path.
  - Use NFS or shared disks and make sure that the same path is accessible in the destination host.
- If you are using disk backups, and if you *cannot* use the same backup directory names on the destination host as the source host, then use of the techniques described in *Oracle Database Backup and Recovery User's Guide*.

If the source database uses a recovery area in ASM storage, then perform one of the following tasks before executing the DUPLICATE command:

- Make a database backup to a location outside the fast recovery area. You can make this backup accessible in the following ways:
  - Use NFS to mount the backup on the destination host with the same name.
  - Use NFS to mount the backup on the destination host with a different name, and then CATALOG the backup while RMAN is connected as TARGET to the source database.
- Back up the fast recovery area to tape and use it for duplication.

Duplication with Oracle Managed Files

If the source database files are in the Oracle Managed Files (OMF) format, then you cannot use the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT initialization parameters or the *fileNameConversionSpec* clause to generate new OMF filenames for the duplicate database. If you do not follow this rule, the new OMF files generated from these three methods can cause problems. For more information on OMF names, see the "Considerations When Renaming OMF Auxiliary Set Files in TSPITR" in the *Oracle Database Backup and Recovery User's Guide*

The only exception to this rule is when changing only an ASM disk group name. Assume that source datafiles and online redo log files are stored in ASM disk group +SOURCEDSK. You want to store the duplicate database files in ASM disk group +DUPDSK. In this case, you can set the initialization parameters as follows:

```
DB_FILE_NAME_CONVERT = ("+SOURCEDSK","+DUPDSK")
LOG_FILE_NAME_CONVERT = ("+SOURCEDSK","+DUPDSK")
```

RMAN uses DB_FILE_NAME_CONVERT or LOG_FILE_NAME_CONVERT to convert the disk group name, and then generates a new, valid filename based on the converted disk group name.

You have the following other supported options for naming datafiles when the source files are in the Oracle Managed Files format:

- Use SET NEWNAME to specify names for individual datafiles.
- Set DB_FILE_CREATE_DEST to make all datafiles of the new database Oracle-managed files, with the exception of the files for which SET NEWNAME is used. You should not set DB_FILE_NAME_CONVERT if you set DB_FILE_CREATE_DEST.

Supported options for naming online redo logs duplicated from Oracle-managed files are DB_CREATE_FILE_DEST, DB_RECOVERY_FILE_DEST, or DB_CREATE_ONLINE_LOG_DEST_*n*.

Tempfile Re-Creation

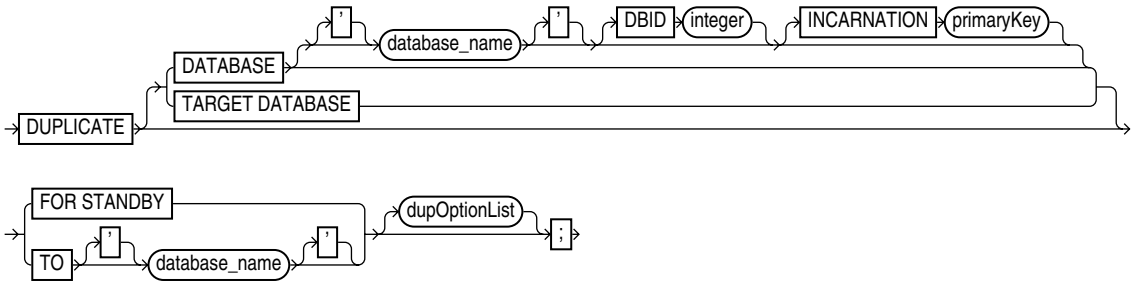
The DB_FILE_NAME_CONVERT parameter can be used to convert the tempfile names for the new database that are not Oracle-managed files (OMF). The only exception to this restriction are Automatic Storage Management (ASM) OMF names where only the name of the disk group can be changed.

The other method for converting tempfile names for the new database is to use SET NEWNAME FOR TEMPFILE TO'filename' or TO NEW. With this latter method, it does not matter if the datafiles are OMF or non-OMF, the tempfiles are re-created in the DB_CREATE_FILE_DEST directory location when the database is opened.

To specify different filenames for the tempfiles, see the discussion of SWITCH TEMPFILE on page 3-92.

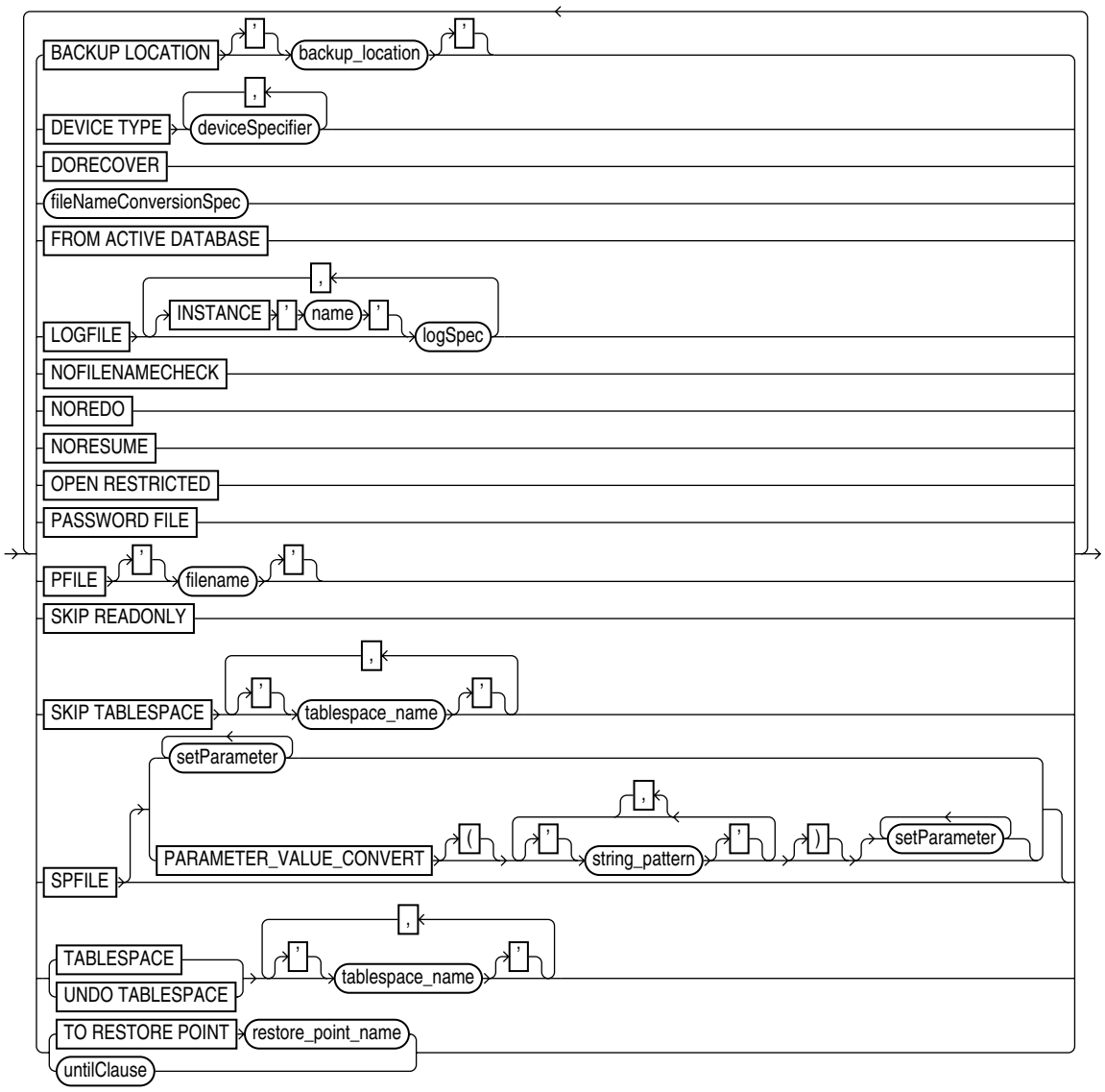
Syntax

duplicate::=



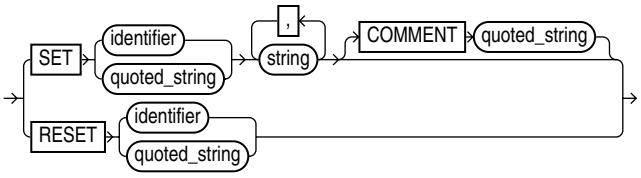
(*dupOptionList::=* on page 2-130)

dupOptionList::=

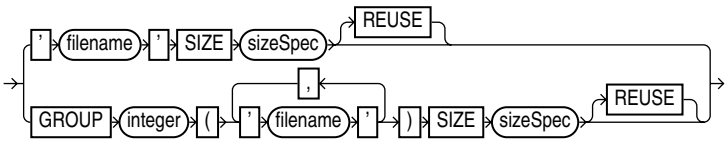


(*deviceSpecifier::=* on page 4-16, *fileNameConversionSpec::=* on page 4-18, *logSpec::=* on page 2-131, *setParameter::=* on page 2-131, *untilClause::=* on page 4-42)

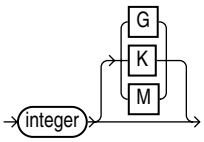
setParameter::=



logSpec::=



sizeSpec::=



Semantics

duplicate

This clause enables you to duplicate a database or tablespace. Refer to the *duplicate::=* diagram on page 2-130 for the syntax.

Syntax Element	Description
`TARGET DATABASE`	Specifies the source database, which is the database you want to duplicate. Starting with Oracle Database 11*g* Release 2 the `TARGET` keyword is optional.
`DATABASE`	Specifies the source database
`'database_name'`	Specifies the name of the source database, which is the database that you want to duplicate. This clause can be used instead of `TARGET DATABASE` when RMAN is connected as `TARGET` to the source database. When performing duplicate with a target connection, then you must specify the currently connected database and not use the INCARNATION subclause. For backup-based duplication without a target connection, you must either specify the database name or run the `SET DATABASE` *'database_name'* command.
`DBID integer`	Specifies the database ID (DBID) of the source database. The `DBID` parameter is required when you duplicate without a `TARGET` connection to the source database and the database name in the recovery catalog is not unique. Another option is to use the `SET DBID` command. When RMAN duplicates the database while connected to the source database as `TARGET`, the `DBID` parameter is not needed. If `DBID` is specified, then the parameter value should match the `DBID` of the source database.

Syntax Element	Description
INCARNATION *primaryKey*	Specifies an orphan incarnation. By default, the DUPLICATE command with the UNTIL clause refers to a point in time in the current database incarnation or the direct ancestor of the current incarnation. This clause allows you to designate an incarnation not in the current incarnation path, known as an orphan incarnation. It is invalid to specify INCARNATION when connected to a target database or when using BACKUP LOCATION. Another option is to use the SET INCARNATION command.
FOR STANDBY	Specifies that database being duplicated is to be used as a standby database (see Example 2–81 on page 2-144). To create a standby database with the DUPLICATE command you must specify the FOR STANDBY option. The DUPLICATE ... FOR STANDBY command creates the standby database by restoring a standby control file and mounting the standby control file. If you specify FROM ACTIVE DATABASE, then RMAN copies the datafiles from the primary to standby database. Otherwise, RMAN restores backups of the source database datafiles to the standby database. RMAN restores the most recent files, unless SET UNTIL is specified. If DORECOVER is specified, then RMAN also recovers the database. The standby database is left mounted after duplication is complete. You cannot use SET NEWNAME or CONFIGURE AUXNAME to transform the filenames for the online redo logs on the standby database. You cannot CONNECT RMAN to the standby database and then use DUPLICATE ... FOR STANDBY to create an additional standby database. To create additional standby databases, connect RMAN to the original primary database and run DUPLICATE ... FOR STANDBY. **Note:** Although you can use the DUPLICATE command to create a standby database, you cannot use this command to activate a standby database. When you connect RMAN to the standby database and the recovery catalog in which the primary database is already registered, RMAN recognizes the standby database and implicitly registers it. Do not attempt to use the REGISTER command for the standby database.
TO *database_name*	Specifies the name of the duplicate database. This duplicate database will not be a standby database. If you do not specify the SPFILE clause, then the specified database name should match the name in the initialization parameter file of the duplicate database instance, which is the instance to which RMAN is connected as AUXILIARY. Otherwise, the database signals an error. You cannot use the same database name for the source database and duplicate database when the duplicate database resides in the same Oracle home as the source database. However, if the duplicate database resides in a different Oracle home from the source database, then its database name just has to differ from other database names in its Oracle home. To simplify administration of duplicate database, Oracle recommends that you use different names for the source and duplicate databases.
dupOptionList	Specifies options for creating a duplicate or standby database. See *dupOptionList* on page 2-132.

dupOptionList

This subclause includes options that control aspects of duplication such as naming the files and determining an end point for the duplication. Refer to the *dupOptionList::=* diagram on page 2-130 for the syntax.

Specify new filenames or convert source database filenames for the datafiles and online redo logs when the filenames of the duplicate database must be different from the filenames of the source database (as when the destination host and source host are

the same). If you do not specify filenames for the online redo logs and datafiles of the duplicate database, then RMAN uses the datafile names from the source database.

Syntax Element	Description
BACKUP LOCATION *backup_location*	Specifies the backup location on disk where the backups and copies of the database to be duplicated have been placed. This option is valid for duplication without a target or a recovery catalog connection.
DEVICE TYPE *deviceSpecifier*	Allocates automatic channels for the specified device only (for example, DISK or sbt). This option is valid only if you have configured automatic channels and have *not* manually allocated channels. For example, if you CONFIGURE automatic disk and tape channels, and if you run DUPLICATE...DEVICE TYPE DISK, then RMAN allocates only disk channels. **See Also:** *deviceSpecifier* on page 4-16
DORECOVER	Specifies that RMAN should recover the standby database after creating it. If you specify an *untilClause*, then RMAN recovers to the specified SCN or time and leaves the database mounted. RMAN leaves the standby database mounted after media recovery is complete, but does not place the standby database in manual or managed recovery mode. After RMAN creates the standby database, you must resolve any gap sequence before placing it in manual or managed recovery mode, or opening it in read-only mode. The checkpoint SCN of the control file must be included in an archived redo log that is either available at the standby site or included in an RMAN backup. For example, assume that you create the standby control file and then immediately afterward archive the current log, which has a sequence of 100. In this case, you must recover the standby database up to at least log sequence 100, or the database signals an ORA-1152 error message because the standby control file backup was taken after the point in time.
fileNameConversionSpec	Specifies one or more patterns to map source database filenames to duplicate database filenames (see Example 2–75 on page 2-139). DB_FILE_NAME_CONVERT set on the DUPLICATE command overrides the initialization parameter DB_FILE_NAME_CONVERT (if set). For example, if the initialization parameter file setting is DB_FILE_NAME_CONVERT=('disk1','disk2'), but you execute DUPLICATE ... DB_FILE_NAME_CONVERT ('disk1','disk3'), then RMAN does not convert the disk1 substring to disk2. Instead, RMAN converts the disk1 substring to disk3. If a file in the specification list is not affected by the conversion parameter in DUPLICATE, then you must rename it by other means, such as SET NEWNAME. **Note:** If you specify the SPFILE clause, then DUPLICATE ... DB_FILE_NAME_CONVERT overrides any conversion parameter specified in the SPFILE syntax. For example, if you specify DB_FILE_NAME_CONVERT twice in the DUPLICATE command, both in the SPFILE clause and outside of the SPFILE clause, then the setting outside of the SPFILE clause takes precedence. **See Also:** *fileNameConversionSpec* on page 4-17
FROM ACTIVE DATABASE	Specifies that the files for the duplicate database should be provided directly from the source database and not from a backup of the source database (see Example 2–72 on page 2-138). **See Also:** "Prerequisites Specific to Active Database Duplication" on page 2-126 for command prerequisites
LOGFILE	Specifies options for creating online redo logs when creating a duplicate database that is not a standby database (see Example 2–75).

Syntax Element	Description
INSTANCE '*inst_name*'	Creates online redo logs for the specified instance in a Real Applications Cluster (Oracle RAC) database. The instance name is a string of up to 80 characters. RMAN automatically uses the thread mapped to the specified instance. If no INSTANCE name is specified, then the log files are for the default instance. This clause is relevant when you use DUPLICATE TARGET DATABASE to duplicate an Oracle RAC database to a single-instance database. Otherwise, you do not need to use INSTANCE. If you use the LOGFILE clause, then use INSTANCE to specify the name of the RAC instance for each thread that was open during the database backup (for backup-based duplication) or during the UNTIL TIME (for active database duplication).
logSpec	Specifies the filenames and groups for the online redo log files. **See Also:** *logSpec* on page 2-137 for the valid options
NOFILENAMECHECK	Prevents RMAN from checking whether the datafiles and online redo logs files of the source database are in use when the source database files share the same names as the duplicate database files. You are 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 source database. For example, assume that you have a small database located in the /dbs directory of srchost: /oracle/dbs/system_prod1.dbf /oracle/dbs/users_prod1.dbf /oracle/dbs/rbs_prod1.dbf Assume that you want to duplicate this database to desthost, which has the same file system /oracle/dbs/*, and you want to use the same filenames in the duplicate database as in the source database. In this case, specify the NOFILENAMECHECK option to avoid an error message. Because RMAN is not aware of the different hosts, RMAN cannot determine automatically that it should not check the filenames. If duplicating a database on the same host as the source database, then make sure that NOFILENAMECHECK is not set. Otherwise, RMAN can potentially overwrite and corrupt the target database datafiles, tempfiles, or online logs. It may also 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
NOREDO	Specifies that no archived redo log files should not be applied when recovering a consistent backup in any of the following scenarios: ■ You do not want to apply archived redo log files to the consistent backup even though the archived redo log files are available. ■ The source database was running in NOARCHIVELOG mode at backup time and DUPLICATE is not connected to the target database. ■ The source database is currently running in ARCHIVELOG mode but the backup was taken when the database was in NOARCHIVELOG mode.

Syntax Element	Description
NORESUME	Disables the ability for RMAN to automatically recover from a failed duplication operation. using NORESUME in the first invocation of duplicate permanently prevents any subsequent duplicate command for the new database from using this automatic optimization. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn more about the automated recovery from a failed DUPLICATE operation.
OPEN RESTRICTED	Enables a restricted session in the duplicate database by issuing the following SQL statement: ALTER SYSTEM ENABLE RESTRICTED SESSION. RMAN issues this statement immediately before the duplicate database is opened.
PASSWORD FILE	Specifies that RMAN should use the password file on the source database to overwrite the password file currently used by the auxiliary instance (see Example 2–72 on page 2-138). This option is only valid when FROM ACTIVE DATABASE is specified; otherwise, RMAN signals an error. If FOR STANDBY is specified, then RMAN copies the password file by default; if not specified, then RMAN does not copy the password file by default. You can use PASSWORD FILE to request that RMAN overwrite the existing password file with the password file from the source database. If you want the duplicate database to contain all the passwords available on your production database, then use the PASSWORD FILE option.
PFILE *filename*	Specifies a text-based initialization parameter file used by the auxiliary instance (see Example 2–75 on page 2-139). RMAN automatically shuts down and restarts the auxiliary instance during duplication. If the auxiliary does not use a server parameter file in the default location, then you must specify the text-based initialization parameter file that RMAN should use when starting the auxiliary instance. The initialization parameter file must reside on the same host as the RMAN client used to perform the duplication. If the auxiliary instance uses a server parameter file in the default location, then you do not need to specify PFILE.
SKIP READONLY	Excludes datafiles in current read-only tablespaces from the duplicate database. By default RMAN duplicates current read-only tablespaces. If a tablespace is currently read/write, but you use *untilClause* to duplicate the database to an SCN at which the tablespace was read-only, then RMAN does not include the tablespace in the duplicate database. Tablespaces that were read-only in the past are considered offline tablespaces and so are not included in the duplication. **Note:** A record for the skipped read-only tablespace still appears in DBA_TABLESPACES. By using this feature, you can activate the read-only tablespace later. For example, you can store the read-only tablespace data on a writable DVD, then mount the DVD later and view the data.
SKIP TABLESPACE *tbs_name*	Excludes the specified tablespace from the duplicate database (see Example 2–75 on page 2-139). **Note:** You must not exclude SYS-owned objects or tablespaces with rollback segments, nor tablespaces containing materialized views. The set of tablespaces to be duplicated must be self-contained. If you need to duplicate a database when some backups of the source database do not exist, then SKIP TABLESPACE is required. If you do not specify SKIP TABLESPACE, then RMAN attempts to duplicate the following: ■ All datafiles in online tablespaces, whether or not the datafiles are online. ■ All tablespaces taken offline with an option *other than* NORMAL. For example, RMAN attempts to duplicate tablespaces taken offline with the IMMEDIATE option. You cannot duplicate OFFLINE NORMAL tablespaces, although you can add these tablespaces manually after duplication.
SPFILE	Copies the server parameter file from the source database to the duplicate database.

Syntax Element	Description
setParameter	Sets the specified initialization parameters to the specified values. Refer to *setParameter* on page 2-136.
PARAMETER_VALUE_CONVERT *string_pattern* [*setParameter*]	Replaces the first string with the second string in all matching initialization parameter values. Refer to the description of PARAMETER_VALUE_CONVERT in *dupOptionList* on page 2-132.
TABLESPACE *tablespace_name*	Specifies which tablespaces should be included in the specified database. Unlike SKIP TABLESPACE, which specifies which tablespaces should be *excluded* from the duplicate database, this option specifies which tablespaces should be included and then skips the remaining tablespaces. **Note:** RMAN automatically includes the SYSTEM, SYSAUX, and undo tablespaces in the duplicate database. These tablespaces cannot be skipped and the set of tablespaces that you want to duplicate must be self-contained.
UNDO TABLESPACE *tablespace_name*	Specifies the names of the tablespaces with undo segments. This option is only required when a subset of tablespaces are being duplicated with the SKIP TABLESPACE and TABLESPACE clauses. You must provide the list of tablespaces with undo segments in the following cases: ■ No connection to the target database or the recovery catalog ■ No connection to a recovery catalog, a connection to the target but the target database is not open.
TO RESTORE POINT *restore_point_name*	Specifies a restore point for backup-based duplication, with the SCN at which the restore point was created as the upper, inclusive limit. Because the limit is inclusive, RMAN selects only files that can be used to duplicate a database up to *and including* the corresponding SCN. **Note:** The same restrictions that apply to *untilClause* also apply to TO RESTORE POINT.
untilClause	Sets the end time, SCN, or log sequence number for point-in-time recovery in backup-based duplication (see Example 2–75 on page 2-139). The UNTIL clause is not supported in active database duplication. You can achieve the same result by running SET UNTIL before the DUPLICATE command. If you specify the UNTIL clause for duplication, then the following restrictions apply: ■ RMAN determines whether to use NOREDO based on the current state of the database. If the database was in an archiving mode at the specified UNTIL time or SCN that is different from the current archiving mode, then RMAN does not use NOREDO. ■ The end point for a DUPLICATE command cannot be before the SCN of the most recent ALTER DATABASE OPEN RESETLOGS. Duplication with a connection to a target does not support previous database incarnations. However, you can specify previous incarnations if you are duplicating without a connection to a target database. ■ You cannot recover the duplicate database to the current point in time, that is, the most recent SCN. RMAN recovers the duplicate database up to or before the most recent available archived log, but cannot recover into the online redo logs.

setParameter

This subclause specifies server parameter file values.

Syntax Element	Description
SET *identifier string*	Sets the specified initialization parameters to the specified values (see Example 2–73 on page 2-138). You can use SET to adjust for differences in memory, turn off replication options, and set other options for the duplicate database. This SET functionality is equivalent to pausing the duplication after restoring the server parameter file and issuing ALTER SYSTEM SET statements to change the initialization parameter file. RMAN processes SET after PARAMETER_VALUE_CONVERT. If PARAMETER_VALUE_CONVERT sets the filename specified by a parameter, and if SET sets the filename specified by the same parameter, then the SET value overrides the PARAMETER_VALUE_CONVERT setting. **Note:** If DB_FILE_NAME_CONVERT is specified on the DUPLICATE command, then its filename settings override competing settings specified by SPFILE SET.
COMMENT '*string*'	Specifies an optional comment for the parameter setting.
RESET *identifier string*	Deletes specified initialization parameters from the parameter file. You can use RESET to remove unneeded initialization parameters. This RESET functionality is equivalent to pausing the duplication after restoring the server parameter file and issuing ALTER SYSTEM RESET statements to change the initialization parameter file.

logSpec

This subclause specifies the online redo logs when creating a duplicate database that is not a standby database. Refer to the *logSpec::=* diagram on page 2-131 for the syntax diagram.

If you do not specify LOGFILE, then RMAN will first look to see if any of the following initialization parameters are set: LOG_FILE_NAME_CONVERT, DB_CREATE_FILE_DEST, DB_RECOVERY_FILE_DEST or DB_CREATE_ONLINE_LOG_DEST_n. If these parameters are set, RMAN directs duplicate database online redo log files to Oracle managed storage based on these parameter settings. If none of these initialization parameters are set, then RMAN uses the original redo log filenames of the source database for redo log files of the duplicate database. You must specify the NOFILENAMECHECK option in this case.

Syntax Element	Description
'*filename*' SIZE *sizeSpec*	Specifies the filename of the online redo log member and the size of the file in kilobytes (K) or megabytes (M). The default is in bytes.
REUSE	Allows the database to reuse an existing file. If the file already exists, then the database verifies that its size matches the value of the SIZE parameter. If the file does not exist, then it is created.
GROUP *integer* ('*filename*', ...) SIZE *sizeSpec*	Specifies the group containing the online redo log members, the filename of the online redo log member, and the size of the file in kilobytes (K) or megabytes (M). The default is in bytes.
REUSE	Allows the database to reuse an existing log.

Examples

Example 2–72 Duplicating from an Active Database to a Host with the Same Directory Structure

Assume that you want to create a test database from database `prod1` on a new host. The new host has the same directory structure as the source host, so the files in the duplicate database can use the same names as the files in the source database. You want to create the database without using RMAN backups and allow `prod1` to remain open during the duplication.

If `prod1` uses a server parameter file, then you can create an initialization parameter file on the destination host that contains only the `DB_NAME` parameter set to an arbitrary value. Before starting the auxiliary instance you should create a password file that has the same `SYSDBA` password as the source database. Afterward, start the auxiliary instance.

By default, RMAN does not duplicate the password file when creating a duplicate database that is not a standby database. The `PASSWORD FILE` option specifies that RMAN should copy the password file to the destination host. If you want the duplicate database to contain all the passwords available on your source database, then use the `PASSWORD FILE` option.

You do not need to configure auxiliary channels because RMAN uses the normal channels configured on the source database to copy the database files. Start the RMAN client, connect to the source and auxiliary database instances, and duplicate the database as follows:

```
% rman
RMAN> CONNECT TARGET SYS@prod1

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

RMAN> CONNECT AUXILIARY SYS@dup1

auxiliary database Password: password
connected to auxiliary database: DUP1 (not mounted)

RMAN> DUPLICATE TARGET DATABASE TO dup1
2> FROM ACTIVE DATABASE
3> NOFILENAMECHECK
4> PASSWORD FILE
5> SPFILE;
```

Example 2–73 Copying the Server Parameter File in Active Database Duplication

Assume that you want to create a standby database from database `prod1` on a new host. The destination host has a different directory structure from the source host, so the standby database files will be stored in `/disk2` rather than `/disk1`. You want to create the standby database without using RMAN backups and let `prod1` remain open during the duplication.

Your first step is to create a minimal initialization parameter file for the standby database and then start the standby instance. This parameter file is minimal because when you use the `SPFILE` option, RMAN copies the server parameter file to the new host and sets various parameters to the new values provided.

Start the RMAN client, `CONNECT` to the source database as `TARGET`, and connect to the auxiliary instance. You do not need to configure auxiliary channels because RMAN

uses the normal channels on the source host to copy the database files. You can then enter the following command:

```
DUPLICATE TARGET DATABASE TO dup1
  FOR STANDBY
  FROM ACTIVE DATABASE
  PASSWORD FILE
  SPFILE
    PARAMETER_VALUE_CONVERT '/disk1', '/disk2'
    SET DB_FILE_NAME_CONVERT '/disk1','/disk2'
    SET LOG_FILE_NAME_CONVERT '/disk1','/disk2'
    SET SGA_MAX_SIZE 200M
    SET SGA_TARGET 125M;
```

Example 2–74 Duplicating a Database Without a Target Connection to a Host with the Same Directory Structure

Assume that you want to duplicate source database `prod` using backups and do not want to connect RMAN as `TARGET` to this database because it is shut down for maintenance. A description of the environment follows:

- The source and destination hosts have identical directory structures. The datafile and online redo log names in the duplicate database should be identical to the names in the source database.
- The same number of online redo log files will be used in the duplicate database.
- A recovery catalog is available. The source database name `prod` is unique in the recovery catalog.
- Auxiliary channels have been configured with the `CONFIGURE CHANNEL` command.

The following commands create a duplicate database named `dupdb`:

```
% rman

RMAN> CONNECT CATALOG rman@catdb;

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CONNECT AUXILIARY SYS@dupdb;

auxiliary database Password: password
connected to auxiliary database: DUPDB (not mounted)

RMAN> DUPLICATE DATABASE 'PROD' TO 'DUPDB' NOFILENAMECHECK;
```

Assume a different scenario in which the database name `prod` is not unique in the recovery catalog. The following `DUPLICATE` command uses the `DBID` parameter to uniquely identify the source database:

```
RMAN> DUPLICATE DATABASE 'PROD' DBID 39525561 TO 'DUPDB' NOFILENAMECHECK;
```

Example 2–75 Setting New Filenames in the DUPLICATE Command

Assume that you want to use tape backups to duplicate the source database `prod` on `srchost` to `newdb` on `desthost`.

In this scenario, the source database does not use a server parameter file. You create a text-based initialization parameter file on `desthost` and use it to start the database

instance. Thus, backup-based duplication must use a target connection (see Table 2–6 on page 2-126).

When executing DUPLICATE on desthost, you must use the PFILE parameter to specify the location of the initialization parameter file. You must use the RMAN client on the same host as the initialization parameter file for the duplicate database.

You do not want the tablespaces example and history to be included in the duplicate database, so you specify DUPLICATE ... SKIP TABLESPACE for these tablespaces. Also, you want the duplicate database to be in the state that the production database was in 24 hours ago, so you use DUPLICATE ... UNTIL TIME.

This example assumes that the datafiles of the source database are on srchost in directory /h1/oracle/dbs/trgt. You intend to duplicate the datafiles to the directory /h2/oracle/oradata/newdb, so you specify DUPLICATE ... DB_FILE_NAME_CONVERT to generate the names for the duplicate datafiles. You use DUPLICATE ... LOGFILE to specify names for the online redo log files in the duplicate database.

Start the RMAN client on desthost, CONNECT to the source database as TARGET, and connect to the auxiliary instance. You can then enter the following RUN command:

```
RUN
{
  ALLOCATE AUXILIARY CHANNEL newdb DEVICE TYPE sbt;
  DUPLICATE TARGET DATABASE TO newdb
    PFILE ?/dbs/initNEWDB.ora
    UNTIL TIME 'SYSDATE-1'  # specifies incomplete recovery
    SKIP TABLESPACE example, history   # skip desired tablespaces
    DB_FILE_NAME_CONVERT ('/h1/oracle/dbs/trgt/','/h2/oracle/oradata/newdb/')
    LOGFILE
      GROUP 1 ('/h2/oradata/newdb/redo01_1.f',
               '/h2/oradata/newdb/redo01_2.f') SIZE 4M,
      GROUP 2 ('/h2/oradata/newdb/redo02_1.f',
               '/h2/oradata/newdb/redo02_2.f') SIZE 4M,
      GROUP 3 ('/h2/oradata/newdb/redo03_1.f',
               '/h2/oradata/newdb/redo03_2.f') SIZE 4M REUSE;
}
```

Example 2–76 Using SET NEWNAME FOR DATABASE to Name Duplicate Files

In this scenario, you intend to use backup-based duplication without a target connection.

The source database prod contains eight datafiles spread out over multiple directories. The datafiles are not Oracle Managed Files. You want to duplicate the source database to dupdb on destination host desthost.

In this scenario, srchost and desthost have different directory structures. You want to store the datafiles in desthost in the /oradata1 subdirectory, so you use SET NEWNAME FOR DATABASE to specify the file names stripped of directory paths. For example, if a source datafile has the name '/oradata/prod/financial.dbf', then %b results in 'financial.dbf'.

The source database does not use a server parameter file, so you cannot use the SPFILE technique to specify names for the duplicate datafiles. You decide to use the SET NEWNAME DATABASE command because you want all duplicate datafiles in the same directory on the destination host.

You want to create two online redo log groups, each with two members of size 200 KB, in the directory /duplogs on the destination host. Assume that srchost and desthost cannot mount each other's file systems by any means such as NFS.

You have disk copies or backup sets stored on disk for all the datafiles and archived redo log files in the source database, and you have manually copied them to desthost by means of an operating system utility. These backups and copies exist in the same location on desthost as they do in srchost.

You use an operating system utility to copy the initialization parameter file from srchost to an appropriate location in desthost. You have reset all initialization parameters that end in _DEST and specify a path name. You do not set DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT because you are specifying names for datafiles and online logs in the RUN command itself. The auxiliary instance uses a server-side initialization parameter file in the default location so the PFILE parameter is not necessary on the DUPLICATE command.

The following sample script creates the duplicate database. A RUN command is necessary because you can only execute SET NEWNAME within RUN.

```
RUN
{
  SET NEWNAME FOR DATABASE TO '/oradata1/%b';
  DUPLICATE TARGET DATABASE TO dupdb
    LOGFILE
      GROUP 1 ('/duplogs/redo01a.log',
               '/duplogs/redo01b.log') SIZE 4M REUSE,
      GROUP 2 ('/duplogs/redo02a.log',
               '/duplogs/redo02b.log') SIZE 4M REUSE;
}
```

Example 2–77 Using SET NEWNAME FOR DATAFILE and SET NEWNAME FOR TABLESPACE to Name Duplicate Files

In this scenario, you intend to duplicate database by using backup-based duplication.

Assume that the source database prod is on srchost and contains nine datafiles, which are spread out over multiple directories. You want to duplicate the source database to database dupdb on remote host desthost. The dupdb database should exclude tablespace tools, but keep all of the other tablespaces.

The source database does not use a server parameter file, so you cannot use the SPFILE technique to specify names for the duplicate datafiles. You decide to use SET NEWNAME commands to specify the filenames because the duplicate datafiles will be spread out across several directories.

In this scenario, srchost and desthost have different directory structures. You want to store the datafiles in desthost in the /oradata1 through /oradata7 subdirectories. You want to place each datafile in a different directory, with the exception of the users tablespace, which contains two datafiles that you intend to duplicate to /oradata7.

You want to create two online redo log groups, each with two members of size 200 KB, in the directory /duplogs on the destination host. Assume that srchost and desthost cannot mount each other's file systems by any means such as NFS.

You have disk copies or backup sets stored on disk for all the datafiles and archived redo log files in the source database, and you have manually copied them to desthost by means of an operating system utility. These backups and copies exist in the same location on desthost as they do in srchost.

You use an operating system utility to copy the initialization parameter file from srchost to an appropriate location in desthost. You have reset all initialization parameters that end in _DEST and specify a path name. You do not set DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT because you are specifying names for datafiles and online logs in the RUN command itself. The auxiliary instance uses a server-side initialization parameter file in the default location so the PFILE parameter is not necessary on the DUPLICATE command.

The following sample scripts creates the duplicate database. A RUN command is necessary because you can only execute SET NEWNAME within RUN.

```
RUN
{
  SET NEWNAME FOR DATAFILE 1 TO '/oradata1/system01.dbf';
  SET NEWNAME FOR DATAFILE 2 TO '/oradata2/undotbs01.dbf';
  SET NEWNAME FOR DATAFILE 3 TO '/oradata3/cwmlite01.dbf';
  SET NEWNAME FOR DATAFILE 4 TO '/oradata4/drsys01';
  SET NEWNAME FOR DATAFILE 5 TO '/oradata5/example01.dbf';
  # because the users command contains 2 datafiles, the following command
  # generates unique names for both datafiles, placing them in /oradata7
  SET NEWNAME FOR TABLESPACE users TO '/oradata7/users%b.dbf';
  DUPLICATE TARGET DATABASE TO dupdb
    SKIP TABLESPACE tools
    LOGFILE
      GROUP 1 ('/duplogs/redo01a.log',
               '/duplogs/redo01b.log') SIZE 4M REUSE,
      GROUP 2 ('/duplogs/redo02a.log',
               '/duplogs/redo02b.log') SIZE 4M REUSE;
}
```

Example 2–78 Using SET NEWNAME FOR DATAFILE to Name Oracle-Managed Files

There are two ways to store specific datafiles or tempfiles in an Oracle-managed files destination that is independent of the locations of the rest of the database files.

1. Set the parameter DB_CREATE_FILE_DEST in the initialization parameter file of the auxiliary instance to the desired location
2. Set the initialization parameters DB_CREATE_FILE_DEST and DB_FILE_NAME_CONVERT. At this point, you can use the SET NEWNAME command for those datafiles that you do not want to be converted by DB_FILE_NAME_CONVERT.

The specified datafiles or tempfiles are created with Oracle-managed file names in the location specified by DB_CREATE_FILE_DEST.

As shown in the following sample script, you can also use SET NEWNAME to direct individual datafiles or tempfiles to a specific ASM disk group.

```
RUN
{
   SET NEWNAME FOR DATAFILE 1 TO "+DGROUP1";
   SET NEWNAME FOR DATAFILE 2 TO "+DGROUP2";
   .
   .
   .
   DUPLICATE TARGET DATABASE
     TO dupdb
     FROM ACTIVE DATABASE
     SPFILE SET DB_CREATE_FILE_DEST +DGROUP3;
}
```

Example 2–79 Using CONFIGURE AUXNAME to Name Duplicate Files

This section assumes the same circumstances described in Example 2–77, "Using SET NEWNAME FOR DATAFILE and SET NEWNAME FOR TABLESPACE to Name Duplicate Files". This example is a variation that uses `CONFIGURE AUXNAME` instead of `SET NEWNAME` to specify the new datafile names. These new filenames are recorded in the control file and used every time you perform the duplication in the future.

This example also uses automatic channels and a client-side initialization parameter file for the database duplication, and uses the `LOGFILE` clause to specify names and sizes for the online redo logs. In this case the `RUN` command is not necessary because you are not using `SET NEWNAME`.

```
CONFIGURE AUXNAME FOR DATAFILE 1 TO '/oradata1/system01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 2 TO '/oradata2/undotbs01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 3 TO '/oradata3/cwmlite01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 4 TO '/oradata4/drsys01';
CONFIGURE AUXNAME FOR DATAFILE 5 TO '/oradata5/example01.dbf';
CONFIGURE AUXNAME FOR DATAFILE 6 TO '/oradata6/indx01.dbf';
DUPLICATE TARGET DATABASE
  TO dupdb
  SKIP TABLESPACE tools
  LOGFILE
    GROUP 1 ('/duplogs/redo01a.log',
             '/duplogs/redo01b.log') SIZE 4M REUSE,
    GROUP 2 ('/duplogs/redo02a.log',
             '/duplogs/redo02b.log') SIZE 4M REUSE;
```

RMAN uses all incremental backups, archived redo log backups, and archived redo log files to perform incomplete recovery and then opens the database with the `RESETLOGS` option to create the online redo logs.

After the duplication is complete, you can clear the configured auxiliary names for the datafiles in the duplicate database, so that they are not overwritten by future operations. For example, enter the following commands:

```
CONFIGURE AUXNAME FOR DATAFILE 1 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 3 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 4 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 5 CLEAR;
CONFIGURE AUXNAME FOR DATAFILE 6 CLEAR;
```

Alternatively, you may want to periodically synchronize a duplicate database with the original database that was duplicated. In this case, you can run the `DUPLICATE` command again, essentially re-creating the duplicate database. This technique requires making complete copies of the datafiles of the duplicate database. Run the following script whenever you want to synchronize the duplicate with the source database. For example, you might run the script daily or weekly.

```
DUPLICATE TARGET DATABASE TO dupdb
SKIP TABLESPACE tools
  LOGFILE
    GROUP 1 ('/duplogs/redo01a.log',
             '/duplogs/redo01b.log') SIZE 4M REUSE,
    GROUP 2 ('/duplogs/redo02a.log',
             '/duplogs/redo02b.log') SIZE 4M REUSE;
```

Example 2–80 Creating a Standby Database with the Same Directory Structure

Assume that you want to use RMAN backups to create a standby database on a remote host with the same directory structure as the source host. The source database is called prod1 and will be the primary database in the Data Guard environment.

First, start the RMAN client, CONNECT to the source database prod1 as TARGET, and connect to the auxiliary instance. You can then CONFIGURE the default device type to sbt for a standby database with the DB_UNIQUE_NAME of standby1:

```
CONFIGURE DEFAULT DEVICE TYPE sbt FOR DB_UNIQUE_NAME standby1;
CONFIGURE DEVICE TYPE sbt PARALLELISM 2 FOR DB_UNIQUE_NAME standby1;
```

Assume all backups needed to create the standby database are on tape. In the standby database initialization parameter file, you set DB_UNIQUE_NAME to standby1.

The default initialization parameter file location is in use on the standby database. After starting the standby instance NOMOUNT, you start the RMAN client, CONNECT to the source database as TARGET, and connect to the auxiliary instance and recovery catalog. You run the following DUPLICATE command, specifying the NOFILENAMECHECK option because the standby and primary datafiles and online redo log files have the same names:

```
DUPLICATE TARGET DATABASE FOR STANDBY
  NOFILENAMECHECK;
```

Example 2–81 Creating a Standby Database in OMF and ASM

Assume that you want to use RMAN backups to create a standby database on a host that uses OMF and ASM. The source database is called prod1 and will be the primary database in the Data Guard environment.

First, start the RMAN client, CONNECT to database prod1 as TARGET, and connect to the recovery catalog. Run the following commands to CONFIGURE the default device type to sbt for a standby database that will have the DB_UNIQUE_NAME of standby1 and the net service name sby1.

```
CONFIGURE CONNECT IDENTIFIER "sby1" FOR DB_UNIQUE_NAME standby1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt FOR DB_UNIQUE_NAME standby1;
CONFIGURE DEVICE TYPE sbt PARALLELISM 2 FOR DB_UNIQUE_NAME standby1;
```

Assume all backups needed to create the standby database are stored on tape. You set the following parameters in the initialization parameter file for database standby1:

- Set DB_UNIQUE_NAME to the value standby1.
- Set DB_CREATE_FILE_DEST and DB_RECOVERY_FILE_DEST to the desired ASM disk groups on the standby host. For example, set DB_CREATE_FILE_DEST to +DATAFILE and DB_RECOVERY_FILE_DEST to +FLASH_REC_AREA.

Ensure that the standby instance is in NOMOUNT mode. Start the RMAN client, CONNECT to database prod1 as TARGET, connect to the standby1 instance as AUXILIARY, and connect to the recovery catalog. Enter the following command to create the standby database:

```
DUPLICATE TARGET DATABASE FOR STANDBY;
```

RMAN automatically generates new OMF/ASM datafile names for the restored datafiles.

Example 2–82 Duplicating a Database Without Connection to Target Database and Recovery Catalog

In this example, all the necessary backups of datafiles, controlfile and archived logs of database `prod` can be accessed from the location: `/net/prod/backups`. This location is where only backups of database `prod` reside. The only connection is to the new instance as `AUXILIARY`.

Enter the following command to create a test database when there is no connection to the recovery catalog or target database:

```
DUPLICATE DATABASE TO 'TEST' BACKUP LOCATION '/net/prod/backups' NOFILENAMECHECK;
```

Example 2–83 Selecting a Specific Database When Duplicating Without Connection to Target Database and Recovery Catalog

In this example, `/backups` contains backups from several databases, including more than two databases with the name `PROD`. In this case, you must specify the `DBNAME` and the `DBID` of the database that you want to duplicate. The only connection is to the auxiliary instance.

```
DUPLICATE DATABASE 'PROD' dbid 8675309 to 'TEST'
  UNTIL TIME "to_date('11/01/2007', 'MM/DD/YYYY')"
  BACKUP LOCATION '/backups' NOFILENAMECHECK
  PFILE='?/dbs/inittest.ora' db_file_name_convert='prod','test';
```

EXECUTE SCRIPT

Purpose

Use the `EXECUTE SCRIPT` command to run a local or global RMAN script stored in the recovery catalog.

See Also:

- *Oracle Database Backup and Recovery User's Guide* to learn how to use stored scripts
- `CREATE SCRIPT` on page 2-105 and `REPLACE SCRIPT` on page 3-25

Prerequisites

Use `EXECUTE SCRIPT` only within the braces of a `RUN` command. RMAN must be connected to the recovery catalog with the `CATALOG` command-line option or the `CONNECT CATALOG` command. The recovery catalog database must be open.

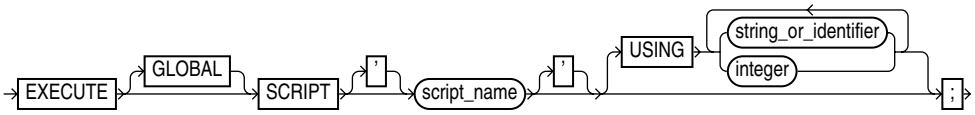
Usage Notes

When you run an `EXECUTE SCRIPT` command within a `RUN` block, RMAN places the contents of the script in the context of that `RUN` block. For this reason, you should not allocate a channel within the `RUN` block if you also allocate it in the script.

If `GLOBAL` is specified, then a global script with this name must already exist in the recovery catalog; otherwise, RMAN returns error `RMAN-06004`. If `GLOBAL` is not specified, then RMAN searches for a local stored script defined for the current target database. If no local script with this name is found, then RMAN searches for a global script by the same name and executes it if one is found.

Syntax

***executeScript*::=**



Semantics

Syntax Element	Description
`GLOBAL`	Specifies the execution of a global stored script instead of a local one. **See Also:** "Usage Notes" on page 2-105 for an explanation of the difference between global and local scripts
`SCRIPT` *`script_name`*	Specifies the name of the stored script to execute. Quotes must be used around the script name when the name contains either spaces or reserved words.
`USING` [*`string_or_identifier`* | *`integer`*]	Specifies one or more values for use in substitution variables in a stored script (see Example 2–85). **See Also:** `CREATE SCRIPT` to learn how to create a stored script with substitution variables, and RMAN on page 3-59 and `@ (at sign)` on page 2-2 to learn how to use substitution variables with RMAN

Example

Example 2–84 Executing a Stored Script

This example uses LIST to list the script stored in the recovery catalog and PRINT SCRIPT to show the contents of global_backup_db, which was created in Example 2–61, "Creating a Global Stored Script". Finally, the example runs global_backup_db to back up the database.

```
RMAN> LIST SCRIPT NAMES;

List of Stored Scripts in Recovery Catalog

    Global Scripts

       Script Name
       Description
       ---------------------------------------------------------------------
       global_backup_db
       back up any database from the recovery catalog, with logs

RMAN> PRINT SCRIPT global_backup_db;

printing stored global script: global_backup_db
{
    BACKUP DATABASE PLUS ARCHIVELOG;
}

RMAN> RUN { EXECUTE GLOBAL SCRIPT global_backup_db; }

executing global script: global_backup_db


Starting backup at 07-JUN-07
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=120 device type=DISK
.
.
.
```

Example 2–85 Creating and Executing a Stored Script That Uses Substitution Variables

After starting RMAN and connecting to a target database and recovery catalog, use REPLACE SCRIPT to create a backup script that includes three substitution variables. RMAN prompts you to enter initial values for the variables (user input is in bold).

```
RMAN> REPLACE SCRIPT
2>  backup_df { BACKUP DATAFILE &1 TAG &2.1 FORMAT '/disk1/&3_%U'; }

Enter value for 1: 1

Enter value for 2: df1_backup

Enter value for 3: df1

starting full resync of recovery catalog
full resync complete
created script backup_df
```

Later, you can execute the `backup_df` script with different values. The following example passes the values `3`, `test_backup`, and `test` to the substitution variables in the stored script:

```
RMAN> RUN { EXECUTE SCRIPT backup_df USING 3 test_backup df3; }
```

After the values are substituted, RMAN executes the `BACKUP` command as follows:

```
BACKUP DATAFILE 3 TAG test_backup1 FORMAT '/disk1/df3_%U';
```

EXIT

Purpose

Use the `EXIT` command to shut down the Recovery Manager utility. This command is functionally equivalent to the QUIT command.

Prerequisites

Execute only at the RMAN prompt.

Syntax

***exit*::=**

Example

Example 2–86 Exiting RMAN

This example terminates RMAN:

```
RMAN> EXIT
```

FLASHBACK DATABASE

Purpose

Use the `FLASHBACK DATABASE` command to rewind the database to a target time, SCN, or log sequence number.

This command works by undoing changes made by Oracle Database to the datafiles that exist when you run the command. Flashback can fix logical failures, but not physical failures. Thus, you cannot use the command to recover from disk failures or the accidental deletion of datafiles.

`FLASHBACK DATABASE` is usually much faster than a `RESTORE` operation followed by point-in-time recovery, because the time needed to perform `FLASHBACK DATABASE` depends on the number of changes made to the database since the desired flashback time. On the other hand, the time needed to do a traditional point-in-time recovery from restored backups depends on the size of the database.

Flashback Database also has a number of uses in a Data Guard environment.

See Also:

- *Oracle Database SQL Language Reference* for a complete list of command prerequisites and usage notes for `FLASHBACK DATABASE`
- *Oracle Data Guard Concepts and Administration* to learn about uses of Flashback Database in a Data Guard environment

Prerequisites

You can run this command from the RMAN prompt or from within a `RUN` command.

RMAN must be connected as `TARGET` to a database, which must be Oracle Database 10*g* or later. The target database must be mounted with a current control file, that is, the control file cannot be a backup or re-created. The database must run in `ARCHIVELOG` mode.

You cannot use `FLASHBACK DATABASE` to return to a point in time before the restore or re-creation of a control file. If the database control file is restored from backup or re-created, then all existing flashback log information is discarded.

The fast recovery area must be configured to enable flashback logging. Flashback logs are stored as Oracle-managed files in the fast recovery area and cannot be created if no fast recovery area is configured. You must have enabled the flashback logging before the target time for flashback by means of the SQL statement `ALTER DATABASE ... FLASHBACK ON`. Query `V$DATABASE.FLASHBACK_ON` to see whether flashback logging has been enabled.

The database must contain no online tablespaces for which flashback functionality was disabled with the SQL statement `ALTER TABLESPACE ... FLASHBACK OFF`.

Usage Notes

A Flashback Database operation applies to the whole database. You cannot flash back individual tablespaces. A Flashback Database operation is similar to a database point-in-time recovery (DBPITR) performed with `RECOVER`, but RMAN uses **flashback logs** to undo changes to a point before the target time or SCN. RMAN automatically restores from backup any archived redo log files that are needed and

recovers the database to make it consistent. Note that RMAN never flashes back data for temporary tablespaces.

The earliest SCN that can be used for a Flashback Database operation depends on the setting of the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter, and on the actual retention of flashback logs permitted by available disk. View the current database SCN in `V$DATABASE.CURRENT_SCN`.

Effect of NOLOGGING Operations on Flashback Database

When using `FLASHBACK DATABASE` with a target time at which a `NOLOGGING` operation was in progress, block corruption is likely in the database objects and datafiles affected by the `NOLOGGING` operation. For example, assume that you do a direct-path `INSERT` operation in `NOLOGGING` mode and that the operation runs from 9:00 to 9:15 on April 3. If you later use Flashback Database to return to 09:07 on this date, then the objects and datafiles updated by the direct-path `INSERT` may be left with block corruption after Flashback Database completes.

If possible, avoid using `FLASHBACK DATABASE` with a target time or SCN that coincides with a `NOLOGGING` operation. Also, perform a full or incremental backup of the affected datafiles immediately after any `NOLOGGING` operation to ensure recoverability to points in time after the operation. If you expect to use `FLASHBACK DATABASE` to return to a point in time during an operation such as a direct-path `INSERT`, then consider performing the operation in `LOGGING` mode.

> **See Also:** The discussion of `logging_clause` in *Oracle Database SQL Language Reference* for more information about operations that support `NOLOGGING` mode

Effect of Datafile Status Changes on Flashback Database

The `FLASHBACK DATABASE` command does not start modifying the database until it has made sure that it has all the files and resources that it needs. A Flashback Database operation should never fail due to missing datafiles, redo log files, or flashback logs.

If a datafile has changed status between the current SCN and the target SCN of the flashback, then the `FLASHBACK DATABASE` command behaves differently depending on the nature of the status change. Refer to Table 2–8 for details.

Table 2–8 How FLASHBACK DATABASE Responds to Datafile Status Changes

If this datafile operation occurred during the flashback window ...	Then the FLASHBACK DATABASE command ...
Added	Removes the datafile record from the control file.
Dropped	Adds the datafile to the control file, but marks it as offline and does not flash it back. You can then restore and recover the datafile to the same time or SCN.
Renamed	Ignores the renaming. The datafile retains its current name.
Resized	May fail. You can take the datafile offline and then rerun the `FLASHBACK DATABASE` command. The datafile will not be flashed back. You can then restore and recover the datafile to the same time or SCN.
Taken offline	Ignores the operation. The datafile retains its current online status.
Brought online	Ignores the operation. The datafile retains its current offline status.

Table 2–8 (Cont.) How FLASHBACK DATABASE Responds to Datafile Status Changes

If this datafile operation occurred during the flashback window ...	Then the FLASHBACK DATABASE command ...
Made read-only or read/write	Changes the status of the datafile in the control file.

Tablespaces with Flashback Logging Disabled

It is possible for the `ALTER TABLESPACE ... FLASHBACK OFF` statement to have been executed for some tablespaces. If `FLASHBACK DATABASE` has insufficient flashback data to rewind a tablespace to the target SCN, then RMAN issues an error and does not modify the database. Whenever `FLASHBACK DATABASE` fails or is interrupted, the database is left mounted.

In this scenario, query `V$TABLESPACE` to determine which tablespaces have flashback logging disabled. You have the following options:

- Take the datafiles in the affected tablespaces offline. Afterwards, run `RESTORE` and then `RECOVER` to bring these datafiles to the same point in time as the rest of the database.
- Drop the affected datafiles with the `ALTER DATABASE DATAFILE ... OFFLINE FOR DROP` statement. You can then open the database with the `RESETLOGS` option. After the database is open, execute `DROP TABLESPACE` statements for the tablespaces that contain the dropped datafiles.

State of the Database After Flashback Database

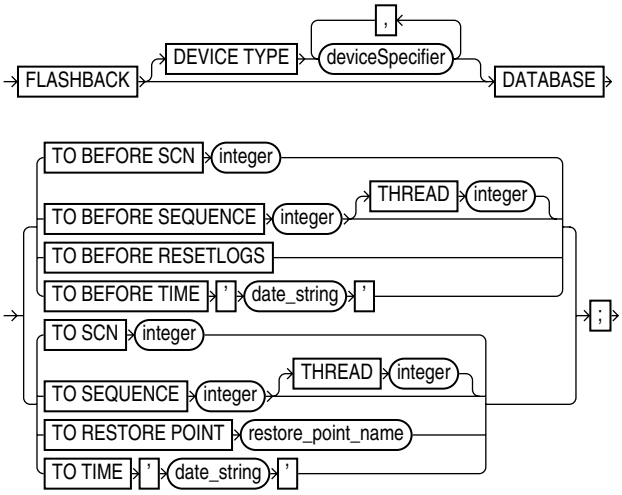
After running `FLASHBACK DATABASE`, the database may not be left at the SCN most immediately before the target time. Events other than transactions can cause the database SCN to be updated. If you use the `FLASHBACK DATABASE TO` form of the command, and if a transaction is associated with the target SCN, then after the flashback the database will include all changes up to and including this transaction. Otherwise, all changes up to but *not* including this transaction will be included in the datafiles, whether you use the `FLASHBACK DATABASE TO` or `FLASHBACK DATABASE TO BEFORE` form of the command. Changes after the specified target SCN are never applied as a result of a `FLASHBACK DATBASE`.

After `FLASHBACK DATABASE` completes, you may want to open the database read-only and run queries to ensure that you achieved the intended result. If you are not satisfied, then you can use `RECOVER DATABASE` to recover the database to its state when you started the flashback. You can then rerun `FLASHBACK DATABASE`.

If you are satisfied with the results of the flashback, then you can `OPEN RESETLOGS` to abandon all changes after the target time. Alternatively, you can use Data Pump to export lost data, use `RECOVER DATABASE` to return the database to its state before the flashback operation, and then use Data Pump to reimport the lost data.

Syntax

***flashback*::=**



(*deviceSpecifier::=* on page 4-16)

Semantics

Syntax Element	Description
`DEVICE TYPE` *`deviceSpecifier`*	Allocates automatic channels for the specified device type only. For example, if you configure automatic disk and tape channels, and issue `FLASHBACK ... DEVICE TYPE DISK`, then RMAN allocates only disk channels. RMAN may need to restore redo logs from backup during the flashback database process. Changes between the last flashback log and the target time must be re-created based on the archived redo log. If no automatic channels are allocated for tape and a needed redo log is on tape, then the `FLASHBACK DATABASE` operation fails. **See Also:** *`deviceSpecifier`* on page 4-16
`TO BEFORE SCN` *`integer`*	Returns the database to its state just before the specified SCN. Any changes at an SCN lower than that specified are applied, but if there is a change associated with the specified SCN it is not applied. By default, the provided SCN resolves to the current or ancestor incarnation. You can override the default by using the `RESET DATABASE INCARNATION` command. Query `OLDEST_FLASHBACK_SCN` in `V$FLASHBACK_DATABASE_LOG` to see the approximate lowest SCN to which you can flash back.
`TO BEFORE SEQUENCE` *`integer`* `[THREAD` *`integer`*`]`	Specifies a redo log sequence number and thread as an upper limit. RMAN applies changes up to (but not including) the last change in the log with the specified sequence and thread number.
`TO BEFORE RESETLOGS`	Returns the database to its state including all changes up to the SCN of the most recent `OPEN RESETLOGS`. **Note:** `FLASHBACK DATABASE` can only return the database to a point before the most recent `OPEN RESETLOGS` operation if your database has been upgraded to Oracle Database 10*g* Release 2 or later.
`TO BEFORE TIME` *`'date_string'`*	Returns the database to its state including all changes up to but not including changes at the specified time. Query `OLDEST_FLASHBACK_TIME` in `V$FLASHBACK_DATABASE_LOG` to see the approximate lowest time to which you can flash back.

Syntax Element	Description
TO SCN *integer*	Returns the database to the point up to (and including) the specified SCN. By default, the provided SCN resolves to the current or ancestor incarnation. You can override the default by using the RMAN RESET DATABASE command to set the recovery target incarnation. Query OLDEST_FLASHBACK_SCN in V$FLASHBACK_DATABASE_LOG to see the approximate lowest SCN to which you can flash back.
TO SEQUENCE *integer* THREAD *integer*	Specifies a redo log sequence number and thread as an upper limit. RMAN applies changes up to (and including) the last change in the log with the specified sequence and thread number.
TO RESTORE POINT *restore_point_name*	Returns the database to the SCN associated with the specified restore point. This can be an ordinary restore point or a guaranteed restore point.
TO TIME '*date_string*'	Returns the database to its state at the specified time. You can use any SQL DATE expressions to convert the time to the current format, for example, FLASHBACK DATABASE TO TIME 'SYSDATE-7'. Query OLDEST_FLASHBACK_TIME in V$FLASHBACK_DATABASE_LOG to see the approximate lowest time to which you can flash back.

Examples

Example 2–87 FLASHBACK DATABASE to a Specific SCN

Assume that you inserted corrupted rows in many tables at 5:00 p.m. on February 14. You connect SQL*Plus to the database and query the earliest SCN in the flashback window:

```
SQL> SELECT OLDEST_FLASHBACK_SCN, OLDEST_FLASHBACK_TIME
  2  FROM   V$FLASHBACK_DATABASE_LOG;

OLDEST_FLASHBACK_SCN OLDEST_FLASHBACK
-------------------- ----------------
              411010 2007/02/14 16:49
```

You then open a new terminal, start the RMAN client, and connect to the target database and recovery catalog. You enter RMAN commands as follows (sample output for the FLASHBACK DATABASE is included):

```
RMAN> SHUTDOWN IMMEDIATE
RMAN> STARTUP MOUNT
RMAN> FLASHBACK DATABASE TO SCN 411010;

Starting flashback at 15-FEB-07
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=104 device type=DISK


starting media recovery
media recovery complete, elapsed time: 00:00:07

Finished flashback at 15-FEB-07

RMAN> ALTER DATABASE OPEN RESETLOGS;
```

Example 2–88 FLASHBACK DATABASE to a Restore Point

Assume that you are preparing to load a massive number of updates to the database. You create a guaranteed restore point before the performing the updates:

```
SQL> CREATE RESTORE POINT before_update GUARANTEE FLASHBACK DATABASE;
```

The bulk update fails, leaving the database with extensive corrupted data. You start an RMAN session, connect to the target database and recovery catalog, and list the guaranteed restore points:

```
RMAN> LIST RESTORE POINT ALL;

SCN              RSP Time  Type       Time      Name
---------------- --------- ---------- --------- ----
412742                     GUARANTEED 15-FEB-07 BEFORE_UPDATE
```

You mount the database, flash back the database to the restore point (sample output included), and then open the database with the `RESETLOGS` option:

```
RMAN> SHUTDOWN IMMEDIATE
RMAN> STARTUP MOUNT
RMAN> FLASHBACK DATABASE TO RESTORE POINT 'BEFORE_UPDATE';

Starting flashback at 15-FEB-07
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=104 device type=DISK


starting media recovery

archived log for thread 1 with sequence 34 is already on disk as file
/disk2/oracle/oradata/prod/arch/archive1_34_614598462.dbf
media recovery complete, elapsed time: 00:00:01
Finished flashback at 15-FEB-07

RMAN> ALTER DATABASE OPEN RESETLOGS;
```

GRANT

Purpose

Use the GRANT command to assign privileges for a virtual private catalog schema to a database user. By default, a virtual catalog user has no access to the base recovery catalog.

Prerequisites

Execute this command at the RMAN prompt.

A base recovery catalog must have been created with CREATE CATALOG before you can use GRANT to assign privileges for a virtual private catalog.

Usage Notes

The best practice is to create a **base recovery catalog** that stores metadata for all databases. You can then create an Oracle Database user that will own the virtual private catalog schema. The virtual private catalog user must be granted the RECOVERY_CATALOG_OWNER role.

Connect RMAN to the base recovery catalog and use the GRANT command to assign recovery catalog privileges to the virtual catalog owner. Afterwards, run CREATE VIRTUAL CATALOG to create a virtual catalog schema for this user. You can use REVOKE to revoke catalog privileges.

Relationship Between Users with CATALOG Privileges on the Same Database

As an illustration of GRANT usage, suppose databases prod1 and prod2 are registered in the base recovery catalog. While logged in as SYS to the base recovery catalog, you create two virtual private catalog users: vpc1 and vpc2. You grant both users CATALOG FOR DATABASE access for database prod1, but not prod2.

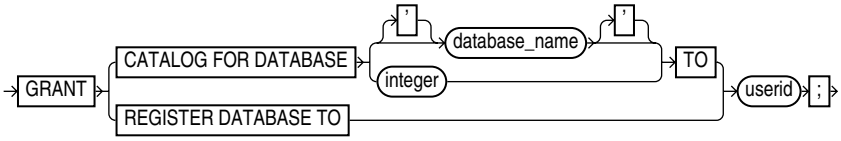
In this scenario, both vpc1 and vpc2 can access the metadata for backups of prod1 made by the base recovery catalog owner. Both users can also access the metadata for backups of prod1 made by each other. Neither vpc1 nor vpc2 can access backup metadata for database prod2.

Relationship Between GRANT REGISTER and GRANT CATALOG

When you grant REGISTER DATABASE to a user, RMAN implicitly grants recovery CATALOG FOR DATABASE privileges for any database registered by this user. If you REVOKE only the REGISTER DATABASE privilege from a user (for example, virtcat), then it does not implicitly revoke the CATALOG FOR DATABASE privilege for a database registered by virtcat (for example, prod). Because the CATALOG FOR DATABASE privilege includes registration privileges for prod, virtcat can continue to unregister and register prod. To prevent virtcat from performing any operations on prod, including reregistering it, REVOKE ALL PRIVILEGES from virtcat.

Syntax

***grant*::=**



Semantics

Syntax Element	Description
`CATALOG FOR DATABASE` [*`database_name`* | *`integer`*] `TO` *`userid`*	Grants recovery catalog access for the specified database to the specified user. **Note:** The catalog operations granted on the specified database include registering and unregistering this database. Specify the database by either database name or DBID. If you specify a name when more than one database with this name is registered in the catalog, then RMAN returns an error. In this case, specify the database by DBID. To grant access to databases that are already registered in the recovery catalog, you must use the `GRANT CATALOG` command. You can also grant access for a target database that is not yet registered in the catalog, thereby enabling a virtual private catalog user to register a database. You must grant access by using the DBID of the database that has not yet been registered.
`REGISTER DATABASE` `TO` *`userid`*	Grants the ability to for the specified user to use `REGISTER DATABASE` to register databases that are currently unknown to the recovery catalog. When you grant `REGISTER DATABASE` to a user, RMAN implicitly grants recovery `CATALOG FOR DATABASE` privileges for any database registered by the user. The `CATALOG FOR DATABASE` privileges on the database include registering and unregistering this database. For example, assume that user `virtcat` is granted `REGISTER DATABASE` and registers database `prod` in the catalog. RMAN implicitly grants recovery `CATALOG FOR DATABASE` privileges for `prod` to `virtcat`.

Examples

Example 2–89 Granting Privileges for a Virtual Private Catalog

Assume that database user `rco` own the base recovery catalog in database `catdb`. This base recovery catalog stores the RMAN metadata for a large number of databases in a data center. Your goal is to create virtual private catalogs for two backup operators in the data center.

You start SQL*Plus and connect to the `catdb` database as `SYS`. You then use the `CREATE USER` statement to create the `bckop2` and `bckop3` users on `catdb`. You can grant recovery catalog ownership to these users as follows:

```
SQL> GRANT recovery_catalog_owner TO bckop2, bckop3;
SQL> EXIT
```

You then start the RMAN client and connect to the recovery catalog database as user `rco`. You use the RMAN `GRANT` command to give `bckop2` the ability to register any database in her virtual private catalog, but grant `bckop3` access to only a subset of the databases in the data center:

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

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

RMAN> GRANT REGISTER DATABASE TO bckop2;
RMAN> GRANT CATALOG FOR DATABASE prod TO bckop3;
RMAN> GRANT CATALOG FOR DATABASE prodb TO bckop3;
RMAN> EXIT;
```

You start a new RMAN session and create the virtual catalog for `bckop2` (sample `CREATE VIRTUAL CATALOG` output included). Note that you must exit and restart RMAN after creating each virtual catalog.

```
RMAN> CONNECT CATALOG bckop2@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CREATE VIRTUAL CATALOG;

found eligible base catalog owned by RCO
created virtual catalog against base catalog owned by RCO

RMAN> EXIT;
```

You start a new RMAN session and create the virtual catalog for `bckop3` (sample `CREATE VIRTUAL CATALOG` output included):

```
RMAN> CONNECT CATALOG bckop3@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CREATE VIRTUAL CATALOG;

found eligible base catalog owned by RCO
created virtual catalog against base catalog owned by RCO

RMAN> EXIT;
```

In the following example, backup operator `dba1` uses her virtual private catalog, which is stored in the `bckop3` schema on `catdb`, to store the metadata for a backup of a target database:

```
RMAN> CONNECT TARGET /
RMAN> CONNECT CATALOG bckop3@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

HOST

Purpose

Use the HOST command to invoke an operating system command-line sub-shell from within RMAN.

Syntax

***host*::=**

Prerequisites

Execute this command at the RMAN prompt or within the braces of a RUN command.

Semantics

Syntax Element	Description
HOST	Displays a command prompt and resumes after you exit the subshell (see Example 2–90 on page 2-159).
'*command*'	Runs the command in the specified string and then continues (see Example 2–91 on page 2-160).

Examples

Example 2–90 Hosting to the Operating System Within a Backup

This example makes an image copy of datafile 3, hosts out to the Linux prompt to check that the copy is in the directory (the Linux session output is indented and displayed in bold), and then resumes the RMAN session:

```
RMAN> BACKUP DATAFILE 3 FORMAT '/disk2/df3.cpy';

Starting backup at 15-FEB-07
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00003 name=/disk1/oracle/oradata/prod/undotbs01.d bf
channel ORA_DISK_1: starting piece 1 at 15-FEB-07
channel ORA_DISK_1: finished piece 1 at 15-FEB-07
piece handle=/disk2/df3.cpy tag=TAG20070215T111326 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 15-FEB-07

RMAN> HOST;

% ls /disk2/df3.copy
/disk2/df3.cpy
% exit
exit
host command complete
```

```
RMAN>
```

Example 2–91 Executing an Operating System Copy Within RMAN

This example makes a backup of datafile `system01.dbf` and then executes the Linux `ls` command to display all files in the `/disk2` directory:

```
BACKUP DATAFILE '?/oradata/prod/system01.dbf'
  FORMAT '/disk2/system01.dbf';
HOST 'ls -lt /disk2/*';
```

IMPORT CATALOG

Purpose

Use the `IMPORT CATALOG` command to import the metadata from one recovery catalog schema into a different catalog schema. If you created catalog schemas of different versions to store metadata for multiple target databases, then this command enables you to maintain a single catalog schema for all databases.

See Also: `CREATE CATALOG` on page 2-102

Prerequisites

RMAN must be connected to the destination recovery catalog, which is the catalog into which you want to import catalog data. This recovery catalog must not be a virtual private catalog.

No target database connection is needed in order to merge catalog schemas. Execute this command at the RMAN prompt or within the braces of a `RUN` command.

The version of the source recovery catalog schema must be equal to the current version of the RMAN executable. If the source schema is a lower version, then upgrade the catalog schema to the current version. If the source schema is a higher version, then retry the operation with a higher version RMAN executable.

Ensure that the same database is not registered in both the source recovery catalog schema and destination catalog schema. If a database is registered in both schemas, then `UNREGISTER` this database from source recovery catalog and execute the `IMPORT` command again.

Usage Notes

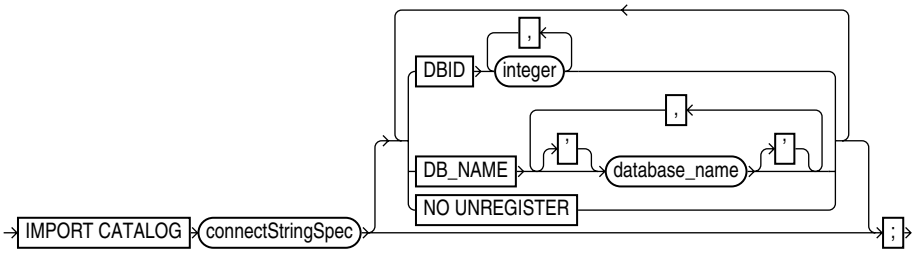
If the operation fails in the middle of the import, then the import is rolled back. Thus, a partial import is not permitted. The unregister operation is separate from the import. By default, the imported database IDs are unregistered from the source recovery catalog schema after a successful import.

Stored scripts are either global or local. It is possible for global scripts, but not local scripts, to have name conflicts during import because the destination schema already contains the script name. In this case, RMAN renames the global script name to `COPY OF` *`script_name`*. For example, RMAN renames `bp_cmd` to `COPY OF bp_cmd`.

If the renamed global script is still not unique, then RMAN renames it to `COPY(2) OF` *`script_name`*. If this script name also exists, then RMAN renames the script to `COPY(3) OF` *`script_name`*. RMAN continues the `COPY(`*`n`*`) OF` pattern until the script is uniquely named.

Syntax

***import*::=**



(*connectStringSpec::=* on page 4-13)

Semantics

Syntax Element	Description
connectStringSpec	Specifies the connection string for the source recovery catalog, which is the catalog whose metadata will be imported.
DBID *integer*	Specifies the list of DBIDs for the databases whose metadata should be imported from the source catalog schema (see Example 2–93). When not specified, RMAN merges metadata for all database IDs from the source catalog schema into the destination catalog schema. RMAN issues an error if the database whose metadata is merged is already registered in the recovery catalog schema.
DB_NAME *database_name*	Specifies the list of databases whose metadata should be imported from the source catalog schema (see Example 2–93). When not specified, RMAN merges metadata for all databases from the source catalog schema into the destination catalog schema. RMAN issues an error if the same DBID is registered in both recovery catalogs.
NO UNREGISTER	Forces RMAN to keep imported database IDs in the source catalog schema. By default, the imported database IDs are unregistered from source recovery catalog schema.

Examples

Example 2–92 Importing Metadata for All Registered Databases

In this example, database `inst1` contains a 10.2 catalog schema owned by user `rcat`, while database `catdb` contains an 11.1 catalog schema owned by user `rco`. RMAN imports metadata for all database IDs registered in `rcat` into the recovery catalog owned by `rco`. All target databases registered in `rcat` are unregistered.

```
RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> IMPORT CATALOG rcat@prod;

Starting import catalog at 15-FEB-07
source recovery catalog database Password: password
connected to source recovery catalog database
import validation complete
database unregistered from the source recovery catalog
```

```
Finished import catalog at 15-FEB-07
```

Example 2–93 Importing Metadata for a Subset of Registered Databases

This example is a variation on Example 2–92. Instead of importing the entire recovery catalog, it imports only the metadata for the database with DBID 1618984270.

```
RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> IMPORT CATALOG rcat@inst1 DBID=1618984270;

Starting import catalog at 15-FEB-07
source recovery catalog database Password: password
connected to source recovery catalog database
import validation complete
database unregistered from the source recovery catalog
Finished import catalog at 15-FEB-07
```

LIST

Purpose

Use the LIST command to display backups and information about other objects recorded in the RMAN repository.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to make lists and reports, and REPORT on page 3-59

Additional Topics

- Prerequisites
- Usage Notes
- Syntax
- Semantics
- LIST Command Output
- Examples

Prerequisites

Execute LIST only at the RMAN prompt. Either of the following conditions must be met:

- RMAN must be connected to a target database. If RMAN is *not* connected to a recovery catalog, and if you are not executing the LIST FAILURE command, then the target database must be mounted or open. If RMAN *is* connected to a recovery catalog, then the target database instance must be started.
- RMAN must be connected to a recovery catalog and SET DBID must have been run.

Usage Notes

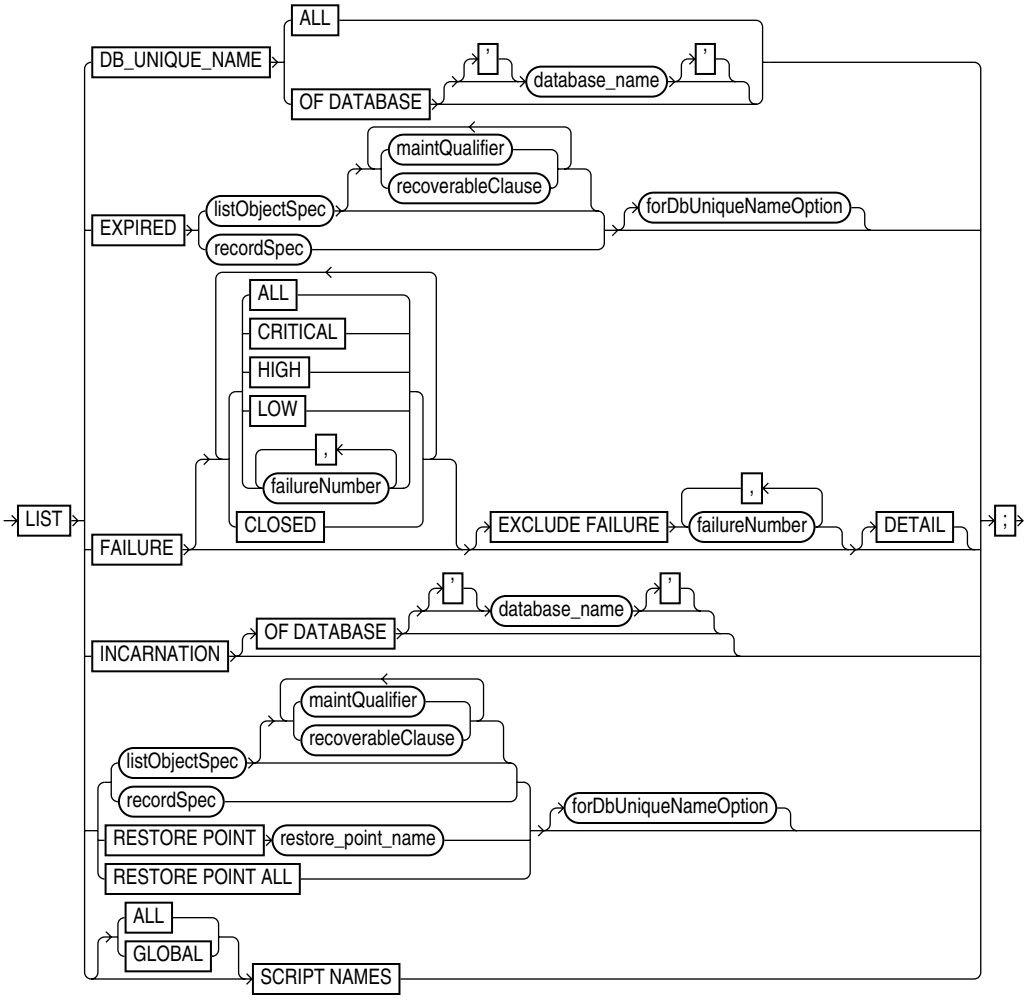
With the exception of LIST FAILURE, the LIST command displays the backups and copies against which you can run CROSSCHECK and DELETE commands. The LIST FAILURE command displays failures against which you can run the ADVISE FAILURE and REPAIR FAILURE commands.

"RMAN Backups in a Data Guard Environment" on page 2-22 explains how RMAN handles backups in a Data Guard environment. In general, RMAN considers tape backups created on one database in the environment as accessible to all databases in the environment, whereas disk backups are accessible only to the database that created them. In a Data Guard environment, LIST displays those files that are accessible to the connected target database.

RMAN prints the LIST output to either standard output or the message log, but not to both at the same time.

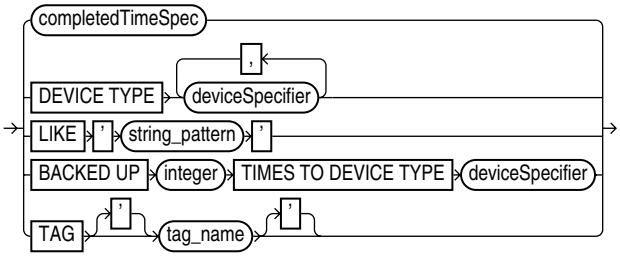
Syntax

list::=



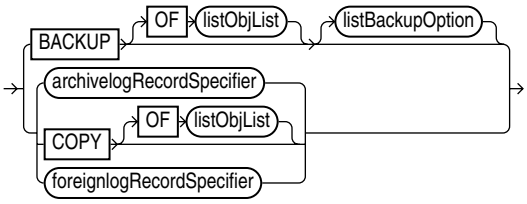
(*listObjectSpec::=* on page 2-166, *recordSpec::=* on page 4-37, *maintQualifier::=* on page 4-32, *forDbUniqueNameOption::=* on page 4-20, *untilClause::=* on page 4-42)

maintQualifier::=



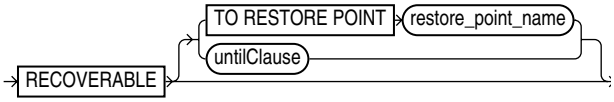
(*completedTimeSpec::=* on page 4-11, *deviceSpecifier::=* on page 4-16)

listObjectSpec::=



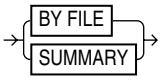
(*listObjList::=* on page 4-29, *listBackupOption::=* on page 2-166, *archivelogRecordSpecifier::=* on page 4-6, *foreignlogRecordSpecifier::=* on page 4-21)

recoverableClause::=



(*untilClause::=* on page 4-42)

listBackupOption::=



Semantics

list

Syntax Element	Description
DB_UNIQUE_NAME	Lists the DB_UNIQUE_NAME of one or more databases registered in the recovery catalog. RMAN must be connected to a recovery catalog. RMAN must also be connected to a mounted or open target database, or you must identify the target database with the SET DBID command. Note that the DBID for a primary database is identical to the DBID of its associated standby databases: they are distinguished by DB_UNIQUE_NAME. **See Also:** Table 2–28 for a description of the output
ALL	Lists the DB_UNIQUE_NAME of every database registered in the RMAN repository.
OF DATABASE *database_name*	Lists the DB_UNIQUE_NAME of every database with the specified DB_NAME.
EXPIRED	Displays backup sets, proxy copies, and image copies marked in the repository as EXPIRED, which means they were not found. See Table 2–9, " List of Backup Sets (for datafile backup sets)" for a description of the output. To ensure that LIST EXPIRED shows up-to-date output, use a CROSSCHECK command periodically. When you use a CROSSCHECK command, RMAN searches disk and tape for backups and copies recorded in the repository. If it does not find them, then it updates their repository records to status EXPIRED.
listObjectSpec	Specifies the type of expired object or objects that you are listing. **See Also:** *listObjectSpec* on page 2-169

Syntax Element	Description
maintQualifier	Restricts the range of the listing. **See Also:** *maintQualifier* on page 4-32
recordSpec	Specifies the expired object or objects that you are listing. **See Also:** *recordSpec* on page 4-37
forDbUniqueNameOption	Lists the expired files in *listObjectSpec* or *recordSpec* that are exclusively associated with the specified DB_UNIQUE_NAME in a Data Guard environment. You can specify a database with *db_unique_name* or use ALL for all uniquely named databases recorded in the catalog for a particular DBID. A database is uniquely identified in the recovery catalog by a DBID and the value of the DB_UNIQUE_NAME initialization parameter. RMAN must be connected to a recovery catalog. RMAN must also be connected to a mounted or open target database, or you must have run the SET DBID command. **See Also:** *forDbUniqueNameOption* on page 4-20 for descriptions of the options in this clause
FAILURE	Lists failures recorded by the Data Recovery Advisor. The database to which RMAN is connected must be a single-instance database and must not be a physical standby database. **See Also:** "Oracle RAC and Data Recovery Advisor" on page 2-6 The Data Recovery Advisor can detect and repair a wide variety of physical problems that cause data loss and corruption. Physical corruptions are typically caused by faulty I/O subsystems or human error. The Data Recovery Advisor may not detect or handle some types of logical corruptions. Corruptions of this type require help from Oracle Support Services. In the context of Data Recovery Advisor, a **failure** is a persistent data corruption that is mapped to a set of repair actions. Data failures are detected by checks, which are diagnostic procedures that assess the health of the database or its components. Each check can diagnose one or more failures, which are mapped to a set of repairs. The typical use case is to run LIST FAILURE to list any failures, then use ADVISE FAILURE to display repair options, and REPAIR FAILURE to fix the failures. Run these commands in the same RMAN session. If no options are specified on LIST FAILURE, then the command lists only the highest priority failures that have status OPEN. Therefore, CRITICAL and HIGH failures are always listed in the command output if they exist. Failures with LOW priority are listed only if no CRITICAL or HIGH priority failures exist. Failures are sorted in reverse order of occurrence, with the most recent failure listed first. The LIST FAILURE command does not initiate checks to diagnose new failures; rather, it lists the results of previously executed assessments. Thus, repeatedly executing LIST FAILURE will reveal new failures only if the database automatically diagnosed them in response to errors that occurred in between command executions. However, LIST FAILURE revalidates all existing failures when the command is issued. If a user fixed failures manually, or if the failures were transient problems that disappeared, then Data Recovery Advisor removes these failures from the LIST FAILURE output. **See Also:** Table 2–27 for an explanation of the column headings of the LIST FAILURE output table and Example 2–99 for an illustration
ALL	Lists failures with all priorities and status OPEN.
CRITICAL	Lists only critical failures with status OPEN.
HIGH	Lists only failures with HIGH priority and status OPEN.
LOW	Lists only failures with LOW priority with status OPEN.
failureNumber	Specifies the failures by failure number.

Syntax Element	Description
CLOSED	Lists only closed failures.
EXCLUDE FAILURE *failureNumber*	Excludes the specified failures from the list.
DETAIL	Lists failures by expanding the consolidated failure. For example, if multiple block corruptions existed in a file, then specifying the DETAIL option would list each of the block corruptions.
INCARNATION	Displays information about the incarnations of a database. Whenever you open a database with the RESETLOGS option, you create a new incarnation of the database. If LIST INCARNATION displays *n* incarnations of a database, then you have reset the online redo logs for this database *n*-1 times. The LIST output includes the primary keys of all database incarnation records for the specified database name (in the column Inc Key, which contains the incarnation key). Use the key in a RESET DATABASE command to change the incarnation that RMAN considers to be current to a previous incarnation. **See Also:** Table 2–24 for an explanation of the column headings of the LIST INCARNATION output table and Example 2–98 for an illustration
OF DATABASE *database_name*	Specifies the name of the database. If you do not specify the OF DATABASE option, then LIST displays all databases registered in the recovery catalog.
listObjectSpec	Specifies the type of expired object or objects that you are listing. **See Also:** *listObjectSpec* on page 2-169
maintQualifier	Restricts the range of the listing. **See Also:** *maintQualifier* on page 4-32
recoverableClause	Restricts the list to datafile backups or copies whose status in the repository is AVAILABLE and which can be used for restore and recovery in the current incarnation of the target database. **See Also:** *recoverableClause* on page 2-169
recordSpec	Specifies the object or objects that you are listing. **See Also:** *recordSpec* on page 4-37
untilClause	Specifies an end time, SCN, or log sequence number. **See Also:** *untilClause* on page 4-42
RESTORE POINT	Displays restore points known to the RMAN repository. **See Also:** Table 2–26 for an explanation of the column headings of the LIST RESTORE POINT output table
restore_point_name	Displays the specified restore point.
ALL	Displays all restore points known to the RMAN repository.
forDbUniqueNameOption	Lists the backups and restore points that are exclusively associated with the specified DB_UNIQUE_NAME in a Data Guard environment. You can specify a database with *db_unique_name* or use ALL for all uniquely named databases recorded in the catalog for a particular DBID. A database is uniquely identified in the recovery catalog by a DBID and the value of the DB_UNIQUE_NAME initialization parameter. RMAN must be connected to a recovery catalog. RMAN must also be connected to a mounted or open target database. **See Also:** *forDbUniqueNameOption* on page 4-20 for descriptions of the options in this clause

Syntax Element	Description
ALL SCRIPT NAMES	RMAN lists all global and local scripts defined for all databases in the connected recovery catalog, along with any descriptive comments. You must be connected to a recovery catalog, but you do not need to be connected to a target database.
GLOBAL SCRIPT NAMES	RMAN lists only global scripts defined in the connected recovery catalog, along with any descriptive comments. You must be connected to a recovery catalog, but you do not need to be connected to a target database.
SCRIPT NAMES	Lists local and global scripts that can be executed on the current target database. You must be connected to a target database and a recovery catalog to use this form of the command. **See Also:** Table 2–25 for a description of the output and Example 3–13, "Replacing a Recovery Catalog Script" for an illustration

listObjectSpec

This subclause specifies the type of object or objects that you are listing.

Syntax Element	Description
BACKUP	Displays information about backup sets (including detail on backup pieces) and proxy copies. **See Also:** Table 2–9 for a description of LIST BACKUP output and Example 2–94 for an illustration
OF *listObjList*	Restricts the list of objects operated on to the object type specified in the *listObjList* clause. If you do not specify an object, then LIST defaults to OF DATABASE CONTROLFILE ARCHIVELOG ALL. **Note:** The LIST BACKUP . . . LIKE command is not valid. The only valid exception is LIST BACKUP OF ARCHIVELOG LIKE. **See Also:** *listObjList* on page 4-29
listBackupOption	Specifies whether to list summary information about backups or detailed information.
archivelogRecordSpecifier	Displays information about a range of archived redo log files.
COPY	Displays only information about datafile copies, archived redo log files, and image copies of archived redo log files. By default, LIST COPY displays copies of all database files and archived redo log files. 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 2–21 on page 2-177 and Table 2–23 on page 2-178 for an explanation of the column headings of the LIST COPY output tables
OF *listObjList*	Restricts the list of objects operated on to the object type specified in the *listObjList* clause. **See Also:** *listObjList* on page 4-29
foreignlogRecordSpecifier	Displays information about a range of foreign archived redo log files.

recoverableClause

This subclause specifies recoverable backups.

Syntax Element	Description
RECOVERABLE	Restricts the list to expired datafile backups or copies whose status in the repository is AVAILABLE and which can be used for restore and recovery in the current incarnation of the target database. This list includes all backups and copies except the incremental backups that have no valid parent to which the incremental can be applied.
TO RESTORE POINT *restore_point_name*	Specifies a restore point, with the SCN at which the restore point was created as the upper, inclusive limit. Because the limit is inclusive, RMAN only lists files that are recoverable up to *and including* the SCN corresponding to the restore point.
untilClause	Specifies an end time, SCN, or log sequence number. **See Also:** *untilClause* on page 4-42

listBackupOption

Specifies whether to summarize backups or list the backups for a particular datafile.

Syntax Element	Description
BY FILE	Lists backups of each datafile, archived redo log file, control file, and server parameter file. **See Also:** Table 2–18, Table 2–19, and Table 2–20 for a description of LIST BACKUP ... BY FILE output
SUMMARY	Gives a one-line summary for each backup. **See Also:** Table 2–16 for a description of LIST BACKUP ... SUMMARY output and Example 2–95 for sample output

LIST Command Output

The information that appears in the output is described in the following tables:

- Table 2–9, " List of Backup Sets (for datafile backup sets)"
- Table 2–10, " List of Backup Pieces (for sets with only one piece)"
- Table 2–11, " List of Datafiles in backup set ..."
- Table 2–12, " List of Archived Logs in backup set ..."
- Table 2–13, " Backup Set Copy ... of backup set ... (only if multiple pieces)"
- Table 2–14, " List of Backup Pieces for backup set ... Copy ... (if multiple pieces)"
- Table 2–15, " List of Proxy Copies"
- Table 2–16, " List of Backup Sets (LIST BACKUP ... SUMMARY)"
- Table 2–17, " List of Backup Pieces (LIST BACKUPPIECE ...)"
- Table 2–18, " List of Datafile Backups (LIST BACKUP ... BY FILE)"
- Table 2–19, " List of Archived Log Backups (LIST BACKUP ... BY FILE)"
- Table 2–20, " List of Control File Backups (LIST BACKUP ... BY FILE)"
- Table 2–21, " List of Datafile Copies"
- Table 2–22, " List of Control File Copies"
- Table 2–23, " List of Archived Log Copies"
- Table 2–24, " List of Database Incarnations"

- Table 2–25, " List of Stored Scripts in the Recovery Catalog (LIST SCRIPT NAMES)"
- Table 2–26, " List of Restore Points (LIST RESTORE POINT)"
- Table 2–27, " List of Failures"
- Table 2–28, " List of Databases (LIST DB_UNIQUE_NAME)"

Table 2–9 List of Backup Sets (for datafile backup sets)

Column	Indicates
`BS Key`	A unique key identifying this backup set. If RMAN is connected to a recovery catalog, then `BS Key` is the primary key of the backup set in the catalog. It corresponds to `BS_KEY` in the `RC_BACKUP_SET` view. If RMAN is connected in the default `NOCATALOG` mode, then `BS Key` displays the `RECID` from `V$BACKUP_SET`.
`Type`	The type of backup: `Full` or `Incr` (incremental). **Note:** Column only included in datafile backup sets.
`LV`	The level of the backup: `NULL` for nonincrementals, level `0` or level 1 for incrementals. **Note:** Column only included in datafile backup sets.
`Size`	The size of the backup in bytes. **Note:** Column only included in datafile backup sets.
`Device Type`	The type of device on which the backup was made, for example, `DISK` or `sbt`.
`Elapsed Time`	The duration of the backup.
`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.
`List of datafiles in backup set ...`	See Table 2–11

Table 2–10 List of Backup Pieces (for sets with only one piece)

Column	Indicates
`BP Key`	A unique identifier for this backup piece in the recovery catalog or target database control file. If RMAN is connected to a recovery catalog, then `BP Key` is the primary key of the backup piece in the catalog. It corresponds to `BP_KEY` in the `RC_BACKUP_PIECE` view. If RMAN is connected in `NOCATALOG` mode, then `BP Key` displays the `RECID` from `V$BACKUP_PIECE`. **Note:** The values for `KEY` in the recovery catalog and the control file are different.
`Status`	The backup piece status: `AVAILABLE`, `UNAVAILABLE`, or `EXPIRED` (refer to the `CHANGE`, `CROSSCHECK`, and `DELETE` commands for an explanation of each status).
`Compressed`	Whether the backup piece is compressed (`YES` or `NO`).
`Tag`	The tag applied to the backup set. Note that tag names are not case sensitive and display in all uppercase.

Table 2–10 (Cont.) List of Backup Pieces (for sets with only one piece)

Column	Indicates
`Piece Name/Handle`	The filename or handle of the backup piece. If the backup piece is on SBT, then the Media ID is displayed with the name.
`SPFILE Included`	A server parameter file is included in the backup.
`Control File Included`	A control file is included in the backup. **Note:** This row appears only if the current control file is included in the backup.
`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. **Note:** This row appears only if the current control file is included in the backup.
`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. **Note:** This row appears only if the current control file is included in the backup.

Table 2–11 List of Datafiles in backup set ...

Column	Indicates
`File`	The number of the file that was backed up.
`LV`	The level of the backup: `NULL` for nonincrementals, level 0 or 1 for incrementals.
`Type`	The type of backup: `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.
`Name`	The location where this file would be restored now if it were restored from this backup set and no `SET NEWNAME` command was entered. **See Also:** `SET` on page 3-70

Table 2–12 List of Archived Logs in backup set ...

Column	Indicates
`Thrd`	The thread number of the redo log.
`Seq`	The log sequence number of the archived log.
`Low SCN`	The lowest SCN in the archived log.
`Low Time`	The time when the database switched into the redo log having this sequence number.
`Next SCN`	The low SCN of the next archived log sequence.
`Next Time`	The low time of the next archived log sequence.

Table 2–13 Backup Set Copy ... of backup set ... (only if multiple pieces)

Column	Indicates
Device Type	The type of device on which the backup was made, for example, DISK or sbt.
Elapsed Time	The duration of the backup.
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.
Tag	The tag applied to the backup set. Note that tag names are not case sensitive and display in all uppercase.

Table 2–14 List of Backup Pieces for backup set ... Copy ... (if multiple pieces)

Column	Indicates
BP Key	A unique identifier for this backup piece in the recovery catalog or target database control file. If RMAN is connected to a recovery catalog, then BP Key is the primary key of the backup piece in the catalog. It corresponds to BP_KEY in the RC_BACKUP_PIECE view. If RMAN is connected in NOCATALOG mode, then BP Key displays the RECID from V$BACKUP_PIECE. **Note:** The values for KEY in the recovery catalog and the control file are different.
Pc#	The number of the backup piece in the backup set.
Status	The backup piece status: AVAILABLE, UNAVAILABLE, or EXPIRED (refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status).
Piece Name	The filename or handle of the backup piece. If the backup piece is stored on SBT, then the media ID is also displayed.

Table 2–15 List of Proxy Copies

Column	Indicates
PC Key	A unique key identifying this proxy copy. If RMAN is connected to a catalog, then PC Key is the primary key of the proxy copy in the catalog. It corresponds to XDF_KEY in the RC_PROXY_DATAFILE view or XCF_KEY in the RC_PROXY_CONTROLFILE view. If RMAN is connected in NOCATALOG mode, then PC Key displays the RECID from V$PROXY_DATAFILE.
File	The absolute datafile number of the file that was copied.
Status	The proxy copy status: AVAILABLE, UNAVAILABLE, or EXPIRED (refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status).
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.
Ckp SCN	The SCN of the proxy copy control file checkpoint. All database changes recorded in the redo records before the specified SCN are reflected in this control file.

Table 2–15 (Cont.) List of Proxy Copies

Column	Indicates
Ckp time	The time of the proxy copy control file checkpoint. All database changes recorded in the redo records before the specified time are reflected in this control file.
Datafile name	The location where this file would be restored now if it were restored from this backup set and no SET NEWNAME command was entered. **See Also:** SET command on page 3-70
Handle	The media manager's handle for the proxy copy. If the object is on sbt, then the media ID is also displayed.
Tag	The tag applied to the proxy copy. Note that tag names are not case sensitive and display in all uppercase.

Table 2–16 List of Backup Sets (LIST BACKUP ... SUMMARY)

Column	Indicates
Key	A unique key identifying this backup set. If RMAN is connected to a recovery catalog, then BS Key is the primary key of the backup set in the catalog. It corresponds to BS_KEY in the RC_BACKUP_SET view. If RMAN is connected in NOCATALOG mode, then BS Key displays the RECID from V$BACKUP_SET.
TY	The type of backup: backup set (B) or proxy copy (P).
LV	For incremental backups, the incremental backup level (0 or 1). For backup sets containing full backups of datafiles, F. For backup sets containing archived redo log files, A.
S	The status of the backup: A (available), U (unavailable), or X (all backup pieces in set expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
Device Type	The type of device on which the backup was made, for example, DISK or sbt.
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.
#Pieces	The number of backup pieces in the backup set.
#Copies	The number of copies made of each backup piece in the set. The number is 1 if no duplexing was performed. Otherwise, the value ranges from 2 to 4.
Compressed	YES if the backup set was compressed by RMAN; NO if not compressed by RMAN.
Tag	The tag applied to the backup set. An asterisk (*) indicates that backup pieces have different tags within the same backup set, which occurs when a user changes the tag when using CATALOG or BACKUP BACKUPSET. Note that tag names are not case sensitive and display in all uppercase.

Table 2–17 List of Backup Pieces (LIST BACKUPPIECE ...)

Column	Indicates
BP Key	A unique identifier for this backup piece in the recovery catalog or target database control file. If RMAN is connected to a catalog, then BP Key is the primary key of the backup piece in the catalog. It corresponds to BP_KEY in the RC_BACKUP_PIECE view. If RMAN is connected in NOCATALOG mode, then BP Key displays the RECID from V$BACKUP_PIECE. **Note:** The values for KEY in the recovery catalog and the control file are different.
BS Key	A unique key identifying this backup set. If RMAN is connected to a recovery catalog, then BS Key is the primary key of the backup set in the catalog. It corresponds to BS_KEY in the RC_BACKUP_SET view. If RMAN is connected in NOCATALOG mode, then BS Key displays the RECID from V$BACKUP_SET.
Pc#	The number of the backup piece in the backup set.
Cp#	The copy number of this backup piece in the backup set. The number is 1 if no duplexing was performed. Otherwise, the value ranges from 2 to 4.
Status	The backup piece status: AVAILABLE, UNAVAILABLE, or EXPIRED (refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status).
Device Type	The type of device on which the backup was made, for example, DISK or sbt.
Piece Name	The filename or handle of the backup piece. If the piece is stored on SBT, then the Handle and media ID are displayed.

Table 2–18 List of Datafile Backups (LIST BACKUP ... BY FILE)

Column	Indicates
File	The absolute datafile number.
Key	A unique key identifying this backup set. If RMAN is connected to a recovery catalog, then Key is the primary key of the backup set in the catalog. It corresponds to BS_KEY in the RC_BACKUP_SET view. If RMAN is connected to a target database in NOCATALOG mode, then Key displays the RECID from V$BACKUP_SET.
TY	The type of backup: backup set (B) or proxy copy (P).
LV	The backup level: F for nonincrementals, level 0 or 1 for incrementals.
S	The status of the backup: A (available), U (unavailable), or X (all backup pieces in set expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
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 2–18 (Cont.) List of Datafile Backups (LIST BACKUP ... BY FILE)

Column	Indicates
#Pieces	The number of backup pieces in the backup set.
#Copies	The number of copies made of each backup piece in the set. The number is 1 if no duplexing was performed. Otherwise, the value ranges from 2 to 4.
Compressed	YES if the backup was compressed by RMAN; NO if not compressed by RMAN.
Tag	The tag applied to the backup set. An asterisk (*) indicates that backup pieces have different tags within the same backup set, which occurs when a user changes the tag when using CATALOG or BACKUP BACKUPSET. Note that tag names are not case sensitive and display in all uppercase.

Table 2–19 List of Archived Log Backups (LIST BACKUP ... BY FILE)

Column	Indicates
Thrd	The thread number of the redo log.
Seq	The log sequence number of the archived log.
Low SCN	The lowest SCN in the archived log.
Low Time	The time when the database switched into the redo log having this sequence number.
BS Key	A unique key identifying this backup set. If RMAN is connected to a recovery catalog, then BS Key is the primary key of the backup set in the catalog. It corresponds to BS_KEY in the RC_BACKUP_SET view. If RMAN is connected in NOCATALOG mode, then BS Key displays the RECID from V$BACKUP_SET.
S	The status of the backup: A (available), U (unavailable), or X (all backup pieces in set expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
#Pieces	The number of backup pieces in the backup set.
#Copies	The number of copies made of each backup piece in the set. The number is 1 if no duplexing was performed. Otherwise, the value ranges from 2 to 4.
Compressed	YES if the backup was compressed by RMAN; NO if not compressed by RMAN.
Tag	The tag applied to the backup set. Note that tag names are not case sensitive and display in all uppercase.

Table 2–20 List of Control File Backups (LIST BACKUP ... BY FILE)

Column	Indicates
CF Ckp SCN	Checkpoint SCN of the control file.
Ckp Time	The log sequence number of the archived log.
BS Key	A unique key identifying this backup set. If RMAN is connected to a recovery catalog, then BS Key is the primary key of the backup set in the catalog. It corresponds to BS_KEY in the RC_BACKUP_SET view. If RMAN is connected in NOCATALOG mode, then BS Key displays the RECID from V$BACKUP_SET.
S	The status of the backup: A (available), U (unavailable), or X (all backup pieces in set expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
#Pieces	The number of backup pieces in the backup set.
#Copies	The number of copies made of each backup piece in the set. The number is 1 if no duplexing was performed. Otherwise, the value ranges from 2 to 4.
Compressed	YES if the backup was compressed by RMAN; NO if not compressed by RMAN.
Tag	The tag applied to the backup set. Note that tag names are not case sensitive and display in all uppercase.

Table 2–21 List of Datafile Copies

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. If RMAN is connected to a recovery catalog, then Key is the primary key of the datafile copy in the catalog. It corresponds to CDF_KEY in the RC_DATAFILE_COPY view. If RMAN is connected in NOCATALOG mode, then Key displays the RECID from V$DATAFILE_COPY. **Note:** The values for KEY in the recovery catalog and the control file are different.
File	The file number of the datafile from which this copy was made.
S	The status of the copy: A (available), U (unavailable), or X (expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
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 this SCN have been written to this datafile.
Ckp TIME	The checkpoint of this datafile when it was copied. All database changes prior to this time have been written to this datafile.
Name	The filename of the datafile copy.
Tag	The tag applied to the datafile copy. Note that tag names are not case sensitive and display in all uppercase.

Table 2–22 List of Control File Copies

Column	Indicates
Key	The unique identifier for the control file copy. Use this value in a CHANGE command to alter the status of the copy. If RMAN is connected to a recovery catalog, then Key is the primary key of the control file copy in the catalog. It corresponds to CCF_KEY in the RC_CONTROLFILE_COPY view. If RMAN is connected in NOCATALOG mode, then Key displays the RECID from V$DATAFILE_COPY. **Note:** The values for Key in the recovery catalog and the control file are different.
S	The status of the copy: A (available), U (unavailable), or X (expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
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 control file when it was copied.
Ckp TIME	The checkpoint of this control file when it was copied.
Name	The filename of the control file copy.
Tag	The tag applied to the control file copy. Note that tag names are not case sensitive and display in all uppercase.

Table 2–23 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. If RMAN is connected to a recovery catalog, then Key is the primary key of the backup set in the catalog. It corresponds to AL_KEY in the RC_ARCHIVED_LOG view. If RMAN is connected in NOCATALOG mode, then Key displays the RECID from V$ARCHIVED_LOG. **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.
S	The status of the copy: A (available), U (unavailable), or X (expired). Refer to the CHANGE, CROSSCHECK, and DELETE commands for an explanation of each status.
Low Time	The time when the database switched into the redo log having this sequence number.
Name	The filename of the archived redo log copy.

Table 2–24 List of Database Incarnations

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 from a recovery catalog, 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 RESET DATABASE TO INCARNATION when recovering the database to a time before the most recent RESETLOGS.
DB Name	The database name as listed in the DB_NAME parameter.
DB ID	The database identification number, which the database generates automatically at database creation.
STATUS	CURRENT for the current incarnation, PARENT for the parent incarnations of the current incarnation, and ORPHAN for orphaned incarnations.
Reset SCN	The SCN at which the incarnation was created.
Reset Time	The time at which the incarnation was created.

Table 2–25 List of Stored Scripts in the Recovery Catalog (LIST SCRIPT NAMES)

Column	Indicates
Script Name	The name of the stored script.
Description	The comment provided when the script was created.

Table 2–26 List of Restore Points (LIST RESTORE POINT)

Column	Indicates
SCN	The SCN for the restore point.
RSP Time	The time specified in the CREATE RESTORE POINT statement; otherwise null.
Type	GUARANTEED if a guaranteed restore point; null if a normal restore point.
Time	The time that the restore point was created.
Name	The name of the restore point.

Table 2–27 List of Failures

Column	Indicates
Failure ID	The unique identifier for a failure.
Priority	The priority of the failure: CRITICAL, HIGH, or LOW. Failures with critical priority require immediate attention because they make the whole database unavailable. Typically, critical failures bring down the instance and are diagnosed during the subsequent startup. The database is not available until all critical failures are fixed (see ADVISE FAILURE). Failures with HIGH priority make a database partially unavailable or unrecoverable, and usually have to be repaired in a reasonably short time. Examples of such failures include physical data block corruptions, nonfatal I/O errors, missing archived redo log files or backup files, and so on. Failures with LOW priority can be ignored until more important failures are fixed. For example, a block corruption will be initially assigned a high priority, but if this block is not important for the database availability, you can use CHANGE FAILURE to change the priority to LOW.
Status	The repair status of the failure. The status of a failure is OPEN (not repaired) until the appropriate repair action is invoked. The failure status changes to CLOSED when the repair is completed.
Time Detected	The date when the failure was diagnosed.
Summary	Summary of the failure.

Table 2–28 List of Databases (LIST DB_UNIQUE_NAME)

Column	Indicates
DB Key	When combined with Inc Key, the unique key by which RMAN identifies the database incarnation in the recovery catalog. Primary and standby databases share the same DB Key value.
DB Name	The DB_NAME of the database. Primary and standby databases share the same DB Name value.
DB ID	The DBID of the database. Primary and standby databases share the same DBID.
Database Role	PRIMARY if the database is a primary database; STANDBY if it is a standby database.
Db_unique_name	The DB_UNIQUE_NAME of the database. Primary and standby databases have different DB_UNIQUE_NAME values.

Examples

Example 2–94 Listing Backups

This example lists all backups. The output shows two backup sets on disk: one containing datafiles and the other containing autobackups. The output also shows one SBT backup containing archived redo log files.

```
RMAN> LIST BACKUP;

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

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
```

```
------- ---- -- ---------- ----------- ------------ ---------------
200     Full    509.78M    DISK        00:01:03     15-FEB-07
        BP Key: 202   Status: AVAILABLE  Compressed: NO  Tag: TAG20070215T171219
        Piece Name:
/disk2/PROD/backupset/2007_02_15/o1_mf_nnndf_TAG20070215T171219_2xb17nbb_.bkp
  List of Datafiles in backup set 200
  File LV Type Ckp SCN    Ckp Time  Name
  ---- -- ---- ---------- --------- ----
  1       Full 421946     15-FEB-07 /disk1/oradata/prod/system01.dbf
  2       Full 421946     15-FEB-07 /disk1/oradata/prod/sysaux01.dbf
  3       Full 421946     15-FEB-07 /disk1/oradata/prod/undotbs01.dbf
  4       Full 421946     15-FEB-07 /disk1/oradata/prod/cwmlite01.dbf
  5       Full 421946     15-FEB-07 /disk1/oradata/prod/drsys01.dbf
  6       Full 421946     15-FEB-07 /disk1/oradata/prod/example01.dbf
  7       Full 421946     15-FEB-07 /disk1/oradata/prod/indx01.dbf
  8       Full 421946     15-FEB-07 /disk1/oradata/prod/tools01.dbf
  9       Full 421946     15-FEB-07 /disk1/oradata/prod/users01.dbf

BS Key  Type LV Size       Device Type Elapsed Time Completion Time
------- ---- -- ---------- ----------- ------------ ---------------
201     Full    7.98M      DISK        00:00:03     15-FEB-07
        BP Key: 203   Status: AVAILABLE  Compressed: NO  Tag: TAG20070215T171219
        Piece Name:
/disk2/PROD/backupset/2007_02_15/o1_mf_ncsnf_TAG20070215T171219_2xb19prg_.bkp
  SPFILE Included: Modification time: 15-FEB-07
  SPFILE db_unique_name: PROD
  Control File Included: Ckp SCN: 421968       Ckp time: 15-FEB-07

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
227     30.50M     SBT_TAPE    00:00:11     15-FEB-07
        BP Key: 230   Status: AVAILABLE  Compressed: NO  Tag: TAG20070215T171334
        Handle: 0bia4rtv_1_1   Media:

  List of Archived Logs in backup set 227
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    5       389156     15-FEB-07 411006     15-FEB-07
  1    6       411006     15-FEB-07 412972     15-FEB-07
  1    7       412972     15-FEB-07 417086     15-FEB-07
  1    8       417086     15-FEB-07 417114     15-FEB-07
  1    9       417114     15-FEB-07 417853     15-FEB-07
  1    10      417853     15-FEB-07 421698     15-FEB-07
  1    11      421698     15-FEB-07 421988     15-FEB-07
```

Example 2–95 Listing a Summary of Backups

This example summarizes the RMAN backups listed in Example 2–94 on page 2-180.

```
RMAN> LIST BACKUP SUMMARY;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------- ---
200     B  F  A DISK        15-FEB-07       1       1       NO         TAG20070215T171219
201     B  F  A DISK        15-FEB-07       1       1       NO         TAG20070215T171219
227     B  A  A SBT_TAPE    15-FEB-07       1       1       NO         TAG20070215T171334
```

Example 2–96 Listing Backups by File

This example groups all backups by file. Note that the tag column of the datafile backup indicates that one backup set contains backup pieces with different tags. The status column of the archived log backup indicates that the backup set is unavailable.

```
RMAN> LIST BACKUP BY FILE;
```

```
List of Datafile Backups
========================

File Key     TY LV S Ckp SCN    Ckp Time  #Pieces #Copies Compressed Tag
---- ------- -  -- - ---------- --------- ------- ------- ---------- ---
1    329     B  F  A 454959     16-FEB-07 1       1       NO         DF1
2    349     B  F  A 454997     16-FEB-07 1       1       NO         DF2
3    527     B  F  A 455218     16-FEB-07 1       1       NO         FRI_BKP
     368     B  F  A 455022     16-FEB-07 1       1       NO         DF3
4    387     B  F  X 455042     16-FEB-07 1       1       NO         DF4
5    407     B  F  A 455063     16-FEB-07 1       1       NO         DF5
6    428     B  F  A 455083     16-FEB-07 1       2       NO         *
7    450     B  F  X 455103     16-FEB-07 1       1       NO         DF7
8    473     B  F  A 455123     16-FEB-07 1       1       NO         DF8
9    497     B  F  A 455143     16-FEB-07 1       1       NO         DF9

List of Archived Log Backups
============================

Thrd Seq     Low SCN    Low Time  BS Key  S #Pieces #Copies Compressed Tag
---- ------- ---------- --------- ------- - ------- ------- ---------- ---
1    5       389156     15-FEB-07 227     U 1       1       NO         TAG20070215T171334
1    6       411006     15-FEB-07 227     U 1       1       NO         TAG20070215T171334
1    7       412972     15-FEB-07 227     U 1       1       NO         TAG20070215T171334
1    8       417086     15-FEB-07 227     U 1       1       NO         TAG20070215T171334
1    9       417114     15-FEB-07 227     U 1       1       NO         TAG20070215T171334
1    10      417853     15-FEB-07 227     U 1       1       NO         TAG20070215T171334
1    11      421698     15-FEB-07 227     U 1       1       NO         TAG20070215T171334

List of Control File Backups
============================

CF Ckp SCN Ckp Time  BS Key  S #Pieces #Copies Compressed Tag
---------- --------- ------- - ------- ------- ---------- ---
454974     16-FEB-07 330     A 1       1       NO         DF1
421968     15-FEB-07 201     A 1       1       NO         TAG20070215T171219
List of SPFILE Backups
======================

Modification Time BS Key  S #Pieces #Copies Compressed Tag
----------------- ------- - ------- ------- ---------- ---
15-FEB-07         330     A 1       1       NO         DF1
15-FEB-07         201     A 1       1       NO         TAG20070215T171219
```

Example 2–97 Listing Image Copies

The following example lists all datafile, control file, and archived redo log copies known to RMAN:

```
RMAN> LIST COPY;

List of Datafile Copies
=======================

Key     File S Completion Time Ckp SCN    Ckp Time
------- ---- - --------------- ---------- ---------------
618     1    A 16-FEB-07       461057     16-FEB-07
        Name: /disk2/PROD/datafile/o1_mf_system_2xdbrg13_.dbf
        Tag: TAG20070216T140706

631     2    A 16-FEB-07       461163     16-FEB-07
        Name: /disk2/PROD/datafile/o1_mf_sysaux_2xdbzybx_.dbf
        Tag: TAG20070216T141109

List of Control File Copies
```

```
==========================

Key     S Completion Time Ckp SCN    Ckp Time
------- - --------------- ---------- ---------------
619     A 16-FEB-07       461133     16-FEB-07
        Name: /disk2/PROD/controlfile/o1_mf_TAG20070216T140706_2xdbz5tb_.ctl
        Tag: TAG20070216T140706

594     A 16-FEB-07       460650     16-FEB-07
        Name: /disk2/PROD/controlfile/o1_mf_TAG20070216T135954_2xdbbz99_.ctl
        Tag: TAG20070216T135954

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

Key     Thrd Seq     S Low Time
------- ---- ------- - ---------
105     1    5       A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_5_614616887.dbf

122     1    6       A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_6_614616887.dbf

123     1    7       A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_7_614616887.dbf

124     1    8       A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_8_614616887.dbf

125     1    9       A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_9_614616887.dbf

185     1    10      A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_10_614616887.dbf

221     1    11      A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_11_614616887.dbf

262     1    12      A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_12_614616887.dbf

263     1    13      A 15-FEB-07
        Name: /disk1/oradata/prod/arch/archive1_13_614616887.dbf
```

Example 2–98 Listing Database Incarnations

This example lists all database incarnations recorded in the recovery catalog.

```
RMAN> LIST INCARNATION;

List of Database Incarnations
DB Key  Inc Key DB Name  DB ID            STATUS  Reset SCN  Reset Time
------- ------- -------- ---------------- --- ---------- ----------
78      94      PROD     1619073740       PARENT  1          14-FEB-07
78      79      PROD     1619073740       CURRENT 388003     15-FEB-07
```

Example 2–99 Listing Failures

This example lists all failures regardless of their priority. If you do not specify `ALL`, then `LIST FAILURE` output does not include failures with `LOW` priority.

```
RMAN> LIST FAILURE ALL;

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

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

PRINT SCRIPT

Purpose

Use the `PRINT SCRIPT` command to print a local or global stored script to standard output or to a file.

Prerequisites

Execute `PRINT SCRIPT` only at the RMAN prompt. RMAN must be connected to a target database and a recovery catalog. The recovery catalog database must be open.

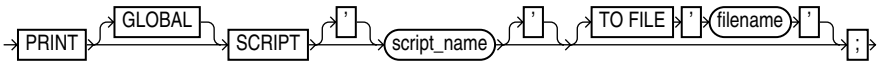
If the specified script is a local script, then RMAN must be connected to the target database that it was connected to when you created or replaced the script.

Usage Notes

If `GLOBAL` is not specified, then RMAN looks for a local or global script *`script_name`* to print. If a local script is found, then it is printed. If no local script is found, but a global script *`script_name`* is found, then the global script is printed.

Syntax

***printScript*::=**



Semantics

Syntax Element	Description
`GLOBAL`	Identifies the script as global. If `GLOBAL` is specified, then a global script with the specified name must already exist in the recovery catalog. Otherwise, RMAN returns error `RMAN-06004`. **See Also:** "Usage Notes" on page 2-105 for an explanation of the difference between global and local scripts
`script_name`	Specifies the name of the script to print. Quotes must be used around the script name when the name contains either spaces or reserved words.
`TO FILE '`*`filename`*`'`	Sends output to the specified file instead of standard output.

Examples

Example 2–100 Printing a Script to a File

This example prints a script to the file `/tmp/global_backup_db.rman`:

```
RMAN> PRINT GLOBAL SCRIPT global_backup_db TO FILE "/tmp/global_backup_db.rman";
```

Example 2–101 Printing a Script to the Screen

This example prints a stored script to standard output (and includes sample output):

```
RMAN> PRINT SCRIPT backup_whole;

printing stored script: backup_whole
```

```
{
    BACKUP
      INCREMENTAL LEVEL 0 TAG backup_whole
      FORMAT "/disk2/backup/%U"
      DATABASE PLUS ARCHIVELOG;
}
```

QUIT

Purpose

Use the `QUIT` command to shut down the Recovery Manager utility. This command is functionally equivalent to the `EXIT` command.

Prerequisites

Execute only at the RMAN prompt.

Syntax

***quit*::=**

Example

Example 2–102 Quitting RMAN

This example terminates RMAN (sample output included):

```
RMAN> QUIT


Recovery Manager complete.
```

3

RMAN Commands: RECOVER to VALIDATE

This chapter describes RMAN commands in alphabetical order. For a summary of the RMAN commands and command-line options, refer to "Summary of RMAN Commands" on page 1-6.

RECOVER

Purpose

Use the RECOVER command to perform one of the following distinct tasks:

- Perform complete recovery of the whole database or one or more restored datafiles
- Perform point-in-time recovery of a database (DBPITR) or tablespace (TSPITR)
- Apply incremental backups to a datafile image copy (*not* a restored datafile) to roll it forward in time
- Recover a corrupt data block or set of data blocks within a datafile

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to recover datafiles

Prerequisites

All redo or incremental changes required for the recovery must exist on disk or in SBT. If RMAN needs to restore incremental backups or archived redo log files during recovery, then you must either have automatic channels configured or manually allocate channels of the same type that created these backups.

If you perform media recovery on an encrypted tablespace, then the Oracle wallet must be open when performing media recovery of this tablespace. See *Oracle Database Administrator's Guide* to learn about encrypted tablespaces.

Prerequisites Specific to RECOVER BLOCK

The following prerequisites apply to RECOVER BLOCK:

- The target database must run in ARCHIVELOG mode and be open or mounted with a current control file.
- RMAN can only recover blocks marked media corrupt. The V$DATABASE_BLOCK_CORRUPTION view indicates which blocks in a file were marked corrupt since the most recent BACKUP or BACKUP ... VALIDATE command was run against the file.
- The backups of the datafiles containing the corrupt blocks must be full backups and not proxy backups. If only proxy backups exist, then you can restore them to a nondefault location on disk, in which case RMAN considers them datafile copies. You can then use the datafile copies for block media recovery.
- RMAN can use only archived redo log files for recovery. Block media recovery cannot survive a missing or inaccessible log, although it can sometimes survive missing or inaccessible records (see *Oracle Database Backup and Recovery User's Guide*).
- For RMAN to be able to search the flashback logs for good copies of corrupt blocks, Flashback Database must be enabled on the target database.
- For RMAN to be able to search a standby database for good copies of corrupt blocks, the target database must be associated with a physical standby database in a Data Guard environment. In addition, the physical standby database must be open read-only in managed recovery.

> **Note:** An active Data Guard license is required for this operation.

Usage Notes

By default, RMAN performs complete recovery. For point-in-time recovery, the best practice is to enter a SET UNTIL command before both the RESTORE and RECOVER commands in a RUN command so that the UNTIL time applies to both commands. If you run SET UNTIL after restoring the database, then you may not be able to recover the database to the target time because the restored files may have timestamps later than the target time.

> **Note:** You must open the database with the RESETLOGS option after incomplete recovery or recovery with a backup control file.

Incremental Backups and Archived Redo Log Files

With the exception of RECOVER BLOCK, RMAN can use both incremental backups and archived redo log files for recovery. RMAN uses the following search order:

1. Incremental backup sets on disk or tape
2. Archived redo log files on disk
3. Archived redo log backups on disk
4. Archived redo log backup sets on tape

When RMAN chooses a destination to restore archived redo log files, it uses the following order of precedence:

1. SET ARCHIVELOG DESTINATION
2. The LOG_ARCHIVE_DEST_*n* parameter whose value is set to LOCATION=USE_DB_RECOVERY_FILE_DEST
3. LOG_ARCHIVE_DEST_1

RMAN can apply incremental backups to datafiles that were not restored from an incremental backup. If overlapping levels of incremental backup exist, then RMAN automatically chooses the level covering the longest period of time.

Recovery Through RESETLOGS

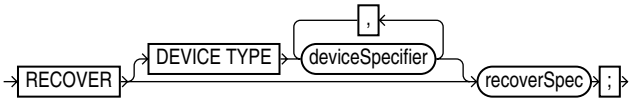
You must RESTORE datafiles before you can recover them. RMAN can recover through RESETLOGS operations transparently if the datafiles to be recovered are from a parent incarnation. If required, the RECOVER command can also restore and apply archived redo log files and incremental backups from previous database incarnations, even if those logs were generated in previous releases of Oracle Database.

When recovering through an OPEN RESETLOGS, ensure that you have all logs needed for recovery. In a previous database incarnation, you must have the logs from the time of the backup until the SCN that is 1 less than the RESETLOGS SCN. The incarnation table must have a complete history of RESETLOGS operations from the creation time of the database backup. If the complete metadata is not found in V$DATABASE_INCARNATION, then you can re-create this metadata by using CATALOG for the archived redo log files from the missing incarnations.

> **See Also:** RESTORE on page 3-37 command for explanation of the default location for restoring archived redo log files. Note that RMAN automatically specifies the MAXSIZE option when staging logs in the fast recovery area.

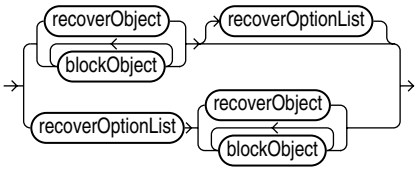
Syntax

recover::=



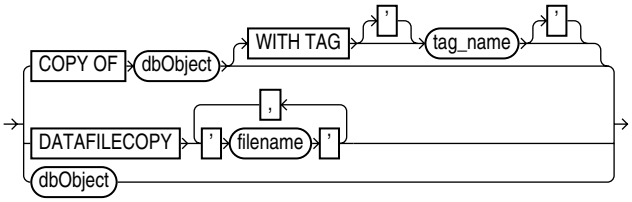
(*deviceSpecifier*::= on page 4-16, *recoverObject*::= on page 3-4, *recoverOptionList*::= on page 3-5)

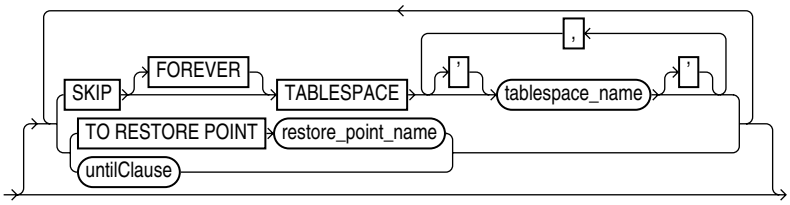
recoverSpec::=



(*recoverObject*::= on page 3-4, *blockObject*::= on page 3-5, *recoverOptionList*::= on page 3-5)

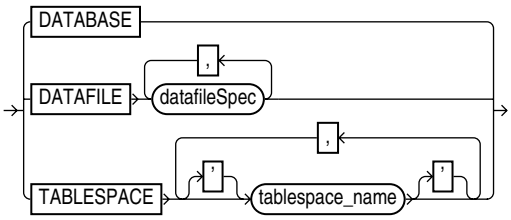
recoverObject::=





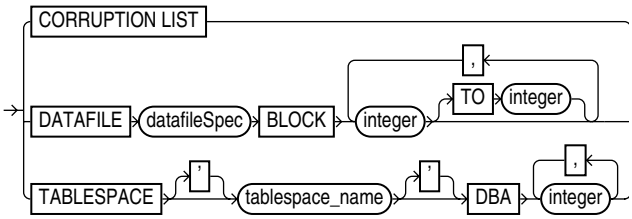
(*dbObject*::= on page 3-4, *blockObject*::= on page 3-5, *untilClause*::= on page 4-42)

dbObject::=



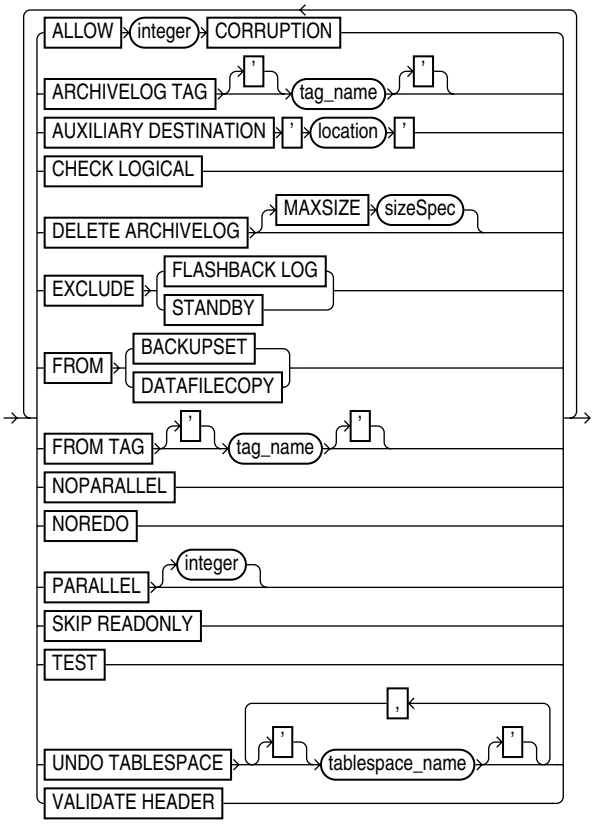
(*datafileSpec*::= on page 4-15)

blockObject::=

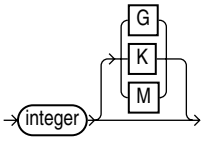


(*datafileSpec::=* on page 4-15)

recoverOptionList::=



sizeSpec::=



Semantics

recover

Syntax Element	Description
DEVICE TYPE *deviceSpecifier*	Allocates automatic channels for the specified device type only. For example, if you configure automatic disk and tape channels, and if you issue RECOVER DEVICE TYPE DISK, then RMAN allocates only disk channels. You must have already configured a device type with the CONFIGURE DEVICE TYPE command (except for DISK, which is preconfigured) before specifying the DEVICE TYPE option. **Note:** You cannot manually allocate channels and then run RECOVER DEVICE TYPE. **See Also:** *deviceSpecifier* on page 4-16
recoverSpec	Specifies the type of object being recovered.

recoverSpec

Syntax Element	Description
recoverObject	Specifies the type of object being recovered.
blockObject	Specifies the blocks to be recovered with block media recovery.
recoverOptionList	Specifies recovery options.

recoverObject

This subclause specifies which files to recover. Refer to *recoverObject*::= on page 3-4 for the syntax diagram.

Syntax Element	Description
COPY OF *dbObject*	Applies incremental backups to the specified image copy to roll it forward to any time equal to or before the most recent incremental backup of the file. The existing image copy is overwritten and remains in a fuzzy state during the recovery. RMAN makes an autobackup after recovering the image copy. This command updates a datafile copy and is *not* a media recovery of a current database file. This command is meant to be used in conjunction with the BACKUP ... FOR RECOVER OF COPY syntax to implement a strategy using incrementally updated backups. The following requirements must be met: ▪ At least one copy of each datafile that you are recovering must exist. ▪ Incremental backups taken after the image copy that you are recovering must exist. RMAN selects one suitable copy if there are multiple possible copies to which the incremental backups can be applied to carry out the operation. **Note:** RMAN issues a warning (not an error) if it cannot recover to the specified time because no incremental backups are available.
WITH TAG *tag_name*	Specifies a tag name to identify the image copy to be rolled forward.
DATAFILECOPY '*filename*'	Applies incremental backups to the specified datafile image copy (see Example 3–4). Refer to description of RECOVER COPY OF.
dbObject	Specifies the data blocks that require recovery. **See Also:** *dbObject* on page 3-7

Syntax Element	Description
SKIP	Takes the datafiles in the specified tablespaces offline before starting media recovery. These files are left offline after the media recovery is complete. This option is useful for avoiding recovery of tablespaces containing only temporary data or for postponing recovery of some tablespaces.
FOREVER	Takes the datafiles offline with the DROP option (see Example 3–3 on page 3-12). Use SKIP FOREVER TABLESPACE when you intend to drop the specified tablespaces after opening the database with the RESETLOGS option. **Note:** If you perform incomplete recovery, then SKIP requires the FOREVER option.
TABLESPACE *tablespace_name*	Specifies the name of the tablespace to take offline.
TO RESTORE POINT *restore_point_name*	Specifies a restore point for termination of the RECOVER command, with the SCN at which the restore point was created as the upper, inclusive limit. Because the limit is inclusive, RMAN selects only files that can be used to recover up to *and including* the SCN corresponding to the restore point.
untilClause	Specifies a past time, SCN, or log sequence number for termination of the RECOVER command. When used with one or more tablespaces, the clause indicates a tablespace point-in-time recovery (TSPITR) operation for the named tablespaces. The clause cannot be used with RECOVER DATAFILE. It should not be used for RECOVER DATABASE (see "Usage Notes" on page 3-3 for details). After database point-in-time recovery (DBPITR), you must open the database with the RESETLOGS option. **See Also:** untilClause on page 4-42

dbObject

This subclause specifies whether to recover the database or a subset of the database. Refer to *dbObject::=* on page 3-4 for the syntax diagram.

Syntax Element	Description
DATABASE	Specifies that the entire database is to be recovered (see Example 3–3 on page 3-12). The database must be mounted but not open. By default, the RECOVER DATABASE command does not recover files that are offline normal at the point in time to which the files are being recovered. RMAN omits offline normal files with no further checking. When recovering after the loss of control files, RMAN automatically updates the control file to point to the actual location of the datafiles on disk (see Example 3–5 on page 3-14). **Note:** If the RMAN encounters redo for adding a datafile, then RMAN automatically creates a new datafile unless the tablespace containing the added datafile is skipped during recovery. This situation can arise when a backup control file is restored prior to recovery and the backup control file does not contain a record of the recently-added datafile.

Syntax Element	Description
DATAFILE *datafileSpec*	Specifies a list of one or more datafiles to recover by either filename or absolute datafile number. The target database must be mounted or open. If the database is open, then the datafiles to be recovered must be offline. If you are not using a recovery catalog, then the filename must be the name of the datafile as recorded 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, even if the name in the control file has been updated more recently. For example, assume that a datafile was renamed in the control file, but the instance fails before you resynchronize the catalog. Specify the old name of the datafile in the RECOVER command because this is the name recorded in the catalog. **Note:** You cannot arbitrarily recover individual datafiles to different points in time, although you can recover the whole database to a single point in time or recover wholly contained tablespaces to a point in time different from the rest of the database (TSPITR). For more information on TSPITR, see the procedure described in *Oracle Database Backup and Recovery User's Guide*. **See Also:** *datafileSpec* on page 4-15
TABLESPACE *tablespace_name*	Specifies a list of one or more tablespaces to recover (see Example 3–1 on page 3-12 and Example 3–2 on page 3-12). The target database must be mounted or open. If the database is open, then the tablespaces to be recovered must be offline. **Note:** If the RMAN encounters redo for adding a datafile, then RMAN automatically creates a new datafile. This situation can arise when a backup control file is restored prior to recovery and the backup control file does not contain a record of the recently-added datafile.

blockObject

This subclause specifies the data blocks that require recovery. Refer to *blockObject::=* on page 3-5 for the syntax diagram. Refer to "Prerequisites Specific to RECOVER BLOCK" on page 3-2 for prerequisites specific to block media recovery.

You can either use RECOVER CORRUPTION LIST to recover all blocks reported in the V$DATABASE_BLOCK_CORRUPTION view, or specify the datafile number and block number or the tablespace and data block address (DBA). RMAN can only perform complete recovery of individual blocks.

By default, if Flashback Database is enabled, then RMAN searches the flashback logs for good copies of corrupt blocks. By default, if the target database exists in a Data Guard environment, then RECOVER BLOCK command can automatically retrieve blocks from a physical standby database to a primary database and vice-versa.

Syntax Element	Description
CORRUPTION LIST	Recovers all physically corrupt blocks listed in the V$DATABASE_BLOCK_CORRUPTION view. Block media recovery may not be able to repair all listed logically corrupt blocks. In these cases, alternate recovery methods, such as tablespace point-in-time recovery, or dropping and recreating the affected objects, may repair the corruption. The V$DATABASE_BLOCK_CORRUPTION view displays blocks marked corrupt by Oracle Database components such as RMAN commands, ANALYZE, dbv, SQL queries, and so on. In short, any process that encounters an ORA-1578 error records the block corruption in this view. The following types of corruption result in rows added to this view: ■ Physical corruption (sometimes called media corruption). The database does not recognize the block at all: the checksum is invalid, the block contains all zeros, or the header and footer of the block do not match. ■ Logical corruption. The block has a valid checksum, the header and footer match, and so forth, but the contents are logically inconsistent. The view does not record corruptions that can be detected by validating relationships between blocks and segments, but cannot be detected by a check of an individual block. **Note:** Any RMAN command that fixes or detects that a block is repaired updates V$DATABASE_BLOCK_CORRUPTION. For example, RMAN updates the repository at end of successful block media recovery. If a BACKUP, RESTORE, or VALIDATE command detects that a block is no longer corrupted, then it removes the repaired block from the view.
DATAFILE *datafileSpec* BLOCK *integer* TO *integer*	Recovers an individual data block or set of data blocks within a datafile. It can copy blocks from either a standby or a primary database. Note that the TO range is inclusive, so that BLOCK 10 TO BLOCK 20 includes both block 10 and block 20. Block media recovery is useful when the data loss or corruption applies to a small number of blocks rather than to an entire datafile. Typically, block corruption is reported in error messages in trace files or by the ADVISE FAILURE command. Block-level data loss usually results from: ■ I/O errors causing minor data loss ■ Memory corruptions that get written to disk If you do not specify an option from *recoverOptionList*, and if Flashback Database is enabled on the database, then RECOVER BLOCK first searches the flashback logs and then the backups for a good version of the block to restore. Blocks marked media corrupt are not accessible until recovery completes. **Note:** You can only perform complete recovery of individual blocks. In other words, you cannot stop recovery before all redo has been applied to the block. **See Also:** *datafileSpec* on page 4-15
TABLESPACE *tablespace_name* DBA *integer*	Specifies the tablespace name or number containing the corrupt blocks and the data block address (DBA) of the corrupt block. You can only perform block media recovery on corrupt blocks. **Note:** The datafile header block (block 1) cannot be recovered.

recoverOptionList

This subclause specifies various recovery options. Refer to *recoverOptionList::=* on page 3-5 for the syntax diagram.

Syntax Element	Description
ALLOW *integer* CORRUPTION	Specifies the number of corrupt blocks that can be tolerated while allowing recovery to proceed.You can set this parameter in case of redo log corruption.

Syntax Element	Description
ARCHIVELOG TAG *tag_name*	Specifies the tag for an archived log backup to be used during recovery. Tag names are not case sensitive and display in all uppercase. If the tagged backup does not contain all the necessary archived redo log files for recovery, then RMAN uses logs or incremental backups as needed from whatever is available.
AUXILIARY DESTINATION '*location*'	Specifies a location where auxiliary set datafiles, control files, and online redo logs are created during TSPITR if another location for an individual file is not explicitly specified. If you do not specify AUXILIARY DESTINATION for a TSPITR, then you must specify the naming of individual auxiliary set datafiles, control files, and online redo logs before executing RECOVER TABLESPACE with the UNTIL clause. Otherwise, TSPITR fails. **See also:** The chapter on TSPITR in *Oracle Database Backup and Recovery User's Guide* for more details about the auxiliary destination
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. The SET MAXCORRUPT setting represents the total number of physical and logical corruptions permitted on a file. By default, MAXCORRUPT is 0, so that if any corrupt blocks exist, media recovery fails. If recovery including corrupt blocks is permissible, then set MAXCORRUPT to the smallest number of corrupt blocks that causes media recovery to fail. For example, to tolerate one corrupt block, set MAXCORRUPT to 1. If the total number of physical and logical corruptions detected for a file is less than its MAXCORRUPT setting, then the RMAN command completes and the database populates V$DATABASE_BLOCK_CORRUPTION with corrupt block ranges. Otherwise, the command terminates without populating V$DATABASE_BLOCK_CORRUPTION.
DELETE ARCHIVELOG	Deletes archived redo log files restored from backups or copies that are no longer needed. RMAN does not delete archived redo log files that were already on disk before the RESTORE command started. If you do not specify MAXSIZE, then RMAN deletes restored archived redo log files as they are applied. **Note:** If archived redo log files are restored to the fast recovery area, then the DELETE ARCHIVELOG option is enabled by default.
MAXSIZE *sizeSpec*	Does not use more than *sizeSpec* amount of disk space for restored archived redo log files. If recovery requires the restore of a log larger than the MAXSIZE value, then RMAN reports an error indicating that you should increase the MAXSIZE value. If MAXSIZE is smaller than the backup set containing the logs, then RMAN must read the backup set more than once to extract the logs. In this situation, RMAN issues a warning that MAXSIZE should be increased.
EXCLUDE FLASHBACK LOG	Specifies that the flashback logs should not be searched for blocks to restore. By default, RMAN searches the flashback logs if Flashback Database is enabled.
EXCLUDE STANDBY	Specifies that a physical standby database should not be searched for blocks to restore. By default, in a Data Guard environment RMAN searches the blocks from a physical standby database.
FROM BACKUPSET	Specifies that only backup sets should be restored.
FROM DATAFILECOPY	Specifies that only datafile image copies should be restored.
FROM TAG *tag_name*	Specifies the tag for an incremental backup to be used during recovery. If the tagged backup does not contain all the necessary incrementals for recovery, then RMAN uses logs or incremental backups as needed from whatever is available. Tag names are not case sensitive and display in all uppercase. **See Also:** BACKUP on page 2-19 to learn how a tag can be applied to an individual copy of a duplexed backup set, and to learn about the default filename format for backup tags

Syntax Element	Description
NOPARALLEL	Specifies that media recovery should not be performed in parallel. Parallel execution is the default for RECOVER (see the description of the RECOVER ... PARALLEL option).
NOREDO	Suppresses the application of redo logs during recovery. Only incremental backups are applied. One use of this option is to use incremental backups to update full backups of NOARCHIVELOG databases (see Example 3–6 on page 3-14). The NOREDO options is required if redo logs are not available. If you do not specify NOREDO when recovering a NOARCHIVELOG database, then RMAN ends recovery and issues an error. **Note:** Incremental backups of NOARCHIVELOG databases can only be taken after a consistent shutdown. Another use is to update standby or duplicate databases. Incremental backups created with the BACKUP INCREMENTAL FROM SCN command can be applied at a standby or duplicate database. The standby database procedure is described in *Oracle Data Guard Concepts and Administration*.
PARALLEL	Specifies parallel recovery (default). By default, the database uses parallel media recovery to improve performance of the roll forward phase of media recovery. To override the default behavior of performing parallel recovery, use the RECOVER with the NOPARALLEL option, or RECOVER PARALLEL 0. In parallel media recovery, the database uses a "division of labor" approach to allocate different processes to different data blocks while rolling forward, thereby making the operation more efficient. The number of processes is derived from the CPU_COUNT initialization parameter, which by default equals the number of CPUs on the system. For example, if parallel recovery is performed on a system where CPU_COUNT is 4, and only one datafile is recovered, then four spawned processes read blocks from the datafile and apply redo. Typically, recovery is I/O-bound on reads from and writes to data blocks. Parallelism at the block level may only help recovery performance if it increases total I/Os, for example, by bypassing operating system restrictions on asynchronous I/Os. Systems with efficient asynchronous I/O see little benefit from parallel media recovery. **Note:** The RECOVERY_PARALLELISM initialization parameter controls instance or crash recovery only. Media recovery is not affected by the value used for RECOVERY_PARALLELISM. **See Also:** The description of the PARALLEL clause in the discussion of CREATE TABLE in *Oracle Database SQL Language Reference*
integer	Specifies the *integer* degree of parallelism. Each parallel thread may use one or two parallel execution servers. Typically, it is not necessary for you to specify the degree of parallelism.
SKIP READONLY	Omits read-only files from the recovery.
TEST	Initiates a trial recovery. A trial recovery is useful if a normal recovery procedure has encountered a problem. It enables the database to look ahead into the redo stream to detect possible problems. The trial recovery applies redo in a way similar to normal recovery, but it does not write changes to disk and it rolls back its changes at the end of the trial recovery. **Note:** You can use this clause only if you have restored a backup taken since the last RESETLOGS operation. Otherwise, the database returns an error.

Syntax Element	Description
UNDO TABLESPACE *tablespace_name*	Specifies a list of tablespaces with undo segments at the target time. Only for use with RECOVER TABLESPACE. During TSPITR, RMAN needs information about which tablespaces had undo segments at the TSPITR target time. This information is usually available in the recovery catalog, if one is used. If there is no recovery catalog, or if the information is not found in the recovery catalog, then RMAN proceeds assuming that the set of tablespaces with undo segments at the target time is the same as the set of tablespaces with undo segments at the present time. If this assumption is not correct, TSPITR fails with an error. In such a case, you can use UNDO TABLESPACE.
VALIDATE HEADER	Reports and validates—but does not restore—the backups that RMAN could use to restore files needed for the recovery. When you run RECOVER with VALIDATE HEADER, RMAN performs the same functions as when you specify the RESTORE . . . PREVIEW option. However, in addition to listing the files needed for restore and recovery, RMAN validates the backup file headers to determine whether the files on disk or in the media management catalog correspond to the metadata in the RMAN repository. **See Also:** The description of the RESTORE . . . PREVIEW option

Examples

Example 3–1 Recovering a Tablespace in an Open Database

Assume that the disk containing the datafiles for tablespace users becomes unavailable because of a hardware error, but is repaired after a few minutes. This example takes tablespace users offline, uses automatic channels to restore the datafiles to their default location and recover them (deleting the logs that it restored from tape), then brings the tablespace back online.

```
SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
RESTORE TABLESPACE users;
RECOVER TABLESPACE users DELETE ARCHIVELOG MAXSIZE 2M;
SQL "ALTER TABLESPACE users ONLINE";
```

Example 3–2 Recovering Datafiles Restored to New Locations

This example uses the preconfigured disk channel and manually allocates one media management channel to use datafile copies on disk and backups on tape, and restores one of the datafiles in tablespace users to a different location.

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt;
  SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
  SET NEWNAME FOR DATAFILE '/disk1/oradata/prod/users01.dbf'
    TO '/disk2/users01.dbf';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;
  RECOVER TABLESPACE users;
  SQL "ALTER TABLESPACE users ONLINE";
}
```

Example 3–3 Performing DBPITR with a Backup Control File and Recovery Catalog

Assume that all datafiles and control files as well as archived redo log 58 were lost due to a disk failure. Also assume that you do not have incremental backups. You need to recover the database with available archived redo log files. You do not need to restore

tablespace `tools` because it has been read-only since before the most recent backup. After connecting RMAN to the target database and recovery catalog, issue the following commands:

```
STARTUP FORCE NOMOUNT;
RUN
{
  SET UNTIL SEQUENCE 40 THREAD 1;  # Recover database until log sequence 40
  RESTORE CONTROLFILE;
  ALTER DATABASE MOUNT;
  RESTORE DATABASE SKIP TABLESPACE temp;
  RECOVER DATABASE SKIP TABLESPACE temp;
}
ALTER DATABASE OPEN RESETLOGS;
```

Note that RMAN automatically skips the restore and recovery of datafile 8, which is the datafile in the read-only tablespace. The following portion of sample output indicates the skip:

```
using channel ORA_DISK_1
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=104 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup

skipping datafile 8; already restored to file /disk1/oradata/prod/tools01.dbf
channel ORA_DISK_1: starting datafile backup set restore
.
.
.
Finished restore at 19-FEB-07

Starting recover at 19-FEB-07
using channel ORA_DISK_1
using channel ORA_SBT_TAPE_1
datafile 8 not processed because file is read-only
```

Example 3–4 Incrementally Updating Backups

By incrementally updating backups, you can avoid the overhead of making full image copy backups of datafiles, while also minimizing time required for media recovery of your database. This example enables you to recover to any SCN within the previous week, but enables you to avoid having to apply more than one day of redo.

Assume you run the following script daily. On first execution, the script creates an image copy backup of the database on disk with the specified tag. On the second through the seventh executions, the script creates a level 1 differential backup of the database. On the eighth and all subsequent executions, RMAN applies the level 1 incremental to the datafile copy made 7 days ago and then makes a new level 1 backup with the changes from the previous day.

```
RUN
{
  RECOVER COPY OF DATABASE
    WITH TAG 'incr_update'
    UNTIL TIME 'SYSDATE - 7';
  BACKUP
    INCREMENTAL LEVEL 1
    FOR RECOVER OF COPY WITH TAG 'incr_update'
    DATABASE;
}
```

Example 3–5 Recovery from Loss of a Control File on a Standby Database

Assume that the standby database dgprod3 control files are lost because of a media failure. The primary and standby database share SBT storage. A backup of the primary database control file exists on tape.

You start the RMAN client and connect to dgprod3 as TARGET and connect to the recovery catalog. The following RMAN commands restore a control file that is usable by the standby database, update the filenames to existing files on disk, and recover the standby database:

```
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
RECOVER DATABASE;
```

You can then start redo apply on the standby database.

Example 3–6 Recovering a NOARCHIVELOG Database

You can perform limited recovery of changes to a database running in NOARCHIVELOG mode by applying incremental backups. The incremental backups must be consistent, like all backups of a database run in NOARCHIVELOG mode, so you cannot back up the database when it is open.

Assume that you run database prod in NOARCHIVELOG mode with a recovery catalog. You shut down the database consistently and make a level 0 backup of database prod to tape on Sunday afternoon. You shut down the database consistently and make a level 1 differential incremental backup to tape at 3:00 a.m. on Wednesday and Friday.

On Saturday, a media failure destroys half of the datafiles as well as the online redo logs. Because the online logs are lost, you must specify the NOREDO option in the RECOVER command. Otherwise, RMAN searches for the redo logs after applying the Friday incremental backup and issues an error message when it does not find them.

After connecting RMAN to prod and the catalog database, recover as follows:

```
STARTUP FORCE NOMOUNT;
RESTORE CONTROLFILE;      # restore control file from consistent backup
ALTER DATABASE MOUNT;
RESTORE DATABASE;         # restore datafiles from consistent backup
RECOVER DATABASE NOREDO;  # specify NOREDO because online redo logs are lost
ALTER DATABASE OPEN RESETLOGS;
```

The recovered database reflects only changes up through the time of the Friday incremental backup. Because there are no archived redo log files, there is no way to recover changes made after the incremental backup.

Example 3–7 Recovering All Block Corruption in the Database

This example runs a backup validation to populate the V$DATABASE_BLOCK_CORRUPTION view, then recovers any corrupt blocks recorded in the view. Sample output is included for both commands.

```
RMAN> VALIDATE DATABASE;

Starting validate at 19-FEB-07
using channel ORA_DISK_1
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
.
.
.
```

```
List of Datafiles
=================
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
1    FAILED 0              4070         57600           555975
  File Name: /disk1/oradata/prod/system01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       1              41550
  Index      0              7677
  Other      0              4303
.
.
.
RMAN> RECOVER CORRUPTION LIST;

Starting recover at 19-FEB-07
using channel ORA_DISK_1
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=104 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
searching flashback logs for block images until SCN 547548
finished flashback log search, restored 1 blocks

starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished recover at 19-FEB-07
```

REGISTER DATABASE

Purpose

Use the `REGISTER DATABASE` command to register the target database in the recovery catalog so that RMAN can maintain its metadata. RMAN obtains all information it needs to register the target database from the target database itself.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* and `CREATE CATALOG` on page 2-102

Prerequisites

Execute this command only at the RMAN prompt. RMAN must be connected to a recovery catalog and a mounted or open target database. The database that you are registering must not be currently registered in the recovery catalog.

You can only register a target database with a DBID that is unique within the recovery catalog. Databases with the same name are permitted if the DBID values are different. The database that you are registering must not be a standby database.

Usage Notes

RMAN automatically registers a new standby database in the recovery catalog when the primary database for the standby database is already registered in the recovery catalog, and either of the following conditions is true:

- RMAN is connected to a database instance that has a `DB_UNIQUE_NAME` unknown to the recovery catalog.
- You execute the `CONFIGURE` `DB_UNIQUE_NAME` command for a database that is not known to the recovery catalog.

The `REGISTER DATABASE` command fails when RMAN detects duplicate DBIDs. This situation can arise when databases are created by copying files from an existing database rather than by using the `DUPLICATE` command. If this failure occurs, then you can change the DBID of the copied database with the DBNEWID utility and then retry the `REGISTER DATABASE` command.

If you open a database with the `RESETLOGS` option and later register this database in the recovery catalog, then the recovery 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.

> **Note:** If you are using RMAN with different target databases that have the same database name and DBID, then be careful to always specify the correct recovery catalog schema when invoking RMAN.

> **See Also:** *Oracle Database Utilities* to learn how to use the DBNEWID utility

Syntax

***register*::=**

Example

Example 3–8 Registering a Database

This example registers a new target database in the recovery catalog. Sample output is included.

```
RMAN> CONNECT TARGET /

connected to target database: PROD (DBID=1619241818)

RMAN> CONNECT CATALOG rman@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> REGISTER DATABASE;

database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

RELEASE CHANNEL

Purpose

Use the `RELEASE CHANNEL` command to release a normal or maintenance channel while maintaining a connection to a target database instance. A normal channel is allocated with `ALLOCATE CHANNEL`, whereas a maintenance channel is allocated with `ALLOCATE CHANNEL FOR MAINTENANCE`.

Prerequisites

To release a normal channel, use the syntax shown in the *release::=* diagram. Execute this form of `RELEASE CHANNEL` only within a `RUN` command and specify the channel name with the same identifier used in the `ALLOCATE CHANNEL` command.

To release a maintenance channel, use the syntax shown in the *releaseForMaint::=* diagram. Execute this form of `RELEASE CHANNEL` only at the RMAN prompt, not within a `RUN` command.

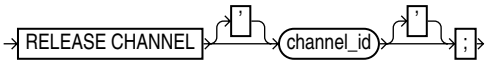
Usage Notes

Maintenance channels are unaffected by `ALLOCATE CHANNEL` and `RELEASE CHANNEL` commands issued within a `RUN` command.

Using `RELEASE CHANNEL` to release channels within `RUN` is optional, because RMAN automatically releases all normal channels when a `RUN` command terminates.

Syntax

***release*::=**



***releaseForMaint*::=**

Semantics

Syntax Element	Description
channel_id	Specifies the case-sensitive channel ID used in the `ALLOCATE CHANNEL` command (see Example 3–9).

Examples

Example 3–9 Releasing a Channel Allocated in a RUN Command

This example allocates an SBT channel named `ch1` with parameters for a set of tapes intended for daily backups, backs up the database, and then releases this channel. The example then allocates an SBT channel named `ch1` with parameters for a set of tapes intended for weekly backups, and makes another database backup:

```
RUN
{
```

```
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt
    PARMS='ENV=(OB_MEDIA_FAMILY=daily_bkp)';
  BACKUP DATABASE;
  RELEASE CHANNEL ch1;
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt
    PARMS='ENV=(OB_MEDIA_FAMILY=weekly_bkp)';
  BACKUP DATABASE;
}
```

Note that a `RELEASE CHANNEL` command is not necessary at the end of the `RUN` command because RMAN automatically releases channel `ch1`.

Example 3–10 Releasing a Maintenance Channel

This example shows the transcript of an RMAN session. The example allocates an SBT maintenance channel and then crosschecks and deletes backups on tape. After the SBT channel is released, RMAN uses the default disk channel to back up the database.

```
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;

allocated channel: ORA_MAINT_SBT_TAPE_1
channel ORA_MAINT_SBT_TAPE_1: SID=105 device type=SBT_TAPE
channel ORA_MAINT_SBT_TAPE_1: Oracle Secure Backup

RMAN> CROSSCHECK BACKUP;

crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=1jiah8ln_1_1 RECID=25 STAMP=615031479
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=1kiah8pk_1_1 RECID=26 STAMP=615031612
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=1niah973_1_1 RECID=28 STAMP=615032036
Crosschecked 3 objects

RMAN> DELETE BACKUP;

List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
------- ------- --- --- ----------- ----------- ----------
1333    1331    1   1   AVAILABLE   SBT_TAPE    1jiah8ln_1_1
1334    1332    1   1   AVAILABLE   SBT_TAPE    1kiah8pk_1_1
1427    1423    1   1   AVAILABLE   SBT_TAPE    1niah973_1_1

Do you really want to delete the above objects (enter YES or NO)? YES
deleted backup piece
backup piece handle=1jiah8ln_1_1 RECID=25 STAMP=615031479
deleted backup piece
backup piece handle=1kiah8pk_1_1 RECID=26 STAMP=615031612
deleted backup piece
backup piece handle=1niah973_1_1 RECID=28 STAMP=615032036
Deleted 3 objects

RMAN> RELEASE CHANNEL;

released channel: ORA_MAINT_SBT_TAPE_1

RMAN> BACKUP DATABASE;

Starting backup at 20-FEB-07
allocated channel: ORA_DISK_1
```

```
channel ORA_DISK_1: SID=105 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
```

REPAIR FAILURE

Purpose

Use the `REPAIR FAILURE` command to repair database failures identified by the Data Recovery Advisor.

The recommended workflow is to run `LIST FAILURE` to display failures, `ADVISE FAILURE` to display repair options, and `REPAIR FAILURE` to fix the failures.

Prerequisites

The target database instance must be started. The database must be a single-instance database and must not be a physical standby database.

Make sure that at most one RMAN session is running the `REPAIR FAILURE` command. The only exception is `REPAIR FAILURE ... PREVIEW`, which is permitted in concurrent RMAN sessions.

To perform an automated repair, the Data Recovery Advisor may require specific backups and archived redo log files. If the files needed for recovery are not available, then the recovery will not be possible.

Usage Notes

Repairs are consolidated whenever possible so that a single repair can fix multiple failures. Be advised that `REPAIR FAILURE` requires you to explicitly run `ADVISE FAILURE` in the current session in order to successfully repair each of the identified failures. You typically iterate through a `REPAIR` session with the following commands:

- `REPAIR FAILURE;`
- `LIST FAILURE;`
- `ADVISE FAILURE;`
- `REPAIR FAILURE;`

RMAN always verifies that failures are still relevant and automatically closes fixed failures. RMAN does not attempt to repair a failure that has already been fixed, nor does it repair a failure that is obsolete because new failures have been introduced since `ADVISE FAILURE` was run.

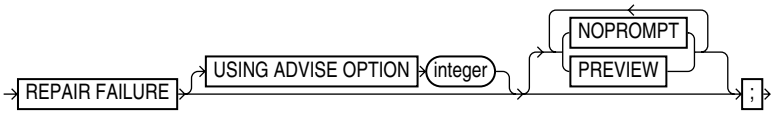
By default, `REPAIR FAILURE` prompts for confirmation before it begins executing. After executing a repair, RMAN reevaluates all existing failures on the chance that they may also have been fixed.

Oracle RAC and Data Recovery Advisor

If a data failure brings down all instances of an Oracle RAC database, then you can mount the database in single-instance mode and use Data Recovery Advisor to detect and repair control file, `SYSTEM` datafile, and dictionary failures. You can also initiate health checks to test other database components for data failures. This approach will not detect data failures that are local to other cluster instances, for example, an inaccessible datafile.

Syntax

***repair*::=**



Semantics

repair

Syntax Element	Description
`REPAIR FAILURE`	Repairs failures recorded in the Automated Diagnostic Repository. If you execute `REPAIR FAILURE` with no other command options, then RMAN uses the first repair option of the most recent `ADVISE FAILURE` command in the current session. The command performs an implicit `ADVISE FAILURE` if this command has not yet been executed in the current session.
`USING ADVISE OPTION` *`integer`*	Specifies a repair option by its option number (*not* its failure number). You can obtain repair option numbers from the `ADVISE FAILURE` command.
`NOPROMPT`	Suppresses the confirmation prompt. This is the default option if you run `REPAIR FAILURE` in a command file.
`PREVIEW`	Does not make any repairs and generates a script with all repair actions and comments. By default the script is displayed to standard output. You can use the `SPOOL` command to write the script to an editable file (see Example 3–12 on page 3-24).

Examples

Example 3–11 Repairing Failures

This example repairs all failures known to the Recovery Data Advisor. The example repairs two failures: missing datafiles and a datafile with corrupt blocks. After the recovery, RMAN asks whether it should open the database (user-entered text is in bold).

```
RMAN> LIST FAILURE;

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

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

RMAN> ADVISE FAILURE;

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

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

```
                                          contains one or more corrupt blocks

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

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

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

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

RMAN> REPAIR FAILURE;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_475549922.hm
contents of repair script:
   # restore and recover datafile
   sql 'alter database datafile 28 offline';
   restore datafile 28;
   recover datafile 28;
   sql 'alter database datafile 28 online';
   # block media recovery
   recover datafile 1 block 56416;

Do you really want to execute the above repair (enter YES or NO)? YES
executing repair script

sql statement: alter database datafile 28 offline

Starting restore at 23-APR-07
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00028 to /disk1/oradata/prod/users01.dbf
channel ORA_DISK_1: reading from backup piece
/disk2/PROD/backupset/2007_04_18/o1_mf_nnndf_TAG20070418T182042_32fjzd3z_.bkp
channel ORA_DISK_1: piece
handle=/disk2/PROD/backupset/2007_04_18/o1_mf_nnndf_TAG20070418T182042_32fjzd3z_.bkp
tag=TAG20070418T182042
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 23-APR-07

Starting recover at 23-APR-07
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:01

Finished recover at 23-APR-07

sql statement: alter database datafile 28 online
```

```
Starting recover at 23-APR-07
using channel ORA_DISK_1
searching flashback logs for block images until SCN 429690
finished flashback log search, restored 1 blocks

starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished recover at 23-APR-07
repair failure complete
```

Example 3–12 Previewing a Repair

The following example previews a repair of the first repair option of the most recent `ADVISE FAILURE` command in the current session. Note that the sample output for the `LIST FAILURE` and `ADVISE FAILURE` commands is not shown in the example.

```
RMAN> LIST FAILURE;
.
.
.
RMAN> ADVISE FAILURE;
.
.
.
RMAN> REPAIR FAILURE PREVIEW;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_3200987003.hm

contents of repair script:
   # block media recovery
   recover datafile 1 block 56416;
```

You can use `SPOOL` in conjunction with `REPAIR FAILURE ... PREVIEW` to write a repair script to a file. You can then edit this script and execute it manually. The following example spools a log a repair preview to `/tmp/repaircmd.dat`.

```
RMAN> SPOOL LOG TO '/tmp/repaircmd.dat';
RMAN> REPAIR FAILURE PREVIEW;
RMAN> SPOOL LOG OFF;
```

REPLACE SCRIPT

Purpose

Use the `REPLACE SCRIPT` command to replace an existing script stored in the recovery catalog. If the script does not exist, then `REPLACE SCRIPT` creates it.

> **See Also:** `CREATE SCRIPT` on page 2-105 to learn how to create stored scripts

Prerequisites

Execute `REPLACE SCRIPT` only at the RMAN prompt. RMAN must be connected to a target database and a recovery catalog. The recovery catalog database must be open.

If you are replacing a local script, then you must be connected to the target database that you connected to when you created the script.

Substitution Variables in Stored Scripts

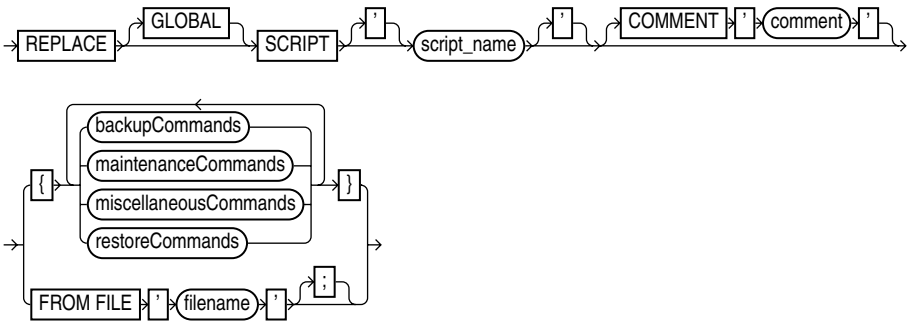
RMAN supports the use of substitution variables in a stored script. `&1` indicates where to place the first value, `&2` indicate where to place the second value, and so on. Special characters must be quoted.

The substitution variable syntax is `&`*`integer`* followed by an optional period, for example, `&1.3`. The optional period is part of the variable and replaced with the value, thus enabling the substitution text to be immediately followed by another integer. For example, if you pass the value `mybackup` to a command file that contains the substitution variable `&1.3`, then the result of the substitution is `mybackup3`. Note that to create the result `mybackup.3`, you would use the syntax `&1..3`.

When you create a stored script with substitution variables, you must provide example values at create time. You can provide these values with the `USING` clause when starting RMAN (see `RMAN` on page 3-59) or enter them when prompted (see Example 2–62 on page 2-107).

Syntax

***replaceScript*::=**



(*backupCommands::=* on page 3-65, *maintenanceCommands::=* on page 3-65, *miscellaneousCommands::=* on page 3-66, *restoreCommands::=* on page 3-66)

Semantics

Syntax Element	Description
GLOBAL	Identifies the script as global. **Note:** A virtual private catalog has read-only access to global scripts. Creating or updating global scripts must be done while RMAN is connected to the base recovery catalog. **See Also:** "Usage Notes" on page 2-105 for an explanation of the difference between global and local scripts
SCRIPT *script_name*	Identifies the local or global script being replaced.
COMMENT '*comment*'	Associates an explanatory comment with the stored script in the recovery catalog. The comment is shown in the output of LIST SCRIPT NAMES.
backupCommands *maintenanceCommands* *restoreCommands* *miscellaneousCommands*	Specifies commands to include in the stored script. The commands allowable within the brackets of the REPLACE SCRIPT '*script_name* ' {...} command are the same commands supported within a RUN block. Any command that is valid within a RUN command is permitted in the stored script. The following commands are not permitted within stored scripts: RUN, @ (at sign), and @@ (double at sign).
FROM FILE '*filename*'	Reads the sequence of commands to define the script from the specified file. The file should look like the body of a valid stored script. The first line of the file must be a left brace ({) and the last line must contain a right brace (}). The RMAN commands in the file must be valid in a stored script.

Example

Example 3–13 Replacing a Recovery Catalog Script

Assume that you start the RMAN client and connect to database prod as TARGET and then connect to a recovery catalog. You use CREATE SCRIPT to create a global script named backup_db as follows:

```
CREATE GLOBAL SCRIPT backup_db
COMMENT "back up any database from the recovery catalog, with logs"
{
    BACKUP DATABASE;
}
```

You then use LIST SCRIPT NAMES to list all scripts known to the recovery catalog:

```
RMAN> LIST SCRIPT NAMES;

List of Stored Scripts in Recovery Catalog


    Global Scripts


       Script Name
       Description
       ---------------------------------------------------------------------
       backup_db
       back up any database from the recovery catalog, with logs
```

You then run the following REPLACE SCRIPT command with the intention of editing the backup_db global script:

```
RMAN> REPLACE SCRIPT backup_db { BACKUP DATABASE PLUS ARCHIVELOG; }
```

```
replaced script backup_db
```

Because you did not specify the GLOBAL keyword, RMAN creates a local script named backup_db in addition to the global script named backup_db. LIST SCRIPT NAMES shows both the global and local script recorded in the recovery catalog:

```
RMAN> LIST SCRIPT NAMES;

List of Stored Scripts in Recovery Catalog


    Scripts of Target Database PROD

       Script Name
       Description
       ---------------------------------------------------------------------
       backup_db


    Global Scripts


       Script Name
       Description
       ---------------------------------------------------------------------
       backup_db
       back up any database from the recovery catalog, with logs
```

You can then delete the local script named backup_db with DELETE SCRIPT and replace the global script as follows:

```
RMAN> DELETE SCRIPT backup_db;

deleted script: backup_db

RMAN> REPLACE GLOBAL SCRIPT backup_db { BACKUP DATABASE PLUS ARCHIVELOG; }

replaced global script backup_db
```

The LIST SCRIPT NAMES command now shows that only one script named backup_db exists in the catalog:

```
RMAN> LIST SCRIPT NAMES;

List of Stored Scripts in Recovery Catalog

    Global Scripts

       Script Name
       Description
       ---------------------------------------------------------------------
       backup_db
```

The PRINT SCRIPT command confirms the changes to the global script:

```
RMAN> PRINT GLOBAL SCRIPT backup_db;

printing stored global script: backup_db
 { BACKUP DATABASE PLUS ARCHIVELOG; }
```

REPORT

Purpose

Use the `REPORT` command to perform detailed analyses of the RMAN repository. RMAN writes the report to standard output or the message log file.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to create RMAN reports

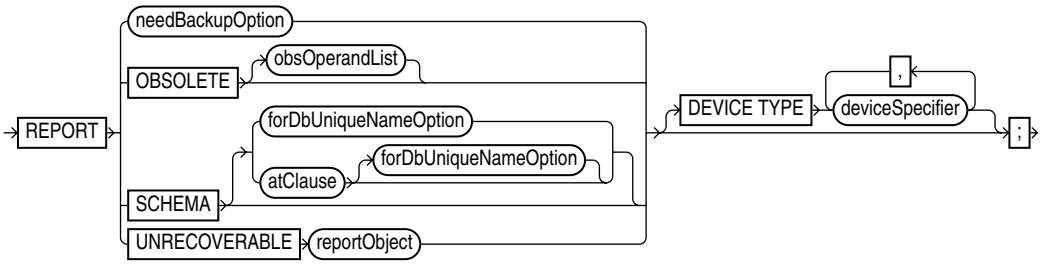
Prerequisites

Execute this command only at the RMAN prompt. Either of the following conditions must be met:

- RMAN must be connected to a target database.
- RMAN must be connected to a recovery catalog and `SET DBID` must have been run.

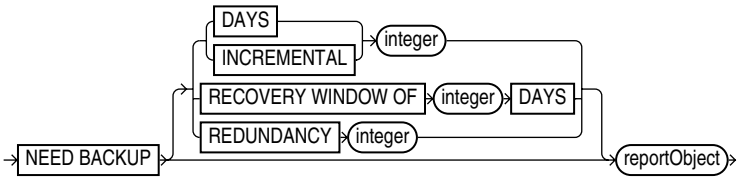
Syntax

report::=



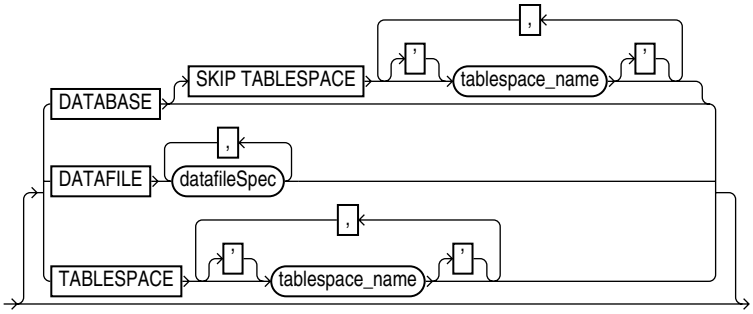
(*needBackupOption::=* on page 3-28, *atClause::=* on page 3-29, *reportObject::=* on page 3-29, *obsOperandList::=* on page 4-36, *deviceSpecifier::=* on page 4-16)

needBackupOption::=



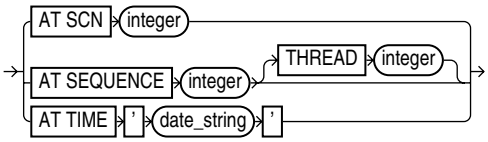
(*reportObject::=* on page 3-29)

reportObject::=



(*datafileSpec::=* on page 4-15)

atClause::=



Semantics

report

This clause specifies the type of report.

Syntax Element	Description
needBackupOption	Lists files that require backups. **See Also:** *needBackupOption* on page 3-30
OBSOLETE *obsOperandList*	Lists full backups, datafile copies, and archived redo log files recorded in the RMAN repository that can be deleted because they are no longer needed. See Table 3–6 for description of output. The command works in two steps: **1.** For each datafile that has been backed up, RMAN identifies the oldest full backup, level 0 backup, or image copy that is not obsolete under the retention policy. Any backup of the datafile older than the one identified in this step is considered obsolete. **2.** Any archived redo log files and level 1 incremental backups that are older than the oldest nonobsolete full backup are considered obsolete. These files are obsolete because no full or level 0 backup exists to which they can be applied. Incremental level 1 backups or archived redo log files are not considered obsolete if they can be applied to nonobsolete level 0 or full backups. The subclause *obsOperandList* describes the criteria that RMAN uses to determine what is obsolete. If you do not specify parameters in *obsOperandList*, then RMAN uses the options specified in `CONFIGURE RETENTION POLICY` (see Example 3–16 on page 3-34). If you use this option in conjunction with `DEVICE TYPE`, then RMAN only considers backups and copies created on the specified device. If the retention policy is disabled, then RMAN does not consider any backups as obsolete. Thus, RMAN issues an error when you run `REPORT OBSOLETE` with no other options and the retention policy is `NONE`. **Note:** A backup made with the `KEEP UNTIL TIME` clause is obsolete after the `KEEP` time passes, regardless of the configured retention policy settings.

Syntax Element	Description
SCHEMA	Lists the names of all datafiles (permanent and temporary) and tablespaces for the target database at the specified point in time. See Table 3–1 for description of output. For REPORT SCHEMA without forDbUniqueNameOption, a target database connection is required, but a recovery catalog connection is optional.
forDbUniqueNameOption	Reports the names of all datafiles and tablespaces for the database specified by its DB_UNIQUE_NAME. You can specify a database with *db_unique_name* or use ALL for all uniquely named databases recorded in the catalog for a particular DBID. A database is uniquely identified in the recovery catalog by a DBID and the value of the DB_UNIQUE_NAME initialization parameter. RMAN must be connected to a recovery catalog. RMAN must be connected to a target database or SET DBID must have been run. **See Also:** *forDbUniqueNameOption* on page 4-20 for descriptions of the options in this clause
atClause	Specifies an SCN, log sequence number, or time.
UNRECOVERABLE *reportObject*	Lists all unrecoverable datafiles. See Table 3–7 on page 3-33 for description of output. 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. In an unrecoverable operation, redo is not generated. Examples are direct load of table data and updates with the NOLOGGING option. **Note:** The nonexistence 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, if redo logs starting from when the file was created still exist.
DEVICE TYPE *deviceSpecifier*	Specifies the type of storage device. RMAN only considers backups and copies available on the specified device for its report.

needBackupOption

This clause reports only on files that need backups.

Syntax Element	Description
NEED BACKUP	Lists all datafiles in the specified *reportObject* that require a new backup. The report assumes that you will restore the most recent backup. If you do not specify any option, then RMAN uses the current retention policy configuration. If the retention policy is disabled (CONFIGURE RETENTION POLICY TO NONE), then RMAN generates an error.
DAYS *integer*	Lists all datafiles requiring more than the specified number of days' worth of archived redo log files for complete recovery. For example, REPORT NEED BACKUP DAYS 7 DATABASE shows the datafiles whose recovery requires more than seven days' worth of archived redo log files. See Table 3–2 for description of output. If the target database control file is mounted and current, then RMAN makes the following optimizations to this report: ▪ Files that are offline and whose most recent backup contains all changes to the file are not included. ▪ Files that were offline and are now online, and whose most recent backup contains all changes up to the offline time, are only reported if they have been online for more than the specified number of days.

Syntax Element	Description
INCREMENTAL *integer*	Specifies a threshold number of incremental backups required for recovery (see Example 3–15 on page 3-34). If complete recovery of a datafile requires more than *integer* incremental backups, then the datafile requires a new full backup. See Table 3–3 for description of output. **Note:** Files for which no backups exist will not appear in this list: issue the REPORT NEED BACKUP REDUNDANCY command to display them.
RECOVERY WINDOW OF *integer* DAYS	Reports datafiles for which there are not sufficient backups to satisfy a recovery window-based retention policy for the specified number of days, that is, datafiles without sufficient backups for point-in-time recovery to any point back to the time SYSDATE - *integer*. See Table 3–4 for description of output.
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. See Table 3–5 for description of output.
reportObject	Specifies the object for which you are generating the report.

reportObject

This subclause 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. Note that RMAN includes objects from prior incarnations.

Syntax Element	Description
DATABASE	Lists backups or datafile copies of all files in the database. **Note:** Specify SKIP TABLESPACE *tablespace_name* to exclude the specified tablespace from the DATABASE specification.
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.

atClause

This subclause specifies a point in time as a time, SCN, or log sequence number. You must be connected to a recovery catalog when issuing a REPORT SCHEMA command with an AT clause.

Syntax Element	Description
AT SCN *integer*	Specifies an SCN.
AT SEQUENCE *integer*	Specifies a log sequence number. The integer indicates the time when the specified log was first opened.
THREAD *integer*	Specifies a redo THREAD number. The integer indicates the time when the thread was first opened.
AT TIME '*date_string*'	Specifies a date (see Example 3–14 on page 3-33). The NLS_LANG and NLS_DATE_FORMAT environment variables specify the format for the time.

Report Output

The information that appears in the output is described in the following tables:

- Table 3–1, " Report of Database Schema"
- Table 3–2, " Report of Files Whose Recovery Needs More Than n Days of Archived Logs"
- Table 3–3, " Report of Files That Need More than n Incrementals During Recovery"
- Table 3–4, " Report of Files That Must Be Backed Up to Satisfy n Days Recovery Window"
- Table 3–5, " Report of Files with Fewer Than n Redundant Backups"
- Table 3–6, " Report of Obsolete Backups and Copies"
- Table 3–7, " Report of Files that Need Backup Due to Unrecoverable Operations"

Table 3–1 Report of Database Schema

Column	Indicates
`File`	The absolute datafile number.
`Size(MB)`	The size of the file in megabytes.
`Tablespace`	The tablespace name.
`RB segs`	For datafiles only. `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.
`Datafile Name`	For permanent datafiles only. The filename of the datafile.
`Maxsize(MB)`	For tempfiles only. The maximum size of the tempfile.
`Tempfile Name`	For tempfiles only. The filename of the tempfile.

Table 3–2 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 log files for recovery.
`Days`	The number of days of archived redo data required for recovery.
`Name`	The name of the datafile.

Table 3–3 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.

Table 3–4 Report of Files That Must Be Backed Up to Satisfy n Days Recovery Window

Column	Indicates
`File`	The absolute file number of a datafile that must be backed up to satisfy a recovery window of *n* days.
`Days`	The number of days required for complete recovery.
`Name`	The name of the datafile that requires backup.

Table 3–5 Report of Files with Fewer 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 3–6 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 3–7 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.

Examples

Example 3–14 Reporting a Database Schema

This example, which requires a recovery catalog, reports the names of all datafiles and tablespaces 20 minutes ago.

```
RMAN> REPORT SCHEMA AT TIME 'sysdate-20/1440';

Report of database schema for database with db_unique_name PROD

List of Permanent Datafiles
===========================
File Size(MB) Tablespace           RB segs Datafile Name
---- -------- -------------------- ------- ------------------------
1    450      SYSTEM               YES     /disk1/oradata/prod/system01.dbf
2    197      SYSAUX               YES     /disk1/oradata/prod/sysaux01.dbf
3    20       UNDOTBS              YES     /disk1/oradata/prod/undotbs01.dbf
4    10       CWMLITE              YES     /disk1/oradata/prod/cwmlite01.dbf
5    10       DRSYS                YES     /disk1/oradata/prod/drsys01.dbf
6    10       EXAMPLE              YES     /disk1/oradata/prod/example01.dbf
7    10       INDX                 YES     /disk1/oradata/prod/indx01.dbf
8    10       TOOLS                YES     /disk1/oradata/prod/tools01.dbf
9    10       USERS                YES     /disk1/oradata/prod/users01.dbf
```

```
List of Temporary Files
=======================
File Size(MB) Tablespace           Maxsize(MB) Tempfile Name
---- -------- -------------------- ----------- --------------------
1    40       TEMP                 32767       /disk1/oradata/prod/temp01.dbf
```

Example 3–15 Reporting Datafiles Needing Incremental Backups

This example reports all datafiles in the database that require the application of one or more incremental backups to be recovered to their current state:

```
RMAN> REPORT NEED BACKUP INCREMENTAL 1;

Report of files that need more than 1 incrementals during recovery
File Incrementals Name
---- ------------ ----------------------------------------------
1    2            /disk1/oradata/prod/system01.dbf
2    2            /disk1/oradata/prod/sysaux01.dbf
3    2            /disk1/oradata/prod/undotbs01.dbf
4    2            /disk1/oradata/prod/cwmlite01.dbf
5    2            /disk1/oradata/prod/drsys01.dbf
6    2            /disk1/oradata/prod/example01.dbf
7    2            /disk1/oradata/prod/indx01.dbf
9    2            /disk1/oradata/prod/users01.dbf
```

Example 3–16 Reporting Obsolete Backups and Copies

The following example reports obsolete backups and copies that are redundant according to the current retention policy. The retention policy is set to redundancy 1.

```
RMAN> REPORT OBSOLETE;

RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
Report of obsolete backups and copies
Type                 Key    Completion Time    Filename/Handle
-------------------- ------ ------------------ --------------------
Archive Log          1022   19-FEB-07          /disk1/prod/arch/archive1_59_614712405.dbf
Archive Log          1023   19-FEB-07          /disk1/prod/arch/archive1_61_614712405.dbf
Archive Log          1024   19-FEB-07          /disk1/prod/arch/archive1_60_614712405.dbf
Archive Log          1025   19-FEB-07          /disk1/prod/arch/archive1_55_614712405.dbf
Backup Set           1032   19-FEB-07
  Backup Piece       1050   19-FEB-07
 /disk2/PROD/backupset/2007_02_19/o1_mf_nnndf_TAG20070216T173839_2xnpmp0l_.bkp
Datafile Copy        1073   19-FEB-07
 /disk2/PROD/datafile/o1_mf_system_2xmz5l5m_.dbf
Backup Set           1035   19-FEB-07
  Backup Piece       1053   19-FEB-07
 /disk2/PROD/backupset/2007_02_19/o1_mf_nnndf_TAG20070219T111434_2xnpozym_.bkp
Datafile Copy        1074   19-FEB-07
 /disk2/PROD/datafile/o1_mf_sysaux_2xmz6zdg_.dbf
Datafile Copy        1075   19-FEB-07
 /disk2/PROD/datafile/o1_mf_undotbs_2xmz7rof_.dbf
Datafile Copy        1076   19-FEB-07
 /disk2/PROD/datafile/o1_mf_cwmlite_2xmz7vrg_.dbf
Datafile Copy        1077   19-FEB-07          /disk2/PROD/datafile/o1_mf_drsys_2xmz7wyc_.dbf
Datafile Copy        1078   19-FEB-07
 /disk2/PROD/datafile/o1_mf_example_2xmz7y5s_.dbf
Datafile Copy        1079   19-FEB-07          /disk2/PROD/datafile/o1_mf_indx_2xmz81jg_.dbf
Datafile Copy        1081   19-FEB-07          /disk2/PROD/datafile/o1_mf_users_2xmz85vo_.dbf
Datafile Copy        1777   20-FEB-07          /disk2/users01.dbf
```

RESET DATABASE

Purpose

Use the `RESET DATABASE TO INCARNATION` command to reset the incarnation of the target database in the RMAN repository to a previous database incarnation. You are only required to use this command in the following scenarios:

- You use `RESTORE` or `RECOVER` to return the database to an SCN before the current `RESETLOGS` timestamp.
- You use `FLASHBACK DATABASE` to rewind the database to an orphaned database incarnation.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn about the circumstances in which it is necessary to use the `RESET DATABASE` command

Prerequisites

Execute `RESET DATABASE TO INCARNATION` only at the RMAN prompt. RMAN must be connected to a target database.

If RMAN runs in `NOCATALOG` mode, then the target database must be mounted. The mounted control file must contain a record of the specified database incarnation.

If RMAN runs in `CATALOG` mode, then the target database can be mounted or unmounted. If the database is mounted, then the control file must contain a record of the specified database incarnation.

Usage Notes

When you use RMAN in `NOCATALOG` mode, the `RESET DATABASE TO INCARNATION` command is persistent across RMAN sessions.

Syntax

***reset*::=**

Semantics

Syntax Element	Description
primaryKey	Changes the current incarnation to a noncurrent database incarnation specified by the primary key of the `DBINC` record for the database incarnation. An incarnation key is used to uniquely tag and identify a stream of redo. Run `LIST INCARNATION OF DATABASE` to obtain possible key values. After you issue `RESET DATABASE TO INCARNATION`, you can run RMAN commands such as `FLASHBACK DATABASE`, `RESTORE`, and `RECOVER`.

Examples

Example 3–17 Resetting RMAN to a Previous Incarnation in NOCATALOG Mode

In NOCATALOG mode, you must mount a control file that contains information about the incarnation that you want to recover. The following scenario resets the database to an abandoned incarnation of database trgt and performs incomplete recovery.

```
CONNECT TARGET / NOCATALOG

# step 1: start and mount a control file that knows about the incarnation to which
# you want to return. Refer to the RESTORE command for appropriate options.
STARTUP NOMOUNT;
RESTORE CONTROLFILE FROM AUTOBACKUP;
ALTER DATABASE MOUNT;

# step 2: obtain the primary key of old incarnation
LIST INCARNATION OF DATABASE trgt;

List of Database Incarnations
DB Key  Inc Key DB Name  DB ID            STATUS   Reset SCN  Reset Time
------- ------- -------- -------------    -------  ---------- ----------
1       2       TRGT     1334358386       PARENT   154381     OCT 30 2007 16:02:12
1       116     TRGT     1334358386       CURRENT  154877     OCT 30 2007 16:37:39

# step 3: in this example, reset database to incarnation key 2
RESET DATABASE TO INCARNATION 2;

# step 4: restore and recover the database to a point before the RESETLOGS
RESTORE DATABASE UNTIL SCN 154876;
RECOVER DATABASE UNTIL SCN 154876;

# step 5: make this incarnation the current incarnation and list incarnations:
ALTER DATABASE OPEN RESETLOGS;
LIST INCARNATION OF DATABASE trgt;

List of Database Incarnations
DB Key  Inc Key DB Name  DB ID            STATUS  Reset SCN  Reset Time
------- ------- -------- ---------------- ------- ---------- ----------
1       2       TRGT     1334358386       PARENT  154381     OCT 30 2007 16:02:12
1       116     TRGT     1334358386       PARENT  154877     OCT 30 2007 16:37:39
1       311     TRGT     1334358386       CURRENT 156234     AUG 13 2007 17:17:03
```

RESTORE

Purpose

Use the RESTORE command to restore, validate, or preview RMAN backups. Typically, you restore backups when a media failure has damaged a current datafile, control file, or archived redo log or before performing a point-in-time recovery.

Prerequisites

To restore datafiles to their current location, the database must be started, mounted, or open with the tablespaces or datafiles to be restored offline.

If you use RMAN in a Data Guard environment, then RMAN should be connected to a recovery catalog.

If you are performing a trial restore of the production database, then perform either of the following actions before restoring the database in the test environment:

- If the test database will use a fast recovery area that is physically *different* from the recovery area used by the production database, then set DB_RECOVERY_FILE_DEST in the test database instance to the new location.
- If the test database will use a fast recovery area that is physically the *same* as the recovery area used by the production database, then set DB_UNIQUE_NAME in the test database instance to a different name from the production database.

If you do not perform either of the preceding actions, then RMAN assumes that you are restoring the production database and deletes flashback logs from the fast recovery area because they are considered unusable.

Usage Notes

The RESTORE command restores full backups, level 0 incremental backups, or image copies. You can restore files to their default location or a different location.

By default, RMAN examines read-only datafiles to make sure they exist, are readable, and have the correct checkpoint. If any of the conditions is not met, then RMAN restores the files. If all of the conditions are met, then RMAN does not restore the files.

Backup Selection

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. RMAN only restores backups created on the same type of channels allocated by the RESTORE command. For example, if you made backups of a datafile with DISK and sbt channels, and if only a DISK channel is allocated for the RESTORE command, then RMAN will not restore the sbt backups. If you do not manually allocate channels, then RMAN allocates all automatic channels that it possibly needs, subject to any restrictions imposed by the DEVICE TYPE option.

In an Oracle RAC configuration, RMAN automatically restores backups, control file copies, and datafile copies from channels that can read the files on tape or a local file system. For example, if channel ch1 connected to inst1 can read log 1000 from its tape drive, but channel ch2 connected to inst2 cannot read the same log from its tape drive, then ch1 cannot participate in restoring the log and so ch2 restores the log. Autolocation is automatically enabled when the channels have different PARMS or CONNECT settings.

If datafile names are symbolic links, then the control file stores 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 re-creating the symbolic link, then RMAN restores the datafile to the location of the link file rather than to the location pointed to by the link file.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* for details on restore failover

Restore Operations Using Encrypted Backup Sets

As explained in "Encryption of Backup Sets" on page 2-21, how RMAN handles encrypted backup sets during restore operations depends on the encryption mode with which the backup was created. You can use `CONFIGURE` and `SET` to manage the RMAN backup encryption settings for your database. Note the following restore considerations:

- For transparent-mode encrypted backups, the required passwords must be available in the database wallet. The same wallet used when creating the backup must be open and available when restoring it. `SET DECRYPTION` is not required.
- For password-mode encrypted backups, the required passwords must be provided with `SET DECRYPTION`.
- For dual-mode encrypted backups, the required passwords must be available in the database wallet or provided with `SET DECRYPTION`.

> **Note:** Wallet-based encryption is more secure than password-based encryption because no passwords are involved. You should use password-based encryption only when absolutely necessary because your backups need to be transportable.

Restore Failover

If a backup piece, image copy or proxy copy is inaccessible or if a block is corrupted, then RMAN performs restore failover. The `RESTORE` command automatically looks for another usable copy of a backup or image copy on the same device and other devices. If no usable copies are available, then RMAN searches for previous backups. RMAN continuously searches for previous usable backups until it has exhausted all possibilities. RMAN automatically uses eligible backups from previous database incarnations if required.

If you are restoring a datafile for which no backups are available, then RMAN will create an empty datafile with the checkpoint change as creation SCN. During recovery, all archived redo log files back to the creation of the datafile will be restored, and all changes during the history of the datafile will be reapplied to re-create its contents.

> **See Also:** "Encryption of Backup Sets" on page 2-21 and the extended discussion in *Oracle Database Backup and Recovery User's Guide*

Location of Restored Datafiles

If you restore datafiles to the default location, then RMAN overwrites files with the same filenames. By default, RMAN will not restore a datafile if the datafile is in the correct place and its header contains the expected data. Note that RMAN does not scan the datafile body for corrupt blocks.

If RMAN detects that the default filename cannot be used (for example, the file may be an Oracle-managed file or on an Automatic Storage Management disk group), then RMAN attempts to create a new file in the same location or disk group.

RMAN restores datafiles to the location currently stored in the recovery catalog. This default behavior eliminates problems with restoring datafiles to locations that may have become obsolete since the time of the original backup. It also means that if you have changed the location of the datafiles from their original backup location, that RMAN restores the files to the most current or changed location.

To restore files to a nondefault location, use `SET NEWNAME` commands to rename the restored files and then use a `SWITCH` command to make the restored files current (as illustrated in Example 3–19 on page 3-49). If you do not issue `SWITCH` commands, then RMAN considers the restored files as valid copies for use in future restore operations. Table 3–8 describes the behavior of the `RESTORE`, `SET NEWNAME`, and `SWITCH` commands.

Table 3–8 SET NEWNAME, SWITCH, and RESTORE

SET NEWNAME Run	SWITCH Run	RESTORE Behavior
No	N/A	RMAN restores the files to the most recent location stored in the recovery catalog.
Yes	Yes	RMAN restores the files to the path names specified by `SET NEWNAME`. RMAN replaces the current datafile names in the control file with the names of the restored files. RMAN records the datafiles with the old names as datafile copies.
Yes	No	RMAN restores the files to the path names specified by `SET NEWNAME`. RMAN does not update the current datafile names in the control file. The restored files are listed in the RMAN repository as datafile copies.

Because tempfiles cannot be backed up and because no redo is ever generated for them, RMAN never restores or recovers tempfiles. RMAN does track the names of tempfiles, but only so that it can automatically re-create them when needed.

RMAN Behavior When Restoring Control Files

The behavior of RMAN when restoring control files depend on a variety of factors, which are summarized in Table 3–9. Required commands and options for restoring autobackups are summarized in Table 3–10.

Table 3–9 RESTORE CONTROLFILE Scenarios

RMAN Connection	RESTORE CONTROLFILE;	RESTORE CONTROLFILE FROM AUTOBACKUP;	RESTORE CONTROLFILE ... TO '*filename*';	RESTORE CONTROLFILE ... FROM '*media_handle*' or TAG '*user_tag*';
No catalog, target database started in NOMOUNT state	Error. Must specify FROM AUTOBACKUP.	Restores to CONTROL_FILES locations. See Table 3–10 for required commands and options.	Must specify FROM AUTOBACKUP. Restores only to *filename*.	First run SET DBID. Restores from specified file (cannot restore from TAG). If TO '*filename*' not used, restores to all CONTROL_FILES locations.
No catalog, target database mounted or open	Error. Must use TO '*filename*', where *filename* is not in CONTROL_FILES list.	Error. Must use TO '*filename*', where *filename* is not in CONTROL_FILES list.	Restores only to *filename*, where *filename* is not in CONTROL_FILES list.	RMAN issues error RMAN-06496. Use TO '*filename*' instead.
Catalog, target database started in NOMOUNT state	Restores to CONTROL_FILES locations. Run SET DBID only if DB_NAME not unique in catalog.	Only use with recovery catalog for testing.	Restores only to *filename*, where *filename* is not in CONTROL_FILES list.	Restores from specified file. If TO '*filename*' not used, restores to all CONTROL_FILES locations.
Catalog, target database mounted or open	Error. Must use TO '*filename*', where *filename* is not in CONTROL_FILES list.	Do not use with recovery catalog.	Restores only to *filename*, where *filename* is not in CONTROL_FILES list.	RMAN issues error RMAN-06496. Use TO '*filename*' instead.

If you use RMAN in a Data Guard environment, then RMAN transparently converts primary control files to standby control files and vice versa. RMAN automatically updates filenames for datafiles, online redo logs, standby redo logs, and temp files when you issue RESTORE and RECOVER. The recovery catalog always contains the correct information about the backup filenames for each database, as explained in "RMAN Backups in a Data Guard Environment" on page 2-22.

Control File and Server Parameter File Autobackup Options

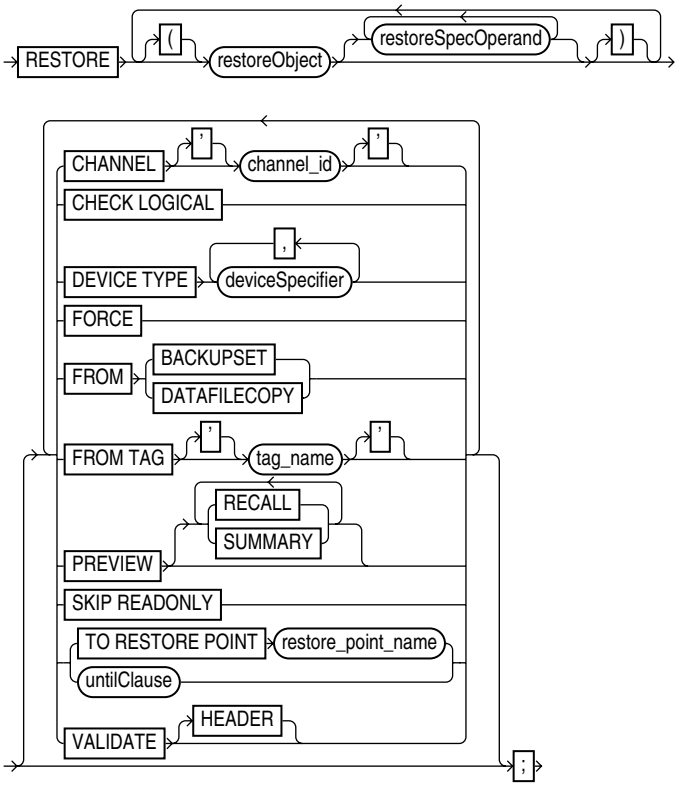
When restoring an autobackup, the commands and options that you use depend on the autobackup type (control file or server parameter file) and location (inside or outside fast recovery area). The options are summarized in Table 3–10.

Table 3–10 RESTORE ... FROM AUTOBACKUP

Restore Object	Autobackup Location	Run SET DBID?	Specify RECOVERY AREA on RESTORE?	Specify DB_NAME or DB_UNIQUE_NAME on RESTORE?	Run SET CONTROLFILE AUTOBACKUP FORMAT?
SPFILE	Recovery area	No	Yes	Yes	No
SPFILE	Outside recovery area	Yes	No	No	Only if autobackup is not in default location
Control file	Recovery area	No	Only if autobackup is in noncurrent recovery area	Only if autobackup is in noncurrent recovery area and uses a noncurrent DB_UNIQUE_NAME	No
Control file	Outside recovery area	Yes	No	No	Only if autobackup is not in default location

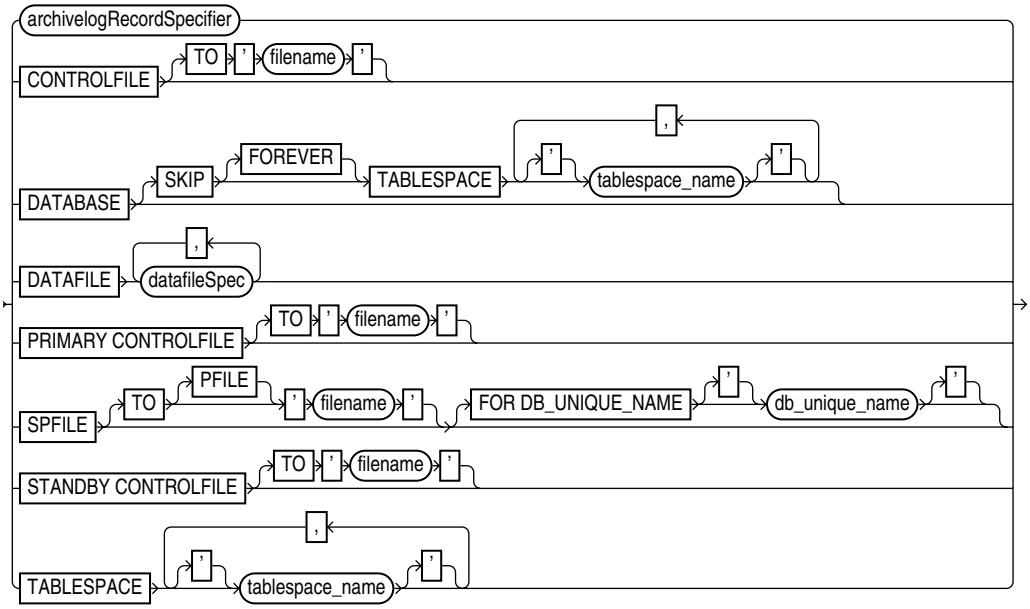
Syntax

restore::=



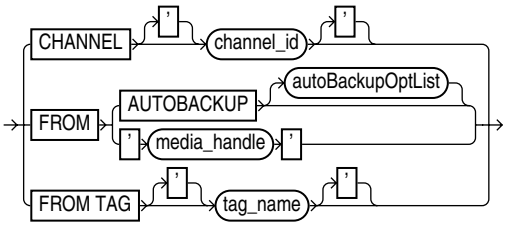
(*restoreObject::=* on page 3-41, *restoreSpecOperand::=* on page 3-42, *deviceSpecifier::=* on page 4-16, *untilClause::=* on page 4-42)

restoreObject::=

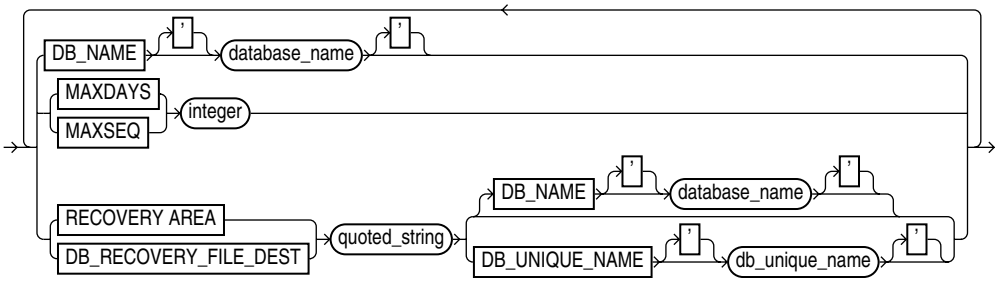


(*archivelogRecordSpecifier::=* on page 4-6, *datafileSpec::=* on page 4-15)

***restoreSpecOperand*::=**



***autoBackupOptList*::=**



Semantics

restore

This clause enables you to select which files you want to restore and specify parameters that control the behavior of the restore operation.

Syntax Element	Description
restoreObject	Specifies the files to be restored.
restoreSpecOperand	Specifies options for the *restoreObject* clause.
CHANNEL *channel_id*	Refer to the *restoreSpecOperand* clause.
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. If the total number of physical and logical corruptions detected in a file is less than its SET MAXCORRUPT setting, then the RMAN command completes and the database populates the V$DATABASE_BLOCK_CORRUPTION view with corrupt block ranges. If MAXCORRUPT is exceeded, then the command terminates without populating the views. When restoring a backup datafile, RMAN honors the DB_BLOCK_CHECKSUM initialization parameter setting. RMAN clears the checksum if DB_BLOCK_CHECKSUM is set to false. If set to typical, then RMAN verifies the checksum when restoring from the backup and writing to the datafile. If the initialization parameter DB_BLOCK_CHECKSUM=typical, and if MAXCORRUPT is not set, then specifying CHECK LOGICAL detects all types of corruption that are possible to detect. **Note:** The MAXCORRUPT setting represents the total number of physical and logical corruptions permitted on a file.

Syntax Element	Description
DEVICE TYPE *deviceSpecifier*	Allocates automatic channels for the specified device type only. For example, if you configure automatic disk and tape channels, and issue RESTORE ... DEVICE TYPE DISK, then RMAN allocates only disk channels. You must have already configured a device type by using CONFIGURE (except for DISK, which is preconfigured) before specifying the DEVICE TYPE option. **Note:** You cannot manually allocate channels within a RUN block and then run RESTORE with the DEVICE TYPE clause. **See Also:** *deviceSpecifier* on page 4-16
FORCE	Overrides the restartable restore feature and restores all files regardless of whether they need to be restored. If you do not specify FORCE, then RMAN restores a file only if its header information does not match the information in the control file.
FROM BACKUPSET	Specifies that RMAN should restore from backup sets only. By default RESTORE chooses the file copy or backup set that needs the least media recovery. If you use the FROM BACKUPSET option, then channels for the appropriate type of storage devices must be allocated for the backup sets that need to be restored. For example, if needed backups are only available on tape, and no sbt channels have been allocated, then RMAN cannot find a candidate backup set to restore, and the RESTORE command fails.
FROM DATAFILECOPY	Specifies that RMAN should restore datafile copies only. By default RESTORE chooses the file copy or backup set that needs the least media recovery. If you use the FROM DATAFILECOPY option, then the allocated channels must be of DEVICE TYPE DISK.
FROM TAG *tag_name*	Refer to the *restoreSpecOperand* clause.
PREVIEW	Reports—but does not restore—the backups and archived redo log files that RMAN could use to restore and recover the database to the specified time. RMAN queries the metadata and does not actually read the backup files. The RESTORE ... PREVIEW output is in the same format as the LIST BACKUP output (see Example 3–24 on page 3-51). Some media managers provide status information to RMAN about which backups are offsite. Offsite backups are stored in a remote location, such as a secure storage facility, and cannot be used without retrieving media. Offsite backups are marked as AVAILABLE in the RMAN repository even though the media must be retrieved from storage before the backup can be restored. If RMAN attempts to restore a offsite backup, then the restore operation fails. RESTORE ... PREVIEW can identify backups needed for a RESTORE operation that are stored on media that requires retrieval. The output indicates whether backups are stored offsite. If a needed backup is stored offsite, but the media manager does not support offsite backups, then your options are: ■ Use CHANGE ... UNAVAILABLE to prevent RMAN from selecting the needed offsite backups, and attempt the RESTORE ... PREVIEW operation again to determine whether RMAN selects another offsite backup. When RMAN does not select any offsite backups, you can perform the restore operation. ■ Use RESTORE ... PREVIEW with the RECALL option. **See Also:** LIST on page 2-164, specifically the BACKUPS and SUMMARY options, and the RECOVER ... VALIDATE HEADER command
RECALL	Instructs the media manager to retrieve the backup media needed for the specified restore operation from offsite storage (see Example 3–25 on page 3-51). **Note:** This option only works if your media manager supports this functionality. You can use RESTORE ... PREVIEW periodically to monitor whether the needed backups are stored locally again.

Syntax Element	Description
SUMMARY	Summarizes the backups that RMAN would restore. The output is in the same format as the output of the `LIST BACKUPS ... SUMMARY` command.
SKIP READONLY	Does not restore read-only files.
TO RESTORE POINT *restore_point_name*	Specifies a restore point, with the SCN at which the restore point was created as the upper, inclusive limit. Because the limit is inclusive, RMAN selects only files that can be used to restore files up to *and including* the SCN corresponding to the restore point.
untilClause	Limits the selection to backup sets or file copies that are suitable for a point-in-time recovery to the specified time, SCN, or log sequence number. In the absence of any other criteria, RMAN selects the most current file copy or backup set to restore. Note that the time specified in the `UNTIL` clause must fall within the current database incarnation. **See Also:** *untilClause* on page 4-42
VALIDATE	Specifies that RMAN should decide which backup sets, datafile copies, and archived redo log files need to be restored, and then validate them (see Example 3–26 on page 3-52). No files are restored. For files on both disk and tape, RMAN reads all blocks in the backup piece or image copy. RMAN also validates offsite backups. The validation is identical to a real restore operation except that RMAN does not write output files. **Note:** If you use `RESTORE` with the `VALIDATE` option, then the database can be open with datafiles online. **See Also:** `VALIDATE` on page 3-107
HEADER	Reports and validates—but does not restore—the backups that RMAN could use to restore to the specified time. When you specify this option, RMAN performs the same functions as when you run `RESTORE` with the `PREVIEW` option. However, in addition to listing the files needed for restore and recovery, RMAN validates the backup file headers to determine whether the files on disk or in the media management catalog correspond to the metadata in the RMAN repository. **See Also:** The descriptions of the `RESTORE PREVIEW` option and the `RECOVER ... VALIDATE HEADER` option

restoreObject

This subclause specifies the objects to be restored: control files, datafiles, archived redo log files, or the server parameter file. Note that RMAN does not support backup and recovery of the change tracking file. RMAN re-creates the change tracking file after database restore and recovery; the next incremental backup after any recovery is able to use the file. Thus, restore and recovery has no user-visible effect on change tracking.

Syntax Element	Description
archivelogRecordSpecifier	Restores the specified range of archived redo log files. The default restore location is DB_RECOVERY_FILE_DEST (if one of LOG_ARCHIVE_DEST_n is configured to USE_DB_RECOVERY_FILE_DEST either implicitly or explicitly). Otherwise, the default restore filenames are constructed with the LOG_ARCHIVE_FORMAT and LOG_ARCHIVE_DEST_1 initialization parameters of the target database. These parameters combine in a port-specific fashion to derive the name of the restored log. You can override the default location with the SET ARCHIVELOG DESTINATION command. Because the RECOVER command automatically restores archived redo log files as needed, you should seldom need to restore logs manually. Possible reasons for manually restoring archived redo log files are to speed up recovery, to stage the logs to multiple destinations, or to analyze the log contents after a point-in-time recovery. To restore logs from a previous incarnation without shutting down the database, you can use RESTORE ARCHIVELOG with: ■ ... FROM SCN ■ ...SCN BETWEEN... AND ■ FROM SCN ... INCARNATION <integer> ■ FROM SCN... INCARNATION ALL **Note:** The database can be started, mounted, or open for this operation. **See Also:** *archivelogRecordSpecifier* on page 4-6.
CONTROLFILE	Restores either a standby or backup control file depending on the target database role. If the control file is lost, then restore the control file (see Table 3–9 on page 3-40) and restore the database after mounting the restored control file. You must always run the RECOVER command after mounting a restored control file and you must open the database with the RESETLOGS option. **Note:** If the target database is not mounted, and if RMAN is not connected to a recovery catalog, then you must specify the FROM AUTOBACKUP clause with RESTORE CONTROLFILE. If the autobackup is in a nondefault format, then first use the SET CONTROLFILE AUTOBACKUP FORMAT command to specify the format. If the target database is mounted or open, then you must specify the TO *filename* clause with RESTORE CONTROLFILE. When you run RESTORE with a backup control file while connected to a recovery catalog (see Example 3–20 on page 3-49), RMAN automatically updates the control file to reflect the structure of the restored database based on the metadata in the catalog.
TO '*filename*'	Restores the control file to the specified filename. Table 3–9 on page 3-40 explains RMAN behavior when restoring the control file with the TO clause.
DATABASE	Restores all datafiles in the database except those that are offline. By default, RMAN restores datafiles in read-only tablespaces. Unlike BACKUP DATABASE, RESTORE DATABASE does *not* automatically include the control file and the server parameter file—you must issue additional RESTORE CONTROLFILE and RESTORE SPFILE commands to restore these files. **Note:** To restore offline datafiles you must use RESTORE DATAFILE or RESTORE TABLESPACE.
SKIP [FOREVER] TABLESPACE *tablespace_name*	Excludes the specifies tablespaces from the restore operation. This option is useful to avoid restoring tablespaces containing temporary data. Specifying the FOREVER keyword does not change the behavior of SKIP. The FOREVER keyword exists solely to maintain compatible syntax between RESTORE SKIP FOREVER and RECOVER SKIP FOREVER.

Syntax Element	Description
DATAFILE *datafileSpec*	Restores the datafiles specified by filename or absolute datafile number (see Example 3–19 on page 3-49). **Note:** Do not specify a datafile more than once in a restore job. For example, the following command is invalid because datafile 1 is both specified explicitly and implied by the SYSTEM tablespace: RESTORE TABLESPACE SYSTEM DATAFILE 1; **See Also:** *datafileSpec* on page 4-15
PRIMARY CONTROLFILE	Restores a control file for a primary database in a Data Guard environment. RMAN restores either a normal or standby control file as appropriate, depending on the most recent database role known to the recovery catalog (RC_SITE.DATABASE_ROLE) for the target database. The purpose of this option to override the default setting in cases where the most recent database role is out-of-date. Assume that you perform a switchover from primary database dgny to standby database dgsf, so that dgsf is the new primary database. You want to restore a control file on dgsf, but the recovery catalog was not resynchronized and still shows dgsf as a standby database. In this case, you can specify PRIMARY CONTROLFILE to override the default RMAN behavior and restore a normal control file.
SPFILE	Restores a primary or standby server parameter file to the location from which it was backed up. RMAN cannot overwrite a server parameter file currently in use by the target database. By default RMAN restores the most current server parameter file. Specify the UNTIL or TAG options to restore older versions of the server parameter file. If the server parameter file is lost, then connect RMAN to the target database (and recovery catalog if used) and run SET DBID. Run STARTUP FORCE NOMOUNT before running RESTORE SPFILE. Then run STARTUP FORCE to restart the database instance with the restored server parameter file. **Note:** If the target database is not mounted, and if RMAN is not connected to a recovery catalog, then you must specify the FROM AUTOBACKUP clause with RESTORE SPFILE. If the autobackup is in a nondefault format, then first use the SET CONTROLFILE AUTOBACKUP FORMAT command to specify the format. If the target database is started, mounted, or open, and if the database was started with a server parameter file, then you must specify the TO *filename* clause with RESTORE SPFILE.
TO [PFILE] '*filename*'	Restores a primary or standby server parameter file to the location specified by the TO clause. Specify PFILE to save the server parameter file as a text-based initialization parameter file.
FOR DB_UNIQUE_NAME *db_unique_name*	Specifies the DB_UNIQUE_NAME for the target database when the instance is not started. This parameter is only useful in a Data Guard environment. When FOR DB_UNIQUE_NAME is specified, RMAN can locate the correct RMAN configurations for the host on which the SPFILE is being restored and use them to access backup devices. Otherwise, RMAN cannot choose the correct channel configurations and returns an RMAN-6758 error. In a Data Guard environment, the primary and standby hosts may have different channel configurations for communicating with their associated SBT backup and disk devices. If both the primary and standby databases are known to the recovery catalog, then the configuration settings for both databases are recorded in the recovery catalog. Because the two databases have the same DB_NAME, the records in the recovery catalog can only be distinguished with the DB_UNIQUE_NAME initialization parameter. **Note:** Using RESTORE SPFILE when the DB_NAME is not unique in the recovery catalog produces an RMAN-6758 error. **See Also:** *Oracle Data Guard Concepts and Administration* for a detailed procedure for restoring the server parameter file in a Data Guard environment

Syntax Element	Description
TO '*filename*'	Restores the standby control file to the specified filename. Table 3–9 explains the RMAN behavior when restoring the control file with the TO clause.
STANDBY CONTROLFILE	Restores a control file for a standby database. RMAN can transparently restore a normal control file backup and make it usable for a standby database. RMAN restores either a normal or standby control file as appropriate, depending on the most recent database role known to the recovery catalog (RC_SITE.DATABASE_ROLE) for the target database. The purpose of this option to override the default setting in cases where the most recent database role is out-of-date. Assume that you perform a switchover from primary database dgny to standby database dgsf, so that dgsf is the new primary database. Later, you make dgny a standby database for dgsf. You want to restore a control file on dgny, but the recovery catalog was not resynchronized and still shows dgny as a primary database. In this case, you can specify STANDBY CONTROLFILE to override the default RMAN behavior and restore a standby control file. If you restore the control file of a database whose DB_UNIQUE_NAME is known to the recovery catalog, then RMAN updates all filenames in the control file to filenames known to the recovery catalog. Any filenames explicitly renamed with ALTER DATABASE RENAME FILE take precedence over the filenames in the recovery catalog. **See Also:** Table 3–9 for restrictions and usage notes **Note:** You must always run the RECOVER command after mounting a restored control file, and must also always open the database with the RESETLOGS option.
TABLESPACE *tablespace_name*	Restores all datafiles in the specified tablespaces (see Example 3–18 on page 3-49). RMAN translates the tablespace name internally into a list of datafiles. If you rename a tablespace (for example, from users to customers), then so long as an additional tablespace with the old name (users) has not been created, you can use either the old name (users) or the new name (customers) for the tablespace. RMAN detects that the tablespace has changed its name and updates the recovery catalog on the next resynchronization. **Note:** RMAN can back up and restore dictionary-managed temporary tablespaces, but it cannot back up locally managed temporary tablespaces. However, RMAN automatically re-creates locally managed temporary tablespaces after restoring the database.

restoreSpecOperand

This subclause specifies options for the *restoreObject* clause. These parameters override the parameters with the same name at the RESTORE command level.

Syntax Element	Description
CHANNEL *channel_id*	Specifies the case-sensitive name of a channel to use for this restore operation. If you do not specify a channel, then RESTORE uses any available channel allocated with the correct device type.

Syntax Element	Description
FROM AUTOBACKUP	Restores a control file autobackup (see Example 3–21 on page 3-50). This option is only valid on the RESTORE CONTROLFILE and RESTORE SPFILE commands. When restoring either type of file in NOCATALOG mode, the FROM AUTOBACKUP clause is required. RMAN begins the search on the current day or on the day specified with the SET UNTIL. On the first day searched, the search begins with sequence number 256 (or the sequence number specified by MAXSEQ, if provided) and counts back to sequence 0. If no autobackup is found in the current or SET UNTIL day, then RMAN checks preceding days, starting with sequence 256 and counting back to 0. The search continues up to MAXDAYS days (default of 7, maximum of 366) prior to the current or SET UNTIL day. If no autobackup is found within MAXDAYS days, then RMAN signals an error and the command stops. **See Also:** Table 3–9 for restrictions and usage notes.
autoBackupOptList	Specifies parameters that control the search for a control file autobackup.
'*media_handle*'	Specifies the name of the control file copy or backup piece containing a control file. The *media_handle* can be any backup piece that contains a backup of a control file: the control file backup does not need to be an autobackup. **See Also:** Table 3–9 for restrictions and usage notes.
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 created with the specified tag. If multiple backup sets or copies have a matching tag, then RMAN selects the most recent one. Tag names are not case sensitive. **See Also:** BACKUP on page 2-19 for a description of how a tag can be applied to an individual copy of a duplexed backup set, and for a description of the default filename format for tags.

autoBackupOptList

This subclause specifies parameters that control the search for a control file autobackup.

Syntax Element	Description
DB_NAME *database_name*	Provides a DB_NAME to be used in searching for control file autobackups. See Table 3–10 on page 3-40 to determine when to set this parameter. The default value of the DB_UNIQUE_NAME initialization parameter is the DB_NAME initialization parameter setting. If no DB_UNIQUE_NAME initialization parameter is set for a target database, then use either RESTORE ... DB_NAME or RESTORE ... DB_UNIQUE_NAME. If the DB_UNIQUE_NAME initialization parameter setting for a target database is different from DB_NAME, then use RESTORE ... DB_UNIQUE_NAME.
MAXDAYS *integer*	Limits the search for a control file autobackup to within the specified number of days in the past.
MAXSEQ *integer*	Specifies the highest sequence number for the control file autobackup search.
RECOVERY AREA '*pathname*'	Specifies a path to the fast recovery area to search for autobackups. RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms. See Table 3–10 on page 3-40 to determine when to set this parameter.
DB_RECOVERY_FILE_DEST '*pathname*'	RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms.

Syntax Element	Description
DB_NAME *database_name*	Provides a DB_NAME to be used in searching for control file autobackups. See Table 3–10 on page 3-40 to determine when to set this parameter. The default value of the DB_UNIQUE_NAME initialization parameter is the DB_NAME initialization parameter setting. If no DB_UNIQUE_NAME initialization parameter is set for a target database, then use either RESTORE ... DB_NAME or RESTORE ... DB_UNIQUE_NAME. If the DB_UNIQUE_NAME initialization parameter setting for a target database is different from DB_NAME, then use RESTORE ... DB_UNIQUE_NAME.
DB_UNIQUE_NAME *db_unique_name*	Specifies the DB_UNIQUE_NAME of the database in the specified fast recovery area that is the target of the restore operation. The default value of the DB_UNIQUE_NAME initialization parameter is the DB_NAME initialization parameter setting. If no DB_UNIQUE_NAME initialization parameter is set for a target database, then use either RESTORE ... DB_NAME or RESTORE ... DB_UNIQUE_NAME. If the DB_UNIQUE_NAME initialization parameter setting for a target database is different from DB_NAME, then use RESTORE ... DB_UNIQUE_NAME.

Examples

Example 3–18 Restoring a Tablespace

This example takes a tablespace offline, restores it, then performs media recovery.

```
SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
RESTORE TABLESPACE users;
RECOVER TABLESPACE users;
SQL "ALTER TABLESPACE users ONLINE";
```

Example 3–19 Setting a New Name for a Restored Datafile

Assume that /disk1, which contains datafile 9, suffers a media failure. This example specifies a new name for the datafile, restores it, updates the control file to use the new name, recovers it, and then brings it online:

```
RUN
{
  SQL "ALTER DATABASE DATAFILE 9 OFFLINE";
  SET NEWNAME FOR DATAFILE 9 TO '/disk2/users01.dbf';
  RESTORE DATAFILE 9;
  SWITCH DATAFILE ALL;
  RECOVER DATAFILE 9;
  SQL "ALTER DATABASE DATAFILE 9 ONLINE";
}
```

Example 3–20 Restoring the Control File When Using a Recovery Catalog

Assume that you want to restore the control file backup with the tag monday_cf_backup. You start the RMAN client, connect to the target and recovery catalog databases, and run the following commands:

```
RUN
{ # SET DBID is not necessary when RMAN is connected to a recovery catalog
  STARTUP FORCE NOMOUNT;
  RESTORE CONTROLFILE FROM TAG 'monday_cf_backup';
  ALTER DATABASE MOUNT;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

```
ALTER DATABASE OPEN RESETLOGS; # required after recovery with backup control file
```

RMAN restores the control file to its default location and replicates it automatically to all CONTROL_FILES locations. RMAN mounts the control file and restores and recovers the database. RMAN automatically updates the control file to reflect the structure of the restored database based on the metadata in the recovery catalog.

Example 3–21 Recovering the Database with a Control File Autobackup

Assume that the control file and some datafiles are lost and need to be restored from tape. Because RMAN does not use a recovery catalog in this scenario, the SET DBID command is necessary to identify the control file to be restored. The example restores the control file from tape, mounts the database, and then restores and recovers the database.

```
CONNECT TARGET /
STARTUP FORCE NOMOUNT;
SET DBID 36508508;  # required when restoring control file in NOCATALOG mode
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  RESTORE CONTROLFILE FROM AUTOBACKUP;
  ALTER DATABASE MOUNT;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
ALTER DATABASE OPEN RESETLOGS;
```

Example 3–22 Restoring a Control File Autobackup to a Nondefault Location

This example is a variation on Example 3–21. In this scenario, the control file autobackup is located on disk in a nondefault location. RMAN starts searching for backups with a sequence number of 20, and searches backward for 5 months:

```
CONNECT TARGET /
STARTUP FORCE NOMOUNT
SET DBID 36508508;  # required when restoring control file in NOCATALOG mode
RUN
{
  SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK
    TO '/disk1/prod_cf_auto_%F';
  RESTORE CONTROLFILE TO '/tmp/cf_auto.dbf' FROM AUTOBACKUP
    MAXSEQ 20 MAXDAYS 150;
  ALTER DATABASE MOUNT;
  RESTORE DATABASE;
  RECOVER DATABASE;
}
ALTER DATABASE OPEN RESETLOGS;
```

Example 3–23 Restoring a Server Parameter File Autobackup to the Current Location

The following series of commands restores the current server parameter file in NOCATALOG mode and then starts the instance with the restored server parameter file.

```
CONNECT TARGET /
SET DBID 1620189241; # set dbid to dbid of target database
STARTUP FORCE NOMOUNT; # start instance with dummy SPFILE
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  RESTORE SPFILE FROM AUTOBACKUP; # FROM AUTOBACKUP needed in NOCATALOG mode
```

```
  STARTUP FORCE; # startup with restored SPFILE
}
```

Example 3–24 Previewing Backups

This example shows the results of a `RESTORE ... PREVIEW` command, which identifies the backup sets RMAN selects for use in restoring archived redo log files.

```
RMAN> RESTORE ARCHIVELOG ALL DEVICE TYPE sbt PREVIEW;

Starting restore at 01-MAR-07
released channel: ORA_SBT_TAPE_1
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=85 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup

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

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
53      1.25M      SBT_TAPE    00:00:18     01-MAR-07
        BP Key: 53   Status: AVAILABLE  Compressed: NO  Tag: TAG20070301T150155
        Handle: 2aibhej3_1_1   Media: RMAN-DEFAULT-000001

  List of Archived Logs in backup set 53
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    8       526376     01-MAR-07 527059     01-MAR-07
  1    9       527059     01-MAR-07 527074     01-MAR-07
  1    10      527074     01-MAR-07 527091     01-MAR-07
  1    11      527091     01-MAR-07 527568     01-MAR-07
  1    12      527568     01-MAR-07 527598     01-MAR-07
validation succeeded for backup piece
Finished restore at 01-MAR-07
```

Example 3–25 Recalling Offsite Backups from Offsite Storage

When used with a media manager that reports information about offsite storage of backups and supports recalling offsite backups, `RESTORE ... PREVIEW RECALL` requests that any media needed to restore archived redo log files from backup be recalled from offsite storage.

```
RMAN> RESTORE ARCHIVELOG ALL PREVIEW RECALL;

Starting restore at 10-JUN-06
using channel ORA_DISK_1
using channel ORA_SBT_TAPE_1


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

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
31      12.75M     SBT_TAPE    00:00:02     10-JUN-06
        BP Key: 33   Status: AVAILABLE  Compressed: NO  Tag: TAG20050610T152755
        Handle: 15gmknbs   Media: /v1,15gmknbs

  List of Archived Logs in backup set 31
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
```

```
  ---- ------- ---------- --------- ---------- ---------
  1    1       221154     06-JUN-06 222548     06-JUN-06
  1    2       222548     06-JUN-06 222554     06-JUN-06
  1    3       222554     06-JUN-06 222591     06-JUN-06
  1    4       222591     06-JUN-06 246629     07-JUN-06
  1    5       246629     07-JUN-06 262451     10-JUN-06

BS Key  Size       Device Type Elapsed Time Completion Time
------- ---------- ----------- ------------ ---------------
32      256.00K    SBT_TAPE    00:00:01     10-JUN-06
        BP Key: 34   Status: AVAILABLE  Compressed: NO  Tag: TAG20050610T153105
        Handle: 17gmknhp_1_1   Media: /v1,17gmknhp_1_1

  List of Archived Logs in backup set 32
  Thrd Seq     Low SCN    Low Time  Next SCN   Next Time
  ---- ------- ---------- --------- ---------- ---------
  1    6       262451     10-JUN-06 262547     10-JUN-06
  1    7       262547     10-JUN-06 262565     10-JUN-06

Initiated recall for the following list of offsite backup files
===========================================================
        Handle: 15gmknbs   Media: /v1,15gmknbs
Finished restore at 10-JUN-06
```

Example 3–26 Validating the Restore of a Backup

The following example illustrates using `RESTORE... VALIDATE` to confirm that backups required to restore the database are present on disk or tape, readable, and not corrupted:

```
RMAN> RESTORE DATABASE VALIDATE;

Starting restore at 01-MAR-07
using channel ORA_DISK_1
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=85 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup

channel ORA_DISK_1: starting validation of datafile backup set
channel ORA_DISK_1: reading from backup piece
/disk2/PROD/backupset/2007_03_01/o1_mf_nnndf_TAG20070301T161038_2ygtvzg0_.bkp
channel ORA_DISK_1: piece
handle=/disk2/PROD/backupset/2007_03_01/o1_mf_nnndf_TAG20070301T161038_2ygtvzg0_.b
kp tag=TAG20070301T161038
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: validation complete, elapsed time: 00:00:16
Finished restore at 01-MAR-07
```

RESYNC CATALOG

Purpose

Use the `RESYNC CATALOG` command to perform a full resynchronization of metadata in a recovery catalog schema with metadata in a target database control file. You can also use the `FROM CONTROLFILECOPY` clause to resynchronize the current control file with the RMAN metadata in a control file copy.

Typically, you run `RESYNC CATALOG` in the following situations:

- The recovery catalog was unavailable when you executed RMAN commands that automatically perform a resynchronization.
- The target database is running in `ARCHIVELOG` mode, because the recovery catalog is *not* updated automatically when an online redo log switch occurs or when a redo log is archived.
- You made changes to the physical structure of the target database such as adding or dropping a tablespace. As with log archiving, the recovery catalog is *not* updated automatically when the physical schema changes.
- RMAN is connected as `TARGET` to a standby database. You want to update the recovery catalog with metadata about RMAN operations performed on this database.
- RMAN is connected as `TARGET` to a standby database. You want to update the recovery catalog with metadata about a physical change on the primary database (see Example 3–29 on page 3-55).

Prerequisites

RMAN must be connected as `TARGET` to a mounted or open database and connected as `CATALOG` to a recovery catalog database.

Usage Notes

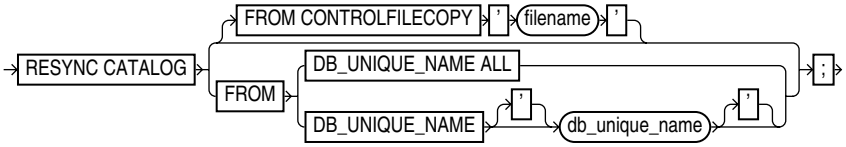
Resynchronizations are full or partial. If full, and if the target database has mounted the current control file (but not a newly created control file or a control file that is less current than a control file that was used previously), then RMAN updates all changed records for the physical schema: datafiles, tablespaces, redo threads, and online redo logs. If the database is open, then RMAN also obtains data about rollback segments. If the resynchronization is partial, then RMAN does not resynchronize metadata about the physical schema or rollback segments.

If the target control file is mounted and the catalog database is available at command execution, then RMAN automatically resynchronizes the recovery catalog as needed when you use RMAN commands. RMAN performs a full resynchronization after structural changes to database (adding or dropping database files, creating new incarnation, and so on) or after changes to the RMAN persistent configuration.

Starting with Oracle Database 11*g*, a single recovery catalog schema can keep track of database filenames for all databases in a Data Guard environment. This catalog schema also keeps track of where the online redo logs, standby redo logs, tempfiles, archived redo log files, backup sets, and image copies are created for all databases. If RMAN is connected as `TARGET` to a standby database, then RMAN implicitly executes a full resynchronization if the standby control file contains information about a physical schema change on the primary database.

Syntax

***resync*::=**



Semantics

Syntax Element	Description
`RESYNC CATALOG`	Updates the recovery catalog with RMAN metadata in the current control file of the target database (default). RMAN creates a snapshot control file in order to obtain a read-consistent view of the control file, then updates the recovery catalog with any new information from the snapshot. The `RESYNC CATALOG` command updates the following classes or records: ▪ Log history records, which are created when a log switch occurs. Note that log history records describe an online log switch, not a log archival. ▪ Archived redo log records, which are associated with archived redo log files created by archiving an online redo log, copying an existing archived log, or restoring backups of archived redo log files. ▪ Backup records, which are records of backup sets, backup pieces, proxy copies, and image copies. ▪ Physical schema records, which are associated with datafiles and tablespaces. If the target database is open, then rollback segment information is also updated.
`FROM CONTROLFILECOPY '`*`filename`*`'`	Updates the current control file and recovery catalog with RMAN metadata from a control file copy (see Example 3–28 on page 3-55). Use *filename* to specify the name of the control file copy to use for resynchronization. The primary use for `FROM CONTROLFILECOPY` occurs when you re-create the control file, which causes you to lose RMAN records stored in the control file. You can then resynchronize the newly created control file with an old copy. Physical schema information is not updated when you use this option. **Note:** The control file copy can either be in the current database incarnation, or created in a prior incarnation (that is, prior to the most recent `OPEN RESETLOGS`).

Syntax Element	Description
FROM DB_UNIQUE_NAME {ALL | *db_unique_name*}	Resynchronizes the recovery catalog with control file metadata in the specified database or databases (see Example 3–30 on page 3-56). You can specify a single database with *db_unique_name* or use ALL for all databases in the recovery catalog that share the DBID of the target database. If you specify ALL, then RMAN resynchronizes all databases in the Data Guard environment that are known to the recovery catalog. **Note:** You must have previously used CONFIGURE DB_UNIQUE_NAME ... CONNECT IDENTIFIER to specify a net service name to be used for an Oracle Net connection to the database specified in FROM DB_UNIQUE_NAME. When you run RESYNC FROM DB_UNIQUE_NAME for a specified database, RMAN performs both a normal resynchronization and a reverse resynchronization. In a normal resynchronization, RMAN updates the recovery catalog with metadata from the control file. In a reverse resynchronization, RMAN updates the persistent configurations in the control file if they do not match the information in the recovery catalog for the specified database. For a sample use case, suppose that you recently connected RMAN as TARGET to the primary database and ran CONFIGURE to create an RMAN configuration for standby database standby_new. However, you have not yet connected RMAN as TARGET to standby_new. In this case, you can run RESYNC CATALOG FROM DB_UNIQUE_NAME standby_new. When you later connect RMAN to standby_new as TARGET, RMAN pushes the configuration from the recovery catalog to the mounted control file of standby_new. **Note:** The password file must be identical on all of the remote databases for this feature to work properly. You must manually copy it to all the remote databases in the configuration.

Examples

Example 3–27 Resynchronizing the Recovery Catalog in ARCHIVELOG Mode

This example performs a full resynchronization of the target database after archiving all unarchived redo log files.

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

recovery catalog database Password: password
connected to recovery catalog database

RMAN> SQL "ALTER SYSTEM ARCHIVE LOG CURRENT";
RMAN> RESYNC CATALOG;
```

Example 3–28 Resynchronizing the Recovery Catalog from a Control File Copy

Suppose you want to retrieve some backup information from a control file copy.

Assume that you start the RMAN client and connect to a target database and recovery catalog. The following commands shut down and mount the target database, update the RMAN repository in the current control file with metadata from a backup control file, and then open the database.

```
STARTUP FORCE MOUNT
RESYNC CATALOG FROM CONTROLFILECOPY '/disk1/cfile.dbf';
ALTER DATABASE OPEN;
```

Example 3–29 Resynchronizing the Recovery Catalog After a Structural Change

Suppose you have the following:

- A Data Guard environment containing primary database prod and standby database standby3
- Both the primary and the standby databases are registered with the catalog
- You take a backup of the database and archivelogs
- You start SQL*Plus, connect to database prod, and add a datafile to tablespace users as follows:

```
SQL> ALTER TABLESPACE users ADD DATAFILE ''?/oradata/prod/users03.dbf''
  2  SIZE 1M AUTOEXTEND ON
  3  NEXT 10K MAXSIZE 10M";
```

The goal is to update the recovery catalog with metadata about this change. After the change is propagated to standby3, you start the RMAN client, connect to standby3 as TARGET, and connect to the recovery catalog.

The next step is to define the connect identifiers for the standby database in your Data Guard environment with the CONNECT IDENTIFIER clause of the CONFIGURE command:

```
CONFIGURE DB_UNIQUE_NAME standby3 CONNECT IDENTIFIER 'inst2';
```

At this point, you use the RESYNC CATALOG ... ALL command to resynchronize the recovery catalog with changes for all the databases in the Data Guard environment. While you can resynchronize the catalog with a specific standby site's information, Oracle recommends you use the ALL option in order to keep the recovery catalog schema current with any database changes that may have occurred within your Data Guard environment:

```
RMAN> RESYNC CATALOG FROM DB_UNIQUE_NAME ALL;
```

The recovery catalog is updated with metadata about the datafile added to the users tablespace of database prod.

Example 3–30 Resynchronizing the Recovery Catalog with a Standby Database

Suppose that primary database prod and standby database dgprod3 exist in a Data Guard environment. Your goal is to create an RMAN configuration for dgprod3.

You connect RMAN to database prod as TARGET and then connect to the recovery catalog. You use CONFIGURE to update the persistent RMAN configuration for dgprod3 in the recovery catalog as follows:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt FOR DB_UNIQUE_NAME dgprod3;
CONFIGURE DB_UNIQUE_NAME dgprod3 CONNECT IDENTIFIER 'inst3';
```

You have not yet performed any backups or other RMAN operations on dgprod3, so the control file of dgprod3 and the recovery catalog metadata for dgprod3 are not synchronized. In the same RMAN session, you synchronize the dgprod3 control file with the recovery catalog as follows:

```
RESYNC CATALOG FROM DB_UNIQUE_NAME dgprod3;
```

RMAN updates the default device type to SBT at dgprod3 and also updates the recovery catalog with the names from the dgprod3 control file.

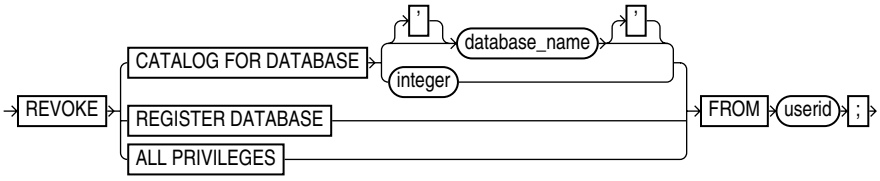
REVOKE

Purpose

Use the REVOKE command to revoke recovery catalog privileges previously granted with the GRANT command.

Syntax

***revoke*::=**



Prerequisites

Execute this command at the RMAN prompt or within the braces of a RUN command.

Usage Notes

Assume that a virtual private catalog user is granted the REGISTER DATABASE privilege, which implicitly grants the CATALOG FOR DATABASE privilege for any registered database. This user registers multiple databases. If you REVOKE the REGISTER DATABASE privilege from this user, then this user retains CATALOG FOR DATABASE privileges for the registered databases. The CATALOG privileges include registering and unregistering the specified databases.

To prevent this user from accessing the metadata for any databases or registering additional databases, execute REVOKE ALL PRIVILEGES for this user. To revoke CATALOG privileges for a subset of the databases registered by this user, execute REVOKE CATALOG FOR DATABASE for each database in the subset.

Semantics

Syntax Element	Description
CATALOG FOR DATABASE {*databasename* | *integer*}	Revokes recovery catalog access for the specified database from the specified user. You can specify the database by either database name or DBID. If you specify a database name when more than one database with this name is registered in the recovery catalog, then RMAN returns an error. In this case, specify the database by DBID.
REGISTER DATABASE	Revokes the ability to for the specified user to register new databases in this recovery catalog (see Example 3–31 on page 3-58).
ALL PRIVILEGES	Revokes all CATALOG and REGISTER privileges from the specified user.
FROM *userid*	Specifies the name of the user from which you are revoking privileges.

Examples

Example 3–31 Revoking Privileges from a Virtual Private Catalog Users

Assume that you connect RMAN to a base recovery catalog as the recovery catalog owner `rco`. As the base catalog owner, you use the RMAN `GRANT` command as follows to give `bckop2` the ability to register any database in her virtual private catalog, but grant `bckop3` access to only a subset of the databases in the data center:

```
RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> GRANT REGISTER DATABASE TO bckop2;
RMAN> GRANT CATALOG FOR DATABASE prod TO bckop3;
RMAN> GRANT CATALOG FOR DATABASE prodb TO bckop3;
RMAN> EXIT;
```

Later, you want to restrict the privileges for user `bckop2` so that this user can no longer register new databases, so you connect to the base catalog as `rco` and execute a `REVOKE` command. Note that `bckop2` retains catalog privileges on the database that this user has already registered.

```
RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> REVOKE REGISTER DATABASE FROM bckop2;
```

RMAN

Purpose

Use the `RMAN` command to start RMAN from the operating system command line.

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

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

See Also: *Oracle Database Backup and Recovery User's Guide* to learn how to start RMAN from the command line

Prerequisites

You must issue the `RMAN` command and any options at the operating system command line rather than at the RMAN prompt.

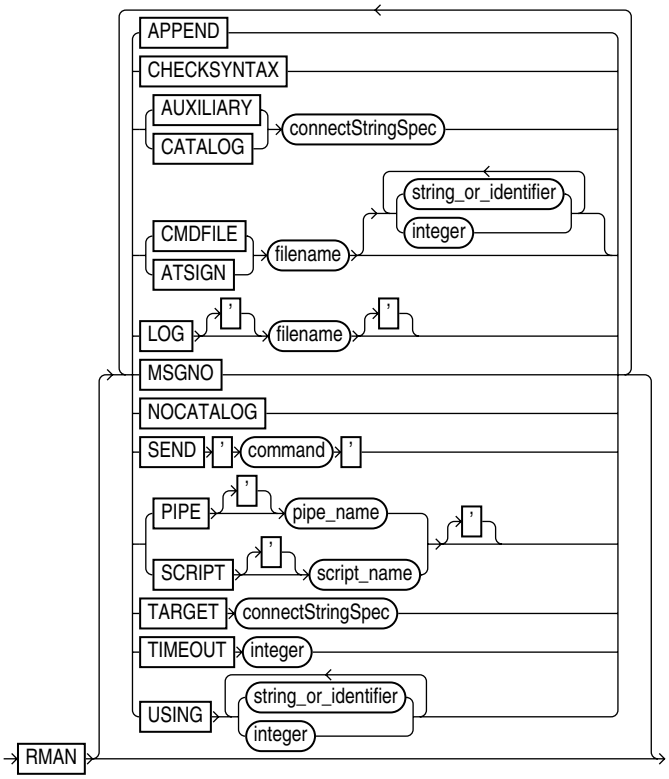
Usage Notes

The command name that you enter at the operating system prompt is operating system-dependent. For example, enter `rman` in lowercase on Linux and UNIX systems.

If you start RMAN without specifying either `CATALOG` or `NOCATALOG` on the operating system command line, then the RMAN session is effectively in `NOCATALOG` mode unless you execute a `CONNECT CATALOG` command (see Example 3–32 on page 3-62). If you maintain a recovery catalog, then the best practice is to connect RMAN to the recovery catalog before performing RMAN operations.

Syntax

***cmdLine*::=**



Semantics

cmdLine

Syntax Element	Description
`APPEND`	Causes new output to be appended to the end of the message log file. If you do not specify this parameter, and if a file with the same name as the message log file already exists, then RMAN overwrites it.
`CHECKSYNTAX`	Causes RMAN to start in a mode in which commands entered are checked for syntax errors, but no other processing is performed (see Example 3–35). If used with a `CMDFILE` or `@` argument, then the RMAN client starts, checks all commands in the file, then exits. If used without specifying a command file, then RMAN prompts the user for input and parses each command until the user exits the RMAN client. RMAN reports an `RMAN-0558` error for each command that is not syntactically correct.
`AUXILIARY` *`connectStringSpec`*	Specifies a connect string to an auxiliary database, for example, `AUXILIARY SYS@dupdb`. **See Also:** *`connectStringSpec`* on page 4-13
`CATALOG` *`connectStringSpec`*	Specifies a connect string to the database containing the recovery catalog, for example, `CATALOG catowner@inst2`. **See Also:** *`connectStringSpec`* on page 4-13

Syntax Element	Description
CMDFILE *filename*	Parses and compiles all RMAN commands in a file and then sequentially executes each command in the file. RMAN exits if it encounters a syntax error during the parse phase or if it encounters a run-time error during the execution phase. If no errors are found, then RMAN exits after the job completes. 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. **Note:** If you run a command file at the RMAN prompt rather than as an option on the operating system command line, then RMAN does *not* run the file as a single job. RMAN reads each line sequentially and executes it, only exiting when it reaches the last line of the script.
@*filename*	Equivalent to CMDFILE.
{*string_or_identifier* | *integer*}	Equivalent to options specified after USING syntax.
LOG *filename*	Specifies the file where RMAN records its output, that is, the commands that were processed and their results. RMAN displays command input at the prompt but does not display command output, which is written to the log file. By default RMAN writes its message log file to standard output. RMAN output is also stored in the V$RMAN_OUTPUT view, which is a memory-only view for jobs in progress, and in V$RMAN_STATUS, which is a control file view for completed jobs and jobs in progress. The LOG parameter does not cause RMAN to terminate if the specified file cannot be opened. Instead, RMAN writes to standard output. **Note:** The easiest way to send RMAN output both to a log file and to standard output is to use the Linux tee command or its equivalent. For example: `% rman | tee rman.log`
MSGNO	Causes RMAN to print message numbers, that is, RMAN-*xxxx*, for the output of all commands. By default, RMAN does not print the RMAN-*xxxx* prefix.
NOCATALOG	Indicates that you are using RMAN without a recovery catalog.
SEND '*command*'	Sends a vendor-specific command string to all allocated channels. **See Also:** Your media management documentation to determine whether this feature is supported, and SEND on page 3-68
PIPE *pipe_name*	Invokes the RMAN pipe interface. RMAN uses two public pipes: one for receiving commands and the other for sending output. The names of the pipes are derived from the value of the PIPE parameter. For example, you can invoke the RMAN pipe interface with the following options: PIPE rpi TARGET /. RMAN opens the following pipes in the target database: ■ ORA$RMAN_RPI_IN, which RMAN uses to receive user commands ■ ORA$RMAN_RPI_OUT, which RMAN uses to send all output All messages on both the input and output pipes are of type VARCHAR2. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to pass commands to RMAN through a pipe

Syntax Element	Description
SCRIPT *script_name*	Specifies the name of a stored script. After connecting to a target database and recovery catalog (which must be specified with the TARGET and CATALOG options), RMAN runs the named stored script from the recovery catalog against the target database. If both a global script and a local stored script exist on the target database with the name *script_name*, then RMAN runs the local script. The single quotes around the stored script name are required when the script name either begins with a number or is an RMAN reserved word (see "RMAN Reserved Words" on page 1-4). You should avoid creating script names that begin with a number or that match RMAN reserved words. **See Also:** CREATE SCRIPT on page 2-105 for more details about stored scripts
TARGET *connectStringSpec*	Specifies a connect string to the target database, for example, TARGET /. **See Also:** *connectStringSpec* on page 4-13
TIMEOUT *integer*	Causes RMAN to exit automatically if it does not receive input from an input pipe within *integer* seconds. The PIPE parameter must be specified when using TIMEOUT. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to pass commands to RMAN through a pipe
USING {*string_or_identifier* | *integer*}	Specifies one or more values for use in substitution variables in a command file. As in SQL*Plus, &1 indicates where to place the first value, &2 indicate where to place the second value, and so on. Example 3–34 on page 3-63 illustrates how to pass values specified in a USING clause to an RMAN command file. The substitution variable syntax is &*integer* followed by an optional dot, for example, &1.3. The optional dot is part of the variable and replaced with the value, thus enabling the substitution text to be immediately followed by another integer. For example, if you pass the value mybackup to a command file that contains the substitution variable &1.3, then the result of the substitution is mybackup3. **See Also:** EXECUTE SCRIPT on page 2-146 to learn how to specify the USING clause when executing a stored script

Examples

Example 3–32 Connecting RMAN to a Target Database in Default NOCATALOG Mode

In this example, you start the RMAN client without specifying database connection options at the operating system prompt. At the RMAN prompt, you run the CONNECT command to connect to a target database. Because CONNECT CATALOG was not run at the RMAN prompt, RMAN connects in default NOCATALOG mode when the first command requiring a repository connection is run, which in this case is the BACKUP DATABASE command.

```
% rman
RMAN> CONNECT TARGET /
RMAN> BACKUP DATABASE;
```

Example 3–33 Connecting RMAN to an Auxiliary Database Instance

This example connects to target database prod and recovery catalog database catdb with net service names, and connects to an auxiliary database instance with operating system authentication.

```
% rman TARGET SYS@prod

Recovery Manager: Release 11.1.0.6.0 - Production
```

```
Copyright (c) 1982, 2007, Oracle.  All rights reserved.

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

RMAN> CONNECT CATALOG rman@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> CONNECT AUXILIARY /
```

Example 3–34 Specifying Substitution Variables

Suppose that you want to create a Linux shell script that backs up the database. You want to use shell variables so that you can pass arguments to the RMAN backup script at run time. Substitution variables solve this problem. First, you create a command file named `whole_db.cmd` with the following contents:

```
cat > /tmp/whole_db.cmd <<EOF
# name: whole_db.cmd
CONNECT TARGET /
BACKUP TAG &1 COPIES &2 DATABASE FORMAT '/disk2/db_%U';
EXIT;
EOF
```

Next, you write the following Linux shell script, which sets `csh` shell variables `tagname` and `copies`. The shell script starts RMAN, connects to target database `prod1`, and runs `whole_db.cmd`. The `USING` clause passes the values in the variables `tagname` and `copies` to the RMAN command file at execution time.

```
#!/bin/csh
# name: runbackup.sh
# usage: use the tag name and number of copies as arguments
set tagname = $argv[1]
set copies = $argv[2]
rman @'/tmp/whole_db.cmd' USING $tagname $copies LOG /tmp/runbackup.out
# note that the preceding line is equivalent to:
# rman @'/tmp/whole_db.cmd' $tagname $copies LOG /tmp/runbackup.out
```

Finally, you execute the shell script `runbackup.sh` from a Linux shell as follows to create two backups of the database with the tag `Q106`:

```
% runbackup.sh Q106 2
```

Example 3–35 Checking the Syntax of a Command File

Suppose that you create command file `backup_db.cmd` as follows:

```
cat > /tmp/backup_db.cmd <<EOF
CONNECT TARGET /
BACKUP DATABASE;
EXIT;
EOF
```

The following example checks the syntax of the contents of command file `backup_db.cmd` (sample output included):

```
% rman CHECKSYNTAX @'/tmp/backup_db.cmd'

Recovery Manager: Release 11.1.0.6.0 - Production on Wed Jul 11 17:51:30 2007
```

```
Copyright (c) 1982, 2007, Oracle.  All rights reserved.

RMAN> CONNECT TARGET *
2> BACKUP DATABASE;
3> EXIT;
The cmdfile has no syntax errors

Recovery Manager complete.
```

Example 3–36 Running a Stored Script and Appending Output to a Message Log

This example connects to a target database using operating system authentication and then runs stored script `wdbb`. RMAN writes output to message log `/tmp/wdbb.log`.

```
% rman TARGET / SCRIPT wdbb LOG /tmp/wdbb.log
```

Example 3–37 Invoking the RMAN Pipe Interface

This example invokes the RMAN pipe `newpipe` with a 90 second timeout option.

```
% rman PIPE newpipe TARGET / TIMEOUT 90
```

RUN

Purpose

Use the RUN command to group a series RMAN commands into a block to be executed sequentially. On reading the closing brace of the RUN block, RMAN compiles the list of job commands into one or more job steps and then executes the steps immediately.

Prerequisites

Execute this command only at the RMAN prompt. You must precede the list of job commands with an opening brace ({) and terminate it with a closing brace (}).

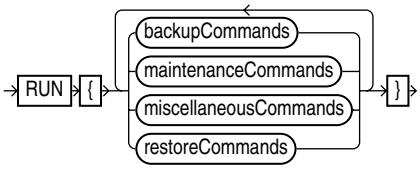
Usage Notes

You can use RUN to create a scope within which a script can override default configurations. For example, you can override configured channels with the ALLOCATE CHANNEL and RELEASE CHANNEL commands and other parameters with the SET command (as shown in Example 3–38 on page 3-66). After executing the commands listed in the RUN block, the channels allocated within the RUN block are released and settings returned to their values.

As shown in Example 3–39, you must use the EXECUTE SCRIPT command within a RUN block.

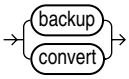
Syntax

***run*::=**

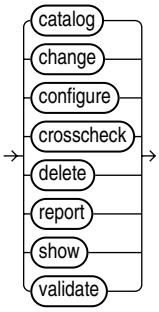


(*backupCommands::=* on page 3-65, *maintenanceCommands::=* on page 3-65, *miscellaneousCommands::=* on page 3-66, *restoreCommands::=* on page 3-66)

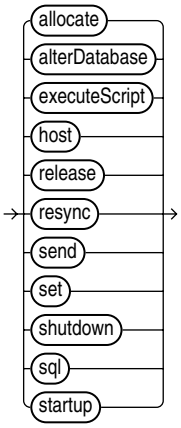
***backupCommands*::=**



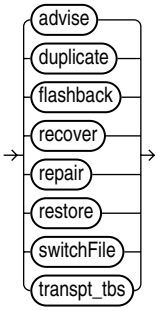
***maintenanceCommands*::=**



***miscellaneousCommands*::=**



***restoreCommands*::=**



Semantics

Refer to individual command entries for information about commands that you can run from the RMAN prompt.

Examples

Example 3–38 Overriding Configured Settings

Assume that your configured device configuration is as follows:

```
RMAN> SHOW DEVICE TYPE;

RMAN configuration parameters for database with db_unique_name PROD1 are:
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DEVICE TYPE SBT_TAPE PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
```

You want to make a backup to a nondefault directory. Instead of changing the configuration, you can override it in the job as follows:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE DISK FORMAT "/disk2/%U";
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

Example 3–39 Executing an RMAN Script

Assume that you use the `CREATE SCRIPT` command to create a backup script named `backup_db`. This example executes the stored script:

```
RUN { EXECUTE SCRIPT backup_db; }
```

SEND

Purpose

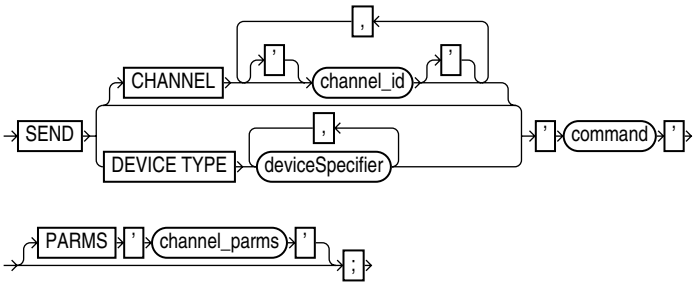
Use the SEND command to send a vendor-specific string to one or more channels supported by a media manager. Refer to your media management documentation to determine which commands are supported.

Usage Notes

Unless you specify DEVICE TYPE or CHANNEL, RMAN uses all allocated channels.

Syntax

***send*::=**



(*deviceSpecifier::=* on page 4-16)

Semantics

Syntax Element	Description
CHANNEL *channel_id*	Specifies which channel to use. You must specify a case-sensitive channel ID, which is the name of the channel, after the CHANNEL keyword. The database uses the channel ID to report I/O errors.
DEVICE TYPE *deviceSpecifier*	Specifies the type of storage device and sends the command to all channels of the specified type. **See Also:** *deviceSpecifier* on page 4-16
'*command*'	Specifies a vendor-specific media management command. **See Also:** Your media management documentation to determine which commands are supported. You must only send commands supported by the media manager. The contents of the string are not interpreted by the database, but are passed unaltered to the media management subsystem.
PARMS '*channel_parms*'	Specifies parameters for the channel communicating with the media manager.

Example

Example 3–40 Specifying a Tape Drive in Oracle Secure Backup

This example uses the SEND command to specify a tape drive for a backup of the users tablespace to Oracle Secure Backup. Note that no equal sign is inserted between the parameter OB_DEVICE and the names of the tape drive.

```
RUN
```

```
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  SEND 'OB_DEVICE stape1';
  BACKUP TABLESPACE users;
}
```

SET

Purpose

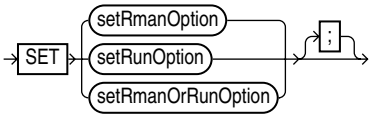
Use the SET command to control RMAN behavior within a job or session. Use CONFIGURE to configure options that persist across sessions.

Prerequisites

You can use the SET command either at the RMAN prompt or within a RUN block. When used at the RMAN prompt, changes made by SET persist until you exit the RMAN client (see *setRmanOption* on page 3-71). When used inside of a RUN block, changes made by SET persist until the end of the RUN block or the next SET command that changes the value of the same attribute (see *setRunOption* on page 3-74).

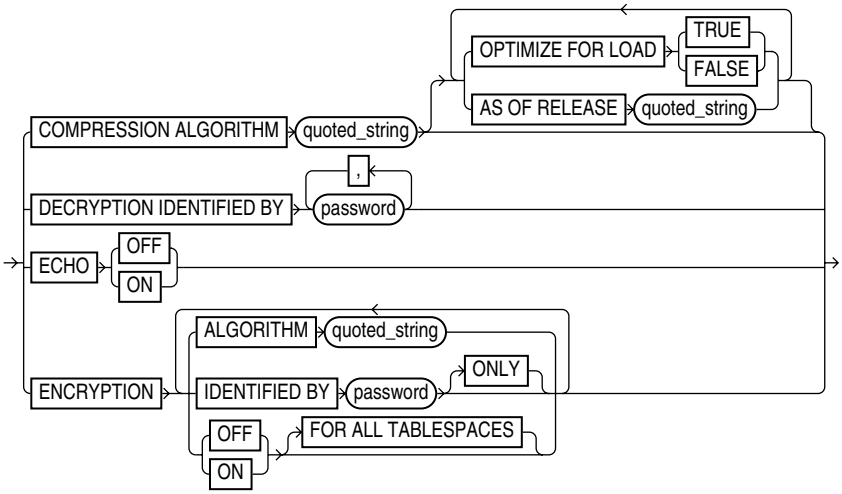
Syntax

***set*::=**



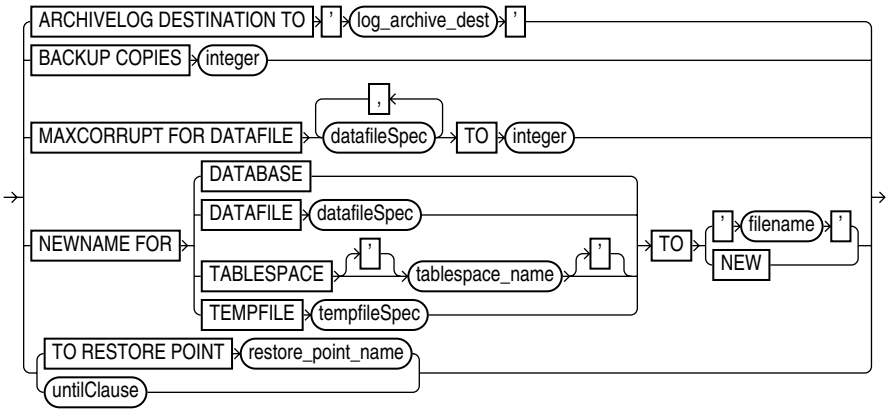
(*setRmanOption::=* on page 3-70, *setRunOption::=* on page 3-71)

***setRmanOption*::=**



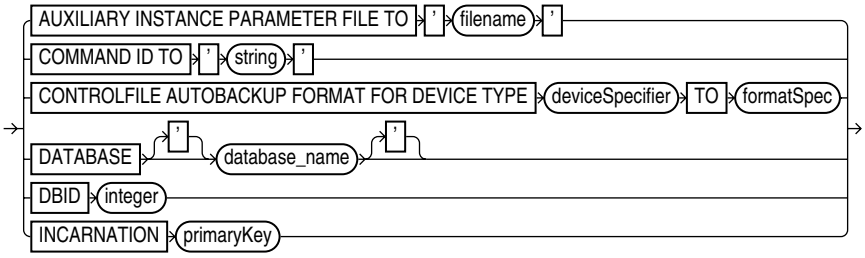
(*deviceSpecifier::=* on page 4-16, *formatSpec::=* on page 4-23)

setRunOption::=



(*deviceSpecifier::=* on page 4-16, *formatSpec::=* on page 4-23, *datafileSpec::=* on page 4-15, *tempfileSpec::=* on page 4-39, *untilClause::=* on page 4-42)

setRmanOrRunOption::=



(*deviceSpecifier::=* on page 4-16, *formatSpec::=* on page 4-23)

Semantics

setRmanOption

This subclause specifies `SET` options that are usable outside of a `RUN` block.

Syntax Element	Description
`COMPRESSION ALGORITHM` *'algorithm_name'*	Specifies the compression algorithm for backup sets. This command overrides the current `CONFIGURE COMPRESSION ALGORITHM` setting for the current RMAN session. You can configure the basic compression level, which does not require the Advanced Compression Option, by specifying `BASIC` for the algorithm_name. If you have enabled the Oracle Database 11*g* Release 2 Advanced Compression Option, you can choose from the following compression levels: ■ `HIGH` – Best suited for backups over slower networks where the limiting factor is network speed ■ `MEDIUM` – Recommended for most environments. Good combination of compression ratios and speed ■ `LOW` - Least impact on backup throughput and suited for environments where CPU resources are the limiting factor. **Note:** The compression ratio generally increases from `LOW` to `HIGH`, with a trade-off of potentially consuming more CPU resources. Since the performance of the various compression levels depends on the nature of the data in the database, network configuration, system resources and the type of computer system and its capabilities, Oracle cannot document universally applicable performance statistics. The decision as to which level is best must factor in on how balanced your system is regarding bandwidth into the CPU as well as the actual speed of the CPU. It is highly recommended that you run tests with the different compression levels on the data in your environment. Choosing a compression level based on your environment, network traffic characteristics (workload) and dataset is the only way to ensure that the backup set compression level can satisfy your organization's performance requirements and any applicable service level agreements. **Note:** `V$RMAN_COMPRESSION_ALGORITHM` describes supported algorithms. **See Also:** *Oracle Database Reference* entry for `$RMAN_COMPRESSION_ALGORITHM`.
`OPTIMIZE FOR LOAD {TRUE|FALSE}`	Specifies whether Oracle performs pre-compression block processing when compressed backups have been requested. `TRUE` is the default and `FALSE` enables pre-compression processing. The default behavior is not to perform pre-compression block processing. Such processing can consume extra CPU resources, and is not needed for blocks that contain all originally loaded data, and have never been the subject of single-row inserts and deletes. Specifying `FALSE` uses additional CPU resources to perform pre-compression block processing which consists of internal block cleanups and defragmentation that can result in improved levels of binary compression. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn more about this option.
`AS OF RELEASE` *'version number'*	Specifies the release version. The version number uses the release number format and may use as many as 5 numbers to fully qualify the release. For example, 10.2.0.3.0 and 11.2 are acceptable values. This option ensures compression algorithm stability across future releases.
`DECRYPTION IDENTIFIED BY` *password*	Specifies one or more decryption passwords to be used when reading dual-mode or password-encrypted backups. Password-encrypted backups require the correct password to be entered before they can be restored. When RMAN reads an encrypted backup piece, it tries each password in the list until it finds the correct one to decrypt this backup piece. RMAN signals an error if none of the specified keys work. **Note:** If restoring from a group of backups created with different passwords, then specify all of the required passwords on the `SET DECRYPTION` command. RMAN will automatically use the correct password with each backup set. **See Also:** "Encryption of Backup Sets" on page 2-21

Syntax Element	Description
ECHO {OFF | ON}	Controls whether RMAN commands appear in the message log. When reading commands from a command file, RMAN automatically echoes them to the message log. When reading commands from standard input, by default RMAN does not echo these commands to the message log. To force RMAN to echo the commands, run the SET ECHO ON command before running your command file. Run SET ECHO OFF to disable echoing to the command log. The command is useful when stdin and stdout have been redirected. For example, in UNIX you can redirect RMAN's input and output in this manner: % rman TARGET / < in_file > out_file By including SET ECHO ON in the in_file, you enable the commands contained in in_file to be visible in out_file.
ENCRYPTION	Specifies encryption-related options that apply to BACKUP commands that create backup sets for the duration of the RMAN session. **See Also:** "Encryption of Backup Sets" on page 2-21
ALGORITHM '*algorithm_name*'	Specifies the algorithm used during this RMAN session. Overrides the configured default encryption algorithm specified by CONFIGURE ALGORITHM. Possible values are listed in V$RMAN_ENCRYPTION_ALGORITHMS.
IDENTIFIED BY *password* [ONLY]	Specifies whether to employ a user-specified password in backup encryption according to the following rules: ■ Omit IDENTIFIED BY *password* clause to specify transparent-mode encrypted backups. ■ Use IDENTIFIED BY *password* ONLY to specify password-mode encrypted backups. ■ Use IDENTIFIED BY *password* without ONLY to specify dual-mode encrypted backups. Create a password that is secure. See *Oracle Database Security Guide* for more information. Note that if the password is *not* surrounded by quotes, then it is translated internally into upper case. Thus, the following clauses are all synonyms for IDENTIFIED BY "SOMEPWD": ■ IDENTIFIED BY somepwd ■ IDENTIFIED BY Somepwd ■ IDENTIFIED BY sOmEpWd **Caution:** Wallet-based encryption is more secure than password-based encryption because no passwords are involved. You should use password-based encryption only when absolutely necessary because your backups need to be transportable. **See Also:** "Encryption of Backup Sets" on page 2-21 for details on the different encryption modes
{OFF | ON}	Specifies whether to encrypt backup sets. If ON, then the default is to encrypt backup sets. If OFF, then the default is not to encrypt backup sets. This option overrides settings made with the CONFIGURE ENCRYPTION FOR command. If no datafiles are configured for encryption, then you must explicitly use ON to encrypt required datafiles. If FOR ALL TABLESPACES is not specified, then this setting controls encryption of backups for tablespaces where CONFIGURE ENCRYPTION FOR TABLESPACE *tablespace_name* has not been used to control encryption behavior.
FOR ALL TABLESPACES	Controls encryption for all tablespaces, overriding any CONFIGURE ENCRYPTION FOR TABLESPACE *tablespace_name* setting.

setRunOption

This subclause specifies SET options that are usable within a RUN block.

Syntax Element	Description
ARCHIVELOG DESTINATION TO '*log_archive_dest*'	Overrides the LOG_ARCHIVE_DEST_1 initialization parameter in the target database when forming names for restored archived redo log files during subsequent RESTORE and RECOVER commands. RMAN restores the logs to the destination specified in '*log_archive_dest*'. You can use this command to stage archived redo log files to different locations while restoring a database. RMAN knows where to find the newly restored archived redo log files; it does not require them to be in the destination specified by LOG_ARCHIVE_DEST_1. For example, if you specify a different destination from the one in the parameter file and restore archived log backups, subsequent restore and recovery operations detect this new location. Use this parameter to restore archived redo log files that are not already on disk. RMAN always looks for logs on disk first before restoring them from backups.
BACKUP COPIES *integer*	Specifies the number of copies of each backup piece that the channels should create: 1, 2, 3, or 4 (see Example 3–42 on page 3-79). RMAN can duplex backups to either disk or tape but cannot duplex backups to tape and disk simultaneously. When backing up to tape, ensure that the number of copies does not exceed the number of available tape devices. Also, if BACKUP COPIES is greater than 1, then the BACKUP_TAPE_IO_SLAVES initialization parameter must be enabled on the target database. The SET BACKUP COPIES command affects all BACKUP commands in the RUN block issued after SET BACKUP COPIES (but not before) and is in effect until explicitly disabled or changed. The SET BACKUP COPIES command affects only BACKUP commands, but does not apply to the BACKUP AS COPY command. The SET BACKUP COPIES command affects all channels allocated in the session. The order of precedence is as follows, with settings higher on the list overriding settings lower on the list: **1.** BACKUP COPIES **2.** SET BACKUP COPIES **3.** CONFIGURE ... BACKUP COPIES The names of the backup pieces are dependent on the FORMAT clause in the BACKUP command. You can specify up to four FORMAT strings. RMAN uses the second, third, and fourth values only when BACKUP COPIES, SET BACKUP COPIES, or CONFIGURE ... BACKUP COPIES is in effect. When choosing which format to use for each backup piece, RMAN uses the first format value for copy 1, the second format value for copy 2, and so on. If the number of format values exceeds the number of copies, then the extra formats are not used. If the number of format values is less than the number of copies, RMAN reuses the format values, starting with the first one. **Note:** BACKUP COPIES option is not valid when files are created in fast recovery area. Backups to the fast recovery area cannot be duplexed. **Note:** Control file autobackups on disk are a special case and are never duplexed: RMAN always writes one and only copy.

Syntax Element	Description
MAXCORRUPT FOR DATAFILE *datafileSpec* TO *integer*	Sets a limit on the number of previously undetected block corruptions that the database will permit in a specified datafile or group of datafiles. The default limit is zero, meaning that RMAN tolerates no corrupt blocks. The SET MAXCORRUPT command specifies the total number of physical and logical corruptions permitted in a datafile during a backup job. If the sum of physical and logical corruptions detected for a datafile is no more than its MAXCORRUPT setting, then the BACKUP command completes. If more than MAXCORRUPT corrupt blocks exist, then RMAN terminates without creating output files. Whether or not the MAXCORRUPT limit is exceeded, RMAN populates the V$DATABASE_BLOCK_CORRUPTION view with any corrupt block ranges that it finds. However, a backup or restore job is terminated after MAXCORRUPT+1 corrupt blocks are found, so in this case RMAN only records MAXCORRUPT+1 corruptions. Any block corruptions beyond the point at which the backup job terminated are not recorded. **Note:** If you specify CHECK LOGICAL, then the MAXCORRUPT limit applies to the sum of logical and physical corruptions detected. Otherwise, MAXCORRUPT only applies to the number of physical block corruptions. **See Also:** *datafileSpec* on page 4-15

NEWNAME FOR DATABASE	Sets new default names for all datafiles and tempfiles in the specified database that have not been named with the SET NEWNAME FOR TABLESPACE, SET NEWNAME FOR TEMPFILE or SET NEWNAME FOR DATAFILE command.
	
	This command enables you to change the names for multiple files in the database and provides a quick alternative to naming each file individually (see Example 2–76 on page 2-140). This command does not set names for tempfiles.
	
	When issuing SET NEWNAME FOR DATABASE, you must specify at least one of the first three of the following substitution variables to avoid name collisions: %b, %f, and %U. See the semantic entry for TO '*filename*' for descriptions of the possible substitution variables.
	
	The new names are used for all subsequent DUPLICATE, RESTORE, SWITCH commands that affect the files in the database and when using the RMAN TSPITR utility. If you do not issue this command before the restore operation, then RMAN restores the files to their default locations.
	
	You do not have to issue the SET NEWNAME commands in any particular order. For example, assume that you run the following series of commands:
	
	`SET NEWNAME FOR DATABASE TO '/oradata1/%b';`
	`SET NEWNAME FOR TABLESPACE users TO '/oradata2/%U';`
	`SET NEWNAME FOR DATAFILE 1 TO '+data';`
	
	The preceding series of commands is equivalent to the following differently ordered series of commands:
	
	`SET NEWNAME FOR DATAFILE 1 TO '+data;`
	`SET NEWNAME FOR DATABASE TO '/oradata1/%b';`
	`SET NEWNAME FOR TABLESPACE users TO '/oradata2/%U';`
	
	Note: The SET NEWNAME command supports ASM disk groups.
	
	See Also: *formatSpec* on page 4-23 to learn about substitution variables that are valid in SET NEWNAME FOR DATABASE

Syntax Element	Description
NEWNAME FOR DATAFILE *datafileSpec*	Sets the default name for the specified datafile. This command enables you to specify the names of each datafile individually (see Example 2–77 on page 2-141). The new names are used for all subsequent DUPLICATE, RESTORE, SWITCH commands that affect the files in the database and when using the RMAN TSPITR utility. If you run SET NEWNAME FOR DATAFILE and then restore a datafile to a new location, then you can run SWITCH to rename the file in the control file to the NEWNAME. If you do not run SWITCH, then RMAN records the restored file as a datafile copy in the RMAN repository. **Note:** The SET NEWNAME command supports ASM disk groups. **See Also:** *datafileSpec* on page 4-15
NEWNAME FOR TABLESPACE *tablespace_name*	Sets new default names for all the files in the specified tablespace that have not been named with the SET NEWNAME FOR DATAFILE command. This command allows you to change the names for multiple files in the duplicate tablespace and provides an alternative to naming each file of the tablespace individually (see Example 2–77 on page 2-141). When issuing SET NEWNAME FOR TABLESPACE, you must specify at least one of the first three of the following substitution variables to avoid name collisions: %b, %f, and %U. See the semantic entry for TO '*filename*' for descriptions of the possible substitution variables. The new names are used for all subsequent DUPLICATE, RESTORE or SWITCH commands that affect the files in the tablespace. If you do not issue this command before the restore operation, then RMAN restores the files to their default locations. **Note:** The SET NEWNAME command supports ASM disk groups. **See Also:** *formatSpec* on page 4-23 to learn about substitution variables that are valid in SET NEWNAME FOR DATABASE
NEWNAME FOR TEMPFILE *tempfileSpec*	Sets the new tempfile name for a subsequent SWITCH command that renames the specified tempfile to the specified name. **Note:** The SET NEWNAME command supports ASM disk groups. **See Also:** *tempfileSpec* on page 4-39

Syntax Element	Description
`TO 'filename'`	Specifies a user-defined filename or ASM disk group for the restored datafile or tempfile. When issuing `SET NEWNAME FOR DATABASE` or `SET NEWNAME FOR TABLESPACE`, you must specify substitution variables within *`filename`* to avoid name collisions. Specify at least one of the first three of the following substitution variables (the `%I` and `%N` variables are optional): ■ `%b` Specifies the filename without the fully qualified directory path. For example, the datafile name `/oradata/prod/financial.dbf` is transformed to `financial.dbf`. This variable enables you to preserve the names of the datafiles while you move them to different directory. During backup, it can be used for the creation of image copies. The variable cannot be used for OMF datafiles or backup sets. ■ `%f` Specifies the absolute file number of the datafile for which the new name is generated. For example, if datafile 2 is duplicated, then `%f` generates the value `2`. ■ `%U` Specifies a system-generated unique filename. The name is in the following format: `data-D-%d_id-%I_TS-%N_FNO-%f`. The `%d` variable specifies the database name. For example, a possible name might be `data-D-prod_id-22398754_TS-users_FNO-7`. ■ `%I` Specifies the DBID. ■ `%N` Specifies the tablespace name. **Note:** Use of other substitution variables defined in formatSpec is undefined: some yield an error, while others result in zero values and are not supported. If you set the `NEWNAME` to an ASM disk group for a datafile and run `RESTORE`, then RMAN restores the file to the disk group. If you specify a filename for a tempfile, then this will the new name of the tempfile after the database is recovered and opened.
`TO NEW`	Creates an Oracle-managed file in the directory specified by the `DB_CREATE_FILE_DEST` initialization parameter. For example, `SET NEWNAME FOR TABLESPACE users TO NEW` sets OMF names for the datafiles of the `users` tablespace to be created in the `DB_CREATE_FILE_DEST` location (see Example 2–78 on page 2-142). If the original file is an Oracle-managed file or is on an ASM disk group, then RMAN attempts to delete the original file. If you specify `TO NEW` for a tempfile, then RMAN creates the tempfile in `DB_CREATE_FILE_DEST` when the database is opened. **See Also:** *Oracle Database Administrator's Guide* for information about Oracle-managed files
`TO RESTORE POINT` *`restore_point_name`*	Specifies a restore point for a subsequent `RESTORE` or `RECOVER` command, with the SCN at which the restore point was created as the upper, inclusive limit. Because the limit is inclusive, RMAN selects only files that can be used to restore or recover up to *and including* the SCN corresponding to the restore point. **Note:** You can only use `SET TO RESTORE POINT` when the database is mounted, because the defined restore points are recorded in the control file. For example, you cannot use `SET TO RESTORE POINT` to specify the target SCN for a `RESTORE CONTROLFILE` operation.

Syntax Element	Description
untilClause	Specifies an end time, SCN, or log sequence number for a subsequent RESTORE or RECOVER command. **See Also:** untilClause on page 4-42

setRmanOrRunOption

This subclause specifies SET options that are usable inside or outside of a RUN block.

Syntax Element	Description
AUXILIARY INSTANCE PARAMETER FILE TO '*filename*'	Specifies the path to the parameter file to use in starting the instance. You can use this parameter when customizing TSPITR with an automatic auxiliary instance or when cloning RMAN tablespaces with RMAN. **Note:** The *filename* is on the host running the RMAN client. **See Also:** *Oracle Database Reference* for more on V$SESSION.CLIENT_INFO
COMMAND ID TO '*string*'	Enters the specified string into the V$SESSION.CLIENT_INFO column of all channels. Use this information to determine which database server sessions correspond to which RMAN channels. The SET COMMAND ID command applies only to channels that are already allocated. 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:** *Oracle Database Reference* for more on V$SESSION.CLIENT_INFO
CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE *deviceSpecifier* TO *formatSpec*	Overrides the default filename format for the control file autobackup on the specified device type. You can use this command either in RUN or at the RMAN prompt. The order of precedence is as follows: **1.** SET CONTROLFILE AUTOBACKUP executed within a RUN block **2.** SET CONTROLFILE AUTOBACKUP executed at the RMAN prompt **3.** CONFIGURE CONTROLFILE AUTOBACKUP FORMAT The %F substitution variable is required to be in the new *formatSpec*. No other substitution variable is valid in a control file autobackup *formatSpec*. **See Also:** *formatSpec* on page 4-23 for the semantics of the %F substitution variable
DATABASE '*database_name*'	Specifies the name of the database that you want to copy (source database) when duplicating without a connection to the target database. Alternatively, you can use the DATABASE clause of the DUPLICATE command to specify the source database when you have chosen not to connect to the target database for the duplicate.

Syntax Element	Description
DBID *integer*	Specifies the DBID, which is a unique 32-bit identification number computed when the database is created. RMAN displays the DBID upon connection to the target database. You can obtain the DBID by querying the V$DATABASE view or the RC_DATABASE and RC_DATABASE_INCARNATION recovery catalog views. You should only run the SET DBID command in the following specialized circumstances: ■ You are not connected to a recovery catalog and want to restore the control file (see Example 3–43 on page 3-80). The same restriction applies when you use the Data Recovery Advisor to restore a control file autobackup. CONFIGURE can locate an autobackup and restore it only if SET DBID is issued before ADVISE FAILURE. ■ You want to restore the server parameter file (see Example 3–44). ■ You are connected to the recovery catalog but not the target database and use the FOR DB_UNIQUE_NAME option on the CONFIGURE, LIST, REPORT, SHOW, or UNREGISTER commands. ■ You are connected to the recovery catalog and want to restore the controlfile but the target database is not mounted and the database name is not unique in the recovery catalog. ■ You are performing a DUPLICATE without a target connection but are not using the DBID subclause in the DUPLICATE command and the database name specified in SET DATABASE or DATABASE clause is not unique in the recovery catalog.
INCARNATION *primaryKey*	Allows you to specify an orphan incarnation when duplicating without a target connection under the following conditions: ■ You have not specified INCARNATION within the DATABASE clause of the DUPLICATE command. ■ You want to duplicate to an incarnation not in the current incarnation path (orphan incarnation).

Examples

Example 3–41 Setting the Command ID

This example sets the command ID to rman, backs up the database, and then archives the online redo logs. You can use the command ID to query V$SESSION with WHERE LIKE '%rman%' for job status information.

```
RUN
{
  ALLOCATE CHANNEL d1 DEVICE TYPE DISK FORMAT '/disk1/%U';
  ALLOCATE CHANNEL d2 DEVICE TYPE DISK FORMAT '/disk2/%U';
  SET COMMAND ID TO 'rman';
  BACKUP INCREMENTAL LEVEL 0 DATABASE;
  SQL 'ALTER SYSTEM ARCHIVE LOG CURRENT';
}
```

Example 3–42 Duplexing a Backup Set

Assume that the current duplexing configuration is as follows:

```
CONFIGURE ARCHIVELOG COPIES FOR DEVICE TYPE sbt TO 3;
CONFIGURE DATAFILE COPIES FOR DEVICE TYPE sbt TO 3;
```

One of the tape drives goes bad, leaving only two available. The guideline for tape backups is that the number of devices should equal the number of copies multiplied

by the number of channels. The following example overrides the persistent duplexing configuration with SET BACKUP COPIES and writes two copies of a database backup to the two functioning tape drives:

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE sbt
    PARMS 'ENV=(OB_DEVICE_1=stape1,OB_DEVICE_2=stape2)';
  SET BACKUP COPIES 2;
  BACKUP DATABASE PLUS ARCHIVELOG;
}
```

Example 3–43 Setting the Control File Autobackup Format During a Restore

Assume that the disk containing the control file fails. You edit the CONTROL_FILES parameter in the initialization parameter file to point to a new location.

In this example, you do not have access to a recovery catalog. The example starts the instance, sets the DBID, and then restores a control file autobackup. After the database is mounted, you can recover the database.

```
CONNECT TARGET /
STARTUP FORCE NOMOUNT
SET DBID 28014364;
RUN
{
  SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/disk2/cf_%F.bak';
  RESTORE CONTROLFILE FROM AUTOBACKUP MAXSEQ 100;
}
ALTER DATABASE MOUNT;
RECOVER DATABASE;
ALTER DATABASE OPEN RESETLOGS;
```

Example 3–44 Restoring the Server Parameter File

Assume that the database is shut down while maintenance is being performed on the database host. During this time, the server parameter file is accidentally deleted. You start the RMAN client, CONNECT as TARGET to the database, and connect to the recovery catalog. The following example restores a server parameter file from an autobackup on tape and then restarts the instance.

```
SET DBID 3257174182;  # set dbid so RMAN can identify the database
STARTUP FORCE NOMOUNT # RMAN starts database with a dummy server parameter file
RUN
{
  ALLOCATE CHANNEL t1 DEVICE TYPE sbt;
  RESTORE SPFILE FROM AUTOBACKUP;
}
STARTUP FORCE; # RMAN restarts database with restored server parameter file
```

Example 3–45 Setting NEWNAME and Duplicating Without a Connection to Target Database

Assume that you want to duplicate a database without a connection to the target database and that you want to duplicate to an incarnation that is not in the current incarnation path (451). The following example uses the various SET NEWNAME commands, sets the DBID and duplicates the database to NEWDB:

```
SET DATABASE PROD
```

```
SET DBID 22398754
SET INCARNATION 451
RUN
{
SET NEWNAME FOR TABLESPACE system TO '/test/oradata/system/%d_%f';
SET NEWNAME FOR TABLESPACE users to '/test/oradata/users/%b';
SET NEWNAME FOR DATAFILE 35 to '/test/oradata/special/%N_%b_%f';
SET NEWNAME FOR DATAFILE 50 to '/test/oradata/special/%N_%b_%f';
SET NEWNAME FOR DATABASE to NEW;
DUPLICATE DATABASE TO newdb
SKIP READONLY
LOGFILE
GROUP 1 ('/test/onlinelogs/redo01_1.f',
'?/test/onlinelogs/redo01_2.f') SIZE 4M,
GROUP 2 ('?/test/onlinelongs/redo02_1.f',
'?/test/onlinelongs/redo02_2.f') SIZE 4M,
GROUP 3 ('?/test/onlinelogs/redo03_1.f',
'?/test/onlinelogs/redo03_2.f') SIZE 4M REUSE;
}
```

Example 3–46 Setting a Compression Level for a Backup

This example assumes that you have a license for Advanced Compression Option (ACO) of the database.

To use the `LOW` compression algorithm for a backup of tablespace `users` that has high volatility:

```
SET COMPRESSION ALGORITHM 'LOW' OPTIMIZE FOR LOAD FALSE;
BACKUP AS COMPRESSED BACKUPSET TABLESPACE USERS;
```

SHOW

Purpose

Use the `SHOW` command to display the `CONFIGURE` commands used to set the current RMAN configuration for one or more databases. RMAN default configurations are suffixed with `#default`.

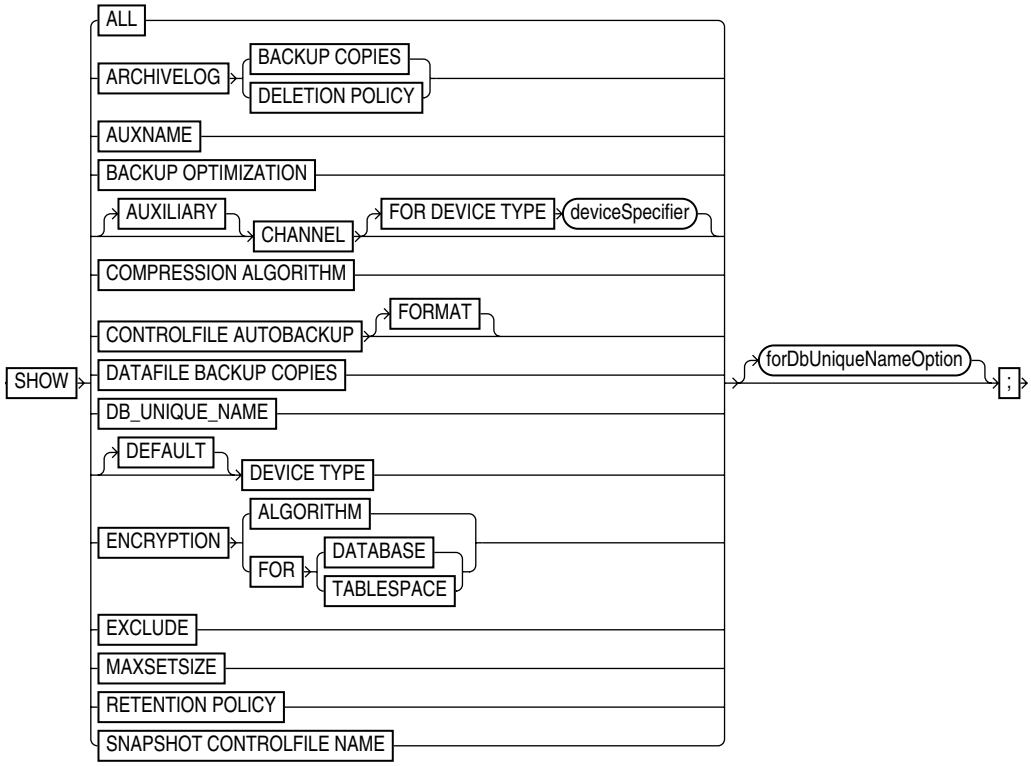
Prerequisites

Execute this command only at the RMAN prompt. Either of the following conditions must be met:

- RMAN must be connected to a target database, which must be mounted or open.
- RMAN must be connected to a recovery catalog and `SET DBID` must have been run.

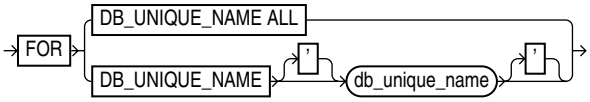
Syntax

***show*::=**



(*deviceSpecifier::=* on page 4-16)

forDbUniqueNameOption::=



Semantics

Syntax Element	Description
ALL	Shows all user-entered CONFIGURE commands as well as default configurations.
ARCHIVELOG BACKUP COPIES	Shows the currently configured degree of duplexing for archived redo log backups.
ARCHIVELOG DELETION POLICY	Shows the CONFIGURE ARCHIVELOG DELETION POLICY setting.
AUXNAME	Shows the CONFIGURE AUXNAME settings.
BACKUP OPTIMIZATION	Shows the CONFIGURE BACKUP OPTIMIZATION settings: ON or OFF (default).
[AUXILIARY] CHANNEL	Shows the CONFIGURE CHANNEL settings. You can specify a normal channel or an AUXILIARY channel.
FOR DEVICE TYPE *deviceSpecifier*	Specifies the device type of the channel. For example, SHOW CHANNEL FOR DEVICE TYPE DISK shows only channel settings for disk channels.
COMPRESSION ALGORITHM	Shows the configured backup compression algorithm.
CONTROLFILE AUTOBACKUP	Shows the CONFIGURE CONTROLFILE AUTOBACKUP settings: ON or OFF.
FORMAT	Shows the format for the control file autobackup file for configured devices.
DATAFILE BACKUP COPIES	Shows the CONFIGURE ... BACKUP COPIES setting for datafiles: 1, 2, 3, or 4.
DB_UNIQUE_NAME	Shows the DB_UNIQUE_NAME values known to the recovery catalog.
[DEFAULT] DEVICE TYPE	Shows the configured device types and parallelism settings. If DEFAULT is specified, then SHOW displays the default device type and settings.
ENCRYPTION	Shows currently configured encryption settings for the database or tablespaces within the database, when used with ALGORITHM or FOR {DATABASE | TABLESPACE}.
ALGORITHM	Shows the configured default algorithm to use for encryption when writing encrypted backup sets. Possible values are listed in V$RMAN_ENCRYPTION_ALGORITHMS.
FOR DATABASE	Shows current encryption settings for the database.
FOR TABLESPACE	Shows current encryption settings for each tablespace.
EXCLUDE	Shows only the tablespaces that you have specified should be excluded.
MAXSETSIZE	Shows the CONFIGURE MAXSETSIZE settings.
RETENTION POLICY	Shows the settings for CONFIGURE RETENTION POLICY for the current target database.
SNAPSHOT CONTROLFILE NAME	Shows the CONFIGURE SNAPSHOT CONTROLFILE settings.

Syntax Element	Description
forDbUniqueNameOption	Shows the configuration in the recovery catalog for a uniquely named database even when RMAN is not connected to this database as TARGET. You can specify a database with *db_unique_name* or use ALL for all uniquely named databases. The unique name for a database is the value of its DB_UNIQUE_NAME initialization parameter setting. The FOR DB_UNIQUE_NAME clause is useful for showing the configurations of standby databases in a Data Guard environment. RMAN must be connected to a recovery catalog. RMAN must be connected to a mounted target database or you must identify the target database with SET DBID. For example, you could run SET DBID with the DBID of the target database and then show the configuration for all standby databases known to the recovery catalog (see Example 2–47 on page 2-84). **See Also:** *forDbUniqueNameOption* on page 4-20 for descriptions of the options in this clause

Examples

Example 3–47 Showing All Configurations for a Target Database

Assume that you want to know all persistent RMAN configurations for a target database. You start the RMAN client, CONNECT to the target database and recovery catalog, and run the SHOW command as follows (sample output included):

```
RMAN> SHOW ALL;

RMAN configuration parameters for database with db_unique_name PROD1 are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/disk1/oracle/dbs/%F';
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE SBT_TAPE TO '%F'; # defa ult
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DEVICE TYPE SBT_TAPE PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE'
  PARMS "SBT_LIBRARY=/usr/local/oracle/backup/lib/libobk.so";
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE ON;
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/disk1/oracle/dbs/cf_snap .f'
```

SHUTDOWN

Purpose

Use the SHUTDOWN command to shut down the target database without exiting RMAN. This command is equivalent to the SQL*Plus SHUTDOWN statement.

> **See Also:** *Oracle Database 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

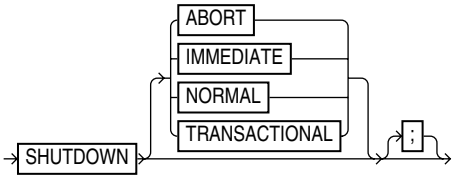
Usage Notes

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.

If the database operates in NOARCHIVELOG mode, then you must shut down the database cleanly and then issue a STARTUP MOUNT before a making a backup.

Syntax

***shutdown*::=**



Semantics

Syntax Element	Description
ABORT	Performs an inconsistent shutdown of the target instance, with the following consequences: ▪ All current client SQL statements are immediately terminated. ▪ Uncommitted transactions are not rolled back until next startup. ▪ All connected users are disconnected. ▪ Instance recovery will be performed on the database at next startup.
IMMEDIATE	Performs an immediate, consistent shutdown of the target database, with the following consequences: ▪ Current client SQL statements being processed by the database are allowed to complete. ▪ Uncommitted transactions are rolled back. ▪ All connected users are disconnected.

Syntax Element	Description
NORMAL	Performs a consistent shutdown of the target database with normal priority (default option), which means: ■ No new connections are allowed after the statement is issued. ■ Before shutting down, the database waits for currently connected users to disconnect ■ The next startup of the database will not require instance recovery.
TRANSACTIONAL	Performs a consistent shut down of 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 terminate 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 terminated, any client still connected is disconnected.

Examples

Example 3–48 Shutting Down a Database with the Immediate Option

This example waits for current SQL transactions to be processed before shutting down, then mounts the database:

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

Example 3–49 Shutting Down a Database in NOARCHIVELOG Mode

This example backs up a database running in NOARCHIVELOG mode:

```
STARTUP FORCE DBA;
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
# executing the preceding commands ensures that database is in proper state
# for NOARCHIVELOG backups
BACKUP DATABASE;
ALTER DATABASE OPEN;
```

SPOOL

Purpose

Use the SPOOL command to direct RMAN output to a log file.

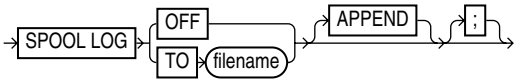
See Also: RMAN on page 3-59 for a description of LOG files

Prerequisites

Execute the SPOOL command at the RMAN prompt.

Syntax

***spool*::=**



Semantics

Syntax Element	Description
OFF	Turns off spooling.
TO *filename*	Specifies the name of the log file to which RMAN directs its output. RMAN creates the file if it does not exist, or overwrites the file if it does exist. If the specified file cannot be opened for writing, then RMAN turns SPOOL to OFF and continues execution.
APPEND	Appends the RMAN output to the end of the existing log.

Example

Example 3–50 Spooling RMAN Output to a File

This example directs RMAN output to standard output for configuration of the default device type, spools output of the SHOW command to log file current_config.log, and then spools output to db_backup.log for the whole database backup:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
SPOOL LOG TO '/tmp/current_config.log';
SHOW ALL;
SPOOL LOG OFF;
SPOOL LOG TO '/tmp/db_backup.log';
BACKUP DATABASE;
SPOOL LOG OFF;
```

SQL

Purpose

Use the `SQL` command to execute a SQL statement or a PL/SQL stored procedure from within RMAN.

See Also: *Oracle Database SQL Language Reference*

Prerequisites

None.

Syntax

***sql*::=**

Semantics

Syntax Element	Description
`CHANNEL` *`channel_id`*	Specifies the case-sensitive name of a channel to use when executing an RMAN command within a `RUN` command. The channel must have been allocated by means of `ALLOCATE CHANNEL` in this `RUN` command. If you do not set this parameter, then RMAN uses the default channel.
`'`*`command`*`'`	Specifies a SQL statement for execution (see Example 3–51 on page 3-88). `SELECT` statements are not permitted. You must use duplicate single quotes to insert a single quote into a quoted string when the quoted string uses the same style of quoting. For example, if the string that RMAN passes to SQL contains a filename, then the filename must be enclosed in duplicate *single* quotes and the entire string following the `SQL` keyword must be enclosed in *double* quotes (see Example 3–52 on page 3-88). **Note:** Because `EXECUTE` is a SQL*Plus command, you cannot execute a PL/SQL program unit by specifying `EXECUTE` within the RMAN `SQL` command. Instead, you must use the `BEGIN` and `END` keywords. For example, to execute the PL/SQL procedure `rman.rman_purge` with the `SQL` command, issue the following command: `SQL 'BEGIN rman.rman_purge; END;';`

Examples

Example 3–51 Archiving the Unarchived Online Logs

This example backs up a tablespace and then archives all unarchived online redo logs.

```
BACKUP TABLESPACE users;
SQL "ALTER SYSTEM ARCHIVE LOG CURRENT";
```

Example 3–52 Specifying a Filename within a Quoted String

This example specifies a filename by using duplicate single quotes within the context of a double-quoted string.

```
SQL "ALTER TABLESPACE users ADD DATAFILE ''/disk1/oradata/users02.dbf''
  SIZE 100K AUTOEXTEND ON NEXT 10K MAXSIZE 100K";
```

STARTUP

Purpose

Use the STARTUP command to start the target database from within the RMAN environment. This command is equivalent to using the SQL*Plus STARTUP command.

Additionally, the RMAN STARTUP command can start an instance in NOMOUNT mode even if no server parameter file or initialization parameter file exists. This feature is useful when you need to restore a lost server parameter file.

> **See Also:** *Oracle Database 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

Prerequisites

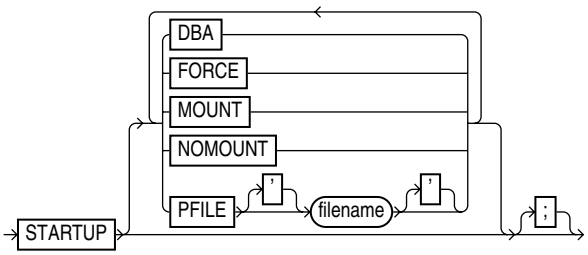
RMAN must be connected to a target database. You can only use this command to start the target database.

Usage Notes

The RMAN STARTUP command can start an instance in NOMOUNT mode even if no server parameter file or initialization parameter file exists. This feature is useful when you need to restore a lost server parameter file (see Example 3–54 on page 3-91).

Syntax

***startup*::=**



Semantics

Syntax Element	Description
STARTUP	If you specify only STARTUP with no other options, then the instance starts the instance with the default server parameter file, mounts the control file, and opens the database.
DBA	Restricts access to users with the RESTRICTED SESSION privilege.
FORCE	If the database is open, then FORCE shuts down the database with a SHUTDOWN ABORT statement before re-opening it. If the database is closed, then FORCE opens the database.
MOUNT	Starts the instance, then mounts the database without opening it
NOMOUNT	Starts the instance without mounting the database. If no parameter file exists, then RMAN starts the instance with a temporary parameter file. You can then run RESTORE SPFILE to restore a backup server parameter file.

Syntax Element	Description
`PFILE` *`filename`*	Specifies the filename of the text-based initialization parameter file for the target database. If `PFILE` is not specified, then the default initialization parameter filename is used.

Examples

Example 3–53 Mounting the Database While Specifying the Parameter File

This example forces a `SHUTDOWN ABORT` and then mounts the database with restricted access, specifying a nondefault initialization parameter file location:

```
CONNECT TARGET /
STARTUP FORCE MOUNT DBA PFILE=/tmp/initPROD.ora;
```

Example 3–54 Starting an Instance Without a Parameter File

Assume that the server parameter file was accidentally deleted from the file system. The following example starts an instance without using a parameter file, then runs `RESTORE SPFILE FROM AUTOBACKUP`. In this example, the autobackup location is the fast recovery area, so `SET DBID` is not necessary.

```
CONNECT TARGET /
STARTUP FORCE NOMOUNT; # RMAN starts instance with dummy parameter file
RESTORE SPFILE TO '?/dbs/spfileprod.ora'
  FROM AUTOBACKUP
  RECOVERY AREA '/disk2' DB_NAME='prod';
STARTUP FORCE; # restart instance with restored server parameter file
```

SWITCH

Purpose

Use the SWITCH command to perform either of the following operations:

- Update the filenames for a database, tablespace, or datafile to the latest image copies available for the specified files
- Update the filenames for datafiles and tempfiles for which you have issued a SET NEWNAME command

A SWITCH is equivalent to the SQL statement ALTER DATABASE RENAME FILE: the names of the files in the RMAN repository are updated, but the database does not rename the files at the operating system level.

Prerequisites

RMAN must be connected to a target database. When switching tablespaces, datafiles, or tempfiles, the files must be offline. When switching the whole database, the database must not be open.

Usage Notes

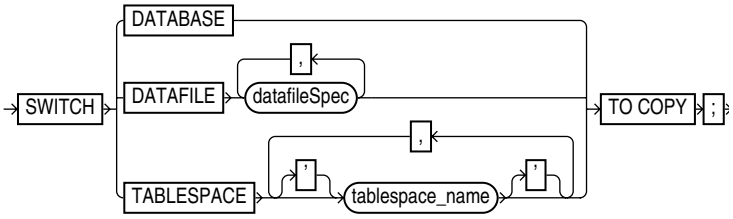
The SWITCH command deletes the RMAN repository records for the datafile copy from the recovery catalog and updates the control file records to status DELETED.

If RMAN is connected to a recovery catalog, and if the database is using a control file restored from backup, then SWITCH updates the control file with records of any datafiles known to the recovery catalog but missing from the control file.

Execute SWITCH ... TO COPY only at the RMAN prompt. Use SWITCH without TO COPY only within a RUN block.

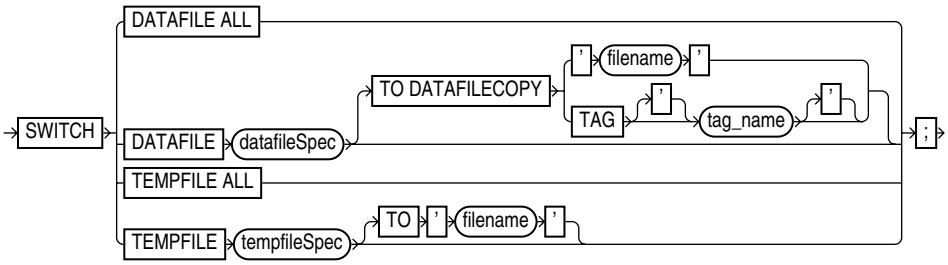
Syntax

***switch*::=**



(*datafileSpec::=* on page 4-15)

***switchFile*::=**



(*datafileSpec*::= on page 4-15, *tempfileSpec*::= on page 4-39)

Semantics

switch

This subclause switches filenames for a database, tablespace, or datafile to the latest image copies available for the specified files. By executing this command, you avoid restoring datafiles from backups. Execute `SWITCH ... TO COPY` only at the RMAN prompt.

Syntax Element	Description
`DATABASE`	Renames the datafiles and control files to use the filenames of image copies of these files. RMAN switches to the latest image copy of each database file. After a database switch, RMAN considers the previous database files as datafile copies.
`DATAFILE` *datafileSpec*	Switches the specified datafiles to the latest image copies. After the switch, the control file no longer considers the specified datafile as current.
`TABLESPACE` *tablespace_name*	Switches all datafiles within the specified tablespace, as with `SWITCH DATAFILE ... TO COPY` (see Example 3–55 on page 3-94).
`TO COPY`	Switches the specified active database files to image copies.

switchFile

This subclause updates the names for datafiles and tempfiles for which you have issued a `SET NEWNAME` command. Use this clause only within a `RUN` block.

Syntax Element	Description
`DATAFILE ALL`	Specifies that all datafiles for which a `SET NEWNAME FOR DATAFILE` command has been issued in this job are switched to their new name (see Example 3–56 on page 3-94).
`DATAFILE` *datafileSpec*	Specifies the datafile that are renaming. After the switch, the control file no longer considers the specified file as current. If you do not specify a `TO` option, then RMAN uses the filename specified on a prior `SET NEWNAME` command in the `RUN` block for this file as the switch target.
`TO DATAFILECOPY {'`*filename*`' | TAG` *tag_name*`}`	Specifies the input copy file for the switch, that is, the datafile copy that you intend to rename (see Example 3–58 on page 3-95).
`TEMPFILE ALL`	Specifies that all tempfiles for which a `SET NEWNAME FOR TEMPFILE` command has been issued in this job are switched to their new name.

Syntax Element	Description
TEMPFILE *tempfileSpec*	Specifies the tempfile that you are renaming. If you do not specify a TO option, then RMAN uses the filename specified on a prior SET NEWNAME command in the RUN block for this file as the switch target. The target database must be mounted but not open.
TO '*filename*'	Renames the tempfile to the specified name (see Example 3–57 on page 3-94). The target database must be mounted but not open.

Examples

Example 3–55 Switching to Image Copies to Avoid Restoring from Backup

Assume that a disk fails, rendering all datafiles in the users tablespace inaccessible. Image copies of all datafiles in this tablespace exist in the fast recovery area. After starting RMAN and connecting to the database as TARGET, you can run SWITCH to point to the control file to the new datafiles and then run RECOVER as follows:

```
SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
SWITCH TABLESPACE users TO COPY;
RECOVER TABLESPACE users;
SQL "ALTER TABLESPACE users ONLINE";
```

Example 3–56 Switching Datafile Filenames After a Restore to a New Location

Assume that a disk fails, forcing you to restore a datafile to a new disk location. After starting RMAN and connecting to the database as TARGET, you can use the SET NEWNAME command to rename the datafile, then RESTORE to restore the missing datafile. You run SWITCH to point to the control file to the new datafile and then RECOVER. This example allocates both disk and tape channels.

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev2 DEVICE TYPE sbt;
  SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
  SET NEWNAME FOR DATAFILE '/disk1/oradata/prod/users01.dbf'
                       TO '/disk2/users01.dbf';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;
  RECOVER TABLESPACE users;
  SQL "ALTER TABLESPACE users ONLINE";
}
```

Example 3–57 Renaming Tempfiles Using SET NEWNAME and SWITCH TEMPFILE ALL

This example demonstrates using SET NEWNAME to specify new names for several tempfiles, and SWITCH TEMPFILE ALL to rename the tempfiles to the specified names. The database must be closed at the beginning of this procedure. The tempfiles are re-created at the new locations when the database is opened.

```
CONNECT TARGET /
STARTUP FORCE MOUNT
RUN
{
  SET NEWNAME FOR TEMPFILE 1 TO '/disk2/temp01.dbf';
  SET NEWNAME FOR TEMPFILE 2 TO '/disk2/temp02.dbf';
  SET NEWNAME FOR TEMPFILE 3 TO '/disk2/temp03.dbf';
  SWITCH TEMPFILE ALL;
  RESTORE DATABASE;
```

```
  RECOVER DATABASE;
  ALTER DATABASE OPEN;
}
```

Example 3–58 Switching to a Datafile Copy

The following command switches the datafile in the `tools` tablespace to the datafile copy named `/disk2/tools.copy`:

```
RUN
{
  SQL "ALTER TABLESPACE tools OFFLINE IMMEDIATE";
  SWITCH DATAFILE '/disk1/oradata/prod/tools01.dbf'
  TO DATAFILECOPY '/disk2/tools.copy';
  RECOVER TABLESPACE tools;
  SQL "ALTER TABLESPACE tools ONLINE";
}
```

TRANSPORT TABLESPACE

Purpose

Use the TRANSPORT TABLESPACE command to create transportable tablespace sets from RMAN backups instead of the live datafiles of the source database.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to transport tablespaces with RMAN

Prerequisites

The limitations on creating transportable tablespace sets described in *Oracle Database Administrator's Guide* apply to transporting tablespaces from backup, with the exception of the requirement to make the tablespaces read-only.

Note that the SYSAUX tablespace must not be part of the recovery set, which is the set of tablespaces to be transported. RMAN enforces inclusion of the SYSAUX tablespace in the auxiliary set, which contains datafiles and other files required for the tablespace transport.

TRANSPORT TABLESPACE does not convert endian formats. If the target platform has a different endian format, then after running TRANSPORT TABLESPACE use the CONVERT command to convert the endian format of the transportable set datafiles.

If you drop a tablespace, then you cannot later use TRANSPORT TABLESPACE to include this tablespace in a transportable tablespace set, even if the SCN for TRANSPORT TABLESPACE is earlier than the SCN at which the table was dropped. If you rename a tablespace, then you cannot use TRANSPORT TABLESPACE to create a transportable tablespace set as of a point in time before the tablespace was renamed.

Backups and Backup Metadata

You must have a backup of all needed tablespaces (including those in the auxiliary set) and archived redo log files needed to recover to the target point in time.

If you do not use a recovery catalog, and if the database has re-used control file records containing metadata about required backups, then the command fails because RMAN cannot locate the backups. You may be able to use CATALOG to add the needed backups to the RMAN repository if they are still available, but if the database is already overwriting control file records you may lose records of other needed backups.

Data Pump Export and Import

Because the RMAN uses the Data Pump Export and Import utilities, you cannot use TRANSPORT TABLESPACE if the tablespaces to be transported use XMLType. In this case you must use the procedure in *Oracle Database Administrator's Guide*.

If a file under the name of the export dump file already in the tablespace destination, then TRANSPORT TABLESPACE fails when it calls Data Pump Export. If you are repeating a previous TRANSPORT TABLESPACE job, then make sure to delete the previous output files, including the export dump file.

Tablespace and Column Encryption

The following database encryption features both use the wallet: transparent data encryption, which functions at the column level, and tablespace encryption. Note the following restrictions for tablespaces that are encrypted or contain encrypted columns:

- If you are transporting an encrypted tablespace, then you must manually copy the wallet to the destination database.
- If the destination database has an existing wallet, then you cannot copy the wallet from the source database to the destination database. Thus, you cannot transport encrypted data to a database that already has a wallet. If you encrypt columns with transparent data encryption, then you can export them into an export file that is password-protected and import the data into the destination database.

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* to learn about transparent data encryption

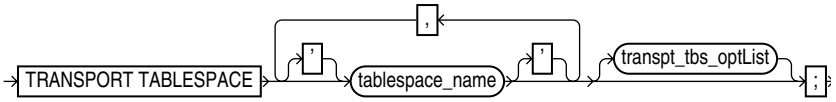
Usage Notes

Because RMAN creates the automatic auxiliary instance used for restore and recovery on the same node as the source instance, there is some performance overhead during the operation of the `TRANSPORT TABLESPACE` command.

If RMAN is not part of the backup strategy for your database, then you can still use `TRANSPORT TABLESPACE` as long as the needed datafile copies and archived redo log files are available on disk. Use the `CATALOG` command to record the datafile copies and archived redo log files in the RMAN repository. You can then use `TRANSPORT TABLESPACE`. You also have the option of using RMAN to back up your database specifically so you can use `TRANSPORT TABLESPACE`.

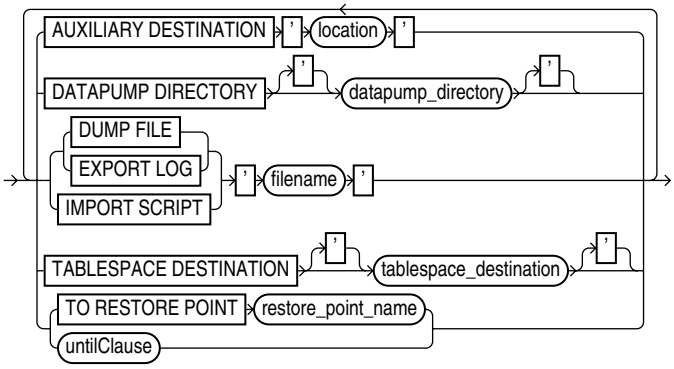
Syntax

transpt_tbs::=



(*transpt_tbs_optlist::=* on page 3-97)

transpt_tbs_optlist::=



(*untilClause::=* on page 4-42)

Semantics

transpt_tbs

Syntax Element	Description
tablespace_name	Specifies the name of each tablespace to transport. You must have a backup of all needed tablespaces (including those in the auxiliary set) and archived redo log files available for use by RMAN that can be recovered to the target time for the TRANSPORT TABLESPACE operation.

transpt_tbs_optlist

This subclause specifies optional parameters that affect the tablespace transport.

Syntax Element	Description
AUXILIARY DESTINATION '*location*'	Specifies the location for files for the auxiliary instance. You can use SET NEWNAME and CONFIGURE AUXNAME to override this argument for individual files. If using your own initialization parameter file to customize the auxiliary instance, then you can use the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT initialization parameters instead of AUXILIARY DESTINATION. **See Also:** *Oracle Database Backup and Recovery User's Guide* for details on the interactions among the different techniques for naming the auxiliary instance files
DATAPUMP DIRECTORY *datapump_directory*	Specifies a database directory object where Data Pump Export outputs are created (see Example 3–60). If not specified, then RMAN creates files in the location specified by TABLESPACE DESTINATION. **See Also:** *Oracle Database Utilities* for more details on Data Pump Export and database directory objects
DUMP FILE '*filename*'	Specifies where to create the Data Pump Export dump file. If not specified, the export dump file is named dmpfile.dmp and stored in the location specified by the DATAPUMP DIRECTORY clause or in the tablespace destination. **Note:** If a file under the name of the export dump file already in the tablespace destination, then TRANSPORT TABLESPACE fails when it calls Data Pump Export. If you are repeating a previous TRANSPORT TABLESPACE job, then make sure to delete the previous output files, including the export dump file.
EXPORT LOG '*filename*'	Specifies the location of the log generated by Data Pump Export. If omitted, the export log is named explog.log and stored in the location specified by the DATAPUMP DIRECTORY clause or in the tablespace destination.
IMPORT SCRIPT '*filename*'	Specifies the filename for the sample input script generated by RMAN for use in plugging in the transported tablespace at the destination database. If omitted, the import script is named impscript.sql. The script is stored in the tablespace destination.
TABLESPACE DESTINATION *tablespace_destination*	Specifies the location of the datafiles for the transported tablespaces after the tablespace transport operation completes.
TO RESTORE POINT *restore_point_name*	Specifies a restore point for tablespace restore and recovery, with the SCN at which the restore point was created as the upper, inclusive limit. Because the limit is inclusive, RMAN selects only files that can be used to restore or recover tablespaces up to *and including* the SCN corresponding to the restore point.

Syntax Element	Description
untilClause	Specifies a past time, SCN, or log sequence number (see Example 3–59 on page 3-99). If specified, RMAN restores and recovers the tablespaces at the auxiliary instance to their contents at that past point in time before export. If you rename a tablespace, then you cannot use this command to create a transportable tablespace set as of a point in time before the tablespace was renamed. RMAN has no knowledge of the previous name of the tablespace. Tablespaces including undo segments as of the `UNTIL` time or SCN for `TRANSPORT TABLESPACE` must be part of the auxiliary set. The control file only contains a record of tablespaces that include undo segments at the current time. If the set of tablespaces with undo segments was different at the `UNTIL` time or SCN, then `TRANSPORT TABLESPACE` fails. Thus, if you use RMAN in `NOCATALOG` mode and specify `UNTIL`, then the set of tablespaces with undo segments at the time `TRANSPORT TABLESPACE` executes must be the same as the set of tablespaces with undo segments at the `UNTIL` time or SCN.

Examples

Example 3–59 Using TRANSPORT TABLESPACE with a Past Time

In this example, the tablespaces for the transportable set are `example` and `tools`, the transportable set files are to be stored at `/disk1/transport_dest`, and the transportable tablespaces are to be recovered to a time 15 minutes ago:

```
TRANSPORT TABLESPACE example, tools
  TABLESPACE DESTINATION '/disk1/transportdest'
  AUXILIARY DESTINATION '/disk1/auxdest'
  UNTIL TIME 'SYSDATE-15/1440';
```

Partial sample output follows:

```
Creating automatic instance, with SID='egnr'

initialization parameters used for automatic instance:
db_name=PROD
compatible=11.0.0
db_block_size=8192
.
.
.
starting up automatic instance PROD
.
.
.
executing Memory Script

executing command: SET until clause

Starting restore at 07-JUN-07
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=44 device type=DISK

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: restoring control file
.
.
.
output file name=/disk1/auxdest/cntrl_tspitr_PROD_egnr.f
Finished restore at 07-JUN-07
```

```
sql statement: alter database mount clone database

sql statement: alter system archive log current

sql statement: begin dbms_backup_restore.AutoBackupFlag(FALSE); end;

starting full resync of recovery catalog
full resync complete
.
.
.
executing Memory Script
.
.
.

Starting restore at 07-JUN-07
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/disk1/auxdest/TSPITR_PROD_EGNR/datafile/o1_mf_system_%u_.dbf
datafile 1 switched to datafile copy
.
.
.
starting media recovery
.
.
.
Finished recover at 07-JUN-07

database opened
.
.
.
executing Memory Script
.
.
.
sql statement: alter tablespace EXAMPLE read only
Removing automatic instance
shutting down automatic instance
Oracle instance shut down
Automatic instance removed
auxiliary instance file /disk1/auxdest/cntrl_tspitr_PROD_egnr.f deleted
.
.
.
```

Example 3–60 Using TRANSPORT TABLESPACE with Customized File Locations

This example illustrates the use of the optional arguments that control the locations of Data Pump-related files such as the dump file. Note that the `DATAPUMP DIRECTORY` must refer to an object that exists in the target database. Use the `CREATE DIRECTORY` SQL statement to create a directory object.

```
TRANSPORT TABLESPACE example
  TABLESPACE DESTINATION '/disk1/transportdest'
```

```
AUXILIARY DESTINATION '/disk1/auxdest'
DATAPUMP DIRECTORY mypumpdir
DUMP FILE 'mydumpfile.dmp'
IMPORT SCRIPT 'myimportscript.sql'
EXPORT LOG 'myexportlog.log';
```

UNREGISTER

Purpose

Use the UNREGISTER command to remove the RMAN metadata for one or more registered databases from the recovery catalog.

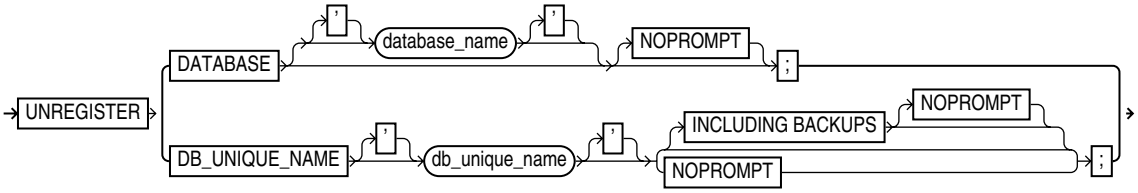
> **See Also:** DROP DATABASE on page 2-123 to learn how to delete a database and unregister it with one command

Prerequisites

Execute this command only at the RMAN prompt. RMAN must be connected to a recovery catalog. The database that you want to unregister must be currently registered in this catalog.

Syntax

***unregister*::=**



Semantics

Syntax Element	Description
DATABASE	Specifies a primary database to be unregistered. RMAN unregisters the primary database as well as any of its associated standby databases (see Example 3–61 on page 3-103). If *database_name* is not specified, then RMAN must identify the database in one of the following ways: ▪ If RMAN is connected to a database as TARGET, then RMAN unregisters the target database and any associated standby databases. ▪ If RMAN is not connected to a database as TARGET, or if the name of the TARGET database is not unique in the recovery catalog, then RMAN uses the value specified in the SET DBID command (see Example 3–62 on page 3-103). RMAN unregisters all databases with this DBID.
database_name	Specifies the name of the primary database that you are unregistering. This database name must be unique in the recovery catalog. Because the database name is unique, RMAN does not need to be connected to a target database or use the SET DBID command.

Syntax Element	Description
DB_UNIQUE_NAME *db_unique_name*	Specifies the removal of all metadata except backup metadata for the Data Guard database specified by *db_unique_name*. The specified database can be either a primary or standby database, although typically you use this clause to unregister a standby database (see Example 3–63 on page 3-104). You can use this clause only when RMAN is connected to a mounted or open target database or the SET DBID command has been run. Typically, the target database is not the database that you are unregistering. For example, you can connect as TARGET to prod1 and use DB_UNIQUE_NAME to unregister standby1. Note that the DBID used in SET DBID is the same for the primary database and its associated standby databases. The backups of a database have the DB_UNIQUE_NAME associated with the backup file names in the recovery catalog. When a database is unregistered, the database name for these backup files is marked as null. You can associate a database name with these files by using the CHANGE ... RESET DB_UNIQUE_NAME command.
INCLUDING BACKUPS	Specifies that backup metadata should also be removed for the database specified by *db_unique_name*. **Note:** Physical backups will not be deleted by the UNREGISTER command. If you want the physical backups removed, then first remove them with the DELETE command and then execute UNREGISTER with the INCLUDING BACKUPS option.
NOPROMPT	Does not prompt for confirmation before unregistering the database.

Example

Example 3–61 Unregistering a Primary Database and Its Standby Databases

Assume that primary database prod has associated standby databases dgprod3 and dgprod4. In this example, you connect RMAN to target database prod, whose database name is unique in the recovery catalog, and unregister it. RMAN removes all metadata about prod, dgprod3, and dgprod4 from the catalog. Sample output is included.

```
RMAN> CONNECT TARGET /

connected to target database: PROD (DBID=1627709917)

RMAN> CONNECT CATALOG rman@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> UNREGISTER DATABASE NOPROMPT;

database name is "PROD" and DBID is 1627709917
database unregistered from the recovery catalog

RMAN> LIST DB_UNIQUE_NAME ALL;

RMAN>
```

Example 3–62 Unregistering a Database That is Not Unique in Catalog

Assume that two databases registered in a recovery catalog have the name prod. Your goal is to unregister the prod database that has the DBID 28014364. Because multiple

databases called prod are registered in the recovery catalog, and because RMAN is not connected as TARGET to the 28014364 database (which has already been deleted from the file system), you run SET DBID before UNREGISTER DATABASE. This example includes sample output.

```
RMAN> CONNECT CATALOG rman@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> SET DBID 28014364;

executing command: SET DBID
database name is "PROD" and DBID is 28014364

RMAN> UNREGISTER DATABASE;

Do you really want to unregister the database (enter YES or NO)? YES
database unregistered from the recovery catalog
```

Example 3–63 Unregistering a Standby Database

Assume that primary database prod has associated standby databases dgprod3 and dgprod4. You want to unregister dgprod4, but not remove the metadata for backups taken on this database because these backups are still usable by other databases in the environment. This example uses SET DBID to specify the DBID of the standby database and then unregisters it (sample output included):

```
RMAN> CONNECT CATALOG rman@catdb

recovery catalog database Password: password
connected to recovery catalog database

RMAN> SET DBID 1627367554;

executing command: SET DBID
database name is "PROD" and DBID is 1627367554

RMAN>  UNREGISTER DB_UNIQUE_NAME dgprod4;

database db_unique_name is "dgprod4", db_name is "PROD" and DBID is 1627367554

Want to unregister the database with target db_unique_name (enter YES or NO)? YES
database with db_unique_name dgprod4 unregistered from the recovery catalog
```

UPGRADE CATALOG

Purpose

Use the `UPGRADE CATALOG` command to upgrade a recovery catalog schema from an older version to the version required by the RMAN client.

Prerequisites

RMAN must be connected to the recovery catalog database, which must be open, as the owner of the recovery catalog. You cannot use the `UPGRADE CATALOG` command while connected to a virtual private catalog (see `CREATE CATALOG` command). Only the base catalog can be upgraded.

The recovery catalog must not already be at a version greater than needed by the RMAN executable; otherwise, RMAN issues an error. RMAN displays all error messages generated during the upgrade in the message log.

The Oracle Database 10*g*R1 version of the recovery catalog schema requires the `CREATE TYPE` privilege. If you created the recovery catalog owner in a release before 10*g*R1, and if you granted the `RECOVERY_CATALOG_OWNER` role to this user when the role did not include the `CREATE TYPE` privilege, then you must grant `CREATE TYPE` to this user explicitly before performing the upgrade.

Usage Notes

You must enter the `UPGRADE CATALOG` command two consecutive times to confirm the upgrade.

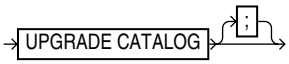
RMAN permits the command to be run if the recovery catalog is already current so that the packages can be re-created if necessary.

If an upgrade to a base recovery catalog requires changes to an existing virtual private catalog, then RMAN makes these changes automatically the next time RMAN connects to that virtual private catalog.

The `UPGRADE CATALOG` command does not run scripts to perform an upgrade. Instead, RMAN sends various SQL DDL statements to the recovery catalog to update the recovery catalog schema with new tables, views, columns, and so on.

Syntax

***upgradeCatalog*::=**



Semantics

None.

Example

Example 3–64 Upgrading a Recovery Catalog

This example connects RMAN to recovery catalog database `catdb` and then upgrades it to a more current version:

```
RMAN> CONNECT CATALOG rco@catdb

recovery catalog database Password: password
connected to recovery catalog database
PL/SQL package rman.DBMS_RCVCAT version 10.01.00 in RCVCAT database is too old

RMAN> UPGRADE CATALOG;

recovery catalog owner is RCO
enter UPGRADE CATALOG command again to confirm catalog upgrade

RMAN> UPGRADE CATALOG;

recovery catalog upgraded to version 11.01.00
DBMS_RCVMAN package upgraded to version 11.01.00
DBMS_RCVCAT package upgraded to version 11.01.00
```

VALIDATE

Purpose

Use the VALIDATE command to check for corrupt blocks and missing files, or to determine whether a backup set can be restored.

If VALIDATE detects a problem during validation, then RMAN displays it and triggers execution of a failure assessment. If a failure is detected, then RMAN logs it into the Automated Diagnostic Repository. You can use LIST FAILURE to view the failures.

Prerequisites

The target database must be mounted or open.

Usage Notes

The options in the VALIDATE command are semantically equivalent to options in the BACKUP VALIDATE command. Unlike BACKUP VALIDATE, however, VALIDATE can check individual backup sets and data blocks.

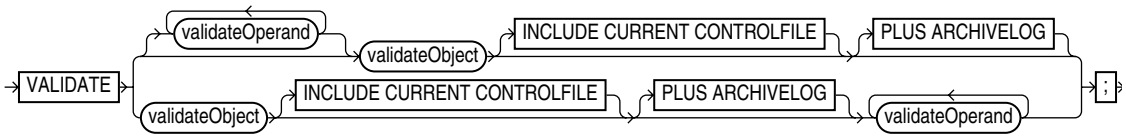
The VALIDATE command skips never-used blocks. RMAN also skips currently unused (as opposed to never used) blocks for locally managed tablespaces when the COMPATIBLE parameter is set to 10.2 or greater. If RMAN does not read a block because of unused block compression, and if the block is corrupt, then RMAN does not detect the corruption. A corrupt unused block is not harmful.

In a physical corruption, the database does not recognize the block at all. In a logical corruption, the contents of the block are logically inconsistent. By default, the VALIDATE command checks for physical corruption only. You can specify CHECK LOGICAL to check for logical corruption as well. RMAN populates the V$DATABASE_BLOCK_CORRUPTION view with its findings.

Block corruptions can be divided into interblock corruption and intrablock corruption. In intrablock corruption, the corruption occurs within the block itself and can be either physical or logical corruption. In interblock corruption, the corruption occurs between blocks and can only be logical corruption. The VALIDATE command checks for intrablock corruptions only.

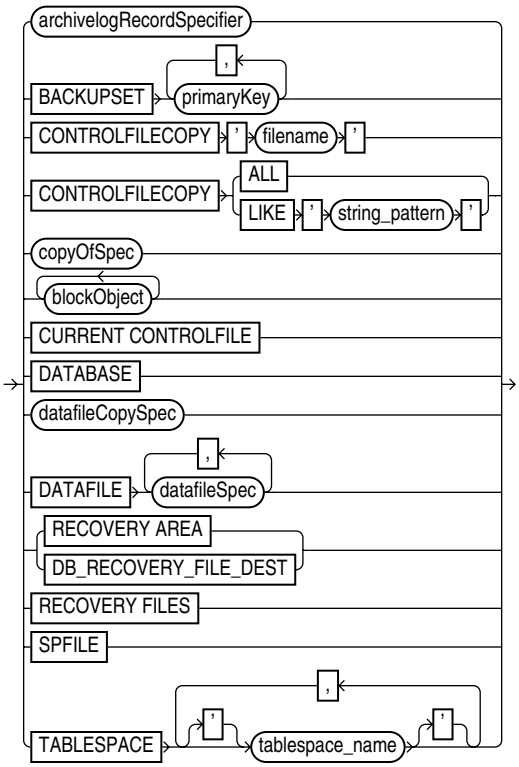
Syntax

***validate*::=**



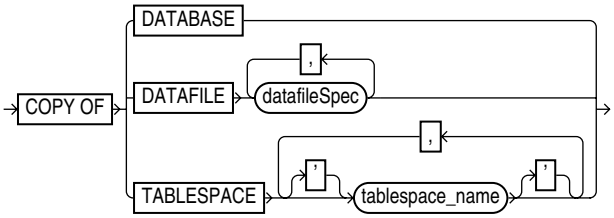
(*validateObject::=* on page 3-108, *validateOperand::=* on page 3-109)

validateObject::=



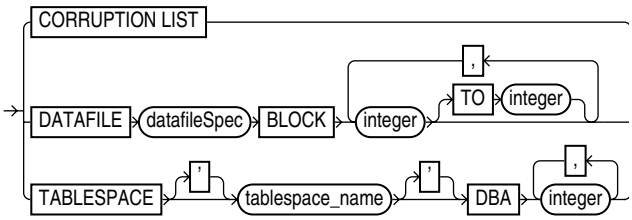
(*archivelogRecordSpecifier::=* on page 4-6, *copyOfSpec::=* on page 2-26, *blockObject::=* on page 3-5, *datafileCopySpec::=* on page 2-27, *datafileSpec::=* on page 4-15)

copyOfSpec::=



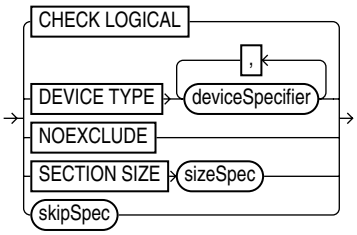
(*datafileSpec::=* on page 4-15)

blockObject::=



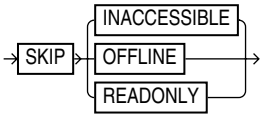
(*datafileSpec::=* on page 4-15)

validateOperand::=

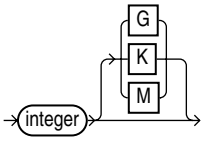


(*deviceSpecifier::=* on page 4-16, *sizeSpec::=* on page 3-109, *skipSpec::=* on page 3-109)

skipSpec::=



sizeSpec::=



Semantics

validate

This subclause specifies backup sets for validation. Refer to *validate::=* on page 3-107 for syntax.

Syntax Element	Description
`validateOperand`	Specifies options that control the validation. **See Also:** `validateOperand` on page 3-111
`validateObject`	Specifies the files to be validated. **See Also:** `validateObject` on page 3-110
`INCLUDE CURRENT CONTROLFILE`	Creates a snapshot of the current control file and validates it.
`PLUS ARCHIVELOG`	Includes archived redo log files in the validation. Causes RMAN to perform the following steps: **1.** Run an `ALTER SYSTEM ARCHIVE LOG CURRENT` statement. **2.** Run the `VALIDATE ARCHIVELOG ALL` command. Note that if backup optimization is enabled, then RMAN only validates logs that have not yet been backed up. **3.** Validate the files specified in the `VALIDATE` command. **4.** Run an `ALTER SYSTEM ARCHIVE LOG CURRENT` statement. **5.** Validate any remaining archived redo log files.

validateObject

This subclause specifies database files for validation. Refer to *validateObject::=* on page 3-108 for syntax.

Syntax Element	Description
archivelogRecordSpecifier	Validates a range of archived redo log files. `VALIDATE ARCHIVELOG` is equivalent to `BACKUP VALIDATE ARCHIVELOG`.
`BACKUPSET` *primary_key*	Checks that the backup sets specified by *primary_key* exist and can be restored. You can 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` recovery catalog view. The `VALIDATE BACKUPSET` command checks every block in the backup set to ensure that the backup is restorable. If RMAN finds block corruption, then it issues an error and terminates the validation. In contrast, the `CROSSCHECK` command examines the headers of the specified files if they are on disk or queries the media management catalog if they are on tape. Use `VALIDATE BACKUPSET` when you suspect that one or more backup pieces in a backup set are missing or have been damaged. Note that `VALIDATE BACKUPSET` selects which backups to test, whereas the `VALIDATE` option of the `RESTORE` command lets RMAN choose which backups to validate. For validating image copies, run `RESTORE VALIDATE FROM DATAFILECOPY`. If you do not have automatic channels configured, then manually allocate at least one channel before executing `VALIDATE BACKUPSET`. **Note:** If multiple copies of a backup set exist, then RMAN validates only the most recent copy. The `VALIDATE` command does not support an option to validate a specific copy. If one copy is on a different device from another copy, however, then you can use `VALIDATE DEVICE TYPE` to validate the copy on the specified device. If both copies exists on the same device, then you can use `CHANGE` to make one copy temporarily `UNAVAILABLE` and then reissue `VALIDATE`.
`CONTROLFILECOPY {'`*filename*`' | ALL | LIKE '`*string_pattern*`'}`	Validates control file copies. You can specify a control file copy in one of the following ways: ▪ `'`*filename*`'` specifies a control file copy by filename ▪ `ALL` specifies that all control file copies should be backed up ▪ `LIKE '`*pattern*`'` specifies a filename pattern. The percent sign (`%`) as a wildcard meaning 0 or more characters; an underscore (_) is a wildcard meaning 1 character. The control file copy can be created with the `BACKUP AS COPY CURRENT CONTROLFILE` command or the SQL statement `ALTER DATABASE BACKUP CONTROLFILE TO '...'`.
copyOfSpec	Validates image copies of datafiles and control files. **See Also:** *copyOfSpec* on page 2-42 for details.
blockObject	Validates individual data blocks. **See Also:** *blockObject* on page 3-8
`CURRENT CONTROLFILE`	Validates the current control file.
`DATABASE`	Validates the database. RMAN validates all datafiles and control files. If the database is currently using a server parameter file, then RMAN validates the server parameter file. **Note:** The online redo log files and tempfiles are not validated.

Syntax Element	Description
datafileCopySpec	Validates one or more datafile image copies. When validating datafile copies, RMAN checks for block corruption but does not terminate the validation if corruption is discovered. Unlike `VALIDATE BACKUPSET`, RMAN proceeds and reports the number of blocks that are corrupted. **See Also:** *datafileCopySpec* on page 2-43 for details
`DATAFILE` *datafileSpec*	Specifies a list of one or more datafiles that contain blocks requiring validation. **Note:** You do not have to take a datafile offline if you are validating it. **See Also:** *datafileSpec* on page 4-15
`RECOVERY AREA`	Validates recovery files created in the current and all previous fast recovery area destinations. Recovery files are full and incremental backup sets, control file autobackups, archived redo log files, and datafile copies. Flashback logs, the current control file, and online redo logs are not validated.
`DB_RECOVERY_FILE_DEST`	Semantically equivalent to `RECOVERY AREA`.
`RECOVERY FILES`	Validates *all* recovery files on disk, whether they are stored in the fast recovery area or other locations on disk. Recovery files include full and incremental backup sets, control file autobackups, archived redo log files, and datafile copies. Flashback logs are not validated.
`SPFILE`	Validates the server parameter file currently used by the database. RMAN cannot validates other copies of the server parameter file, and cannot validate the server parameter file when the instance was started with an initialization parameter file.
`TABLESPACE` *tablespace_name*	Validates the specified tablespaces. RMAN translates tablespace names internally into a list of datafiles, then validates all datafiles that are currently part of the tablespaces. RMAN validates all datafiles that are currently members of the specified tablespaces.

validateOperand

This subclause specifies modifiers for the validation. Refer to *validateOperand::=* on page 3-109 for syntax.

Syntax Element	Description
`CHECK LOGICAL`	Tests data and index blocks in the files 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. The RMAN command completes and `V$DATABASE_BLOCK_CORRUPTION` is populated with corrupt block ranges. **Note:** `VALIDATE` does not use `MAXCORRUPT`.
`DEVICE TYPE` *deviceSpecifier*	Allocates automatic channels for the specified device type only. This option is valid only if you have configured automatic channels and have not manually allocated channels. For example, if you configure automatic disk and tape channels, and run `VALIDATE ...DEVICE TYPE DISK`, RMAN allocates only disk channels. **See Also:** *deviceSpecifier* on page 4-16
`NOEXCLUDE`	When specified on a `VALIDATE DATABASE` or `VALIDATE COPY OF DATABASE` command, RMAN validates all tablespaces, including any for which a `CONFIGURE EXCLUDE` command has been entered. This option does not override `SKIP OFFLINE` or `SKIP READONLY`.

Syntax Element	Description
SECTION SIZE *sizeSpec*	Parallelizes the validation by dividing each file into the specified section size. Only specify this parameter when multiple channels are configured or allocated and you want the channels to parallelize the validation, so that multiple channels can validate a single datafile. This parameter applies only when validating datafiles. If you specify a section size that is larger than the size of the file, then RMAN does not parallelize validation for the file. If you specify a small section size that would produce more than 256 sections, then RMAN increases the section size to a value that results in exactly 256 sections. **See Also:** BACKUP SECTION SIZE to learn how to make multisection backups
skipSpec	Excludes the specified files from the validation.

skipSpec

This subclauses specifies files to be excluded from the validation.

Syntax Element	Description
SKIP	Excludes datafiles or archived redo log files from the validation if they are inaccessible, offline, or read-only.
INACCESSIBLE	Specifies that datafiles and archived redo log files that cannot be read due to I/O errors should be excluded from the validation. 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.
OFFLINE	Specifies that offline datafiles should be excluded from the validation.
READONLY	Specifies that read-only datafiles should be excluded from the validation.

VALIDATE Command Output

Table 3–11 List of Datafiles

Column	Indicates
File	Absolute number of the datafile being validated.
Status	OK if no corruption, or FAILED if block corruption is found.
Marked Corrupt	Number of blocks marked corrupt. These blocks were previously marked corrupt by the database. For example, the database may intentionally mark blocks corrupt during a recovery involving a NOLOGGING operation. Also, an RMAN backup may contain corrupt blocks permitted by the SET MAXCORRUPT command. When this backup is restored, the file contains blocks that are marked corrupt.
Empty Blocks	Number of blocks that either have never been used.
Blocks Examined	Total number of blocks in the file.
High SCN	The highest SCN recorded in the file.
File Name	The name of the file being validated.
Block Type	The type of block validated: Data, Index, or Other.
Blocks Failing	The number of blocks that fail the corruption check. These blocks are newly corrupt.
Blocks Processed	The number of blocks checked for corruption.

Table 3–12 List of Control File and SPFILE

Column	Indicates
`File TYPE`	Type of file: `SPFILE` or `Control File`.
`Status`	`OK` if no corruption, or `FAILED` if block corruption is found.
`Blocks Failing`	The number of blocks that fail the corruption check. These blocks are newly corrupt.
`Blocks Examined`	Total number of blocks in the file.

Table 3–13 List of Archived Logs

Column	Indicates
`Thrd`	The redo thread number.
`Seq`	The log sequence number.
`Status`	`OK` if no corruption, or `FAILED` if block corruption is found.
`Blocks Failing`	The number of blocks that fail the corruption check. These blocks are newly corrupt.
`Blocks Examined`	Total number of blocks in the file.
`Name`	The name of the archived redo log file.

Examples

Example 3–65 Validating a Backup Set

This example lists all available backup sets and then validates them. As the sample output indicates, RMAN confirms that it is possible to restore the backups.

```
RMAN> LIST BACKUP SUMMARY;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------- ---
3871    B  F  A DISK        08-MAR-07       1       1       NO         TAG20070308T092426
3890    B  F  A DISK        08-MAR-07       1       1       NO         TAG20070308T092534

RMAN> VALIDATE BACKUPSET 3871, 3890;

Starting validate at 08-MAR-07
using channel ORA_DISK_1
channel ORA_DISK_1: starting validation of datafile backup set
channel ORA_DISK_1: reading from backup piece
 /disk2/PROD/backupset/2007_03_08/o1_mf_nnndf_TAG20070308T092 426_2z0kpc72_.bkp
channel ORA_DISK_1: piece
 handle=/disk2/PROD/backupset/2007_03_08/o1_mf_nnndf_TAG20070308T092426_2z0kpc72_.bkp ta
 g=TAG20070308T092426
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: validation complete, elapsed time: 00:00:18
channel ORA_DISK_1: starting validation of datafile backup set
channel ORA_DISK_1: reading from backup piece
 /disk2/PROD/autobackup/2007_03_08/o1_mf_s_616670734_2z0krhjv_.bkp
channel ORA_DISK_1: piece
 handle=/disk2/PROD/autobackup/2007_03_08/o1_mf_s_616670734_2z0krhjv_.bkp
 tag=TAG20070308T092534
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: validation complete, elapsed time: 00:00:00
Finished validate at 08-MAR-07
```

Example 3–66 Validating the Database

This example validates the database and includes sample output. The validation finds one corrupt block in datafile 1. The `VALIDATE` output indicates that more information about the corruption can be found in the specified trace file.

```
RMAN> VALIDATE DATABASE;

Starting validate at 26-FEB-07
using channel ORA_DISK_1
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
input datafile file number=00001 name=/disk1/oradata/prod/system01.dbf
input datafile file number=00002 name=/disk1/oradata/prod/sysaux01.dbf
input datafile file number=00003 name=/disk1/oradata/prod/undotbs01.dbf
input datafile file number=00004 name=/disk1/oradata/prod/cwmlite01.dbf
input datafile file number=00005 name=/disk1/oradata/prod/drsys01.dbf
input datafile file number=00006 name=/disk1/oradata/prod/example01.dbf
input datafile file number=00007 name=/disk1/oradata/prod/indx01.dbf
input datafile file number=00008 name=/disk1/oradata/prod/tools01.dbf
input datafile file number=00009 name=/disk1/oradata/prod/users01.dbf
channel ORA_DISK_1: validation complete, elapsed time: 00:01:25
List of Datafiles
=================
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
1    FAILED 0              4140         57600           498288
  File Name: /disk1/oradata/prod/system01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       1              41508
  Index      0              7653
  Other      0              4299

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
2    OK     0              8918         20040           498237
  File Name: /disk1/oradata/prod/sysaux01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              2473
  Index      0              2178
  Other      0              6471

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
3    OK     0              36           2560            498293
  File Name: /disk1/oradata/prod/undotbs01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              2524

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
4    OK     0              1            1280            393585
  File Name: /disk1/oradata/prod/cwmlite01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              1279

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
5    OK     0              1            1280            393644
```

```
  File Name: /disk1/oradata/prod/drsys01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              1279

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
6    OK     0              1            1280            393690
  File Name: /disk1/oradata/prod/example01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              1279

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
7    OK     0              1            1280            393722
  File Name: /disk1/oradata/prod/indx01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              1279

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
8    OK     0              1            1280            393754
  File Name: /disk1/oradata/prod/tools01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              1279

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
9    OK     0              1272         1280            393785
  File Name: /disk1/oradata/prod/users01.dbf
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              0
  Index      0              0
  Other      0              8

validate found one or more corrupt blocks
See trace file /disk2/oracle/log/diag/rdbms/prod/prod/trace/prod_ora_10609.trc for details
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
including current control file for validation
including current SPFILE in backup set
channel ORA_DISK_1: validation complete, elapsed time: 00:00:01
List of Control File and SPFILE
===============================
File Type    Status Blocks Failing Blocks Examined
------------ ------ -------------- ---------------
SPFILE       OK     0              2
Control File OK     0              506
Finished validate at 26-FEB-07
```

4

RMAN Subclauses

This chapter describes, in alphabetical order, Recovery Manager subclauses referred to within RMAN commands. For a summary of the RMAN commands and command-line options, refer to "Summary of RMAN Commands" on page 1-6.

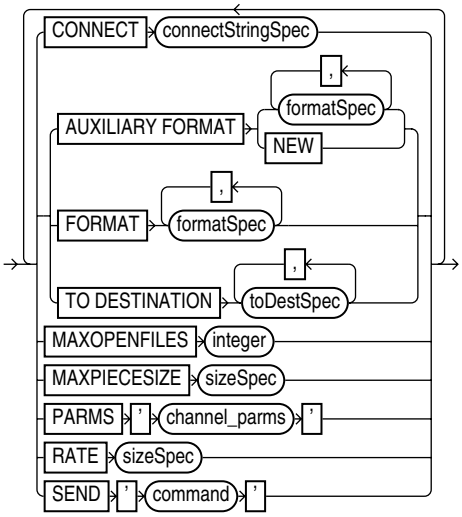
allocOperandList

Purpose

Use the `allocOperandList` subclause to control options on a **channel**, which is a connection between RMAN and a database instance.

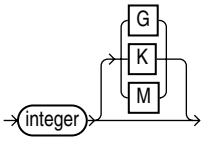
Syntax

allocOperandList::=



(*connectStringSpec::=* on page 4-13, *formatSpec::=* on page 4-23, *toDestSpec::=* on page 4-40)

sizeSpec::=



Semantics

allocOperandList

Syntax Element	Description
`CONNECT` *`connectStringSpec`*	Specifies a connect string to the database instance where RMAN should conduct the backup or restore operations. Use this parameter to spread the work across different instances in an Oracle RAC configuration. If you do not specify this parameter, and if you did not specify the `AUXILIARY` option, then RMAN conducts all operations on the target database instance specified by the command-line `CONNECT` parameter or `CONNECT` command. Typically, you should not use the `CONNECT` parameter in conjunction with the `AUXILIARY` option. **See Also:** *`connectStringSpec`* on page 4-13 and `RMAN` on page 3-59

Syntax Element	Description
AUXILIARY FORMAT	Specifies the format of image copies created on an auxiliary instance. RMAN must be connected to the auxiliary instance with CONNECT AUXILIARY and have access to auxiliary channels.
formatSpec	Specifies a pattern for image copy names on an auxiliary instance. The path must be valid on the auxiliary host. **See Also:** *formatSpec* on page 4-23 for valid substitution variables
NEW	Creates an image copy in the directory specified by the DB_CREATE_FILE_DEST initialization parameter of the auxiliary instance. The image copy name is in an Oracle Managed Files format.
FORMAT *formatSpec*	Specifies the format to use for the names backup pieces or image copies created on this channel. Example 4–1 illustrates this technique. The FORMAT parameter is useful if you allocate multiple disk channels and want each channel to write to a different directory. The FORMAT parameter specified in CONFIGURE CHANNEL or ALLOCATE CHANNEL is semantically equivalent to the FORMAT parameter specified in the BACKUP command (not the DATAFILECOPY FORMAT parameter in *forRecoveryOfSpec* on page 2-44). If you specify the FORMAT parameter in the BACKUP command, then it overrides the FORMAT parameter specified in CONFIGURE CHANNEL or ALLOCATE CHANNEL. If you do not specify FORMAT, then RMAN uses %U by default, which guarantees a unique name for the names of the backup files. If the fast recovery area is configured, then RMAN creates the backup files in the default disk location. Otherwise, the default disk location is operating system-specific (for example, ?/dbs on Solaris). You can specify up to four FORMAT strings. RMAN uses the second, third, and fourth values only when BACKUP COPIES, SET BACKUP COPIES, or CONFIGURE ... BACKUP COPIES is in effect. When choosing which format to use for each backup piece, RMAN uses the first format value for copy 1, the second format value for copy 2, and so forth. If the number of format values exceeds the number of copies, then the extra formats are not used. If the number of format values is less than the number of copies, then RMAN reuses the format values, starting with the first one. Because the channels correspond to server sessions on the target database, the FORMAT string must use the conventions of the target host, not the client host. For example, if the RMAN client runs on a Windows host and the target database runs on a Linux host, then the FORMAT string must adhere to the naming conventions of a Linux file system or raw device. **See Also:** *formatSpec* on page 4-23 for available FORMAT parameters
TO DESTINATION *toDestSpec*	Specifies the directory where the backup is created. This parameter is valid for disk and not SBT channels. The backup files are created in a Oracle Managed Files (OMF) directory. Backup skips the files only when backups do not exist in the specified TO DESTINATION.
MAXOPENFILES *integer*	Controls the maximum number of input files that a BACKUP command can have open at any given time (the default is 8). Use this parameter to prevent "Too many open files" error messages when backing up a large number of files into a single backup set.
MAXPIECESIZE *sizeSpec*	Specifies the maximum size of each backup piece created on this channel. Example 4–2 illustrates this technique. Specify the size in bytes, kilobytes (K), megabytes (M), or gigabytes (G). The default setting is in bytes and is rounded down into kilobytes. For example, if you set MAXPIECESIZE to 5000, RMAN sets the maximum piece size at 4 kilobytes, which is the lower kilobyte boundary of 5000 bytes. **Note:** You cannot use BACKUP ... SECTION SIZE in conjunction with MAXPIECESIZE.

Syntax Element	Description
PARMS '*channel_parms*'	Specifies parameters for the sbt channel. Example 4–3 illustrates this technique. Do not use this port-specific string if you have specified DEVICE TYPE DISK. You can use the following formats for the channel parameters: ■ 'ENV=(*var1=val1, var2=val2,...*)' Specifies one or more environment variables required by the media manager in the server session corresponding to this RMAN client. Because RMAN is a client program, the ENV parameter can be used to set server session-specific variables that perform operations on behalf of the RMAN client, for example, PARMS 'ENV=(OB_DEVICE_1=tape1)'. ■ 'SBT_LIBRARY=*lib_name*' In cases where the media management library supports it, (for example, the OSB Cloud Module) you must use the SBT_PARMS parameter to specify environment variables. In all other scenarios, you use the ENV variable. Specifies which media library should be used on this sbt channel, for example, PARMS="SBT_LIBRARY=/oracle/lib/mmv.so". Note that the default library is operating system-specific (for example, libobk.so on Linux and ORASBT.DLL on Windows. **See Also:** *Oracle Database Backup and Recovery User's Guide* to learn how to integrate media management libraries
RATE *sizeSpec*	Sets the maximum number of bytes (default), kilobytes (K), megabytes (M), or gigabytes (G) that RMAN reads each second on this channel. This parameter sets an upper limit for bytes read so that RMAN does not consume too much disk bandwidth and degrade performance.
SEND '*command*'	Sends a vendor-specific command string to all allocated channels. For example, you could specify an Oracle Secure Backup media family with SEND 'OB_MEDIA_FAMILY datafile_mf'. **See Also:** Your media manager documentation to determine whether this feature is supported and when it should be used

sizeSpec

This subclause specifies the size of data. Refer to *sizeSpec::=* on page 4-2 for the syntax diagram.

Syntax Element	Description
integer [G | K | M]	Specifies the size of data in gigabytes (G), kilobytes (K), or megabytes (M).

toDestSpec

This subclause specifies either a directory or an Automated Storage Management (ASM) diskgroup for a backup piece or image copy. Refer to *toDestSpec::=* on page 4-40 for the syntax diagram.

Examples

Example 4–1 Specifying the Default Location for Disk Backups

This example allocates a disk channel that specifies a nondefault format, and then backs up the database to the nondefault location.

```
RUN
{
  ALLOCATE CHANNEL d1 DEVICE TYPE DISK FORMAT = '/disk1/bkup_%U';
```

```
  BACKUP DATABASE;
}
```

Example 4–2 Setting the Maximum Size of a Backup Piece

This example manually allocates an SBT channel, which specifies an Oracle Secure Backup tape drive, and makes a whole database backup. The `MAXPIECESIZE` parameter specifies that no backup piece written to tape should exceed 800 MB.

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
    PARMS 'SBT_LIBRARY=/usr/local/oracle/backup/lib/libobk.so,
ENV=(OB_DEVICE_1=stape1)'
    MAXPIECESIZE 800M;
  BACKUP DATABASE;
}
```

Example 4–3 Setting SBT Channel Parameters

This example configures the default SBT channel to use the Oracle Secure Backup tape drive named `stape1` and makes a database backup with the default channel:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(OB_DEVICE_1=stape1)';
BACKUP DATABASE;
```

Later you decide to back up the database to drive `stape2`. The following examples uses a manually allocated SBT channel to back up the database to `stape2`.

```
RUN
{
  ALLOCATE CHANNEL st2 DEVICE TYPE sbt
    PARMS 'ENV=(OB_DEVICE_1=stape2)';
  BACKUP DATABASE;
}
```

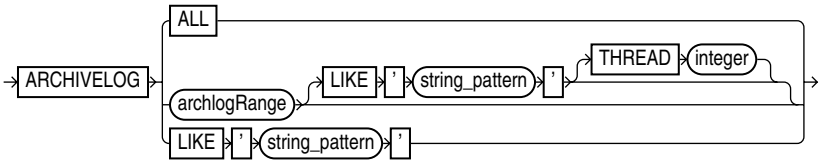
archivelogRecordSpecifier

Purpose

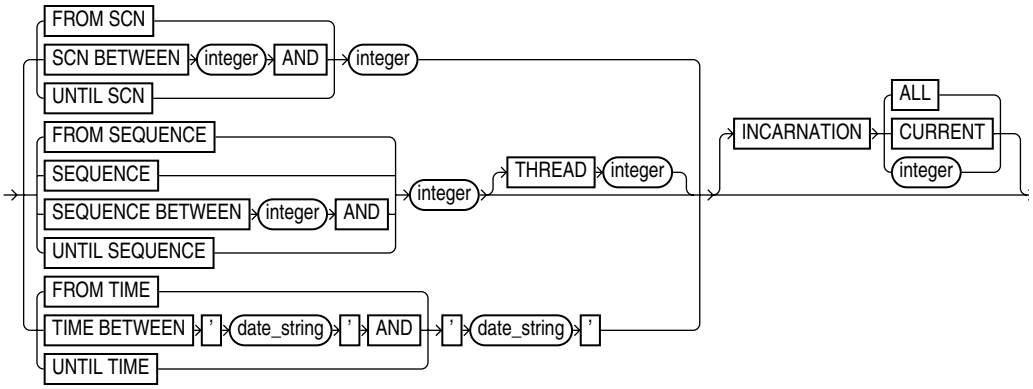
Use the *archivelogRecordSpecifier* subclause to specify a set of archived redo log files for use in RMAN operations.

Syntax

***archivelogRecordSpecifier*::=**



***archlogRange*::=**



Semantics

archivelogRecordSpecifier

This subclause specifies either all archived redo log files or logs whose filenames match a specified pattern.

Syntax Element	Description
`ALL`	Specifies all available archived logs.
`LIKE '`*`string_pattern`*`'`	Specifies all archived logs that match the specified *`string_pattern`*. You can use the same pattern matching characters that are valid in the `LIKE` operator in the SQL language to match multiple files. **See Also:** *Oracle Real Application Clusters Administration and Deployment Guide* to learn about using RMAN in an Oracle RAC configuration

archlogRange

This subclause specifies a range of archived redo log files by SCN, time, restore point (which is a label for a timestamp or SCN), or log sequence number. This subclause is useful for identifying the point to which you want the database to be recoverable.

RMAN queries the V$ARCHIVED_LOG or RC_ARCHIVED_LOG view to determine which logs to include in the range. When you specify a time, SCN, or restore point, RMAN determines the range according to the contents of the archived redo log files, not when the logs were created or backed up. When you specify the range by log sequence number, then RMAN uses the sequence number to determine the range.

Table 4–1 explains how RMAN determines which logs are in the range. The columns FIRST_TIME, NEXT_TIME, and so on refer to columns in V$ARCHIVED_LOG. Additionally, the column INCARNATION corresponds to the incarnation column in the v$database_incarnation table which should be joined with v$archived_log on the resetlogs_change# and resetlogs_time columns.

For example, if you specify FROM SCN 1000 UNTIL SCN 2000, then RMAN includes all logs whose V$ARCHIVED_LOG.NEXT_SCN value is greater than 1000 and whose V$ARCHIVED_LOG.FIRST_SCN value is less than or equal to 2000.

Table 4–1 Determining Logs for Inclusion in the Range

Subclause	FIRST_TIME	NEXT_TIME	FIRST_SCN	NEXT_SCN	SEQUENCE#	INCARNATION#
FROM TIME t1	n/a	Later than t1	n/a	n/a	n/a	n/a
FROM TIME t1 UNTIL TIME t2	Earlier than or the same as t2	Later than t1	n/a	n/a	n/a	n/a
UNTIL TIME t2	Earlier than or the same as t2	n/a	n/a	n/a	n/a	n/a
FROM SCN s1	n/a	n/a	n/a	Greater than s1	n/a	n/a
FROM SCN s1 UNTIL SCN s2	n/a	n/a	Less than or equal to s2	Greater than s1	n/a	n/a
UNTIL SCN s2	n/a	n/a	Less than or equal to s2	n/a	n/a	n/a
FROM SEQUENCE q1	n/a	n/a	n/a	n/a	Between q1 and the maximum sequence (inclusive)	n/a
FROM SEQUENCE q1 UNTIL SEQUENCE q2	n/a	n/a	n/a	n/a	Between q1 and q2 (inclusive)	n/a
UNTIL SEQUENCE q2	n/a	n/a	n/a	n/a	Between 0 and q2 (inclusive)	n/a
INCARNATION ALL	n/a	n/a	n/a	n/a	n/a	ANY
INCARNATION CURRENT	n/a	n/a	n/a	n/a	n/a	Active incarnation number
INCARNATION i	n/a	n/a	n/a	n/a	n/a	Equals i

The time must be formatted according to the Globalization Technology date format specification currently in effect. If your current Globalization settings do not specify the time, then string dates default to 00 hours, 00 minutes, and 00 seconds.

The *date_string* can be any SQL expression of type DATE, such as SYSDATE. You can use TO_DATE to specify hard-coded dates that work regardless of the current Globalization Technology settings. SYSDATE always has seconds precision regardless of the Globalization settings. Thus, if today is March 15, 2007, then SYSDATE-10 is not equivalent to 05-MAR-07 or TO_DATE('03/15/2007','MM/DD/YYYY')-10.

Specifying a sequence of archived redo log files does not guarantee that RMAN includes all redo data in the sequence. For example, the last available archived log file may end before the end of the sequence, or an archived log file in the range may be missing from all archiving destinations. RMAN includes the archived redo log files it finds and does not issue a warning for missing files.

> **See Also:** *Oracle Database Reference* to learn how to use the NLS_LANG and NLS_DATE_FORMAT environment variables to specify time format

Syntax Element	Description
FROM SCN *integer*	Specifies the beginning SCN for a range of archived redo log files. If you do not specify the UNTIL SCN parameter, then RMAN includes all available log files beginning with SCN specified in the FROM SCN parameter.
SCN BETWEEN *integer* AND *integer*	Specifies the beginning and ending SCN for a range of logs. This syntax is semantically equivalent to FROM SCN *integer1* UNTIL SCN *integer2*.
UNTIL SCN *integer*	Specifies the ending SCN for a range of archived redo log files (see Table 4–1 for the rules determining the range).
FROM SEQUENCE *integer*	Specifies the beginning log sequence number for a sequence of archived redo log files (see Table 4–1 for the rules determining the range). **Note:** You can specify all log sequence numbers in a thread by using the syntax shown in Example 4–6.
SEQUENCE *integer*	Specifies a single log sequence number.
SEQUENCE BETWEEN *integer* AND *integer*	Specifies a range of log sequence numbers. This syntax is semantically equivalent to FROM SEQUENCE *integer1* UNTIL SEQUENCE *integer2*.
UNTIL SEQUENCE *integer*	Specifies the terminating log sequence number for a sequence of archived redo log files (see Table 4–1 for the rules determining the range).
THREAD *integer*	Specifies the thread containing the archived redo log files you want to include. This parameter is only necessary if the database is in an Oracle RAC configuration. Although the SEQUENCE parameter does not require that THREAD be specified, a given log sequence always implies a thread. The thread defaults to 1 if not specified. Query V$ARCHIVED_LOG to determine the thread number for a log.
INCARNATION ALL|CURRENT|*integer*	Specifies the database incarnation for archived redo logs for the BACKUP, RESTORE and LIST commands. INCARNATION allows you to designate either current archive redo logs, archive redo logs from a previous incarnation of the database or all redo logs. CURRENT is the default. **Note:** The INCARNATION subclause is not valid for foreign archived redo log files that are received by a logical standby database for a LogMiner session. Unlike normal archived redo log files, foreign archived redo log files have a different DBID and cannot be backed up or restored on a logical standby database.
FROM TIME '*date_string*'	Specifies the beginning time for a range of archived redo log files (see Table 4–1 for the rules determining the range). Example 4–5 shows FROM TIME in a LIST command.
TIME BETWEEN '*date_string*' AND '*date_string*'	Specifies a range of times. This syntax is semantically equivalent to FROM TIME '*date_string*' UNTIL TIME '*date_string*'.
UNTIL TIME '*date_string*'	Specifies the end time for a range of archived redo log files (see Table 4–1 for the rules determining the range). Example 4–4 shows UNTIL TIME in a BACKUP command.

Examples

Example 4–4 Specifying Records by Recovery Point-in-Time

Assume that you want to be able to recover the database to a point in time 5 days before now. You want to back up a range of archived redo log files that makes this point-in-time recovery possible.

You can use the `UNTIL TIME 'SYSDATE-5'` clause to specify that all logs in the range have a first time (shown by `V$ARCHIVED_LOG.FIRST_TIME`) that is earlier than or the same as `SYSDATE-5`. This function resolves to a time with seconds precision five days before now.

```
CONNECT TARGET /
BACKUP ARCHIVELOG UNTIL TIME 'SYSDATE-5';
```

Example 4–5 Listing Archived Log Backups by Time

As shown in Table 4–1 on page 4-7, when you specify *`date_string`* for a range of archived redo log files, you are not specifying the backup time or creation time of the log. Assume that archived log 32 has a next time of March 6.

```
SQL> SELECT SEQUENCE#, NEXT_TIME
  2  FROM V$ARCHIVED_LOG
  3  WHERE SEQUENCE#='32';

 SEQUENCE# NEXT_TIME
---------- ---------
        50 06-MAR-07
```

On March 8 you run the following `BACKUP` and `LIST` commands:

```
RMAN> BACKUP ARCHIVELOG SEQUENCE 32;

Starting backup at 08-MAR-07
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=109 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
channel ORA_SBT_TAPE_1: starting archived log backup set
channel ORA_SBT_TAPE_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=32 RECID=125 STAMP=616528547
channel ORA_SBT_TAPE_1: starting piece 1 at 08-MAR-07
channel ORA_SBT_TAPE_1: finished piece 1 at 08-MAR-07
piece handle=6kic3fkm_1_1 tag=TAG20070308T111014 comment=API Version 2.0,MMS Version 10.1.0.3
channel ORA_SBT_TAPE_1: backup set complete, elapsed time: 00:00:25
Finished backup at 08-MAR-07

Starting Control File and SPFILE Autobackup at 08-MAR-07
piece handle=c-28014364-20070308-08 comment=API Version 2.0,MMS Version 10.1.0.3
Finished Control File and SPFILE Autobackup at 08-MAR-07

RMAN> LIST BACKUP OF ARCHIVELOG FROM TIME 'SYSDATE-1';

RMAN>
```

Because the next time of log 32 is earlier than the range of times specified in the `FROM TIME` clause, the preceding `LIST BACKUP` command does not show the backup of archived log 32.

Example 4–6 Crosschecking All Logs in a Redo Thread

Assume that you are managing an Oracle RAC database with two threads of redo. This example crosschecks all archived redo log files in thread `1` only.

```
CROSSCHECK ARCHIVELOG FROM SEQUENCE 0 THREAD 1;
```

completedTimeSpec

Purpose

Use the *completedTimeSpec* subclause to specify when a backup or copy completed.

Usage Notes

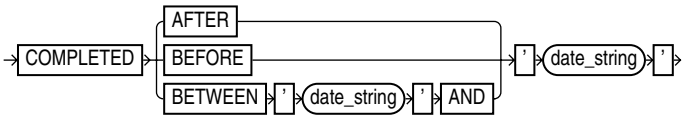
All date strings must be either:

- Formatted according to the Global Technology date format specification currently in effect.
- Created by a SQL expression that returns a DATE value, as in the following examples:
  - 'SYSDATE-30'
  - TO_DATE('09/30/2000 08:00:00','MM/DD/YY HH24:MI:SS')

  The TO_DATE function specifies dates independently of the current Global Technology environment variable settings.

Syntax

***completedTimeSpec*::=**



Semantics

Syntax Element	Description
AFTER '*date_string*'	Specifies the time after which the backup was completed (see Example 4–7).
BEFORE '*date_string*'	Specifies the time before which the backup was completed (see Example 4–9).
BETWEEN '*date_string*' AND '*date_string*'	Specifies a time range during which the backup was completed (see Example 4–8). Note that BETWEEN '*date1*' AND '*date2*' is exactly equivalent to AFTER '*date1*' BEFORE '*date2*'.

Examples

Example 4–7 Crosschecking Backups Within a Time Range

This example crosschecks the backup sets of the database made last month:

```
CROSSCHECK BACKUP OF DATABASE
  COMPLETED BETWEEN 'SYSDATE-62' AND 'SYSDATE-31';
```

Example 4–8 Deleting Expired Backups

This example deletes expired backups of archived logs made in the last two weeks:

```
DELETE EXPIRED BACKUP OF ARCHIVELOG ALL
  COMPLETED AFTER 'SYSDATE-14';
```

Example 4–9 Listing Copies

This example lists image copies of datafile `/disk1/oradata/prod/users01.dbf` made before March 9, 2007:

```
RMAN> LIST COPY OF DATAFILE '/disk1/oradata/prod/users01.dbf' COMPLETED BEFORE
'9-MAR-07';

List of Datafile Copies
=======================

Key     File S Completion Time Ckp SCN    Ckp Time
------- ---- - --------------- ---------- ---------------
3794    28   A 06-MAR-07       1010097    06-MAR-07
        Name: /disk1/oradata/prod/users01.dbf
3793    28   A 06-MAR-07       1009950    06-MAR-07
        Name: /disk2/PROD/datafile/o1_mf_users_2yvg4v6o_.dbf
        Tag: TAG20070306T105314
```

connectStringSpec

Purpose

Use the *connectStringSpec* subclause to specify the user name, 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.

Prerequisites

You must have SYSDBA privileges to CONNECT to a target or auxiliary database. Do not connect to the recovery catalog database as user SYS.

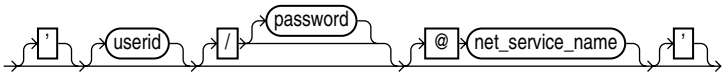
Usage Notes

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

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

Syntax

***connectStringSpec*::=**



Semantics

Syntax Element	Description
/	If you do not specify a user ID or password when connecting to a target database, then a forward slash establishes a connection as user SYS by using operating system authentication (see Example 4–12). **Note:** The forward slash depends on platform-specific environment variables.
userid	Establishes a connection to the database for the specified user. If you do not specify a password, then RMAN obtains the password interactively by displaying a prompt (see Example 4–11). The characters will not be echoed to the terminal. **Note:** The connect string must not contain any white space, but it can contain punctuation characters such as a forward slash (/) and an at sign (@).
/password	Establishes a connection for the specified user by using a password. If the target database is not open, then a password file must exist. **Caution:** Passwords entered in plain text on the command line are a security vulnerability. More secure alternatives, such as entering passwords in response to a prompt, are available.

Syntax Element	Description
@net_service_name	Establishes a connection to the database through an optional Oracle Net net service name (see Example 4–10).

Examples

Example 4–10 Connecting to a Target Database Without a Recovery Catalog

This example starts RMAN without specifying a database connection. The CONNECT command connects to a target database by using the Oracle Net service name prod in the default NOCATALOG mode. The SYS password is entered in response to a prompt.

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

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

Example 4–11 Connecting to a Target Database at the Operating System Command Line

This example connects to the target database as user SYS at the operating system command line, but without specifying a password. RMAN prompts for the password.

```
% rman TARGET SYS

Recovery Manager: Release 11.1.0.6.0 - Production on Wed Jul 11 17:51:30 2007

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

target database Password: password
```

Example 4–12 Connecting to a Target Database with Operating System Authentication

This example starts RMAN and then connects to the target database prod using operating system authentication. The example also connects to the recovery catalog database catdb using a net service name.

```
% rman
RMAN> CONNECT TARGET /

connected to target database: PROD (DBID=39525561)

RMAN> CONNECT CATALOG rman@catdb

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

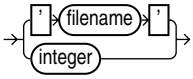
datafileSpec

Purpose

Use the *datafileSpec* subclause to specify a datafile by filename or absolute file number.

Syntax

***datafileSpec*::=**



Semantics

Syntax Element	Description
'filename'	Specifies the datafile by using either the full path or a relative filename. If you specify a relative filename, then the filename is qualified in a port-specific manner by the target database. Note that you can use `?` to represent the Oracle home and `@` for the Oracle SID (see Example 4–14). Double and single quotes are both valid (although only single quotes are shown in the diagram). **See Also:** "Quotes in RMAN Syntax" on page 1-3 to learn about the difference between single and double quotes, as well as the behavior of environment variables in RMAN quoted strings
integer	Specifies the datafile by using its absolute file number (see Example 4–13). Obtain the file number from the `V$DATAFILE`, `V$DATAFILE_COPY`, or `V$DATAFILE_HEADER` views or `REPORT SCHEMA` command output.

Examples

Example 4–13 Specifying a Datafile by Filename

This example copies datafile `/disk1/oradata/prod/users01.dbf` to disk, specifying it by filename:

```
BACKUP AS COPY
  DATAFILE '/disk1/oradata/prod/users01.dbf'
  FORMAT '/disk2/users01.cpy';
```

Example 4–14 Specifying a Datafile by Absolute File Number

This example copies datafiles 1 and 2 to disk, specifying them by file number:

```
BACKUP AS COPY
  DATAFILE 1 FORMAT '/disk2/df1.cpy'
  DATAFILE 2 FORMAT '/disk2/df1.cpy';
```

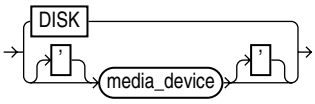
deviceSpecifier

Purpose

Use the *deviceSpecifier* subclause to specify the type of storage for a backup.

Syntax

***deviceSpecifier*::=**



Semantics

Syntax Element	Description
DISK	Specifies a disk storage device (see Example 4–16 on page 4-16).
media_device	Specifies a sequential I/O device or access method for storage (see Example 4–15 on page 4-16). The *media_device* variable specifies a case-insensitive name for a media manager. The syntax and semantics of sequential I/O device types are platform-specific. The most common value is sbt or sbt_tape (which are synonymous values). **Note:** RMAN stores the value sbt in the recovery catalog as sbt_tape for backward compatibility.

Examples

Example 4–15 Allocating a Tape Channel

This example allocates a maintenance channel for a media management device:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP;
RELEASE CHANNEL;
```

Example 4–16 Backing Up the Database to Disk

This example backs up the database to disk:

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

fileNameConversionSpec

Purpose

Use the *fileNameConversionSpec* subclause to specify one or more patterns to be used in generating new filenames based on old filenames. Used with BACKUP, CONVERT, and DUPLICATE as one way of generating output filenames.

Usage Notes

The rules for string patterns and how they affect file naming are the same as those for the initialization parameter DB_FILE_NAME_CONVERT. In parentheses, provide an even number of string patterns.

When RMAN generates a new filename based on an old one, it compares the original filename to the first member of each pair of string patterns. The first time that RMAN finds a pattern that is a substring of the original filename, RMAN generates the new filename by substituting the second member of the pair for the substring that matched.

The pattern does not have to match at the beginning of the filename. The following command creates an image copy of datafile /disk1/dbs/users.dbf as /disk1/newdbs/users.dbf:

```
BACKUP AS COPY
    DB_FILE_NAME_CONVERT = ('dbs','newdbs');
    TABLESPACE users;
```

When multiple possible matches exist for a given filename being converted, RMAN uses the first match in the list of patterns to generate the new filename. The following command has the same effect as the previous example because the pattern dbs matches the filename, so that the filename is never compared to the second pattern /disk1:

```
BACKUP AS COPY
    DB_FILE_NAME_CONVERT = ('dbs','newdbs','/disk1','/newdisk')
    TABLESPACE users;
```

For the CONVERT TABLESPACE, CONVERT DATABASE, and BACKUP AS COPY commands, if the source files for these operations are Oracle-managed files, then you cannot use *fileNameConversionSpec* to convert the source filenames into new output filenames. For Oracle-managed files, either in Automated Storage Management (ASM) or in ordinary file system storage, the database must be allowed to generate the filenames for the output files. For example, an OMF filename for a datafile in non-ASM storage might be of the following form:

```
/private/boston/datafile/01_mf_system_ab12554_.dbf
```

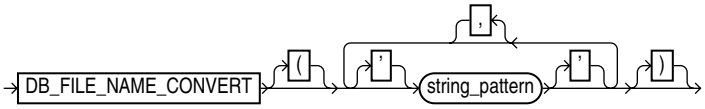
An OMF filename in ASM storage might be of the following form:

```
+DISK/boston/datafile/system.256.4543080
```

Only the database can generate and manage OMF filenames. Typically, substituting the name of a different disk group or a different OMF location into an OMF filename does not produce a valid filename in the new destination. To convert OMF filenames for storage in another OMF location, use an alternative such as a FORMAT clause with these commands to specify the new output location and allow the database to manage the specific output filenames.

Syntax

***fileNameConversionSpec*::=**



Semantics

Syntax Element	Description
string_pattern	Specifies pairs of strings used to convert the filenames. You can use as many pairs of primary and standby replacement strings as required. For example, you could set the string pattern to a value such as: `DB_FILE_NAME_CONVERT = ('str1','str2','str3', 'str4' ...)` In this example, `str1` is a pattern matching the original filename, `str2` is the pattern replacing `str1` in the generated filename, `str3` is a pattern matching the original filename, and `str4` is the pattern replacing `str3` in the generated filename.

Examples

Example 4–17 Using DB_FILE_NAME_CONVERT with a Single Conversion Pair

In this example, the tablespace `users` contains datafiles in directory `/disk1/oradata/prod/`, whereas tablespace `tools` contains datafiles in `/disk1/oradata/prod/`. For each datafile to be converted, if `disk1/oradata/prod` is a substring of the datafile name, then the image copy name is created by replacing the string with `disk2`.

```
BACKUP AS COPY
  DB_FILE_NAME_CONVERT = ('disk1/oradata/prod','disk2')
  TABLESPACE users, tools;
```

Example 4–18 Using DB_FILE_NAME_CONVERT with Multiple Conversion Pairs

This example creates image copies of the same datafiles described in Example 4–17. The first string in each pair specifies a pattern to match in the name of the source datafiles. The second string in each pair is the substitution pattern to use when generating the names of the image copies.

```
BACKUP AS COPY
    DB_FILE_NAME_CONVERT=('/disk1/oradata/prod/users','/disk2/users',
                          '/disk1/oradata/prod/tools','/tmp/tools')
    TABLESPACE tools, users;
```

The following sample output for this command demonstrates how RMAN uses the conversion pairs to name the output image copies:

```
Starting backup at 08-MAR-07
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
input datafile file number=00027 name=/disk1/oradata/prod/tools01.dbf
output file name=/tmp/tools01.dbf tag=TAG20070308T143300 RECID=33 STAMP=616689181
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00028 name=/disk1/oradata/prod/users01.dbf
output file name=/disk2/users01.dbf tag=TAG20070308T143300 RECID=34 STAMP=616689182
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 08-MAR-07
```

```
Starting Control File and SPFILE Autobackup at 08-MAR-07
piece handle=/disk2/PROD/autobackup/2007_03_08/o1_mf_s_616689184_2z13s1kx_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 08-MAR-07
```

forDbUniqueNameOption

Purpose

Use the *forDbUniqueNameOption* subclause to specify either all databases or a specific database in a Data Guard environment.

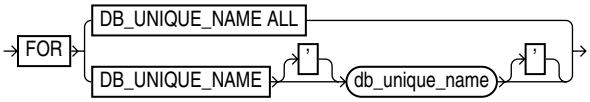
Usage Notes

The DBID for a primary database is identical to the DBID of its associated physical standby databases. A database is uniquely identified in the recovery catalog by a DBID and the value of its DB_UNIQUE_NAME initialization parameter.

When you specify *forDbUniqueNameOption* for any command, RMAN restricts its operations to the objects that are associated exclusively with the database with the specified DB_UNIQUE_NAME. For example, if you use this option with the LIST command, then RMAN lists only the objects associated exclusively with the database with the specified DB_UNIQUE_NAME. Note that objects that are not associated with any database (SITE_KEY column in the recovery catalog view is null) are not listed.

Syntax

***forDbUniqueNameOption*::=**



Semantics

Syntax Element	Description
FOR DB_UNIQUE_NAME ALL	Specifies all primary and standby databases in the recovery catalog that share the DBID of the target database (or DBID specified by the SET DBID command).
FOR DB_UNIQUE_NAME *db_unique_name*	Specifies the primary or standby database in the recovery catalog with the specified *db_unique_name*.

Examples

Example 4–19 Listing Expired Backups Associated with a Standby Database

This example connects to a recovery catalog and sets the DBID for the Data Guard environment. All primary and standby databases in this environment share the same DBID. The LIST command lists all expired backups associated with database standby1:

```
RMAN> CONNECT CATALOG rman@catdb;

recovery catalog database Password: password
connected to recovery catalog database

RMAN> SET DBID 3257174182;
RMAN> LIST EXPIRED BACKUP FOR DB_UNIQUE_NAME standby1;
```

foreignlogRecordSpecifier

Purpose

Use the *foreignlogRecordSpecifier* subclause to specify a set of foreign archived redo log files for use in RMAN operations.

Usage Notes

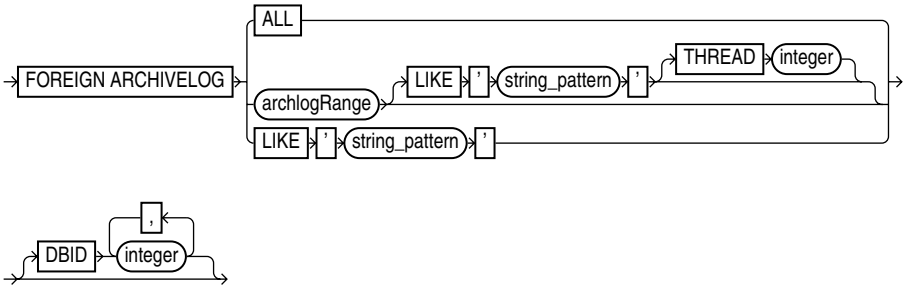
Foreign archived redo log files are received by a logical standby database for a LogMiner session. Unlike normal archived logs, foreign archived logs have a different DBID. For this reason, they cannot be backed up or restored on a logical standby database.

A logical standby database creates foreign archived logs in the fast recovery area if the following conditions are met:

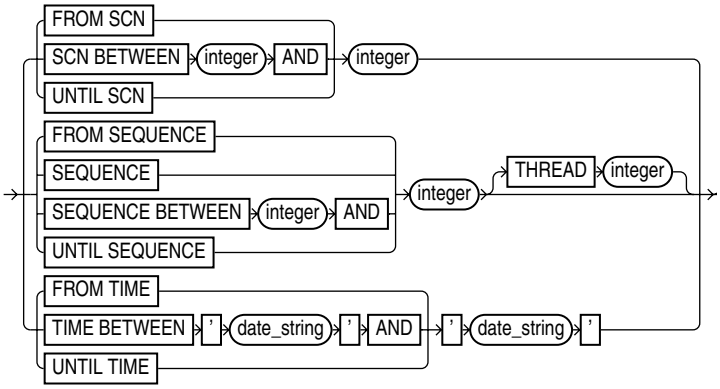
- A fast recovery area is configured on the logical standby database.
- One of the `LOG_ARCHIVE_DEST_n` initialization parameters is set to `'LOCATION=USE_DB_RECOVERY_FILE_DEST'` with a proper `VALID_FOR` setting to receive foreign archived logs.
- The `COMPATIBLE` initialization parameter is set to a value greater than or equal to 11.0.0.

Syntax

***foreignlogRecordSpecifier*::=**



***archlogRange*::=**



Semantics

The semantics are exactly the same as for *archivelogRecordSpecifier* on page 4-6 except that the logs are foreign archived redo log files.

Examples

Example 4–20 Crosschecking Foreign Archived Redo Log Files

This example crosschecks all foreign archived redo log files:

```
RMAN> CROSSCHECK FOREIGN ARCHIVELOG ALL;
```

formatSpec

Purpose

Use the *formatSpec* subclause to specify a filename format or an Automatic Storage Management disk group for a backup piece or image copy. If you do not specify a value for the FORMAT parameter, then RMAN either creates the backup in the fast recovery area if it is enabled, or in a platform-specific directory (for example, ?/dbs on UNIX) if a fast recovery area is not enabled. In either case, RMAN uses the variable %U to name the backup.

> **Tip:** *Oracle Database SQL Language Reference* to learn how to create and name Automated Storage Manager disk groups

Usage Notes

Any name that is valid as a sequential filename on the platform is allowed, so long as each backup piece or copy has a unique name. If backing up to disk, then any valid disk filename is allowed, provided it is unique.

You cannot specify an Oracle Managed Files filename as the format for a backup. For example, if +DISK1/datafile/system.732.609791431 is an OMF filename, then you cannot specify this filename in the FORMAT parameter.

Environment variables are not valid in the FORMAT parameter.

The entire *format_string* is processed in a port-specific manner by the target instance to derive the final backup piece name. The substitution variables listed in "Semantics" on page 4-24 are available in FORMAT strings to aid in generating unique filenames. The formatting of this information varies by platform.

You can specify up to four FORMAT strings. RMAN uses the second, third, and fourth values only when BACKUP COPIES, SET BACKUP COPIES, or CONFIGURE ... BACKUP COPIES is in effect. When choosing the format for each backup piece, RMAN uses the first format value for copy 1, the second format value for copy 2, and so on. If the number of format values exceeds the number of copies, then the extra formats are not used. If the number of format values is less than the number of copies, then RMAN reuses the format values, starting with the first one.

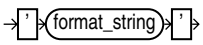
Specify *format_string* in any of the following places, listed in order of precedence:

1. The *backupSpec* clause
2. The BACKUP command
3. The ALLOCATE CHANNEL command
4. The CONFIGURE CHANNEL command

If specified in more than one of these places, then RMAN searches for the FORMAT parameter in the order shown.

Syntax

***formatSpec*::=**



Semantics

formatSpec

The following table lists RMAN substitution variables that are valid in format strings.

Syntax Element	Description
`%a`	Specifies the activation ID of the database.
`%b`	Specifies the file name stripped of directory paths. It is only valid for `SET NEWNAME` and backup when producing image copies It will yield errors if used as a format specification for a backup that produces backup pieces.
`%c`	Specifies the copy number of the backup piece within a set of duplexed backup pieces. If you did not duplex a backup, then this variable is 1 for backup sets and 0 for proxy copies. If one of these commands is enabled, then the variable shows the copy number. The maximum value for `%c` is 256.
`%d`	Specifies the name of the database (see Example 4–22 on page 4-25).
`%D`	Specifies the current day of the month from the Gregorian calendar in format `DD`.
`%e`	Specifies the archived log sequence number.
`%f`	Specifies the absolute file number (see Example 4–22 on page 4-25).
`%F`	Combines the DBID, day, month, year, and sequence into a unique and repeatable generated name. This variable translates into *`c-IIIIIIIIII-YYYYMMDD-QQ`*, where: ■ *`IIIIIIIIII`* stands for the DBID. The DBID is printed in decimal so that it can be easily associated with the target database. ■ *`YYYYMMDD`* is a time stamp in the Gregorian calendar of the day the backup is generated ■ *`QQ`* is the sequence in hexadecimal number that starts with 00 and has a maximum of 'FF' (256)
`%h`	Specifies the archived redo log thread number.
`%I`	Specifies the DBID.
`%M`	Specifies the month in the Gregorian calendar in format `MM`.
`%N`	Specifies the tablespace name. This substitution variable is only valid when backing up datafiles as image copies.
`%n`	Specifies the name of the database, padded on the right with `x` characters to a total length of eight characters. For example, if `prod1` is the database name, then the padded name is `prod1xxx`.
`%p`	Specifies the 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. **Note:** If you specify `PROXY`, then the `%p` variable must be included in the `FORMAT` string either explicitly or implicitly within `%U`.
`%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`.
`%t`	Specifies the backup set time stamp, 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.
`%T`	Specifies the year, month, and day in the Gregorian calendar in this format: `YYYYMMDD`.
`%u`	Specifies an 8-character name constituted by compressed representations of the backup set or image copy number and the time the backup set or image copy was created.

Syntax Element	Description
%U	Specifies a system-generated unique filename (default). The meaning of %U is different for image copies and backup pieces. For a backup piece, %U specifies a convenient shorthand for %u_%p_%c that guarantees uniqueness in generated backup filenames. For an image copy of a datafile, %U means the following: data-D-%d_id-%I_TS-%N_FNO-%f_%u For an image copy of an archived redo log, %U means the following: arch-D_%d-id-%I_S-%e_T-%h_A-%a_%u For an image copy of a control file, %U means the following: cf-D_%d-id-%I_%u
%Y	Specifies the year in this format: YYYY.
%%	Specifies the percent (%) character. For example, %%Y translates to the string %Y.

Example

Example 4–21 Specifying an ASM Disk Group

This example copies the database to ASM disk group DISK1:

```
BACKUP AS COPY DATABASE FORMAT '+DATAFILE';
```

Example 4–22 Specifying a Format for Datafile Copies

This example copies two datafiles with tag LATESTCOPY to directory /disk2:

```
BACKUP AS COPY
  COPY OF DATAFILE 27, 28
  FROM TAG 'LATESTCOPY'
  FORMAT '/disk2/Datafile%f_Database%d';
```

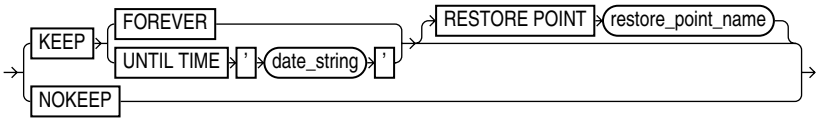
keepOption

Purpose

Use the *keepOption* subclause to specify the status of a backup or copy in relation to a retention policy.

Syntax

***keepOption*::=**



Usage Notes

RMAN does not consider backup pieces with the KEEP option when computing the backup retention policy. If available, RMAN uses these backups for disaster recovery restore operations, but their purpose is to produce a snapshot of the database that can be restored on another system for testing or historical usage.

Semantics

Syntax Element	Description
KEEP	Specifies the backup as an **archival backup**, which is a self-contained backup that is exempt from the configured retention policy. An archival backup is self-contained because is contains all files necessary to restore the backup and recover it to a consistent state. If the database is open during the backup, then RMAN automatically generates and backs up the archived redo log files needed to make the database backup consistent (see Example 2–26 on page 2-50). RMAN does not consider backup pieces with the KEEP option when computing the retention policy. If available, RMAN uses these backups for disaster recovery restore operations, but their purpose is to produce a snapshot of the database that can be restored on another system for testing or historical usage. **Note:** You cannot use KEEP to override the retention policy for files stored in the fast recovery area. If you specify KEEP when backing up to the recovery area, then RMAN issues an error. When KEEP is specified, RMAN creates multiple backup sets. RMAN backs up datafiles, archived redo log files, the control file, and the server parameter file with the options specified in the first *backupOperand*. Since a backup of the control file is already created, an autobackup of the control file is not created. RMAN uses the FORMAT, POOL, and TAG parameters for all the backups. For this reason, the FORMAT string must allow for the creation of multiple backup pieces. Specifying %U is the easiest way to meet this requirement. **See Also:** CONFIGURE and *Oracle Database Backup and Recovery User's Guide* to learn more about autobackup of the control file. When KEEP is specified with INCREMENTAL LEVEL, the parent backup must be a KEEP backup and have the same TAG string. Unless both these criteria are met, the backup cannot be created. Backups that meet these criteria will not interfere with other nightly or archival backup tasks. **Note:** A recovery catalog is only required for KEEP FOREVER. No other KEEP options require a catalog.

Syntax Element	Description
FOREVER	Specifies that the backup or copy never becomes obsolete (see Example 2–27 on page 2-50). A recovery catalog is required when FOREVER is specified because the backup records eventually age out of the control file.
UNTIL TIME *'date_string'*	Specifies the time until which the backup or copy must be kept. After this time the backup is obsolete, regardless of the backup retention policy settings. You can either specify a specific time by using the current NLS_DATE_FORMAT, or a SQL date expression such as 'SYSDATE+365'. If you specify a KEEP TIME such as 01-JAN-07, then the backup becomes obsolete one second after midnight on this date. If you specify a KEEP time such as 9:00 p.m, then the backup becomes obsolete at 9:01 p.m.
RESTORE POINT *restore_point_name*	Creates a normal restore point matching the SCN to which RMAN must recover the backup to a consistent state (see Example 2–26 on page 2-50). The restore point name must not already exist. The SCN is captured just after the datafile backups complete. The restore point is a label for the SCN to which this archival backup can be restored and recovered, enabling the database to be opened. In contrast, the UNTIL TIME clause specifies the date until which the backup must be kept. **Note:** The RESTORE POINT parameter is not valid with the CHANGE command.
NOKEEP	Specifies that any KEEP attributes no longer apply to the backup. Thus, the backup is a normal backup that is subject to the configured backup retention policy. This is the default behavior if no KEEP option is specified.

Examples

Example 4–23 Creating a Consistent Database Backup for Archival

This example makes a database backup with tag Q107 and specifies that it should never be considered obsolete (partial sample output included). The archived redo log files necessary to make the datafiles consistent are included in the backup set.

```
RMAN> BACKUP TAG Q107 DATABASE KEEP FOREVER;

Starting backup at 24-JAN-07

current log archived
allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=105 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
backup will never be obsolete
archived logs required to recover from this backup will be backed up
channel ORA_SBT_TAPE_1: starting full datafile backup set
channel ORA_SBT_TAPE_1: specifying datafile(s) in backup set
.
.
.
```

Example 4–24 Removing the KEEP Attributes for a Backup

This example backs up all archived redo log files. The KEEP clause specifies that one second after midnight on January 1, 2008 the backup is considered obsolete.

```
RMAN> BACKUP KEEP UNTIL TIME '01-JAN-08' ARCHIVELOG ALL;
```

The following command removes the KEEP attributes of all archived redo log backups (sample output included):

```
RMAN> CHANGE BACKUP OF ARCHIVELOG ALL NOKEEP;
```

```
using channel ORA_SBT_TAPE_1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=77 device type=DISK
keep attributes for the backup are deleted
backup set key=330 RECID=19 STAMP=612722760
keep attributes for the backup are deleted
backup set key=397 RECID=22 STAMP=612722884
```

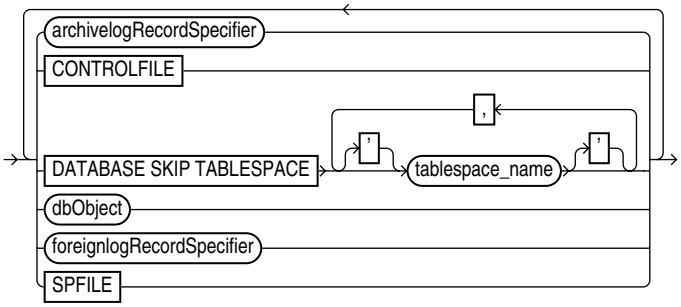
listObjList

Purpose

Use the `listObjList` subclause to specify database files and archived redo log files.

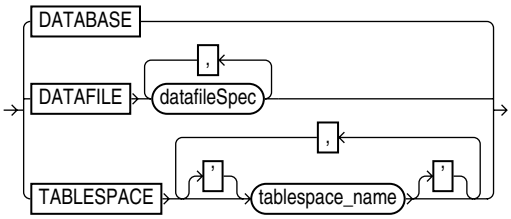
Syntax

listObjList::=



(*archivelogRecordSpecifier::=* on page 4-6)

dbObject::=



(*datafileSpec::=* on page 4-15)

Semantics

listObjList

Syntax Element	Description
archivelogRecordSpecifier	Specifies a range of archived redo log files. **See Also:** *archivelogRecordSpecifier* on page 4-6
CONTROLFILE	Specifies the current control file.
DATABASE SKIP TABLESPACE *tablespace_name*	Omits the specified tablespaces from the DATABASE specification.
foreignlogRecordSpecifier	Processes the specified foreign archived redo log files. **See Also:** *foreignlogRecordSpecifier* on page 4-6
SPFILE	Specifies the current server parameter file.

dbObject

Syntax Element	Description
DATABASE	Specifies backup sets or image copies of all files in the current database.
DATAFILE *datafileSpec*	Specifies datafiles by filename or file number. The clause specifies datafile image copies or backup sets that contain at least one of the datafiles. **See Also:** *datafileSpec* on page 4-15
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.

Examples

Example 4–25 Listing Datafile Copies

The following command lists image copies of all the files in the database, skipping the `temp` tablespace, which is a dictionary-managed temporary tablespace:

```
LIST COPY OF DATABASE
  SKIP TABLESPACE temp;
```

Example 4–26 Crosschecking Archived Redo Log Files

The following example queries the media manager for the status of server parameter file and archived redo log backups created in the last three months. The example includes sample output.

```
RMAN> ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;

allocated channel: ORA_MAINT_SBT_TAPE_1
channel ORA_MAINT_SBT_TAPE_1: SID=103 device type=SBT_TAPE
channel ORA_MAINT_SBT_TAPE_1: Oracle Secure Backup

RMAN> CROSSCHECK BACKUP OF SPFILE ARCHIVELOG FROM TIME 'SYSDATE-90';

crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=8cic4031_1_1 RECID=195 STAMP=616693857
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-28014364-20070308-15 RECID=196 STAMP=616693875
Crosschecked 2 objects

RMAN> RELEASE CHANNEL;

released channel: ORA_MAINT_SBT_TAPE_1
```

Example 4–27 Deleting Expired Backups

The following command performs a crosscheck of all backups. One backup is found to be expired. The example then deletes all expired backups (sample output included).

```
RMAN> CROSSCHECK BACKUP;

allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=104 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=103 device type=DISK
crosschecked backup piece: found to be 'EXPIRED'
backup piece handle=/disk2/PROD/autobackup/2007_03_08/o1_mf_s_616690991_2z15kl5h_.bkp
 RECID=191 STAMP=616690994
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=8cic4031_1_1 RECID=195 STAMP=616693857
```

```
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=c-28014364-20070308-15 RECID=196 STAMP=616693875
Crosschecked 3 objects

RMAN> DELETE EXPIRED BACKUP;

using channel ORA_SBT_TAPE_1
using channel ORA_DISK_1

List of Backup Pieces
BP Key  BS Key  Pc# Cp# Status      Device Type Piece Name
------- ------- --- --- ----------- ----------- ----------
7678    7677    1   1   EXPIRED     DISK
 /disk2/PROD/autobackup/2007_03_08/o1_mf_s_616690991_2z15kl5h_.bkp

Do you really want to delete the above objects (enter YES or NO)? YES
deleted backup piece
backup piece handle=/disk2/PROD/autobackup/2007_03_08/o1_mf_s_616690991_2z15kl5h_.bkp
 RECID=191 STAMP=616690994
Deleted 1 EXPIRED objects
```

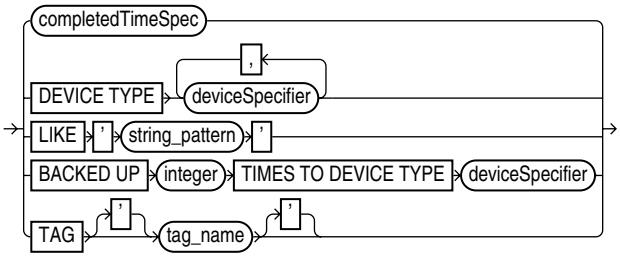
maintQualifier

Purpose

Use the *maintQualifier* subclause to specify database files and archived redo log files.

Syntax

***maintQualifier*::=**



(*completedTimeSpec::=* on page 4-11, *deviceSpecifier::=* on page 4-16)

Semantics

maintQualifier

Syntax Element	Description
completedTimeSpec	Specifies a range of time for completion of the backup or copy. **See Also:** *completedTimeSpec* on page 4-11
DEVICE TYPE *deviceSpecifier*	Allocates automatic channels for the specified device type only. This option is valid only if you have configured automatic channels and have not manually allocated channels. For example, if you configure automatic disk and tape channels, and issue CHANGE...DEVICE TYPE DISK, then RMAN allocates only disk channels. **See Also:** *deviceSpecifier* on page 4-16
LIKE '*string_pattern*'	Restricts datafile copies by specifying a filename pattern. The pattern can contain Oracle pattern matching characters percent sign (%) and underscore (_). RMAN only operates on those files whose name matches the pattern. **Note:** You cannot use the LIKE option with the LIST ... ARCHIVELOG command or with backup pieces.
BACKED UP *integer* TIMES TO DEVICE TYPE *deviceSpecifier*	Restricts the command to archived logs that have been successfully backed up *integer* or more times to the specified media. This option applies only to archived redo log files. When the BACKED UP option is used with the DELETE ARCHIVELOG command, RMAN uses the BACKED UP setting rather than the configured settings to determine whether an archived log can be deleted. That is, CONFIGURE ARCHIVELOG DELETION POLICY is overridden. Use FORCE with DELETE ARCHIVELOG to override this configuration as well as any mismatches between media and repository.

Syntax Element	Description
TAG *tag_name*	Specifies the datafile copies and backup sets by tag. Tag names are not case sensitive and display in all uppercase. **See Also:** BACKUP on page 2-19 for a description of how a tag can be applied to an individual copy of a duplexed backup set, and also for a description of the default filename format for tags

Example

Example 4–28 Listing Backups in a Specific Location

The following command lists all image copies located in directory `/disk2`:

```
RMAN> LIST COPY LIKE '/disk2/%';

List of Datafile Copies
=======================

Key     File S Completion Time Ckp SCN    Ckp Time
------- ---- - --------------- ---------- ---------------
9855    1    A 08-MAR-07       1394701    08-MAR-07
        Name: /disk2/data_D-PROD_I-28014364_TS-SYSTEM_FNO-1_8eic410j
        Tag: TAG20070308T160643

9856    2    A 08-MAR-07       1394735    08-MAR-07
        Name: /disk2/data_D-PROD_I-28014364_TS-SYSAUX_FNO-2_8fic412a
        Tag: TAG20070308T160643
```

Example 4–29 Deleting Archived Logs That Are Already Backed Up

The following command deletes only those archived logs that have been successfully backed up two or more times to tape. In this example, only the sequence 36 archived log meets these criteria.

```
RMAN> DELETE ARCHIVELOG ALL BACKED UP 2 TIMES TO DEVICE TYPE sbt;

released channel: ORA_SBT_TAPE_1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=104 device type=DISK
RMAN-08138: WARNING: archived log not deleted - must create more backups
archived log file name=/disk1/oradata/prod/arch/archiver_1_37_616443024.arc thread=1
sequence=37
List of Archived Log Copies for database with db_unique_name PROD
=====================================================================

Key     Thrd Seq     S Low Time
------- ---- ------- - ---------
9940    1    36      A 08-MAR-07
        Name: /disk1/oradata/prod/arch/archiver_1_36_616443024.arc


Do you really want to delete the above objects (enter YES or NO)? Y
deleted archived log
archived log file name=/disk1/oradata/prod/arch/archiver_1_36_616443024.arc RECID=129
STAMP=616695115
Deleted 1 objects
```

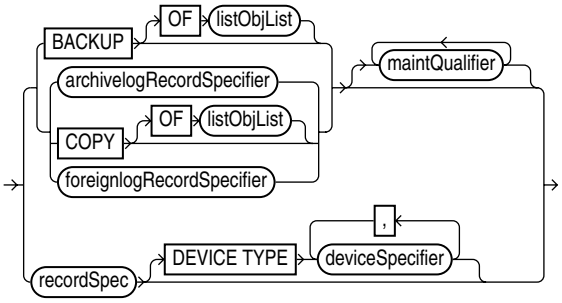
maintSpec

Purpose

Use the *maintSpec* subclause to specify the backup files operated on by the CHANGE, CROSSCHECK, and DELETE commands.

Syntax

maintSpec::=



(*listObjList::=* on page 4-29, *maintQualifier::=* on page 4-32, *archivelogRecordSpecifier::=* on page 4-6, *recordSpec::=* on page 4-37, *deviceSpecifier::=* on page 4-16)

Semantics

maintSpec

Syntax Element	Description
BACKUP	Processes backup sets. For processing image copies, use the option COPY. If you do not use the OF clause with CHANGE BACKUP, then RMAN changes all backup sets recorded in the repository. If you do not use the OF clause with CROSSCHECK BACKUP, then RMAN crosschecks backups of the whole database. If you do not use the OF clause with DELETE BACKUP, then RMAN deletes backups of the whole database.
OF *listObjList*	Restricts the list of files operated on to the object type specified in the *listObjList* clause. **See Also:** *listObjList* on page 4-29
archivelogRecordSpecifier	Processes the specified archived redo log files. If you specify DELETE ARCHIVELOG without the BACKED UP clause, then RMAN uses the configured settings to determine whether an archived log can be deleted (CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP). If you specify DELETE ARCHIVELOG with the BACKED UP clause, then RMAN uses the DELETE settings to determine whether an archived log can be deleted. Use FORCE with DELETE ARCHIVELOG to override a configured deletion policy as well as any mismatches between media and repository. **See Also:** *archivelogRecordSpecifier* on page 4-6

Syntax Element	Description
COPY	Processes datafile copies, control file copies, and archived redo log files. If you do not specify an option for CHANGE COPY, then the command operates on all image copies recorded in the repository. If you are using CROSSCHECK COPY, then by default the command checks all image copies of all files in the database with status AVAILABLE or EXPIRED. If you are using DELETE COPY, then by default COPY removes copies of all files in the database. Specify the EXPIRED option to remove only copies that are marked EXPIRED in the repository.
OF *listObjList*	Restricts the list of objects operated on to the object type specified in the *listObjList* clause. If you do not specify an object, then the command defaults to all copies. Note that CHANGE COPY OF DATABASE includes datafiles but not control files. **See Also:** *listObjList* on page 4-29
foreignlogRecordSpecifier	Processes the specified foreign archived redo log files. **See Also:** *foreignlogRecordSpecifier* on page 4-6
maintQualifier	Restricts the command based on the specified options. **See Also:** *maintQualifier* on page 4-32
recordSpec	Specifies the file that you are performing maintenance on. If you use the BACKUPSET parameter in *recordSpec*, then the keys identify backup sets for use with the CHANGE, CROSSCHECK and DELETE commands. For more details, see "LIST Command Output" on page 2-170 for an explanation of the column headings of the LIST output tables. Use the KEY column of the output to obtain the primary key usable in the CHANGE and DELETE commands. **See Also:** *recordSpec* on page 4-37
DEVICE TYPE *deviceSpecifier*	Allocates automatic channels for the specified device type only. This option is valid only if you have configured automatic channels and have not manually allocated channels. For example, if you configure automatic disk and tape channels and run CROSSCHECK...DEVICE TYPE DISK, then RMAN allocates only disk channels. **See Also:** *deviceSpecifier* on page 4-16

Examples

Example 4–30 Crosschecking Backups

The following command crosschecks backups of archived redo log files:

```
RMAN> CROSSCHECK BACKUP OF ARCHIVELOG ALL;

allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=103 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
using channel ORA_DISK_1
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=8cic4031_1_1 RECID=195 STAMP=616693857
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=8oic41ad_1_1 RECID=198 STAMP=616695118
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=8qic41c3_1_1 RECID=200 STAMP=616695171
Crosschecked 3 objects
```

obsOperandList

Purpose

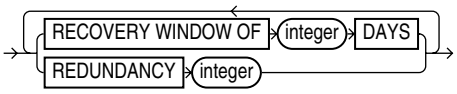
Use the *obsOperandList* subclause used to specify which criteria are used to mark backups as obsolete.

Usage Notes

Using both RECOVERY WINDOW and REDUNDANCY in a single REPORT OBSOLETE or DELETE OBSOLETE command is not supported.

Syntax

***obsOperandList*::=**



Semantics

obsOperandList

Syntax Element	Description
RECOVERY WINDOW OF *integer* DAYS	Specifies that RMAN should report as obsolete those backup sets and image copies that are not needed to recover the database to any point within the last *integer* days. **See Also:** CONFIGURE on page 2-65 for an explanation of the recovery window
REDUNDANCY *integer*	Specifies the minimum level of redundancy considered necessary for a backup set or image copy to be obsolete. A datafile copy is obsolete if there are at least *integer* more recent backup sets or image copies of this file; a datafile backup set is obsolete if there are at least *integer* more recent backup sets or image copies of each datafile contained in the backup set. For example, REDUNDANCY 2 means that there must be at least two more recent backup sets or image copies of a datafile for any other backup set or image copy to be obsolete.

Example

Example 4–31 Deleting Obsolete Backups

The following command deletes all backups and copies not needed to recover the database to an SCN within the last 30 days:

```
DELETE OBSOLETE RECOVERY WINDOW OF 30 DAYS;
```

recordSpec

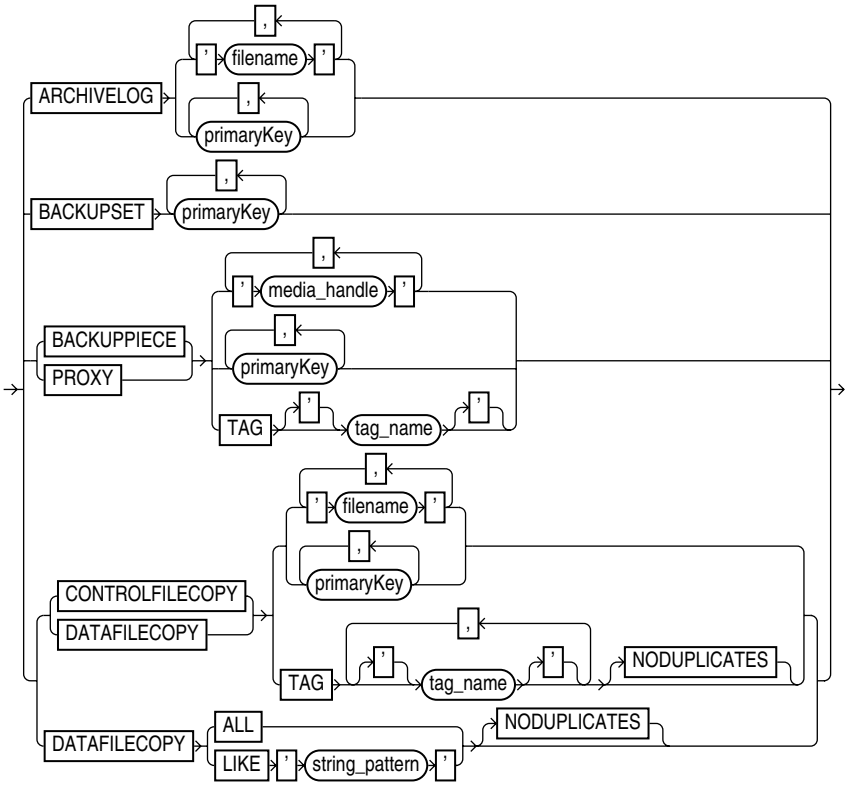
Purpose

Use the *recordSpec* subclause to specify which backups or copies the CHANGE, CROSSCHECK, DELETE, and LIST commands should process.

Most *recordSpec* options allow you to specify a primary key. Use the output of the LIST command to obtain primary keys.

Syntax

***recordSpec*::=**



Semantics

Syntax Element	Description
ARCHIVELOG	Specifies an archived redo log by either primary key or filename.
BACKUPSET	Specifies a backup set by primary key.
BACKUPPIECE	Specifies a backup piece by media handle, primary key, or tag name.
PROXY	Specifies a proxy copy by media handle, primary key, or tag name.
CONTROLFILECOPY	Specifies a control file copy by primary key, filename pattern (`'filename'`), or TAG *tag_name*. If you crosscheck a control file copy, then you must specify a filename rather than a primary key.

Syntax Element	Description
DATAFILECOPY	Specifies a datafile copy by primary key, filename pattern (`'filename'`), tag (`TAG tag_name`), or matching string (`LIKE 'string_pattern'`). Specify `ALL` to indicate all datafile copies recorded in the RMAN repository.
NODUPLICATES	Specifies that only one copy of the control file or datafile copy specified by the rest of the clause should be the target of the operation, even when there are multiple copies.

Examples

Example 4–32 Crosschecking Backups

This example crosschecks backup sets specified by primary key:

```
RMAN> LIST BACKUP SUMMARY;

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
------- -- -- - ----------- --------------- ------- ------- ---------- ---
8504    B  A  A SBT_TAPE    08-MAR-07       1       1       NO         TAG20070308T155057
8558    B  F  A SBT_TAPE    08-MAR-07       1       1       NO         TAG20070308T155114
9872    B  F  A DISK        08-MAR-07       1       1       NO         TAG20070308T160830
9954    B  A  A SBT_TAPE    08-MAR-07       1       1       NO         TAG20070308T161157
9972    B  F  A SBT_TAPE    08-MAR-07       1       1       NO         TAG20070308T161224
10021   B  A  A SBT_TAPE    08-MAR-07       1       1       NO         TAG20070308T161251
10042   B  F  A SBT_TAPE    08-MAR-07       1       1       NO         TAG20070308T161308
10185   B  F  A DISK        08-MAR-07       1       1       NO         TAG20070308T170532
10210   B  F  A DISK        08-MAR-07       1       1       NO         TAG20070308T170535

RMAN> CROSSCHECK BACKUPSET 9872, 10185, 10210;

allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: SID=103 device type=SBT_TAPE
channel ORA_SBT_TAPE_1: Oracle Secure Backup
using channel ORA_DISK_1
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/disk2/PROD/autobackup/2007_03_08/o1_mf_s_616694910_2z19d0wg_.bkp
RECID=197 STAMP=616694912
crosschecked backup piece: found to be 'AVAILABLE'
backup piece
handle=/disk2/PROD/backupset/2007_03_08/o1_mf_nnsnf_TAG20070308T170532_2z1dpwz6_.bkp
RECID=202 STAMP=616698332
crosschecked backup piece: found to be 'AVAILABLE'
backup piece handle=/disk2/PROD/autobackup/2007_03_08/o1_mf_s_616698335_2z1dq0d0_.bkp
RECID=203 STAMP=616698336
Crosschecked 3 objects
```

Example 4–33 Deleting Datafile Copies

This example deletes the specified datafile copy:

```
RMAN> DELETE NOPROMPT DATAFILECOPY '/disk1/oradata/prod/users01.dbf';
```

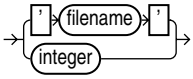
tempfileSpec

Purpose

Use the *tempfileSpec* subclause to specify a tempfile by name or absolute file number.

Syntax

***tempfileSpec*::=**



Semantics

Syntax Element	Description
'filename'	Specifies the datafile by using either the full path or a relative filename. If you specify a relative filename, the filename is qualified in a platform-specific manner by the target database. You can specify an absolute path name, or a path name relative to the Oracle home. Double and single quotes are both valid (although only single quotes are shown in the diagram). Use a question mark (?) to represent the Oracle home and the at sign (@) for the Oracle SID.
integer	Specifies the datafile by absolute file number. Obtain the file number from the V$TEMPFILE view or REPORT SCHEMA output.

Examples

Example 4–34 Specifying a Tempfile by Filename

This example renames tempfile /disk1/oradata/prod/temp01.dbf to /disk2/temp01.dbf, specifying it by filename:

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
RUN
{
  SWITCH TEMPFILE '/disk1/oradata/prod/temp01.dbf'
               TO '/disk2/temp01.dbf';
}
ALTER DATABASE OPEN;
```

Note that the database must be mounted when performing this example.

toDestSpec

Purpose

Use the `toDestSpec` subclause to specify a directory or an Automatic Storage Management disk group for disk backups. If you do not specify a value for the `TO DESTINATION`, then RMAN either creates the backup in the fast recovery area if it is enabled, or in a platform-specific directory (for example, `?/dbs` on UNIX) if a fast recovery area is not enabled. RMAN uses Oracle Managed Filenames when the `TO DESTINATION` parameter is used.

Usage Notes

This subclause can only be used with disk devices and cannot be used with the `FORMAT` option.

If backup optimization is enabled, RMAN only skips backups of files that already have identical backups in the location designated by the `TO DESTINATION` field.

You can specify up to four `TO DESTINATION` strings. RMAN uses the second, third, and fourth values only when `BACKUP COPIES`, `SET BACKUP COPIES`, or `CONFIGURE ... BACKUP COPIES` is in effect. When choosing the format for each backup piece, RMAN uses the first format value for copy 1, the second format value for copy 2, and so on. If the number of format values exceeds the number of copies, then the extra formats are not used. If the number of format values is less than the number of copies, then RMAN reuses the format values, starting with the first one.

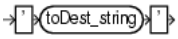
Specify *`todest_string`* in any of the following places, listed in order of precedence:

1. The *`backupSpec`* clause
2. The `BACKUP` command
3. The `ALLOCATE CHANNEL` command
4. The `CONFIGURE CHANNEL` command

If specified in more than one of these places, then RMAN searches for the `TO DESTINATION` parameter in the order shown.

Syntax

***toDestSpec*::=**



Examples

Example 4–35 Specifying an ASM Disk Group

This example copies the database to ASM disk group `DISK1`:

```
BACKUP AS COPY DATABASE TO DESTINATION '+DATAFILE';
```

Example 4–36 Specifying a Destination for Datafile Copies

This example copies two datafiles with tag `LATESTCOPY` to directory `/disk2`:

```
BACKUP AS COPY
 COPY OF DATAFILE 27, 28
 FROM TAG 'LATESTCOPY'  TO DESTINATION '/disk2';
```

Example 4–37 Specifying a Destination for Recovery Area files

This example copies all recovery area files to `/disk2`:

Because `Backup Recovery Area` has backup optimization enabled by default, RMAN only skips backups of files that previously exist on `/disk2` and not ones that reside in other locations.

Note: This subclause, when used in conjunction with the `BACKUP RECOVERY AREA`, allows you to designate a disk channel as a location.

```
BACKUP RECOVERY AREA TO DESTINATION '/disk2';
```

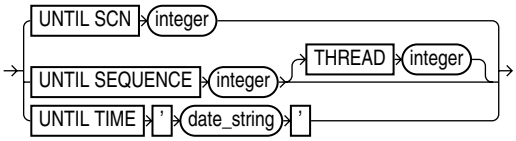
untilClause

Purpose

Use the *untilClause* subclause to specify an upper limit by time, SCN, restore point or log sequence number for various RMAN operations.

Syntax

***untilClause*::=**



Semantics

Syntax Element	Description
`UNTIL SCN` *integer*	Specifies an SCN as an upper, noninclusive limit. RMAN selects only files that can be used to restore or recover up to but not including the specified SCN (see Example 4–38 on page 4-43). For example, `RESTORE DATABASE UNTIL SCN 1000` chooses only backups that could be used to recover to SCN 1000.
`UNTIL SEQUENCE` *integer*	Specifies a redo log sequence number and thread as an upper, noninclusive limit. RMAN selects only files that can be used to restore or recover up to but not including the specified sequence number. For example, `REPORT OBSOLETE UNTIL SEQUENCE 8000` reports only backups that could be used to recover through log sequence 7999.
`THREAD` *integer*	Specifies the number of the redo thread.

Syntax Element	Description

`UNTIL TIME 'date_string'`	Specifies a time as an upper, noninclusive limit (see Example 4–39 on page 4-43).
	RMAN selects only files that can be used to restore and recover up to but not including the specified time. For example, `LIST BACKUP UNTIL TIME 'SYSDATE-7'` lists all backups that could be used to recover to a point one week ago.
	When specifying dates in RMAN commands, the date string must 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('01/30/2007', '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('09/20/06','MM/DD/YY')";`
	Note: The granularity of time-based recovery is dependent on time stamps in the redo log. For example, suppose that you specify the following command:
	`RECOVER DATABASE UNTIL TIME '2007-07-26 17:45:00';`
	If no redo was written with a time stamp of `17:45:00`, then recovery proceeds until it finds the next redo time stamp that is higher. For example, the next redo time stamp may be `17:45:04`. You can check for the nearest time for a specific SCN by querying the `FIRST_TIME` and `FIRST_CHANGE#` columns in `V$LOG_HISTORY TABLE`.

Examples

Example 4–38 Performing Incomplete Recovery to a Specified SCN

This example, which assumes a mounted database, recovers the database up to (but not including) the specified SCN:

```
STARTUP FORCE MOUNT
RUN
{
  SET UNTIL SCN 1418901;  # set to 1418901 to recover database through SCN 1418900
  RESTORE DATABASE;
  RECOVER DATABASE;
}
ALTER DATABASE OPEN RESETLOGS;
```

Example 4–39 Reporting Obsolete Backups

This example assumes that you want to be able to recover to any point within the last week. It considers as obsolete all backups that could be used to recover the database to a point one week ago:

```
REPORT OBSOLETE UNTIL TIME 'SYSDATE-7';
```

5

Recovery Catalog Views

This chapter contains descriptions of recovery catalog views. You can only access these views if you have created a recovery catalog (see CREATE CATALOG). For a summary of the recovery catalog views, refer to "Summary of RMAN Recovery Catalog Views" on page 5-1.

> **Note:** These views are not normalized, but are optimized for RMAN and Enterprise Manager usage. Hence, most catalog views have redundant values that result from joining of several underlying tables.
>
> The views intended for use by Enterprise Manager are generally less useful for direct querying than the other views.

Summary of RMAN Recovery Catalog Views

The following table provides a functional summary of RMAN recovery catalog views.

Table 5–1 Recovery Catalog Views

Recovery Catalog View	Corresponding V$ View	Catalog View Describes ...
RC_ARCHIVED_LOG	V$ARCHIVED_LOG	Archived and unarchived redo log files
RC_BACKUP_ARCHIVELOG_DETAILS	V$BACKUP_ARCHIVELOG_DETAILS	Details about archived redo log backups for Enterprise Manager
RC_BACKUP_ARCHIVELOG_SUMMARY	V$BACKUP_ARCHIVELOG_SUMMARY	Summary of information about archived redo log backups for Enterprise Manager
RC_BACKUP_CONTROLFILE	V$BACKUP_DATAFILE	Control files backed up in backup sets
RC_BACKUP_CONTROLFILE_DETAILS	V$BACKUP_CONTROLFILE_DETAILS	Details about control file backups for Enterprise Manager
RC_BACKUP_CONTROLFILE_SUMMARY	V$BACKUP_CONTROLFILE_SUMMARY	Summary of information about control file backups for Enterprise Manager
RC_BACKUP_COPY_DETAILS	V$BACKUP_COPY_DETAILS	Details about datafile image copy backups for Enterprise Manager
RC_BACKUP_COPY_SUMMARY	V$BACKUP_COPY_SUMMARY	Summary of information about datafile image copy backups for Enterprise Manager
RC_BACKUP_CORRUPTION	V$BACKUP_CORRUPTION	Corrupt block ranges in datafile backups
RC_BACKUP_DATAFILE	V$BACKUP_DATAFILE	Datafiles in backup sets

Table 5–1 (Cont.) Recovery Catalog Views

Recovery Catalog View	Corresponding V$ View	Catalog View Describes ...
RC_BACKUP_DATAFILE_DETAILS	V$BACKUP_DATAFILE_DETAILS	Details about datafile backups for Enterprise Manager
RC_BACKUP_DATAFILE_SUMMARY	V$BACKUP_DATAFILE_SUMMARY	Summary of information about datafile backups for Enterprise Manager
RC_BACKUP_FILES	V$BACKUP_FILES	RMAN backups and copies known to the repository.
RC_BACKUP_PIECE	V$BACKUP_PIECE	Backup pieces
RC_BACKUP_PIECE_DETAILS	V$BACKUP_PIECE_DETAILS	Details about backup pieces for Enterprise Manager
RC_BACKUP_REDOLOG	V$BACKUP_REDOLOG	Archived redo log files in backup sets
RC_BACKUP_SET	V$BACKUP_SET	Backup sets for all incarnations of databases registered in the catalog
RC_BACKUP_SET_DETAILS	V$BACKUP_SET_DETAILS	Details about backup sets for Enterprise Manager
RC_BACKUP_SET_SUMMARY	V$BACKUP_SET_SUMMARY	Summary of information about backup sets for Enterprise Manager
RC_BACKUP_SPFILE	V$BACKUP_SPFILE	Server parameter files in backups
RC_BACKUP_SPFILE_DETAILS	V$BACKUP_SPFILE_DETAILS	Details about server parameter file backups for Enterprise Manager
RC_BACKUP_SPFILE_SUMMARY	V$BACKUP_SPFILE_SUMMARY	Summary of information about server parameter file backups for Enterprise Manager
RC_CHECKPOINT	n/a	Deprecated in favor of RC_RESYNC
RC_CONTROLFILE_COPY	V$DATAFILE_COPY	Control file copies on disk
RC_COPY_CORRUPTION	V$COPY_CORRUPTION	Corrupt block ranges in datafile copies
RC_DATABASE	V$DATABASE	Databases registered in the recovery catalog
RC_DATABASE_BLOCK_CORRUPTION	V$DATABASE_BLOCK_CORRUPTION	Database blocks marked as corrupted in the most recent RMAN backup or copy
RC_DATABASE_INCARNATION	V$DATABASE_INCARNATION	Database incarnations registered in the recovery catalog
RC_DATAFILE	V$DATAFILE	Datafiles registered in the recovery catalog
RC_DATAFILE_COPY	V$DATAFILE_COPY	Datafile copies on disk
RC_LOG_HISTORY	V$LOG_HISTORY	Online redo log history indicating when log switches occurred
RC_OFFLINE_RANGE	V$OFFLINE_RANGE	Offline ranges for datafiles
RC_PROXY_ARCHIVEDLOG	V$PROXY_ARCHIVEDLOG	Archived log backups taken with the proxy copy functionality
RC_PROXY_ARCHIVELOG_DETAILS	V$PROXY_ARCHIVELOG_DETAILS	Details about proxy archived redo log files for Enterprise Manager
RC_PROXY_ARCHIVELOG_SUMMARY	V$PROXY_ARCHIVELOG_SUMMARY	Summary of information about proxy archived redo log files for Enterprise Manager
RC_PROXY_CONTROLFILE	V$PROXY_DATAFILE	Control file backups taken with the proxy copy functionality
RC_PROXY_COPY_DETAILS	V$PROXY_COPY_DETAILS	Details about datafile proxy copies for Enterprise Manager
RC_PROXY_COPY_SUMMARY	V$PROXY_COPY_SUMMARY	Summary of information about datafile proxy copies for Enterprise Manager

Table 5–1 (Cont.) Recovery Catalog Views

Recovery Catalog View	Corresponding V$ View	Catalog View Describes ...
RC_PROXY_DATAFILE	V$PROXY_DATAFILE	Datafile backups that were taken using the proxy copy functionality
RC_REDO_LOG	V$LOG and V$LOGFILE	Online redo logs for all incarnations of the database since the last catalog resynchronization
RC_REDO_THREAD	V$THREAD	All redo threads for all incarnations of the database since the last catalog resynchronization
RC_RESTORE_POINT	V$RESTORE_POINT	All restore points for all incarnations of the database since the last catalog resynchronization
RC_RESYNC	n/a	Recovery catalog resynchronizations
RC_RMAN_BACKUP_JOB_DETAILS	V$RMAN_BACKUP_JOB_DETAILS	Details about backup jobs for Enterprise Manager
RC_RMAN_BACKUP_SUBJOB_DETAILS	V$RMAN_BACKUP_SUBJOB_DETAILS	Details about backup subjobs for Enterprise Manager
RC_RMAN_BACKUP_TYPE	V$RMAN_BACKUP_TYPE	Used internally by Enterprise Manager
RC_RMAN_CONFIGURATION	V$RMAN_CONFIGURATION	RMAN configuration settings
RC_RMAN_OUTPUT	V$RMAN_OUTPUT	Output from RMAN commands for use in Enterprise Manager
RC_RMAN_STATUS	V$RMAN_STATUS	Historical status information about RMAN operations.
RC_SITE	n/a	Databases in a Data Guard environment
RC_STORED_SCRIPT	n/a	Names of scripts stored in the recovery catalog
RC_STORED_SCRIPT_LINE	n/a	Contents of the scripts stored in the recovery catalog
RC_TABLESPACE	V$TABLESPACE	All tablespaces registered in the recovery catalog, all dropped tablespaces, and tablespaces that belong to old incarnations
RC_TEMPFILE	V$TEMPFILE	All tempfiles registered in the recovery catalog
RC_UNUSABLE_BACKUPFILE_DETAILS	V$UNUSABLE_BACKUPFILE_DETAILS	Unusable backup files registered in the recovery catalog

RC_ARCHIVED_LOG

This view contains historical information about archived and unarchived redo log files. It corresponds to the V$ARCHIVED_LOG view in the target database control file.

Oracle inserts an archived redo log record after the online redo log is successfully archived. If a log that has not been archived is cleared, then a record is inserted with the NAME column set to NULL.

If the log is archived multiple times, then the view will contain multiple archived log records with the same THREAD#, SEQUENCE#, and RESETLOGS_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 log can have no record if the record ages out of the control file.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database to which this record belongs. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
AL_KEY	NUMBER	The primary key of the archived redo log in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The archived redo log RECID from V$ARCHIVED_LOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The archived redo log stamp from V$ARCHIVED_LOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
NAME	VARCHAR2(1024)	The filename of the archived redo log.
THREAD#	NUMBER	The number of the redo thread.
SEQUENCE#	NUMBER	The log sequence number.
RESETLOGS_CHANGE#	NUMBER	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS when the record was created.
FIRST_CHANGE#	NUMBER	The first SCN of this redo log.
FIRST_TIME	DATE	The time when Oracle switched into the redo log.
NEXT_CHANGE#	NUMBER	The first SCN of the next redo log in the thread.
NEXT_TIME	DATE	The first time stamp of the next redo log in the thread.
BLOCKS	NUMBER	The size of this archived log in operating system blocks.
BLOCK_SIZE	NUMBER	The size of the block in bytes.
COMPLETION_TIME	DATE	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)	The status of the archived redo log: A (available), U (unavailable), D (deleted), or X (expired).

Column	Data Type	Description
IS_STANDBY	VARCHAR2(3)	The database that archived this log: Y (belongs to a standby database) or N (belongs to the primary database).
DICTIONARY_BEGIN	VARCHAR2(3)	Indicates whether this archived log contains the start of a LogMiner dictionary: YES or NO. If both DICTIONARY_BEGIN and DICTIONARY_END are YES, then this log contains a complete LogMiner dictionary. If DICTIONARY_BEGIN is YES but DICTIONARY_END is NO, this log contains the start of the dictionary, and it continues through each subsequent log of this thread and ends in the log where DICTIONARY_END is YES.
DICTIONARY_END	VARCHAR2(3)	Indicates whether this archived log contains the end of a LogMiner dictionary: YES or NO. See the description of DICTIONARY_BEGIN for an explanation of how to interpret this value.
IS_RECOVERY_DEST_FILE	VARCHAR2(3)	This copy is located in the fast recovery area: YES or NO.
COMPRESSED	VARCHAR2(3)	Internal only
CREATOR	VARCHAR2(7)	Creator of the archived redo log: ■ ARCH - Archiver process ■ FGRD - Foreground process ■ RMAN - Recovery Manager ■ SRMN - RMAN at standby ■ LGWR - Log Writer process
TERMINAL	VARCHAR2(3)	Indicates whether this log was created during terminal recovery of a standby database. Values are YES or NO.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.

RC_BACKUP_ARCHIVELOG_DETAILS

RC_BACKUP_ARCHIVELOG_DETAILS provides detailed information about backups of archived redo log files.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
BTYPE	CHAR(9)	The backup type container, which can be BACKUPSET, IMAGECOPY, or PROXYCOPY.
BTYPE_KEY	NUMBER	A unique identifier for the backup type. For backup sets, it is VS_KEY; for image copies, it is COPY_KEY; for proxy copies it is XAL_KEY.
SESSION_KEY	NUMBER	Unique identifier for this session. Use for joins with RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	The RECID from the control file for the target database which corresponds to this RMAN session.
SESSION_STAMP	NUMBER	The STAMP from the control file for the target database which corresponds to this RMAN session.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
ID1	NUMBER	For archived logs in backup sets, this column contains SET_STAMP. For proxy copy or image copy backups, this column contains the RECID from the control file.
ID2	NUMBER	For archived logs in backup sets, this column contains SET_COUNT. For image copy or proxy copy backups, this column contains STAMP.
THREAD#	NUMBER	Thread number of this archived redo log file.
SEQUENCE#	NUMBER	Sequence number of this archived redo log file.
RESETLOGS_CHANGE#	NUMBER	SCN of OPEN RESETLOGS branch for this archived log.
RESETLOGS_TIME	DATE	Time of OPEN RESETLOGS branch for this archived log.
FIRST_CHANGE#	NUMBER	Starting SCN for this archived log file.
FIRST_TIME	DATE	Time corresponding to starting SCN for this archived redo log file.
NEXT_CHANGE#	NUMBER	Ending SCN for this archived redo log file.
NEXT_TIME	DATE	Time corresponding to the ending SCN for this archived redo log file.
FILESIZE	NUMBER	Size of backed up redo log file, in bytes.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
FILESIZE_DISPLAY	VARCHAR2(4000)	Same value as FILESIZE, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_ARCHIVELOG_SUMMARY

RC_BACKUP_ARCHIVELOG_SUMMARY summarizes the backup of archived redo log files for a single or for multiple RMAN jobs.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
NUM_FILES_BACKED	NUMBER	The number of archived redo log files backed up. If an archived log is included in more than on backup job, then RMAN counts each backup separately. For example, if the view summarizes two RMAN backup jobs, each of which backs up only archived log 1000, then the value of this column is 2 and the value of NUM_DISTINCT_FILES_BACKED is 1.
NUM_DISTINCT_FILES_BACKED	NUMBER	The number of distinct archived redo log files backed up, where files are distinguished by unique log sequence number, thread number, and RESETLOGS branch. For example, if the view summarizes two RMAN backup jobs, each of which backs up only archived log 1000, then the value of this column is 1 and the value of NUM_FILES_BACKED is 2.
MIN_FIRST_CHANGE#	NUMBER	The lowest SCN in the range of archived redo log files that have been backed up.
MAX_NEXT_CHANGE#	NUMBER	The highest SCN in the range of archived redo log files that have been backed up.
MIN_FIRST_TIME	DATE	The earliest point in time covered by the archived redo log files that have been backed up.
MAX_NEXT_TIME	DATE	The latest point in time covered by the archived redo log files that have been backed up.
INPUT_BYTES	NUMBER	The total size in bytes of all archived redo log files that have been backed up.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES, but converted to a user-displayable format, for example *n*M, *n*G, *n*T, *n*P.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_CONTROLFILE

This view lists information about control files in backup sets. Note that the V$BACKUP_DATAFILE view contains both datafile and control file records: a backup datafile record with file number 0 represents the backup control file. In the recovery catalog, the RC_BACKUP_CONTROLFILE view contains only control file records, while the RC_BACKUP_DATAFILE view contains only datafile records.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
BCF_KEY	NUMBER	The primary key of the control file backup in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The RECID value from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The STAMP value from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$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	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS when the record was created.
CHECKPOINT_CHANGE#	NUMBER	The control file checkpoint SCN.
CHECKPOINT_TIME	DATE	The control file checkpoint time.
CREATION_TIME	DATE	The control file creation time.
BLOCK_SIZE	NUMBER	The size of the blocks in bytes.
OLDEST_OFFLINE_RANGE	NUMBER	Internal use only.
STATUS	VARCHAR2(1)	The status of the backup set: A (available), U (unavailable), or D (deleted).
BS_RECID	NUMBER	The control file RECID of the backup set that contains this backup control file.
BS_STAMP	NUMBER	The control file stamp of the backup set that contains this control file.
BS_LEVEL	NUMBER	The incremental level (NULL, 0, 1) of the backup set that contains this backup control file. Although an incremental backup set can contain the control file, it is always contains a complete copy of the control file. There is no such thing as an incremental control file backup.
COMPLETION_TIME	DATE	The date that the control file backup completed.
CONTROLFILE_TYPE	VARCHAR2(1)	The type of control file backup: B (normal backup) or S (standby backup).
BLOCKS	NUMBER	The number of blocks in the file.
AUTOBACKUP_DATE	DATE	The date of the control file autobackup.

Column	Data Type	Description
AUTOBACKUP_SEQUENCE	NUMBER	The sequence of the control file autobackup: 1 - 255.

RC_BACKUP_CONTROLFILE_DETAILS

RC_BACKUP_CONTROLFILE_DETAILS provides detailed information about control file backups that can be restored, including backups in control file image copies, backup sets, and proxy copies.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
BTYPE	CHAR(9)	The type of this control file backup. Possible values are BACKUPSET, IMAGECOPY or PROXYCOPY.
BTYPE_KEY	NUMBER	Unique identifier for the backup type. If BTYPE is BACKUPSET, then this value is the BS_KEY value for the backup set. If BTYPE is IMAGECOPY, this value is the value of COPY_KEY for the copy. If BTYPE is PROXYCOPY, this is the value of XCF_KEY for the proxy copy.
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	The RECID from the control file for the target database which corresponds to this RMAN session.
SESSION_STAMP	NUMBER	The STAMP from the control file for the target database which corresponds to this RMAN session.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
ID1	NUMBER	For backups taken as backup sets, this column contains SET_STAMP. For proxy copy or image copy backups, this column contains the RECID from the control file.
ID2	NUMBER	For backups taken as backup sets, this column contains SET_COUNT. For proxy copy or image copy backups, this column contains the value of STAMP.
CREATION_TIME	DATE	File creation time for the control file that was backed up.
RESETLOGS_CHANGE#	NUMBER	SCN of the RESETLOGS branch where this control file was backed up.
RESETLOGS_TIME	DATE	Time of the RESETLOGS branch where this control file was backed up.
CHECKPOINT_CHANGE#	NUMBER	Most recent checkpoint change SCN for the control file that was backed up.
CHECKPOINT_TIME	DATE	Most recent checkpoint time for the control file that was backed up.
FILESIZE	NUMBER	File size, in bytes, for the output of backing up this control file.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
FILESIZE_DISPLAY	VARCHAR2(4000)	Same value as the FILESIZE column, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_CONTROLFILE_SUMMARY

RC_BACKUP_CONTROLFILE_SUMMARY provides summary information about control file backups that can be restored, including backups in control file image copies, backup sets, and proxy copies.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
NUM_FILES_BACKED	NUMBER	Total number of control file backups.
NUM_DISTINCT_FILES_BACKED	NUMBER	Number of distinct control files backed up.
MIN_CHECKPOINT_CHANGE#	NUMBER	Lowest checkpoint SCN among all backed up control file s.
MAX_CHECKPOINT_CHANGE#	NUMBER	Highest checkpoint SCN among all backed up control files.
MIN_CHECKPOINT_TIME	DATE	Earliest checkpoint time of any control file in the summary.
MAX_CHECKPOINT_TIME	DATE	Latest checkpoint time of any control file in the summary.
INPUT_BYTES	NUMBER	Total size of input files, in bytes.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_COPY_DETAILS

RC_BACKUP_COPY_DETAILS contains detailed information all AVAILABLE control file and datafile copies. Columns SESSION_KEY, SESSION_RECID, SESSION_STAMP, and COPY_KEY uniquely identify an RMAN session and datafile copy. Other columns for this view have the same semantics as in RC_DATAFILE_COPY.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS corresponding to the job that created this copy.
COPY_KEY	NUMBER	Unique identifier for this datafile or control file copy.
FILE#	NUMBER	The absolute file number for the datafile, for datafile copies.
NAME	VARCHAR2(1024)	The filename of the datafile or control file copy.
TAG	VARCHAR2(32)	The tag, if any, for this datafile or control file copy.
CREATION_CHANGE#	NUMBER	The creation SCN of the datafile, for datafiles.
CREATION_TIME	DATE	The creation time of the file.
CHECKPOINT_CHANGE#	NUMBER	The SCN of the most recent datafile checkpoint.
CHECKPOINT_TIME	DATE	The time of the most recent datafile checkpoint.
MARKED_CORRUPT	NUMBER	Number of blocks in the datafile that are marked corrupt, based on RC_DATABASE_BLOCK_CORRUPTION view.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
COMPLETION_TIME	DATE	The completion time for this file copy.
CONTROLFILE_TYPE	VARCHAR2(1)	The type of control file backed up, for control file copies: B (normal copy) or S (standby copy). Otherwise, NULL.
KEEP	VARCHAR2(3)	YES, if this copy has a retention policy different from the value for CONFIGURE RETENTION POLICY. Otherwise, NO.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the data after which this file copy becomes obsolete. If the column is NULL and KEEP_OPTIONS is not null, then this copy never becomes obsolete.
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this datafile copy. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup while the database was open, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the backup retention policy.
IS_RECOVERY_DEST_FILE	VARCHAR2(3)	YES if this copy is located in the fast recovery area. Otherwise, NO.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_COPY_SUMMARY

RC_BACKUP_COPY_SUMMARY contains summary information about all AVAILABLE control file and datafile copies for each database.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
NUM_COPIES	NUMBER	Total number of image copy backups.
NUM_DISTINCT_COPIES	NUMBER	Number of distinct image copy backups.
MIN_CHECKPOINT_CHANGE#	NUMBER	Minimum checkpoint SCN among all image copy backups described in this view.
MAX_CHECKPOINT_CHANGE#	NUMBER	Maximum checkpoint SCN among all image copy backups described in this view.
MIN_CHECKPOINT_TIME	DATE	Earliest checkpoint time among all copies described in this view.
MAX_CHECKPOINT_TIME	DATE	Latest checkpoint time among all copies described in this view.
OUTPUT_BYTES	NUMBER	Sum of sizes of all datafile and control file copies.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_CORRUPTION

This view lists corrupt block ranges in datafile backups, which may be detected when BACKUP VALIDATE is used or the MAXCORRUPT parameter is greater than 0. 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	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	The record identifier from V$BACKUP_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp propagated from V$BACKUP_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$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	The backup piece that contains this corrupt block.
BDF_KEY	NUMBER	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 RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
BDF_RECID	NUMBER	The RECID value from V$BACKUP_DATAFILE.
BDF_STAMP	NUMBER	The STAMP value from V$BACKUP_DATAFILE.
FILE#	NUMBER	The absolute file number for the datafile that contains the corrupt blocks.
CREATION_CHANGE#	NUMBER	The creation SCN of the datafile containing the corrupt blocks.
BLOCK#	NUMBER	The block number of the first corrupted block in this range of corrupted blocks.
BLOCKS	NUMBER	The number of corrupted blocks found beginning with BLOCK#.
CORRUPTION_CHANGE#	NUMBER	For media corrupt blocks, this value is zero. For logically corrupt blocks, this value is the lowest SCN in the blocks in this corrupt range.
MARKED_CORRUPT	VARCHAR2(3)	YES if this corruption was not previously detected by Oracle, or NO if Oracle had already discovered this corrupt block and marked it as corrupt in the database. Note that when a corrupt block is encountered in a backup, and was not already marked corrupt by Oracle, then the backup process does not mark the block as corrupt in the production datafile. Thus, this field may be YES for the same block in more than one backup set.
CORRUPTION_TYPE	VARCHAR2(9)	Same as RC_DATABASE_BLOCK_CORRUPTION.CORRUPTION_TYPE.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
BDF_KEY	NUMBER	The primary key of the datafile backup in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The backup datafile RECID from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The backup datafile stamp from V$BACKUP_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$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	The RECID from V$BACKUP_SET.
BS_STAMP	NUMBER	The STAMP from V$BACKUP_SET.
BACKUP_TYPE	VARCHAR2(1)	The type of the backup: D (full or level 0 incremental) or I (incremental level 1).
INCREMENTAL_LEVEL	NUMBER	The level of the incremental backup: NULL, 0, or 1.
COMPLETION_TIME	DATE	The completion time of the backup.
FILE#	NUMBER	The absolute file number of the datafile. Note that when FILE#=0, the record refers to the control file. See the note following this table for special semantics of other columns when FILE#=0.
CREATION_CHANGE#	NUMBER	The creation SCN of the datafile.
RESETLOGS_CHANGE#	NUMBER	The SCN of the most recent RESETLOGS in the datafile header.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS in the datafile header.
INCREMENTAL_CHANGE#	NUMBER	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	The checkpoint SCN of this datafile in this backup set.
CHECKPOINT_TIME	DATE	The time associated with CHECKPOINT_CHANGE#.
ABSOLUTE_FUZZY_CHANGE#	NUMBER	The absolute fuzzy SCN. See the note following this table for special semantics when FILE#=0.
DATAFILE_BLOCKS	NUMBER	The number of data blocks in the datafile.

Column	Data Type	Description
BLOCKS	NUMBER	The number of data blocks written to the backup. This value is often less than DATAFILE_BLOCKS because for full backups, blocks that have never been used are not included in the backup, and for incremental backups, blocks that have not changed are not included in the backup. This value is never greater than DATAFILE_BLOCKS.
BLOCK_SIZE	NUMBER	The size of the data blocks in bytes.
STATUS	VARCHAR2(1)	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).
BS_LEVEL	NUMBER	The incremental level (NULL, 0, or 1) specified when this backup was created. This value can be different from the INCREMENTAL_LEVEL column because if you run, for example, a level 1 incremental backup, but no previous level 0 backup exists for some files, a level 0 backup is automatically taken for these files. In this case, BS_LEVEL is 1 and INCREMENTAL_LEVEL is 0.
PIECES	NUMBER	The number of backup pieces in the backup set that contains this backup datafile.
BLOCKS_READ	NUMBER	Number of blocks that were scanned while taking this backup. If this was an incremental backup, and change tracking was used to optimize the backup, then the value of this column will be smaller than DATAFILE_BLOCKS. Otherwise, the value of this column will be the same as DATAFILE_BLOCKS. Even when change tracking data is used, the value of this column may be larger than BLOCKS, because the data read by change tracking is further refined during the process of creating an incremental backup.
CREATION_TIME	DATE	Creation timestamp of the datafile.
MARKED_CORRUPT	NUMBER	Number of blocks marked corrupt.
USED_CHANGE_TRACKING	VARCHAR2(3)	Whether change tracking data was used to accelerate this incremental backup (YES) or was not used (NO).
USED_OPTIMIZATION	VARCHAR2(3)	Whether backup optimization was applied (YES) or not (NO).
PCT_NOTREAD	NUMBER	The percentage of the file that was skipped during this backup. For incremental backups, this value indicates the efficiency of the block change tracking file.
FOREIGN_DBID	NUMBER	Foreign DBID of the database from which this datafile was transported. The value is 0 if the file backed up is not a foreign database file.
PLUGGED_READONLY	VARCHAR2(3)	YES if this is a backup of a transported read-only foreign file; otherwise NO.
PLUGIN_CHANGE#	NUMBER	SCN at which the foreign datafile was transported into the database. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_TIME	DATE	The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.
SECTION_SIZE	NUMBER	Specifies the number of blocks in each section of a multisection backup. Value is 0 for whole file backups.

RC_BACKUP_DATAFILE_DETAILS

RC_BACKUP_DATAFILE_DETAILS provides detailed information about available datafile backups for databases registered in the recovery catalog.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
BTYPE	CHAR(9)	The backup type container, which can be BACKUPSET, IMAGECOPY, or PROXYCOPY.
BTYPE_KEY	NUMBER	Unique identifier for the backup type. If BTYPE is BACKUPSET, then this value is the BS_KEY value for the backup set. If BTYPE is IMAGECOPY, then this value is the value of COPY_KEY for the copy.
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
ID1	NUMBER	For backups taken as backup sets, this column contains SET_STAMP. For proxy copy or image copy backups, this column contains the RECID from the control file.
ID2	NUMBER	For backups taken as backup sets, this column contains SET_COUNT. For proxy copy or image copy backups, this column contains STAMP.
FILE#	NUMBER	The number of this datafile.
CREATION_CHANGE#	NUMBER	The checkpoint SCN at the time that this datafile was created.
CREATION_TIME	DATE	The time at which this datafile was created.
RESETLOGS_CHANGE#	NUMBER	The checkpoint SCN of the most recent RESETLOGS operation affecting this datafile.
RESETLOGS_TIME	DATE	The time of the most recent RESETLOGS operation affecting this datafile
INCREMENTAL_LEVEL	NUMBER	For incremental backups, the level of the incremental backup (0 or 1). Otherwise, NULL.
INCREMENTAL_CHANGE#	NUMBER	For incremental backups, the incremental backup SCN. Otherwise, NULL.
CHECKPOINT_CHANGE#	NUMBER	The current checkpoint SCN for the datafile at the time it was backed up.
CHECKPOINT_TIME	DATE	The time corresponding to the current checkpoint SCN for the datafile at the time it was backed up.
MARKED_CORRUPT	NUMBER	Number of datafile blocks marked corrupt.
FILESIZE	NUMBER	The size of the datafile at the time it was backed up.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
TS#	NUMBER	Tablespace number.
TSNAME	VARCHAR2(30)	The name of the tablespace containing this datafile.
FILESIZE_DISPLAY	VARCHAR2(4000)	Same value as FILESIZE, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_DATAFILE_SUMMARY

RC_BACKUP_DATAFILE_SUMMARY provides summary information about available backups of datafiles.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
NUM_FILES_BACKED	NUMBER	Number of datafiles backed up for this value of DB_KEY and DB_NAME.
NUM_DISTINCT_FILES_BACKED	NUMBER	Number of distinct files backed up for this value of DB_KEY and DB_NAME.
NUM_DISTINCT_TS_BACKED	NUMBER	Number of distinct tablespaces backed up for this value of DB_KEY and DB_NAME.
MIN_CHECKPOINT_CHANGE#	NUMBER	Minimum checkpoint of any datafile backed up for this value of DB_KEY and DB_NAME.
MAX_CHECKPOINT_CHANGE#	NUMBER	Maximum checkpoint change number of any datafile backed up for this value of DB_KEY and DB_NAME.
MIN_CHECKPOINT_TIME	DATE	Minimum checkpoint time for any datafile backed up for this value of DB_KEY and DB_NAME.
MAX_CHECKPOINT_TIME	DATE	Maximum checkpoint time for any datafile backed up for this value of DB_KEY and DB_NAME.
INPUT_BYTES	NUMBER	Total input bytes read for all files backed up for this value of DB_KEY and DB_NAME.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by backups for this value of DB_KEY and DB_NAME.
COMPRESSION_RATIO	NUMBER	Compression ratio across all backups.
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_FILES

This view lists backups known to the RMAN repository as reflected in the recovery catalog. This view corresponds to the V$BACKUP_FILES control file view.

Note:

- It is usually more convenient to access this information using the LIST BACKUP and LIST COPY commands from within RMAN.
- You must use DBMS_RCVMAN.SetDatabase to select a database from the recovery catalog schema before you can use this view, even if only one database is registered in the recovery catalog. *Oracle Database Backup and Recovery User's Guide* explains how to perform this task.

Column	Data Type	Description
PKEY	NUMBER	The primary key for the backup.
BACKUP_TYPE	VARCHAR2(32)	The type of the backup: BACKUP SET, COPY, or PROXY COPY.
FILE_TYPE	VARCHAR2(32)	Type of the file backed up: DATAFILE, CONTROLFILE, SPFILE, REDO LOG, ARCHIVED LOG, COPY (for an image copy backup) or PIECE (for a backup piece).
KEEP	VARCHAR2(3)	Whether this backup has KEEP attributes set that override the backup retention policy. Values are YES or NO.
KEEP_UNTIL	DATE	Date after which this backup is considered obsolete.
KEEP_OPTIONS	VARCHAR2(13)	Attributes affecting retention for this backup. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
STATUS	VARCHAR2(16)	Status of the backup. Possible values are: AVAILABLE, UNAVAILABLE, EXPIRED.
FNAME	VARCHAR2(1024)	Filename of this piece, copy, or filename of the file included in this backup set. For example, for a row whose BACKUP_TYPE is BACKUP SET and FILE_TYPE is DATAFILE, FNAME is the name of the datafile in the backup. On the other hand, if BACKUP_TYPE is BACKUP SET and FILE_TYPE is PIECE, then FNAME shows the name of backup piece.
TAG	VARCHAR2(32)	The tag for this backup piece or image copy. This column can only have a value if FILE_TYPE is PIECE or COPY.
MEDIA	VARCHAR2(80)	Media ID for the media on which the backup is stored. This column can only have a value if BACKUP_TYPE is BACKUP SET and FILE_TYPE is PIECE.
RECID	NUMBER	ID of the control file record corresponding to this row.
STAMP	NUMBER	Timestamp of the control file record corresponding to this row.
DEVICE_TYPE	VARCHAR2(255)	Device type on which this backup is stored. This column populated only if FILE_TYPE is PIECE or COPY.
BLOCK_SIZE	NUMBER	Block size for the backup or copy (in bytes).
COMPLETION_TIME	NUMBER	Time when this backup was completed. This column populated only if FILE_TYPE is PIECE or COPY.
COMPRESSED	VARCHAR2(3)	Whether the backup piece represented by this row is compressed. This column populated only if file-type is PIECE (since image copies cannot be compressed, by definition).

Column	Data Type	Description
OBSOLETE	VARCHAR2(3)	Whether this backup piece or copy is obsolete. Possible value: YES, NO. This column populated only if FILE_TYPE is PIECE or COPY.
BYTES	NUMBER	Size of file described by this row. If BACKUP_TYPE is BACKUP SET, this represents the total size of the backup set. If FILE_TYPE is PIECE or COPY, then this represents the size of the individual file. If FILE_TYPE is DATAFILE, ARCHIVED LOG, SPFILE or CONTROL FILE, the value represents how much data was incorporated into the backup set (but note that the corresponding backup set may be smaller, if compression was used in creating the backup set).
BS_KEY	NUMBER	Backup set key. This column populated only if BACKUP_TYPE is BACKUP SET. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
BS_COUNT	NUMBER	Backup set count. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_STAMP	NUMBER	Backup set timestamp. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_TYPE	VARCHAR2(32)	Type of backup set contents (datafiles or archived redo log files). This column populated only if BACKUP_TYPE is BACKUP SET.
BS_INCR_TYPE	VARCHAR(32)	Backup set incremental type (full or not). This column populated only if BACKUP_TYPE is BACKUP SET.
BS_PIECES	NUMBER	Number of pieces in backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_COPIES	NUMBER	Number of copies of this backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_COMPLETION_TIME	DATE	Completion time of the backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_STATUS	VARCHAR2(16)	Status of the backup set. Possible values are AVAILABLE, UNAVAILABLE, EXPIRED, or OTHER. (OTHER means that not all pieces of the backup set have the same status, which can happen if some are AVAILABLE and others UNAVAILABLE.) This column populated only if BACKUP_TYPE is BACKUP SET.
BS_BYTES	NUMBER	Sum of the sizes of all backup pieces in the backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_COMPRESSED	VARCHAR2(3)	Whether the backup pieces of this backup set are compressed. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_TAG	VARCHAR2(1024)	Tag or tags of the backup pieces of this backup set. If pieces have different tags, then BS_TAGS will contain a comma-delimited list of all tags for pieces in the backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
BS_DEVICE_TYPE	VARCHAR2(255)	Type of device on which this backup set is stored. If multiple copies of this backup set exist and are stored on different devices, then this field will contain a comma-delimited list of all device types. For example, for a backup set that is on disk and also backed up on tape, BS_DEVICE_TYPE might contain DISK, SBT_TAPE. This column populated only if BACKUP_TYPE is BACKUP SET.
BP_PIECE#	NUMBER	Number of backup pieces that make up this backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
BP_COPY#	NUMBER	Number of copies of this backup set. This column populated only if BACKUP_TYPE is BACKUP SET.
DF_FILE#	NUMBER	File number of the datafile described by this row. This column populated only if FILE_TYPE is DATAFILE.
DF_TABLESPACE	VARCHAR2(30)	Tablespace name for the datafile described by this row. This column populated only if FILE_TYPE is DATAFILE.
DF_RESETLOGS_CHANGE#	NUMBER	RESETLOGS change of the datafile described by this row. This column populated only if FILE_TYPE is DATAFILE.
DF_CREATION_CHANGE#	NUMBER	Creation change number of the datafile described by this row. This column populated only if FILE_TYPE is CONTROLFILE, DATAFILE or SPFILE.

Column	Data Type	Description
DF_CHECKPOINT_CHANGE#	NUMBER	Checkpoint change number of the datafile described by this row. This column populated only if FILE_TYPE is CONTROLFILE, DATAFILE or SPFILE.
DF_CKP_MOD_TIME	DATE	Checkpoint time of the datafile described by this row. Valid only if FILE_TYPE is CONTROLFILE, DATAFILE or SPFILE.
RL_THREAD#	NUMBER	Redo log thread number of the archived redo log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.
RL_SEQUENCE#	NUMBER	Redo log sequence number of the archived redo log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.
RL_RESETLOGS_CHANGE#	NUMBER	RESETLOGS change number of the archived redo log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.
RL_FIRST_CHANGE#	NUMBER	First change number in the archived redo log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.
RL_FIRST_TIME	DATE	Time of the first change in the archived redo log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.
RL_NEXT_CHANGE#	NUMBER	Change number after the archived log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.
RL_NEXT_TIME	DATE	Time of the first change after the archived log described by this row. Valid only if FILE_TYPE is ARCHIVED LOG.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_ID	NUMBER	The database identifier.
BP_KEY	NUMBER	The primary key for the backup piece in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The backup piece RECID from V$BACKUP_PIECE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The backup piece stamp propagated from V$BACKUP_PIECE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET, RC_BACKUP_CONTROLFILE, RC_BACKUP_DATAFILE, and so on.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$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)	The type of the backup: D (full or level 0 incremental), I (incremental level 1), L (archived redo log).
INCREMENTAL_LEVEL	NUMBER	The level of the incremental backup: NULL, 0, or 1.
PIECE#	NUMBER	The number of the backup piece. The first piece has the value of 1.
COPY#	NUMBER	The copy number of the backup piece.
DEVICE_TYPE	VARCHAR2(255)	The type of backup device, for example, DISK.
HANDLE	VARCHAR2(1024)	The filename of the backup piece.
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 tag for the backup piece. Refer to description in BACKUP for default format for tag names.
START_TIME	DATE	The time when RMAN started to write the backup piece.
COMPLETION_TIME	DATE	The time when the backup piece was completed.
ELAPSED_SECONDS	NUMBER	The duration of the creation of the backup piece.
STATUS	VARCHAR2(1)	The status of the backup piece: A (available), U (unavailable), D (deleted), or X (expired). Note that status D will not appear unless an older recovery catalog is upgraded.
BYTES	NUMBER	The size of the backup piece in bytes.
IS_RECOVERY_DEST_FILE	VARCHAR2(3)	This backup piece is located in the fast recovery area: YES or NO.

Column	Data Type	Description
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS that created this backup piece.
COMPRESSED	VARCHAR2(3)	Indicates whether the backup piece is compressed (YES) or not (NO).
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
ENCRYPTED	VARCHAR2(3)	Indicates whether the backup piece is encrypted (YES) or not (NO).
BACKED_BY_OSB	VARCHAR2(3)	Indicates whether the backup piece is backed up to Oracle Secure Backup (YES) or not (NO).

RC_BACKUP_PIECE_DETAILS

RC_BACKUP_PIECE_DETAILS contains detailed information about all available backup pieces recorded in the recovery catalog. The semantics of most columns are the same as for the RC_BACKUP_PIECE recovery catalog view.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_ID	NUMBER	The DBID of the database incarnation to which this record belongs.
BP_KEY	NUMBER	The primary key for the backup piece in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The backup piece RECID from RC_BACKUP_PIECE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The backup piece stamp propagated from V$BACKUP_PIECE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
SET_STAMP	NUMBER	The SET_STAMP value from V$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	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)	The type of backup. Possible values are D for datafile or control file backups, I for incremental backups, and L for archived log file backups.
INCREMENTAL_LEVEL	NUMBER	For incremental backups, indicates the level of incremental backup. Possible values are NULL (for full backups), 0 or 1.
PIECE#	NUMBER	The number of the backup piece. The first piece has the value of 1.
COPY#	NUMBER	Indicates the copy number for backup pieces created with duplex enabled. 1 if the backup piece is not duplexed.
DEVICE_TYPE	VARCHAR2(255)	Type of the device on which the backup piece resides. Set to DISK for backup sets on disk.
HANDLE	VARCHAR2(1024)	The filename of the backup piece.
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. 0 indicates no media pool.
CONCUR	VARCHAR2(3)	Specifies whether backup media supports concurrent access: YES or NO.

Column	Data Type	Description
TAG	VARCHAR2(32)	The tag associated with this backup piece.
START_TIME	DATE	The time when RMAN started to write the backup piece.
COMPLETION_TIME	DATE	The time when the backup piece was completed.
ELAPSED_SECONDS	NUMBER	The duration of the creation of the backup piece.
STATUS	VARCHAR2(1)	The status of the backup piece: A for backup pieces that are AVAILABLE. (The value is always A, because this view shows only available backup pieces.)
BYTES	NUMBER	The size of the backup piece in bytes.
IS_RECOVERY_DEST_FILE	VARCHAR2(3)	YES if this backup piece is located in the fast recovery area. NO otherwise.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS corresponding to the job that created this backup piece.
COMPRESSED	VARCHAR2(3)	YES if this backup piece is compressed. NO otherwise.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value.You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
ENCRYPTED	VARCHAR2(3)	Indicates whether the backup piece is encrypted (YES) or not (NO).
BACKED_BY_OSB	VARCHAR2(3)	Indicates whether the backup piece is backed up to Oracle Secure Backup (YES) or not (NO).
PIECES_PER_SET	NUMBER	Number of backup pieces in the backup set containing this backup piece.
SIZE_BYTES_DISPLAY	VARCHAR2(4000)	Same value as BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_REDOLOG

This view lists information about archived redo log files 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	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
BRL_KEY	NUMBER	The primary key of the archived redo log in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The record identifier propagated from V$BACKUP_REDOLOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp from V$BACKUP_REDOLOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$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)	The type of the backup: L (archived redo log).
COMPLETION_TIME	DATE	The time when the backup completed.
THREAD#	NUMBER	The thread number of the redo log.
SEQUENCE#	NUMBER	The log sequence number.
RESETLOGS_CHANGE#	NUMBER	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS when the record was created.
FIRST_CHANGE#	NUMBER	The SCN generated when Oracle switched into the redo log.
FIRST_TIME	DATE	The time when Oracle switched into the redo log.
NEXT_CHANGE#	NUMBER	The first SCN of the next redo log in the thread.
NEXT_TIME	DATE	The first time stamp of the next redo log in the thread.
BLOCKS	NUMBER	The number of operating system blocks written to the backup.
BLOCK_SIZE	NUMBER	The number of bytes in each block of this redo log.
STATUS	VARCHAR2(1)	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).
BS_RECID	NUMBER	The RECID value from V$BACKUP_SET.
BS_STAMP	NUMBER	The STAMP value from V$BACKUP_SET. Note that BS_STAMP is different from SET_STAMP. BS_STAMP is the stamp of the backup set record when created in the control file, whereas SET_STAMP joins with SET_COUNT to make a unique identifier.
PIECES	NUMBER	The number of pieces in the backup set.

Column	Data Type	Description
TERMINAL	VARCHAR2(3)	Indicates whether this log was created during terminal recovery of a standby database. Values are YES or NO.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_ID	NUMBER	The unique database identifier.
BS_KEY	NUMBER	The primary key of the backup set in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output. Use this column to join with RC_BACKUP_PIECE.
RECID	NUMBER	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	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	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	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)	The type of the backup: D (full backup or level 0 incremental), I (incremental level 1), L (archived redo log).
INCREMENTAL_LEVEL	NUMBER	The level of the incremental backup: NULL, 0, or 1.
PIECES	NUMBER	The number of backup pieces in the backup set.
START_TIME	DATE	The time when the backup began.
COMPLETION_TIME	DATE	The time when the backup completed.
ELAPSED_SECONDS	NUMBER	The duration of the backup in seconds.
STATUS	VARCHAR2(1)	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).
CONTROLFILE_INCLUDED	VARCHAR2(7)	Possible values are NONE (backup set does not include a backup control file), BACKUP (backup set includes a normal backup control file), and STANDBY (backup set includes a standby control file).
INPUT_FILE_SCAN_ONLY	VARCHAR2(3)	This backup set record was created by the BACKUP VALIDATE command. No real backup set exists. This record is only a placeholder used to keep track of which datafiles were scanned and which corrupt blocks (if any) were found in those files. If COMPATIBLE is set to 11.0.0 or greater, then RMAN does not populate this column.
KEEP	VARCHAR2(3)	Indicates whether this backup set has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which this backup becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, then the backup never becomes obsolete.

Column	Data Type	Description
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this backup set. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
BLOCK_SIZE	NUMBER	Block size of the backup set.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this backup set. Each database in a Data Guard environment has a unique SITE_KEY value.You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
MULTI_SECTION	VARCHAR2(3)	Y if this is a multisection backup; otherwise null.

RC_BACKUP_SET_DETAILS

RC_BACKUP_SET_DETAILS provides details about currently available backup sets, including backup sets created by the use of the BACKUP BACKUPSET command.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
RECID	NUMBER	The backup set RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The backup set RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies this record in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies this record in the target database control file.
BACKUP_TYPE	VARCHAR2(1)	The type of the backup: D (full backup or level 0 incremental), I (incremental level 1), L (archived redo log).
CONTROLFILE_INCLUDED	VARCHAR2(7)	Possible values are NONE (backup set does not include a backup control file), BACKUP (backup set includes a normal backup control file), and STANDBY (backup set includes a standby control file).
INCREMENTAL_LEVEL	NUMBER	The level of the incremental backup: NULL, 0, or 1.
PIECES	NUMBER	The number of backup pieces in the backup set.
START_TIME	DATE	The time when the backup began.
COMPLETION_TIME	DATE	The time when the backup completed
ELAPSED_SECONDS	NUMBER	The duration of the backup in seconds.
BLOCK_SIZE	VARCHAR2	The block size used when creating the backup pieces in the backup set.
KEEP	VARCHAR2(3)	Indicates whether this backup set has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which this backup becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, the backup never becomes obsolete.

Column	Data Type	Description
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this backup set. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
DEVICE_TYPE	VARCHAR2(255)	Device type on which the backup is stored. If the backup set is stored on more than one type of device (for example, if a backup set created on disk and still present on disk has also been backed up to tape using BACKUP BACKUPSET), then this column contains an asterisk (*). Values are DISK or SBT_TAPE.
COMPRESSED	VARCHAR2(3)	YES if RMAN's binary compression was used in creating the backup set. NO, otherwise.
NUM_COPIES	NUMBER	Number of identical copies of this backup set created during the backup, for example if duplexing was used.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
ORIGINAL_INPUT_BYTES	NUMBER	Sum of sizes of all input files backed up for this job.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
STATUS	CHAR(1)	The status of the backup set: always A (all backup pieces available), because this view only reflects available backup sets.
ORIGINAL_INPRATE_BYTES	NUMBER	Number of bytes read each second when the backup set was initially created.
OUTPUT_RATE_BYTES	NUMBER	Number of bytes written each second when the backup set was initially created.
ORIGINAL_INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as ORIGINAL_INPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
ORIGINAL_INPRATE_BYTES_DISPLAY	VARCHAR2(4000)	Same value as ORIGINAL_INPRATE_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_RATE_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_RATE_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
TIME_TAKEN_DISPLAY	VARCHAR2(4000)	Same value as ELAPSED_SECONDS, but converted to a user-displayable format in hours, minutes and seconds.
ENCRYPTED	VARCHAR2(3)	Indicates whether the backup piece is encrypted (YES) or not (NO).
BACKED_BY_OSB	VARCHAR2(3)	Indicates whether the backup piece is backed up to Oracle Secure Backup (YES) or not (NO).

RC_BACKUP_SET_SUMMARY

RC_BACKUP_SET_SUMMARY provides aggregate information about available backup sets for each database registered in the recovery catalog.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
NUM_BACKUPSETS	NUMBER	Total number of available backup sets recorded in the recovery catalog for this database.
OLDEST_BACKUP_TIME	DATE	Creation time of the oldest available backup set recorded in the recovery catalog for this database.
NEWEST_BACKUP_TIME	DATE	Creation time of the newest available backup set recorded in the recovery catalog for this database.
OUTPUT_BYTES	NUMBER	Sum of sizes of all backup pieces for all available backup sets recorded in the recovery catalog for this database.
ORIGINAL_INPUT_BYTES	NUMBER	Sum of sizes of all input files for all available backup sets recorded in the recovery catalog for this database.
ORIGINAL_INPRATE_BYTES	NUMBER	Average input rate in bytes for the creation of all available backup sets recorded in the recovery catalog for this database.
OUTPUT_RATE_BYTES	NUMBER	Average output rate in bytes for the creation of all available backup sets recorded in the recovery catalog for this database.
COMPRESSION_RATIO	NUMBER	Aggregate compression ratio for all available backup sets recorded in the recovery catalog for this database.
ORIGINAL_INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Total size of all input files stored in all available backup sets recorded in the recovery catalog for this database.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
ORIGINAL_INPRATE_BYTES_DISPLAY	VARCHAR2(4000)	Same value as ORIGINAL_INPRATE_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_RATE_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_RATE_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_SPFILE

This view lists information about server parameter files in backup sets.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
BSF_KEY	NUMBER	The primary key of the server parameter file in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The record identifier propagated from V$BACKUP_SPFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp from V$BACKUP_SPFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
BS_KEY	NUMBER	The primary key of the backup set to which this record belongs in the recovery catalog. Use this column to join with RC_BACKUP_SET.
SET_STAMP	NUMBER	The SET_STAMP value from V$BACKUP_SET. SET_STAMP and SET_COUNT form a concatenated key that uniquely identifies the backup set to which this record belongs in the target database control file.
SET_COUNT	NUMBER	The SET_COUNT value from V$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.
MODIFICATION_TIME	DATE	The time when the server parameter file was last modified.
STATUS	VARCHAR2(1)	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).
BS_RECID	NUMBER	The RECID value from V$BACKUP_SET.
BS_STAMP	NUMBER	The STAMP value from V$BACKUP_SET. Note that BS_STAMP is different from SET_STAMP. BS_STAMP is the stamp of the backup set record when created in the control file, whereas SET_STAMP joins with SET_COUNT to make a unique identifier.
COMPLETION_TIME	DATE	The time when the backup set completed.
BYTES	NUMBER	The size of the backup set in bytes.
DB_UNIQUE_NAME	VARCHAR2(30)	The DB_UNIQUE_NAME of the database to which this record belongs.

RC_BACKUP_SPFILE_DETAILS

RC_BACKUP_SPFILE_DETAILS provides detailed information about SPFILE backups for each database registered in the recovery catalog.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
BS_KEY	NUMBER	Unique backup set identifier. Use this column to join with RC_BACKUP_SET or RC_BACKUP_PIECE.
SET_STAMP	NUMBER	Set stamp.
SET_COUNT	NUMBER	Set count.
MODIFICATION_TIME	DATE	Modification time.
FILESIZE	NUMBER	Size in bytes of the SPFILE that was backed up.
FILESIZE_DISPLAY	VARCHAR2(4000)	Same value as FILESIZE, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_BACKUP_SPFILE_SUMMARY

RC_BACKUP_SPFILE_SUMMARY provides summary information about server parameter file backups for databases registered in the recovery catalog.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
NUM_FILES_BACKED	NUMBER	Number of files backed up.
NUM_DISTINCT_FILES_BACKED	NUMBER	Number of distinct files backed up (based on differing modification timestamps).
MIN_MODIFICATION_TIME	DATE	Earliest modification time of any SPFILE backed up for this database.
MAX_MODIFICATION_TIME	DATE	Latest modification time of any SPFILE backed up for this database.
INPUT_BYTES	NUMBER	Total number of bytes in the input files backed up.
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_CHECKPOINT

This view is deprecated. See RC_RESYNC on page 5-62 instead.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
CCF_KEY	NUMBER	The primary key of the control file copy in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The record identifier from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
NAME	VARCHAR2(1024)	The control file copy filename.
TAG	VARCHAR2(32)	The tag of the control file copy. NULL if no tag used.
RESETLOGS_CHANGE#	NUMBER	The SCN of the most recent RESETLOGS when the record was created.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS when the record was created.
CHECKPOINT_CHANGE#	NUMBER	The control file checkpoint SCN.
CHECKPOINT_TIME	DATE	The control file checkpoint time.
CREATION_TIME	DATE	The control file creation time.
BLOCKS	NUMBER	The number of blocks in the control file.
BLOCK_SIZE	NUMBER	The block size in bytes.
MIN_OFFR_RECID	NUMBER	Internal use only.
OLDEST_OFFLINE_RANGE	NUMBER	Internal use only.
COMPLETION_TIME	DATE	The time when the copy was generated.
STATUS	VARCHAR2(1)	The status of the copy: A (available), U (unavailable), X (expired), or D (deleted).
CONTROLFILE_TYPE	VARCHAR2(1)	The type of control file copy: B (normal copy) or S (standby copy).
KEEP	VARCHAR2(3)	Indicates whether this copy has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the COPY command was specified, then this column shows the date after which this file becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, the file never becomes obsolete.
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this control file copy. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
IS_RECOVERY_DEST_FILE	VARCHAR2(3)	This copy is located in the fast recovery area: YES or NO.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS that created this backup piece.

Column	Data Type	Description
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value.You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.

RC_COPY_CORRUPTION

This view lists corrupt block ranges in datafile copies. It corresponds to the V$COPY_CORRUPTION view.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	The record identifier from V$COPY_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp from V$COPY_CORRUPTION. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
CDF_KEY	NUMBER	The primary key of the datafile copy in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output. Use this column to join with RC_DATAFILE_COPY.
COPY_RECID	NUMBER	The RECID from RC_DATAFILE_COPY. This value is propagated from the control file.
COPY_STAMP	NUMBER	The STAMP from RC_DATAFILE_COPY. This value is propagated from the control file.
FILE#	NUMBER	The absolute file number of the datafile.
CREATION_CHANGE#	NUMBER	The creation SCN of this data file. Because file numbers can be reused, FILE# and CREATION_CHANGE# are both required to uniquely identify a specified file over the life of the database.
BLOCK#	NUMBER	The block number of the first corrupted block in the file.
BLOCKS	NUMBER	The number of corrupted blocks found beginning with BLOCK#.
CORRUPTION_CHANGE#	NUMBER	For media corrupt blocks, this value is zero. For logically corrupt blocks, this value is the lowest SCN in the blocks in this corrupt range.
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.
CORRUPTION_TYPE	VARCHAR2(9)	Same as RC_DATABASE_BLOCK_CORRUPTION.CORRUPTION_TYPE.

RC_DATABASE

This view gives information about the databases registered in the recovery catalog. It corresponds to the V$DATABASE view.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the current incarnation. Use this column to join with RC_DATABASE_INCARNATION.
DBID	NUMBER	Unique identifier for the database obtained from V$DATABASE.
NAME	VARCHAR2(8)	The DB_NAME of the database for the current incarnation.
RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS of the current database incarnation.
RESETLOGS_TIME	DATE	The timestamp of the RESETLOGS of the current database incarnation.

RC_DATABASE_BLOCK_CORRUPTION

This view gives information about database blocks that were corrupted after the last backup. It corresponds to the `V$DATABASE_BLOCK_CORRUPTION` view, which is populated in real time with corruption information.

Column	Data Type	Description
`DB_KEY`	`NUMBER`	The primary key for the database. Use this column to join with almost any other catalog view.
`DBINC_KEY`	`NUMBER`	The primary key for the current incarnation. Use this column to join with `RC_DATABASE_INCARNATION`.
`FILE#`	`NUMBER`	The absolute file number of the datafile.
`BLOCK#`	`NUMBER`	The block number of the first corrupted block in this range of corrupted blocks.
`BLOCKS`	`NUMBER`	The number of corrupted blocks found beginning with `BLOCK#`.
`CORRUPTION_CHANGE#`	`NUMBER`	For media corrupt blocks, this value is zero. For logically corrupt blocks, this value is the lowest SCN in the blocks in this corrupt range.
`CORRUPTION_TYPE`	`VARCHAR2(9)`	The type of block corruption in the datafile. Possible values are: ■ `ALL ZERO`. The block header on disk contained only zeros. The block may be valid if it was never filled and if it is in an Oracle7 file. The buffer will be reformatted to the Oracle8 standard for an empty block. ■ `FRACTURED`. The block header looks reasonable, but the front and back of the block are different versions. ■ `CHECKSUM`. The optional check value shows that the block is not self-consistent. It is impossible to determine exactly why the check value fails, but it probably fails because sectors in the middle of the block are from different versions. ■ `CORRUPT`. The block is wrongly identified or is not a data block (for example, the data block address is missing) ■ `LOGICAL`. Specifies the range for logically corrupt blocks. ■ `NOLOGGING`. The block does not have redo log entries (for example, `NOLOGGING` operations on primary database can introduce this type of corruption on a physical standby)

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. Records about the current and immediately previous incarnation are also contained in the V$DATABASE view.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to join with almost any other catalog view.
DBID	NUMBER	Unique identifier for the database.
DBINC_KEY	NUMBER	The primary key for the incarnation.
NAME	VARCHAR2(8)	The DB_NAME for the database at the time of the RESETLOGS. The value is UNKNOWN if you have done at least one RESETLOGS before registering the target database with RMAN, because RMAN does not know the DB_NAME prior to the RESETLOGS.
RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS operation that created this incarnation.
RESETLOGS_TIME	DATE	The time stamp 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.
PRIOR_RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS operation that created the parent of this incarnation.
PRIOR_RESETLOGS_TIME	DATE	The time stamp of the RESETLOGS operation that created the parent of this incarnation.
STATUS	VARCHAR2(8)	CURRENT if this incarnation is the current database incarnation. PARENT if this is a noncurrent incarnation that is a direct ancestor of the current incarnation. ORPHAN if this is a noncurrent incarnation that is not a direct ancestor of the current incarnation.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
TS#	NUMBER	The number of the tablespace to which the datafile belongs. The TS# may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
TABLESPACE_NAME	VARCHAR2(30)	The tablespace name. The name may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
FILE#	NUMBER	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	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 size of the datafile in blocks.
BLOCK_SIZE	NUMBER	The size of the data blocks in bytes.
NAME	VARCHAR2(1024)	The datafile filename.
STOP_CHANGE#	NUMBER	For offline or read-only datafiles, the SCN value such that no changes in the redo stream at an equal or greater SCN apply to this file.
STOP_TIME	DATE	For offline normal or read-only datafiles, the time beyond which there are no changes in the redo stream that apply to this datafile.
READ_ONLY	NUMBER	1 if the file is read-only; otherwise 0.
RFILE#	NUMBER	The relative file number of this datafile within its tablespace.
INCLUDED_IN_DATABASE_BACKUP	VARCHAR2(3)	Indicates whether this tablespace is included in whole database backups: YES or NO. The NO value occurs only if CONFIGURE EXCLUDE was run on the tablespace that owns this datafile.
AUX_NAME	VARCHAR2(1024)	Indicates the auxiliary name for the datafile as set by CONFIGURE AUXNAME.
ENCRYPT_IN_BACKUP	VARCHAR2(3)	YES if this datafile has been configured to be transparently encrypted when backed up; otherwise NULL.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.

Column	Data Type	Description
DB_UNIQUE_NAME	VARCHAR2(512)	The DB_UNIQUE_NAME of the database incarnation to which this record belongs. All databases in a Data Guard environment share the same DBID but different DB_UNIQUE_NAME values. The value in this column is null when the database name is not known for a specific file. For example, rows for databases managed by versions of RMAN before Oracle Database 11*g* are null.
FOREIGN_DBID	NUMBER	Foreign DBID from which this data file came from. The value is 0 if this file is not a foreign database file.
FOREIGN_CREATION_CHANGE#	NUMBER	Creation SCN of a foreign datafile. The value is 0 if this file is not a foreign database file.
FOREIGN_CREATION_TIME	DATE	Creation time of a foreign datafile. The value is 0 if this file is not a foreign database file.
PLUGGED_READONLY	VARCHAR2(3)	YES if this is a transported read-only foreign file; otherwise NO.
PLUGIN_CHANGE#	NUMBER	SCN at which the foreign datafile was transported into the database. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_TIME	DATE	The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.

RC_DATAFILE_COPY

This view lists information about datafile copies on disk. It corresponds to the V$DATAFILE_COPY view.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
CDF_KEY	NUMBER	The primary key of the datafile copy in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The datafile copy record from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The datafile copy stamp from V$DATAFILE_COPY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
NAME	VARCHAR2(1024)	The filename of the datafile copy.
TAG	VARCHAR2(32)	The tag for the datafile copy.
FILE#	NUMBER	The absolute file number for the datafile.
CREATION_CHANGE#	NUMBER	The creation SCN of the datafile.
CREATION_TIME	DATE	Datafile creation timestamp.
RESETLOGS_CHANGE#	NUMBER	The SCN of the most recent RESETLOGS when the datafile was created.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS in the datafile header.
INCREMENTAL_LEVEL	NUMBER	The incremental level of the copy: 0 or NULL.
CHECKPOINT_CHANGE#	NUMBER	The SCN of the most recent datafile checkpoint.
CHECKPOINT_TIME	DATE	The time of the most recent datafile checkpoint.
ABSOLUTE_FUZZY_CHANGE#	NUMBER	The highest SCN in any block of the file, if known. Recovery must proceed to at least this SCN for the file to become not fuzzy.
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 that is associated with the RECOVERY_FUZZY_CHANGE#.
ONLINE_FUZZY	VARCHAR2(3)	YES/NO. If set to YES, this copy was made after an instance failure or OFFLINE IMMEDATE (or is 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, the recovery process 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	The number of blocks in the datafile copy (also the size of the datafile when the copy was made).
BLOCK_SIZE	NUMBER	The size of the blocks in bytes.
COMPLETION_TIME	DATE	The time when the copy completed.
STATUS	VARCHAR2(1)	The status of the copy: A (available), U (unavailable), X (expired), or D (deleted).

Column	Data Type	Description
KEEP	VARCHAR2(3)	Indicates whether this copy has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the COPY command was specified, then this column shows the date after which this datafile copy becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, the copy never becomes obsolete.
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this datafile copy. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
SCANNED	VARCHAR2(3)	Whether RMAN scanned the file (YES or NO). If YES, then this copy was created by a server process that examined every block in the file, for example, by the RMAN COPY or RESTORE command. If NO, then RMAN did not examine every block in the file, as when RMAN inspects a non-RMAN generated image copy or restores by proxy copy. Whenever RMAN creates or restores a datafile copy, it adds rows to the V$DATABASE_BLOCK_CORRUPTION view and RC_DATABASE_BLOCK_CORRUPTION view if it discovers corrupt blocks in the file. If RMAN has scanned the entire file, then the absence of corruption records for this copy means that no corrupt blocks exist in the file. If RMAN did not scan the file, then the absence of corruption records means that corrupt blocks may or may not exist in the file.
IS_RECOVERY_DEST_FILE	VARCHAR2(3)	This datafile copy is located in the fast recovery area: YES or NO.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS that created this backup piece.
MARKED_CORRUPT	NUMBER	The number of blocks that were discovered to be corrupt during the course of this backup. These blocks were reformatted as known-corrupt blocks in the output image copy.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
DB_UNIQUE_NAME	VARCHAR2(512)	The DB_UNIQUE_NAME of the database incarnation to which this record belongs. All databases in a Data Guard environment share the same DBID but different DB_UNIQUE_NAME values. The value in this column is null when the database name is not known for a specific file. For example, rows for databases managed by versions of RMAN before Oracle Database 11*g* are null.
FOREIGN_DBID	NUMBER	Foreign DBID of the database from which this datafile was transported. The value is 0 if the file backed up is not a foreign database file.
PLUGGED_READONLY	VARCHAR2(3)	YES if this is a copy of a transported read-only foreign file; otherwise NO.
PLUGIN_CHANGE#	NUMBER	SCN at which the foreign datafile was transported into the database. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_TIME	DATE	The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.

RC_LOG_HISTORY

This view lists historical information about the online redo logs. RMAN adds a new row during a catalog resynchronization whenever Oracle has switched out of the online redo log. This catalog view corresponds to the V$LOG_HISTORY view.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	The redo log history RECID from V$LOG_HISTORY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The redo log history stamp from V$LOG_HISTORY. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
THREAD#	NUMBER	The thread number of the online redo log.
SEQUENCE#	NUMBER	The log sequence number of the redo log.
FIRST_CHANGE#	NUMBER	The SCN generated when switching into the redo log.
FIRST_TIME	DATE	The time stamp when switching into the redo log.
NEXT_CHANGE#	NUMBER	The first SCN of the next redo log in the thread.
CLEARED	VARCHAR2(3)	YES 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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	The record identifier for the offline range from V$OFFLINE_RANGE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp for the offline range from V$OFFLINE_RANGE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
FILE#	NUMBER	The absolute file number of the datafile.
CREATION_CHANGE#	NUMBER	The SCN at datafile creation.
OFFLINE_CHANGE#	NUMBER	The SCN taken when the datafile was taken offline.
ONLINE_CHANGE#	NUMBER	The online checkpoint SCN.
ONLINE_TIME	DATE	The online checkpoint time.
CF_CREATE_TIME	DATE	The time of control file creation.

RC_PROXY_ARCHIVEDLOG

This view contains descriptions of archived log backups that were taken using the proxy copy functionality. It corresponds to the V$PROXY_ARCHIVEDLOG 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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
XAL_KEY	NUMBER	The proxy copy primary key in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The proxy copy record identifier from V$PROXY_ARCHIVEDLOG. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The proxy copy stamp from V$PROXY_ARCHIVEDLOG. 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.
DEVICE_TYPE	VARCHAR2(255)	The type of media device that stores the proxy copy.
HANDLE	VARCHAR2(1024)	The name or "handle" for the proxy copy. RMAN passes this value to the media manager that created the proxy copy of the archived redo log.
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.
STATUS	VARCHAR2(1)	The status of the backup set: A (available), U (unavailable), X (expired), or D (deleted).
THREAD#	NUMBER	The number of the redo thread.
SEQUENCE#	NUMBER	The log sequence number.
RESETLOGS_CHANGE#	NUMBER	The RESETLOGS SCN of the database incarnation to which this archived log belongs.
RESETLOGS_TIME	DATE	The RESETLOGS time stamp of the database incarnation to which this archived log belongs.
FIRST_CHANGE#	NUMBER	The first SCN of this redo log.
FIRST_TIME	DATE	The time when Oracle switched into the redo log.
NEXT_CHANGE#	NUMBER	The first SCN of the next redo log in the thread.
NEXT_TIME	DATE	The first time stamp of the next redo log in the thread.
BLOCKS	NUMBER	The size of this archived redo log in operating system blocks.
BLOCK_SIZE	NUMBER	The block size for the copy in bytes.
DEVICE_TYPE	VARCHAR2(255)	The type of sequential media device.
START_TIME	DATE	The time when proxy copy was initiated.
COMPLETION_TIME	DATE	The time when the proxy copy was completed.
ELAPSED_SECONDS	NUMBER	The duration of the proxy copy.

Column	Data Type	Description
RSR_KEY	NUMBER	The primary key from the RMAN status record. Use this column to perform a join with RC_RMAN_STATUS.
TERMINAL	VARCHAR2(3)	YES if this record corresponds to a terminal archived redo log, as defined in V$ARCHIVED_LOG.
KEEP	VARCHAR2(3)	Indicates whether this proxy copy has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this proxy copy. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which this proxy copy becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, then the proxy copy never becomes obsolete.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.

RC_PROXY_ARCHIVELOG_DETAILS

RC_PROXY_ARCHIVELOG_DETAILS provides detailed information about proxy copy backups of archived redo log for each database registered in the recovery catalog.

This view shows one record for each database registered in the recovery catalog. Thus, if only one database is registered, then this view shows one row regardless of the number of proxy copies of archived redo log files that have been performed. This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this backup job from RC_RMAN_OUTPUT.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
COPY_KEY	NUMBER	Unique identifier for this proxy copy.
THREAD#	NUMBER	Redo thread number for the archived redo log file that was backed up.
SEQUENCE#	NUMBER	Log sequence number for the archived redo log file that was backed up.
RESETLOGS_CHANGE#	NUMBER	Checkpoint SCN of OPEN RESETLOGS for incarnation of the database of the archived redo log file that was backed up.
RESETLOGS_TIME	DATE	Time corresponding to RESETLOGS_CHANGE#.
HANDLE	VARCHAR2(1024)	A filename for 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 backup is stored.
TAG	VARCHAR2(32)	Tag specified for this backup.
FIRST_CHANGE#	NUMBER	First change SCN included in the archived redo log file.
NEXT_CHANGE#	NUMBER	Next change SCN after this archived redo log file.
FIRST_TIME	DATE	Time corresponding to FIRST_CHANGE#.
NEXT_TIME	DATE	Time corresponding to NEXT_CHANGE#.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
COMPLETION_TIME	DATE	The time at which the job was completed.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_PROXY_ARCHIVELOG_SUMMARY

RC_PROXY_ARCHIVELOG_SUMMARY contains a summary of proxy copy backups of archived redo log files.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
NUM_FILES_BACKED	NUMBER	Total number of archived redo log files backed up.
NUM_DISTINCT_FILES_BACKED	NUMBER	Number of distinct archived redo log files backed up.
MIN_FIRST_CHANGE#	NUMBER	Minimum value for the first SCN of any redo log file in this summary.
MAX_NEXT_CHANGE#	NUMBER	Maximum value for the NEXT_CHANGE# SCN of any redo log file in this summary.
MIN_FIRST_TIME	DATE	Minimum value for the time of the first change in any redo log. Along with MAX_NEXT_TIME, forms the redo range.
MAX_NEXT_TIME	DATE	Maximum value for the time of the next change after any redo logs in this session.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
XCF_KEY	NUMBER	The proxy copy primary key in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The proxy copy record identifier from V$PROXY_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The proxy copy stamp from V$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	The RESETLOGS SCN of the database incarnation to which this datafile belongs.
RESETLOGS_TIME	DATE	The RESETLOGS time stamp of the database incarnation to which this datafile belongs.
CHECKPOINT_CHANGE#	NUMBER	Datafile checkpoint SCN when this copy was made.
CHECKPOINT_TIME	DATE	Datafile checkpoint time when this copy was made.
CREATION_TIME	DATE	The control file creation time.
BLOCK_SIZE	NUMBER	The block size for the copy in bytes.
BLOCKS	NUMBER	The number of blocks in the copy.
MIN_OFFR_RECID	NUMBER	Internal use only.
OLDEST_OFFLINE_RANGE	NUMBER	Internal use only.
DEVICE_TYPE	VARCHAR2(255)	The type of sequential media device.
HANDLE	VARCHAR2(1024)	The name or "handle" for the proxy copy. RMAN passes this value 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	The time when proxy copy was initiated.
COMPLETION_TIME	DATE	The time when the proxy copy was completed.
ELAPSED_SECONDS	NUMBER	The duration of the proxy copy.
STATUS	VARCHAR2(1)	The status of the backup set: A (available), U (unavailable), X (expired), or D (deleted).
KEEP	VARCHAR2(3)	Indicates whether this proxy copy has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.

Column	Data Type	Description
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this control file backup. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which this control file backup becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, the backup never becomes obsolete.
CONTROLFILE_TYPE	VARCHAR2(1)	The type of control file copy: B (normal copy) or S (standby copy).
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS that created this backup piece.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.

RC_PROXY_COPY_DETAILS

RC_PROXY_COPY_DETAILS contains detailed information about proxy copy backups for databases registered in the recovery catalog.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_STAMP, uniquely identifies output for this proxy copy operation from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID, uniquely identifies output for this proxy copy operation from RC_RMAN_OUTPUT.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS corresponding to the job that created this proxy copy.
COPY_KEY	NUMBER	Unique identifier for this proxy copy.
FILE#	NUMBER	The absolute file number of the datafile that is proxy copied.
HANDLE	VARCHAR2(1024)	The proxy copy handle identifies the copy for purposes of restore operations.
COMMENTS	VARCHAR2(255)	A comment that contains further information about the media manager that stores this backup.
MEDIA	VARCHAR2(80)	Identifies the media manager that stores this backup.
MEDIA_POOL	NUMBER	The number of the media pool in which the copy is stored. This is the same value that was entered in the POOL operand of the Recovery Manager BACKUP command.
TAG	VARCHAR2(32)	Tag associated with this proxy copy.
CREATION_CHANGE#	NUMBER	The datafile creation SCN.
CREATION_TIME	DATE	The time corresponding to CREATION_CHANGE#.
CHECKPOINT_CHANGE#	NUMBER	Checkpoint SCN when the proxy copy was made.
CHECKPOINT_TIME	DATE	The time corresponding to CHECKPOINT_CHANGE#.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this proxy copy operation.
COMPLETION_TIME	DATE	Time when the proxy copy was completed.
CONTROLFILE_TYPE	VARCHAR2(1)	Possible values are: B for a normal control file, and S for a standby control file.
KEEP	VARCHAR2(3)	Indicates whether this backup has a retention policy different from the value for CONFIGURE RETENTION POLICY. Possible values are YES and NO.
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause was specified, then this column shows the date after which this backup becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, the backup never becomes obsolete.
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this backup. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

RC_PROXY_COPY_SUMMARY

RC_PROXY_COPY_SUMMARY contains aggregate information about all available proxy copy backups for databases registered in the recovery catalog.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for this database.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
NUM_COPIES	NUMBER	Number of copies created by all proxy copy operations for this database.
NUM_DISTINCT_COPIES	NUMBER	Number of distinct copies created by all proxy copy operations for this database.
MIN_CHECKPOINT_CHANGE#	NUMBER	Minimum checkpoint SCN among any proxy copies for this database.
MAX_CHECKPOINT_CHANGE#	NUMBER	Maximum checkpoint SCN among any proxy copies for this database.
MIN_CHECKPOINT_TIME	DATE	The oldest checkpoint time among any proxy copies for this database.
MAX_CHECKPOINT_TIME	DATE	The most recent checkpoint time among any proxy copies for this database.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output files generated by proxy copies.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
XDF_KEY	NUMBER	The proxy copy primary key in the recovery catalog. If you issue the LIST command while RMAN is connected to the recovery catalog, then this value appears in the KEY column of the output.
RECID	NUMBER	The proxy copy record identifier from V$PROXY_DATAFILE. RECID and STAMP form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The proxy copy stamp from V$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	The absolute file number of the datafile that is proxy copied.
CREATION_CHANGE#	NUMBER	The datafile creation SCN.
CREATION_TIME	DATE	The datafile creation time.
RESETLOGS_CHANGE#	NUMBER	The SCN of the most recent RESETLOGS in the datafile header.
RESETLOGS_TIME	DATE	The time stamp 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	Checkpoint SCN when the copy was made.
CHECKPOINT_TIME	DATE	Checkpoint time when the copy was made.
ABSOLUTE_FUZZY_CHANGE#	NUMBER	The highest SCN in any block of the file, if known. Recovery must proceed to at least this SCN for the file to become not fuzzy.
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 that is associated with the RECOVERY_FUZZY_CHANGE#.
ONLINE_FUZZY	VARCHAR2(3)	YES/NO. If set to YES, this copy was made after an instance failure 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 must 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	Size of the datafile copy in blocks (also the size of the datafile when the copy was made).
BLOCK_SIZE	NUMBER	The block size for the copy in bytes.
DEVICE_TYPE	VARCHAR2(255)	The type of sequential media device.

Column	Data Type	Description
HANDLE	VARCHAR2(1024)	The name or "handle" for the proxy copy. RMAN passes this value 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	The time when proxy copy was initiated.
COMPLETION_TIME	DATE	The time when the proxy copy was completed.
ELAPSED_SECONDS	NUMBER	The duration of the proxy copy.
STATUS	VARCHAR2(1)	The status of the backup set: A (available), U (unavailable), X (expired), or D (deleted).
KEEP	VARCHAR2(3)	Indicates whether this proxy copy has a retention policy different from the value for CONFIGURE RETENTION POLICY (YES or NO).
KEEP_UNTIL	DATE	If the KEEP UNTIL TIME clause of the BACKUP command was specified, then this column shows the date after which this datafile backup becomes obsolete. If the column is NULL and KEEP OPTIONS is not NULL, the backup never becomes obsolete.
KEEP_OPTIONS	VARCHAR2(11)	The KEEP options specified for this backup. Possible values are NOLOGS, BACKUP_LOGS, LOGS, and NULL. NOLOGS indicates a consistent backup made when the database was mounted. BACKUP_LOGS indicates that the backup was made in open mode, so archived log backups must be applied to make it consistent. LOGS indicates a long-term backup made with the LOGS keyword, which is now deprecated. NULL indicates that this backup has no KEEP options and becomes obsolete based on the retention policy.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS corresponding to the job that created this proxy copy.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
FOREIGN_DBID	NUMBER	Foreign DBID of the database from which this datafile was transported. The value is 0 if the file backed up is not a foreign database file.
PLUGGED_READONLY	VARCHAR2(3)	YES if this is a proxy copy of a transported read-only foreign file; otherwise NO.
PLUGIN_CHANGE#	NUMBER	SCN at which the foreign datafile was transported into the database. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_CHANGE#	NUMBER	The SCN of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.
PLUGIN_RESETLOGS_TIME	DATE	The time of the RESETLOGS operation for the incarnation into which this foreign file was transported. The value is 0 if this file is not a foreign database file.

RC_REDO_LOG

This view lists information about the online redo logs for all incarnations of the database since the last catalog resynchronization. This view corresponds to a join of the V$LOG and V$LOGFILE views.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
THREAD#	NUMBER	The number of the redo thread.
GROUP#	NUMBER	The number of the online redo log group.
NAME	VARCHAR2(1024)	The name of the online redo log file.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
BYTES	NUMBER	The size of the file in bytes.
TYPE	VARCHAR2(7)	The type of redo log: ONLINE or STANDBY.

RC_REDO_THREAD

This view lists data about all redo threads for all incarnations of the database since the last catalog resynchronization. This view corresponds to V$THREAD.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
THREAD#	NUMBER	The redo thread number for the database incarnation.
STATUS	VARCHAR2(1)	The status of the redo thread: D (disabled), E (enabled), or O (open).
SEQUENCE#	NUMBER	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 most recent SCN at which this thread was disabled. If the thread is still disabled, then no redo at or beyond this SCN exists for this thread. If the thread is now enabled, then no redo exists between the DISABLE_CHANGE# and the ENABLE_CHANGE# for this thread.
DISABLE_TIME	DATE	The most recent time at which this thread was disabled.

RC_RESTORE_POINT

This view lists all restore points for all incarnations of the database since the last catalog resynchronization. This view corresponds to V$RESTORE_POINT.

Column	Data Type	Description
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
RECID	NUMBER	The recid of the corresponding row in the control file.
STAMP	NUMBER	The timestamp of the row in the control file. (Because control file records are reused, you must combine the timestamp and recid to get a value unique across all records in RC_RMAN_STATUS.)
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this restore point. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
NAME	VARCHAR2(128)	The name of the restore point.
RESTORE_POINT_TIME	DATE	The database time associated with the restore point SCN.
CREATION_TIME	DATE	The date when the restore point was created.
SCN	NUMBER	The database SCN when the restore point was created
LONG_TERM	VARCHAR2(3)	YES if the restore point is associated with a backup created with the KEEP option specified; otherwise NO.
PRESERVED	VARCHAR2(3)	YES if the restore point must be explicitly deleted; otherwise NO.
GUARANTEE_FLASHBACK_DATABASE	VARCHAR2(3)	YES if the flashback logs must be retained to guarantee a flashback to this point; otherwise NO.

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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RESYNC_KEY	NUMBER	The primary key for the resynchronization.
CONTROLFILE_CHANGE#	NUMBER	The control file checkpoint SCN from which the catalog was resynchronized.
CONTROLFILE_TIME	DATE	The control file checkpoint time stamp from which the catalog was resynchronized.
CONTROLFILE_SEQUENCE#	NUMBER	The control file sequence number.
CONTROLFILE_VERSION	DATE	The creation time for the version of the control file from which the catalog was resynchronized.
RESYNC_TYPE	VARCHAR2(7)	The type of resynchronization: FULL or PARTIAL.
DB_STATUS	VARCHAR2(7)	The status of the target database: OPEN or MOUNTED.
RESYNC_TIME	DATE	The time of the resynchronization.

RC_RMAN_BACKUP_JOB_DETAILS

RC_RMAN_BACKUP_JOB_DETAILS provides detailed information on RMAN backup jobs. An RMAN job is the set of commands executed within an RMAN session. An RMAN backup job is the set of BACKUP commands executed in one RMAN job. For example, a BACKUP DATABASE and BACKUP ARCHIVELOG ALL command executed in the same RMAN job make up a single RMAN backup job.

This view contains one row for each RMAN session, even if multiple BACKUP commands are executed in the same session. The SESSION_KEY column is the unique key for the RMAN session in which the backup job occurred. Details for operations performed during an RMAN session are available in the RC_RMAN_BACKUP_SUBJOB_DETAILS view.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for this database.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
SESSION_KEY	NUMBER	Identifier for the RMAN session. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_KEY and SESSION_STAMP, used to uniquely identify job's output from RC_RMAN_OUTPUT
SESSION_STAMP	NUMBER	Together with SESSION_RECID and SESSION_KEY, used to uniquely identify job's output from RC_RMAN_OUTPUT
COMMAND_ID	VARCHAR2(33)	Either a user-specified value set using SET COMMAND ID, or an RMAN-generated unique command id.
START_TIME	DATE	Start time of the first backup command in the RMAN job.
END_TIME	DATE	End time of the last backup command in the RMAN job.
INPUT_BYTES	NUMBER	Sum of sizes of all input files backed up during this RMAN job.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output backup pieces generated by this RMAN job.
STATUS_WEIGHT	NUMBER	Used internally by Enterprise Manager.
OPTIMIZED_WEIGHT	NUMBER	Used internally by Enterprise Manager.
INPUT_TYPE_WEIGHT	NUMBER	Used internally by Enterprise Manager.
OUTPUT_DEVICE_TYPE	VARCHAR2(17)	DISK, SBT_TAPE, or *. An asterisk indicates a backup job that wrote its output to more than one device type.
AUTOBACKUP_COUNT	NUMBER	Number of autobackups performed by this RMAN job.
BACKED_BY_OSB	VARCHAR2(3)	Indicates whether the backup piece is backed up to Oracle Secure Backup (YES) or not (NO).
AUTOBACKUP_DONE	VARCHAR2(3)	YES or NO, depending upon whether a control file autobackup was done as part of this backup job.
STATUS	VARCHAR2(23)	One of the following values: RUNNING, RUNNING WITH WARNINGS, RUNNING WITH ERRORS, COMPLETED, COMPLETED WITH WARNINGS, COMPLETED WITH ERRORS, FAILED. Note that a failed job does not entail that no backup sets were created. It is possible that the job failed after RMAN created some backup sets. Thus, the RC_BACKUP_SET_DETAILS view may contain rows describing backup sets created successfully by the backup job. You can join this view with RC_BACKUP_SET_DETAILS to obtain more information about the failed backup job.

Column	Data Type	Description
INPUT_TYPE	VARCHAR2(13)	Contains a value indicating the type of input for this backup. For possible values, see the RC_RMAN_BACKUP_TYPE view. When a job includes backups corresponding to more than one of these values, multiple rows will appear in the view, corresponding to the different INPUT_TYPE values for each type of input, with corresponding values for the INPUT_BYTES, OUTPUT_BYTES, INPUT_BYTES_DISPLAY, OUTPUT_BYTES_DISPLAY fields.
OPTIMIZED	VARCHAR2(3)	YES or NO, depending on whether backup optimization was applied.
ELAPSED_SECONDS	NUMBER	Number of seconds elapsed during execution of this backup job.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
INPUT_BYTES_PER_SEC	NUMBER	Input read rate, in bytes/second.
OUTPUT_BYTES_PER_SEC	NUMBER	Output write rate, in bytes/second.
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.
INPUT_BYTES_PER_SEC_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES_PER_SEC, but converted to a user-displayable format, for example, 798.01M or 5.25G.
OUTPUT_BYTES_PER_SEC_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES_PER_SEC, but converted to a user-displayable format, for example, 798.01M or 5.25G.
TIME_TAKEN_DISPLAY	VARCHAR2(4000)	Total time for the RMAN job, converted to a user-displayable format, such as *hh*:*mm*:*ss*.

RC_RMAN_BACKUP_SUBJOB_DETAILS

RC_RMAN_BACKUP_SUBJOB_DETAILS provides details for groups of similar operations within an RMAN session.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for this database.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
SESSION_RECID	NUMBER	Together with SESSION_KEY and SESSION_STAMP, used to uniquely identify the job output from RC_RMAN_OUTPUT.
SESSION_STAMP	NUMBER	Together with SESSION_RECID and SESSION_KEY, used to uniquely identify the job output from RC_RMAN_OUTPUT.
OPERATION	VARCHAR2(33)	The possible values are BACKUP, ROLLFORWARD, VALIDATE, or BACKUPSET. For each operation type in a session, the view contains one row with each value representing all operations of that type during the session.
COMMAND_ID	VARCHAR2(33)	Either a user-specified value set using SET COMMAND ID, or an RMAN-generated unique command id.
START_TIME	DATE	Start time of the first backup command in the job
END_TIME	DATE	End time of the last backup job in the job.
INPUT_BYTES	NUMBER	Sum of sizes of all input files backed up during this job.
OUTPUT_BYTES	NUMBER	Sum of sizes of all output pieces generated by this job.
STATUS_WEIGHT	NUMBER	Used internally by Enterprise Manager.
OPTIMIZED_WEIGHT	NUMBER	Used internally by Enterprise Manager.
INPUT_TYPE_WEIGHT	NUMBER	Used internally by Enterprise Manager.
OUTPUT_DEVICE_TYPE	VARCHAR2(17)	Type of output device: DISK, SBT_TAPE, or *. An asterisk (*) indicates a backup job that wrote its output to more than one device type.
BACKED_BY_OSB	VARCHAR2(3)	Indicates whether the backup piece is backed up to Oracle Secure Backup (YES) or not (NO).
AUTOBACKUP_DONE	VARCHAR2(3)	'Whether a control file autobackup was done as part of this job (YES) or not done (NO).
STATUS	VARCHAR2(23)	One of the following values: RUNNING, RUNNING WITH WARNINGS, RUNNING WITH ERRORS, COMPLETED, COMPLETED WITH WARNINGS, COMPLETED WITH ERRORS, FAILED.
INPUT_TYPE	VARCHAR2(13)	Contains one of the following values: DATABASE FULL, RECOVERY AREA, DATABASE INCR, DATAFILE FULL, DATAFILE INCR, ARCHIVELOG, CONTROLFILE, SPFILE. When a subjob includes backups corresponding to more than one of these values, multiple rows will appear in the view, corresponding to the different INPUT_TYPE values for each type of input, with corresponding values for the INPUT_BYTES, OUTPUT_BYTES, INPUT_BYTES_DISPLAY, OUTPUT_BYTES_DISPLAY fields.
OPTIMIZED	VARCHAR2(3)	If the value of the OPERATION column is BACKUP, then OPTIMIZED is YES or NO, depending on whether backup optimization was applied.
AUTOBACKUP_COUNT	NUMBER	Number of autobackups performed by this job.
COMPRESSION_RATIO	NUMBER	Compression ratio for this backup.
INPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as INPUT_BYTES, but converted to a user-displayable format, for example, 798.01M or 5.25G.

Column	Data Type	Description
OUTPUT_BYTES_DISPLAY	VARCHAR2(4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable for example, 798.01M or 5.25G.

RC_RMAN_BACKUP_TYPE

This view is used internally by Enterprise Manager.

It contains information used in filtering the other Enterprise Manager views when generating reports on specific backup types.

Column	Data Type	Description
WEIGHT	NUMBER	Used internally by Enterprise Manager to set precedence order of different backup types in reports.
INPUT_TYPE	VARCHAR2(13)	Used internally by Enterprise Manager to represent possible filters used in creating various reporting screens.

RC_RMAN_CONFIGURATION

This view lists information about RMAN persistent configuration settings. It corresponds to the V$RMAN_CONFIGURATION view.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database corresponding to this configuration. Use this column to join with almost any other catalog view.
CONF#	NUMBER	A unique key identifying this configuration record within the target database that owns it.
NAME	VARCHAR2(65)	The type of configuration. All options of CONFIGURE command are valid types except: CONFIGURE EXCLUDE, (described in RC_TABLESPACE), CONFIGURE AUXNAME (described in RC_DATAFILE), and CONFIGURE SNAPSHOT CONTROLFILE (stored only in control file).
VALUE	VARCHAR2(1025)	The CONFIGURE command setting. For example: RETENTION POLICY TO RECOVERY WINDOW OF 1 DAYS.
DB_UNIQUE_NAME	VARCHAR2(512)	The DB_UNIQUE_NAME of the database incarnation to which this record belongs. All databases in a Data Guard environment share the same DBID but different DB_UNIQUE_NAME values. The value in this column is null when the database name is not known for a specific file. For example, rows for databases managed by versions of RMAN before Oracle Database 11*g* are null.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this configuration. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.

RC_RMAN_OUTPUT

RC_RMAN_OUTPUT corresponds to the control file view V$RMAN_OUTPUT.

This view is primarily for internal use by Enterprise Manager.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS corresponding to the job that created this output.
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
RECID	NUMBER	Contains the value displayed in V$RMAN_OUTPUT.RECID for this database.
STAMP	NUMBER	Stamp (used for ordering) of when the row for this line out output was added.
OUTPUT	VARCHAR2(130)	RMAN output text.

RC_RMAN_STATUS

This view contains information about the history of RMAN operations on all databases associated with this recovery catalog. It contains essentially the same information as V$RMAN_STATUS, except that it does not contain information about current sessions.

All RMAN operations such as backups, restores, deletion of backups, and so on are logged in this table. The table is organized to show the status of each RMAN session (the invocation of an RMAN client, including all actions taken until the RMAN client exits), operations executed during the session, and recursive operations.

RC_RMAN_STATUS also contains the RSR_KEY, PARENT_KEY and SESSION_KEY columns, which do not appear in V$RMAN_STATUS.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the database incarnation.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
RECID	NUMBER	The recid of the corresponding row in the control file.
STAMP	NUMBER	The timestamp of the row in the control file. (Because control file records are reused, you must combine the timestamp and recid to get a value unique across all records in RC_RMAN_STATUS.)
RSR_KEY	NUMBER	Unique key for this row.
PARENT_KEY	NUMBER	The value of RSR_KEY for the parent row of this row.
SESSION_KEY	NUMBER	The value of RSR_KEY for the session row associated with this row. Use in joins with RC_RMAN_BACKUP_JOB_DETAILS.
ROW_TYPE	VARCHAR2(33)	This is the type of operation represented by this row. Possible values are: ■ SESSION for rows at level 0 ■ COMMAND for rows at level 1 ■ RECURSIVE OPERATION for rows at level >1
ROW_LEVEL	NUMBER	The level of this row. ■ If 0 then this is a session row, that is, ROW_TYPE=SESSION and the row represents an invocation of the RMAN client. ■ If 1 then this row represents a command entered in the RMAN client and executed. ROW_TYPE=COMMAND for rows at level 1. ■ If >1 then this row represents a recursive operation, which is a suboperation of an RMAN command such as a control file autobackup performed in conjunction with a database backup. ROW_TYPE=RECURSIVE OPERATION for rows at levels >1.
OPERATION	VARCHAR2(33)	The name of the operation presented by this row. For SESSION operations, this column is set to RMAN. For COMMAND operations, it describes the command executed, such as BACKUP, RESTORE, CONFIGURE, REPORT and so on.

Column	Data Type	Description
STATUS	VARCHAR2(33)	The status of the operation described by this row. Possible values are: ▪ COMPLETED - Job completed successfully - no warnings or errors during execution. ▪ COMPLETED WITH WARNINGS- Job completed successfully, but there were some warning messages during execution. For example, a warning message like the following: RMAN-05019: WARNING: no channel of required type allocated to recover copy of datafile1 ▪ COMPLETED WITH ERRORS - Job completed successfully, but there were some errors which were overcome using failover features. For example, RESTORE FAILOVER or BACKUP FAILOVER was used to schedule the job on another channel. ▪ FAILED - Job failed. ▪ RUNNING - Job is executing and there are no errors or warnings during execution so far. ▪ RUNNING WITH ERRORS - Job is executing with error messages. ▪ RUNNING WITH WARNINGS - Job is executing with warning messages.
COMMAND_ID	VARCHAR2(33)	The user-specified ID of the operation. The user can change this using the SET COMMAND ID syntax in RMAN. By default, the command ID is set to the time at which RMAN is invoked, in ISO standard format.
MBYTES_PROCESSED	NUMBER	If the operation represented by this row performed some data transfer (such as backing up or restoring data), then this column contains the number of megabytes processed in the operation. Otherwise, this row contains NULL.
START_TIME	DATE	The start time for the operation represented by this row.
END_TIME	DATE	The end time for the operation represented by this row.
JOB_KEY	NUMBER	The key of the RMAN session. Identical to SESSION_KEY.
INPUT_BYTES	NUMBER	Number of input bytes read.
OUTPUT_BYTES	NUMBER	Number of input bytes written.
OPTIMIZED	VARCHAR2(3)	YES if backup optimization was applied during the backup job. Otherwise, NO.
OBJECT_TYPE	VARCHAR2(80)	Contains one of the following values: DATABASE FULL, RECOVERY AREA, DATABASE INCR, DATAFILE FULL, DATAFILE INCR, ARCHIVELOG, CONTROLFILE, SPFILE.
SESSION_RECID	NUMBER	If ROW_TYPE=SESSION, that is, this row has no parents and represents an RMAN session, then this column contains NULL. Otherwise, it contains the recid of the row representing the session associated with this row.
SESSION_STAMP	NUMBER	If ROW_TYPE=SESSION, that is, this row has no parents and represents an RMAN session, then this column contains NULL. Otherwise, it contains the timestamp of the row representing the session associated with this row.
OUTPUT_DEVICE_TYPE	VARCHAR2(17)	The type of output device: DISK, SBT_TAPE, or *. An asterisk (*) indicates that output was written to more than one device type.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with the RMAN status information. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
OSB_ALLOCATED	VARCHAR2(3)	YES if this session allocated an SBT channel for Oracle Secure Backup; otherwise, NO.

RC_SITE

This view lists information about all databases in a Data Guard environment that are known to the recovery catalog. You can use this view to obtain the DB_UNIQUE_NAME value for views which do not have this column.

Column	Data Type	Description
SITE_KEY	NUMBER	The unique key of this database. You can join the RC_SITE.SITE_KEY column with the RC_SITE column of other views to determine which DB_UNIQUE_NAME is associated with a backup.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
DATABASE_ROLE	VARCHAR2(7)	The role of the database in the Data Guard environment.
CF_CREATE_TIME	DATE	The creation date of the control file.
DB_UNIQUE_NAME	VARCHAR2(30)	The DB_UNIQUE_NAME of the database. All databases in a Data Guard environment share the same DBID but different DB_UNIQUE_NAME values.

RC_STORED_SCRIPT

This view lists information about scripts stored in the recovery catalog. The view contains one row for each stored script. Note that RMAN's commands for script management such as `LIST SCRIPT NAMES` and `LIST SCRIPT` provide more convenient ways of viewing this information.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the database that owns this stored script. Use this column to join with almost any other catalog view.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
SCRIPT_NAME	VARCHAR2(100)	The name of the stored script.
SCRIPT_COMMENT	VARCHAR2(255)	The comment that the user entered while creating this script. NULL if no comment was entered.

RC_STORED_SCRIPT_LINE

This view lists information about individual lines of stored scripts in the recovery catalog. The view contains one row for each line of each stored script.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the database that owns this stored script. Use this column to join with almost any other catalog view.
SCRIPT_NAME	VARCHAR2(100)	The name of the stored script.
LINE	NUMBER	The number of the line in the stored script. Each line of a stored script is uniquely identified by SCRIPT_NAME and LINE.
TEXT	VARCHAR2(1024)	The text of the line of the stored 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	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
TS#	NUMBER	The tablespace ID 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)	The tablespace name. The name may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
CREATION_CHANGE#	NUMBER	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.
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.
INCLUDED_IN_DATABASE_BACKUP	VARCHAR2(3)	Indicates whether this tablespace is included in whole database backups: YES or NO. The YES value occurs only if CONFIGURE EXCLUDE was run on the tablespace that owns this datafile.
BIGFILE	VARCHAR2(3)	Indicates whether this tablespace is a tablespace created with the BIGFILE option.
TEMPORARY	VARCHAR2(3)	YES if tablespace is a locally managed temporary tablespace. Otherwise, NO.
ENCRYPT_IN_BACKUP	VARCHAR2(3)	Possible values are: ■ ON. Encryption is turned ON at tablespace level ■ OFF. Encryption is turned OFF at tablespace level ■ NULL. Encryption is neither explicitly turned on or off at tablespace level (default or when CLEARED)

RC_TEMPFILE

This view lists information about all tempfiles registered in the recovery catalog. It corresponds to the V$TEMPFILE view. A tempfile is shown as dropped if its tablespace is dropped.

Column	Data Type	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
TS#	NUMBER	The tablespace ID 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)	The tablespace name. The name may exist multiple times in the same incarnation if the tablespace is dropped and re-created.
FILE#	NUMBER	The absolute file number of the tempfile. The same tempfile number may exist in the same incarnation of the tempfile is dropped and re-created.
CREATION_CHANGE#	NUMBER	The SCN when the tempfile is created.
CREATION_TIME	DATE	The time when the tempfile is created.
DROP_CHANGE#	NUMBER	The SCN recorded when the tempfile was dropped. If a new tempfile with the same FILE# is discovered, then the DROP_CHANGE# is set to CREATION_CHANGE# for the tempfile; otherwise, the value is set to RC_CHECKPOINT.CKP_SCN.
DROP_TIME	DATE	The time when the tempfile was dropped. If a new tempfile with the same FILE# is discovered, then the DROP_TIME is set to CREATION_TIME for the tempfile; otherwise the value is RC_CHECKPOINT.CKP_TIME.
BYTES	NUMBER	The size of the tempfile in bytes.
BLOCKS	NUMBER	Size of the file in blocks.
BLOCK_SIZE	NUMBER	The block size of the tempfile in bytes.
NAME	VARCHAR2(1024)	The tempfile name.
RFILE#	NUMBER	The relative file number of this tempfile within the tablespace.
AUTOEXTEND	VARCHAR2(3)	ON if the tempfile is autoextensible. Otherwise OFF.
MAXSIZE	NUMBER	Maximum file size in blocks to which the file can be extended. Valid only when AUTOEXTEND is ON. Always 0 when AUTOEXTEND is OFF.
NEXTSIZE	NUMBER	Amount of incremental size for file extensible in blocks. Valid only when AUTOEXTEND is ON. Always 0 when AUTOEXTEND is OFF.
BIGFILE	VARCHAR2(3)	YES if the tablespace is a bigfile tablespace; otherwise, NO.
SITE_KEY	NUMBER	Primary key of the Data Guard database associated with this file. Each database in a Data Guard environment has a unique SITE_KEY value. You can use SITE_KEY in a join with the RC_SITE view to obtain the DB_UNIQUE_NAME of the database.
TABLESPACE_CREATION_CHANGE#	NUMBER	SCN at which this tablespace was created.
TABLESPACE_CREATION_TIME	DATE	Timestamp at which this tablespace was created.
TABLESPACE_DROP_CHANGE#	NUMBER	SCN at which this tablespace was dropped.
TABLESPACE_DROP_TIME	DATE	Timestamp at which the tablespace was dropped.

Column	Data Type	Description
DB_UNIQUE_NAME	VARCHAR2(512)	The DB_UNIQUE_NAME of the database incarnation to which this record belongs. All databases in a Data Guard environment share the same DBID but different DB_UNIQUE_NAME values. The value in this column is null when the database name is not known for a specific file. For example, rows for databases managed by versions of RMAN before Oracle Database 11*g* are null.

RC_UNUSABLE_BACKUPFILE_DETAILS

This view lists all backup files (backup pieces, proxy copies or image copies) that are marked UNAVAILABLE or EXPIRED. You can select one of the rows and, using BTYPE_KEY or FILETYPE_KEY, change the status of a backup set or specific file to AVAILABLE.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Data Type	Description
DB_NAME	VARCHAR2(8)	The DB_NAME of the database incarnation to which this record belongs.
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to join with almost any other catalog view.
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT and RC_RMAN_BACKUP_JOB_DETAILS.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS corresponding to the job that created this file.
BTYPE	CHAR(9)	The backup type container, which can be BACKUPSET, IMAGECOPY, or PROXYCOPY.
BTYPE_KEY	NUMBER	Unique identifier for the backup type. It is BS_KEY/COPY_KEY.
ID1	NUMBER	For backups taken as backup sets, this column contains SET_STAMP. For proxy copy or image copy backups, this column contains the RECID from the control file.
ID2	NUMBER	For backups taken as backup sets, ID2 contains SET_COUNT. For image copy and proxy copy backups ID2 contains STAMP.
FILETYPE	VARCHAR2(15)	The type of this backup file. Possible values are BACKUPPIECE, COPY, or PROXYCOPY.
FILETYPE_KEY	NUMBER	Backup piece key if the file is a backup piece, other wise COPY_KEY. This key can be used to directly change the status of the file to available.
STATUS	VARCHAR2(1)	Can be either U (for unavailable backups) or X (for expired backups).
FILESIZE	NUMBER	Size in bytes of the unusable backup file.
DEVICE_TYPE	VARCHAR2(255)	Device type storing this unusable backup file. Possible values are DISK and SBT_TAPE.
FILENAME	VARCHAR2(1024)	File name.
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.

A

Deprecated RMAN Syntax

This appendix describes Recovery Manager syntax that is deprecated and describes preferred syntax if any exists.

Deprecated RMAN syntax continues to be supported in subsequent releases for backward compatibility. For example, the `SET AUXNAME` command replaced the `SET CLONENAME` command in Oracle8*i*, and the `CONFIGURE AUXNAME` command replaced the `SET AUXNAME` command in Oracle9*i*. However, you can continue to run both `SET CLONENAME` and `SET AUXNAME` in all subsequent RMAN releases.

Table A–1 Deprecated RMAN Syntax

Deprecated in Release	Deprecated Syntax	Preferred Current Syntax
11.2.0	CONFIGURE or SET `COMPRESSION ALGORITHM 'ZLIB'`	CONFIGURE or SET `COMPRESSION ALGORITHM 'MEDIUM'`
11.2.0	CONFIGURE or SET `COMPRESSION ALGORITHM 'BZIP2'`	CONFIGURE or SET `COMPRESSION ALGORITHM 'BASIC'`
11.2.0	`V$FLASH_RECOVERY_AREA_USAGE`	`V$RECOVERY_AREA_USAGE`
11.1.0	`CONVERT ON TARGET PLATFORM`	`CONVERT ON DESTINATION PLATFORM`
11.1.0	`UNTIL RESTORE POINT`	`TO RESTORE POINT`
11.1.0	`BACKUP ... AS STANDBY`	n/a
11.1.0	`... KEEP [LOGS | NOLOGS]`	`... KEEP`
11.1.0	`BLOCKRECOVER`	`RECOVER`
10.0.1	`BACKUP ... INCREMENTAL LEVEL [2,3,4]`	Levels *other than* `0` and `1` are deprecated.
10.0.1	`BACKUP ... PARMS`	`CONFIGURE CHANNEL ... PARMS`
10.0.1	`COPY`	`BACKUP AS COPY`
10.0.1	`CREATE CATALOG TABLESPACE`	`CREATE CATALOG`
10.0.1	`LIST ... BY BACKUP [SUMMARY]`	n/a
10.0.1	`LIST ... VERBOSE`	n/a
10.0.1	`RESTORE ... PARMS`	`CONFIGURE CHANNEL ... PARMS`
10.0.1	`SEND ... PARMS`	`CONFIGURE CHANNEL ... PARMS`
9.2	`REPLICATE`	`RESTORE CONTROLFILE FROM ...`
9.2	`SET AUTOLOCATE`	Enabled by default
9.0.1	`ALLOCATE CHANNEL FOR DELETE`	n/a
9.0.1	`ALLOCATE CHANNEL ... TYPE`	`CONFIGURE CHANNEL ... DEVICE TYPE`

Table A–1 (Cont.) Deprecated RMAN Syntax

Deprecated in Release	Deprecated Syntax	Preferred Current Syntax
9.0.1	ALLOCATE CHANNEL ... KBYTES	CONFIGURE CHANNEL ... MAXPIECESIZE
9.0.1	ALLOCATE CHANNEL ... READRATE	CONFIGURE CHANNEL ... RATE
9.0.1	... ARCHIVELOG ... LOGSEQ	... ARCHIVELOG ... SEQUENCE
9.0.1	BACKUP ... SETSIZE	BACKUP ... MAXSETSIZE
9.0.1	CHANGE ... CROSSCHECK	CROSSCHECK
9.0.1	CHANGE ... DELETE	DELETE
9.0.1	REPORT ... AT LOGSEQ	REPORT ... AT SEQUENCE
9.0.1	SET AUXNAME	CONFIGURE AUXNAME
9.0.1	SET DUPLEX	SET BACKUP COPIES CONFIGURE BACKUP COPIES
9.0.1	SET LIMIT CHANNEL ...	ALLOCATE CHANNEL ... CONFIGURE CHANNEL ...
9.0.1	SET SNAPSHOT	CONFIGURE SNAPSHOT
9.0.1	UNTIL LOGSEQ (see *untilClause* on page 4-42)	UNTIL SEQUENCE (see *untilClause* on page 4-42)
8.1.7	CONFIGURE COMPATIBLE	n/a
8.1.5	ALLOCATE CHANNEL CLONE	CONFIGURE AUXILIARY CHANNEL
8.1.5	CHANGE ... VALIDATE	CROSSCHECK
8.1.5	CLONE (see RMAN)	AUXILIARY (see RMAN)
8.1.5	CONFIGURE CLONE	CONFIGURE AUXILIARY
8.1.5	MSGLOG (see RMAN)	LOG (see RMAN)
8.1.5	RCVCAT (see RMAN)	CATALOG (see RMAN)

B

RMAN Compatibility

This appendix describes the requirements for compatibility among the different components of the Recovery Manager (RMAN) environment. This appendix contains these topics:

- About RMAN Compatibility
- Determining the Recovery Catalog Schema Version
- RMAN Compatibility Matrix
- Cross-Version Compatibility of Recovery Catalog Exports and Imports
- RMAN Compatibility: Scenario

See Also: *Oracle Database Upgrade Guide*

About RMAN Compatibility

Table B–1 describes the components of an RMAN environment. Each component has a release number associated with it.

Table B–1 Components of an RMAN Environment

Component	Release Number Refers to ...
RMAN client	Version of RMAN client (displayed when you start RMAN)
Recovery catalog database	Version of Oracle Database
Recovery catalog schema in recovery catalog database	Version of RMAN client used to create the recovery catalog
Target database	Version of Oracle Database
Auxiliary database	Version of Oracle Database

For example, you can use a release 9.0.1 RMAN client with:

- A release 9.0.1 target database
- A release 9.0.1 auxiliary database
- A release 8.1.7 recovery catalog database whose catalog tables were created with RMAN release 9.0.1

Determining the Recovery Catalog Schema Version

To determine the current release of the catalog schema, you must run a SQL query.

1. Use SQL*Plus to connect to the recovery catalog database as the catalog owner. For example, enter:

   ```
   % sqlplus rman@catdb
   ```

2. Query the `rcver` catalog table. For example, run this query:

   ```
   SQL> SELECT * FROM rcver;

        VERSION
        ------------
        09.00.01.00
        10.02.01.00
        11.01.00.03
   ```

   If multiple versions are listed, then the last row is the current version, and the rows before it are prior versions. In the preceding example, the current recovery catalog schema version is 11.1 and the previous version was 10.2.

   Note that for releases 10.2 and later, the last two digits in the `rcver` output indicate patch level. For earlier releases, they are always zeros.

RMAN Compatibility Matrix

In general, the rules of RMAN compatibility are as follows:

- You can create an 8.*x* or 9.*x* recovery catalog schema in any Oracle database release 8.1.*x* (or higher), and a 10.0.1 (or higher) recovery catalog schema in any Oracle database release 9.0.1 (or higher).
- The recovery catalog schema version must be greater than or equal to the RMAN client version.
- If the recovery catalog is a **virtual private catalog** (see `CREATE CATALOG`), then the RMAN client connecting to it must be at patch level 10.1.0.6 or 10.2.0.3. Oracle9*i* RMAN clients cannot connect to a virtual private catalog. This version restriction does not affect RMAN client connections to an Oracle Database 11*g* base recovery catalog, even if the base catalog has virtual private catalog users.
- Ideally, the versions of the RMAN client and the target database should be the same (although there are other valid combinations, listed in Table B–2). The RMAN client cannot be a greater version than the target or auxiliary database.
- While backing up an Oracle Database 10*g* or later database with the Oracle9*i* RMAN client, you cannot include a control file that was created using `COMPATIBLE=10.0.0` in a datafile backup set. The workaround is to turn control file autobackup `ON`.
- The version of an auxiliary database instance must be equal to the version of the RMAN client.
- Any release of Oracle database can restore backup sets and copies created by any prior release back to Oracle8*i*.

Table B–2 shows version requirements for RMAN components. Note that the symbol >= before a release means all Oracle Database releases from this release or later along with their patches.

Table B–2 RMAN Compatibility Table

Target/Auxiliary Database	RMAN client	Recovery Catalog Database	Recovery Catalog Schema
8.0.6	8.0.6	>=8.1.7	>=8.0.6
8.1.7	8.0.6.1	>=8.1.7	>=8.1.7
8.1.7	8.1.7	>=8.1.7	>=RMAN client
8.1.7.4	8.1.7.4	>=8.1.7	8.1.7.4
8.1.7.4	8.1.7.4	>=8.1.7	>= 9.0.1.4
9.0.1	9.0.1	>=8.1.7	>= RMAN client
9.2.0	>=9.0.1.3 and <= target database executable	>=8.1.7	>= RMAN client
10.1.0	>=9.0.1.3 and <= target database executable	>=9.0.1	>= RMAN client
10.2.0	>=9.0.1.3 and <= target database executable	>=9.0.1	>= RMAN client
11.1.0	>=9.0.1.3 and <= target database executable	>=10.2.0.3	>= RMAN client
11.2.0	>=9.0.1.3 and <= target database executable	>=10.2.0.3	>= RMAN client

When using an older version of the RMAN client with a newer version of the database, you do not get the features of the newer version. For example, when using the Oracle9*i* RMAN client to back up an Oracle Release 10*g* database, you will not have access to features like the fast recovery area, flashback database, TSPITR with an RMAN-managed auxiliary instance, or recovery through `RESETLOGS`.

Cross-Version Compatibility of Recovery Catalog Exports and Imports

Data Pump Exports of the recovery catalog are often used as a way to backup its contents. When planning to use Data Pump Export to make a logical backup of the recovery catalog, see *Oracle Database Utilities* for details on compatibility issues relating to the use of database exports across versions of Oracle Database.

Exports from a later version of the database cannot be imported into databases running under earlier versions. You must export your recovery catalog data using the export utility from the earliest version of Oracle Database that you need to use for a recovery catalog.

For example, if you want to export recovery catalog data from a 9.2.0.5 database and you expect to import it into an 8.1.7.4 version of Oracle for disaster recovery, you must use the export utility from the 8.1.7.4 release of Oracle to perform the export operation. Otherwise, the import operation will fail.

RMAN Compatibility: Scenario

Assume that you maintain a production databases of the following releases:

- 9.2.0
- 10.2.0
- 11.2.0

You want to record RMAN repository data about these databases in a single recovery catalog database. According to Table B–2 on page B-3, you can use a single 11.2.0 recovery catalog database with a 11.2.0 catalog schema for all target databases. Ensure that the version of the RMAN client used to back up each target database meets the following requirements:

- Use the 9.2.0 RMAN executable to back up the 9.2.0 database.
- Use either the 9.2.0 or 10.2.0 RMAN executable to back up the 10.2.0 database.
- Use any RMAN executable to back up the 11.2.0 database.

C

Oracle Secure Backup (OSB) Cloud Module

The Oracle Secure Backup (OSB) Cloud Module allows you to take advantage of internet-based data storage services offered by Amazon Simple Storage Service (S3) for RMAN backup and recovery tasks. The Oracle Secure Backup Cloud Module uses the RMAN SBT (system backup to tape) interface to extend the Amazon S3 functionality for Oracle backup operations. The OSB Cloud Module offers an easy-to-manage, cost efficient, and scalable alternative to maintaining in-house data storage and managing a local, fully configured backup infrastructure.

The OSB Cloud Module has several advantages over traditional tape-based offsite backups:

- Continuous Accessibility

  OSB Cloud Module backups stored on Amazon S3 storage are always accessible. The cloud storage services availability and access model helps an organization to streamline recovery operations. For example, there is no need to ship or load tapes before a restore can be performed. You can still use familiar and standard tools like Enterprise Manager and your organization's current scripts continue to execute backup and restore tasks. With the ability to continually and easily access backups, the time spent restoring backups may be substantially reduced.

- Improved Reliability

  Because S3 storage is disk based, it is inherently more reliable than tape media. Internet storage service providers keep multiple, redundant copies of your data for availability and scalability purposes and the benefit of this practice to your organization and your data is increased reliability.

Backup On the Cloud

The OSB Cloud Module is part of the OSB product family and now provides the flexibility to back up your database to the Amazon S3 Cloud as well as to tape. With this cloud offering, local disk backups are sent directly to Amazon S3 for offsite storage and are fully integrated with RMAN's features and functionality. In addition, OSB Cloud Module backups work with tools like Oracle Enterprise Manager and your customized RMAN scripts.

The OSB Cloud Module efficiently handles the backing up of Oracle databases to S3 storage. You can backup Oracle databases starting with Oracle Database 9*i* Release 2 or higher. The OSB Cloud Module does not backup operating system files.

Getting Started

In order to use OSB Cloud Module on Amazon S3, you must set up both an Oracle Technical Network (OTN) and an Amazon Web Services (AWS) account. You also need the S3 Backup installer and a compatible version of Java software.

Here are the steps to set up, install, verify and run the OSB Cloud Module:

Steps	Description
1	Hardware and Software Prerequisites
2	Registering for An Oracle Technology Network (OTN) Account
3	Signing Up For Amazon S3 - AWS Account
4	Getting Your AWS Identifiers
5	Installing the OSB Cloud Module Library
6	Running the S3 Backup Installer ■ Verifying Java Version ■ Verifying `$ORACLE_HOME`
7	Storing Configuration in RMAN Repository - optional
8	Using the OSB Web Services Library and First Backup

Hardware and Software Prerequisites

The prerequisites for the OSB Cloud Module:

Hardware/Software	Version
Java	JDK 1.5 or later on the computer where you plan to run the S3 Backup Installer
Supported Platforms	■ Linux x86 ■ Linux x86-64 ■ Microsoft Windows (32-bit)
Oracle Database	You can backup databases starting with Oracle Database 9*i* Release 2 or later. Operating system files cannot be backed up with RMAN or the RMAN SBT interface.
S3 Backup Installer File	`osbws_install.jar` The installer downloads the library that is appropriate for the platform it is running on. It also creates the library configuration file and the Oracle Wallet where the AWS credentials are stored. If you are using Oracle provided Amazon Machine Images (AMIs) to run the Oracle Database on Amazon's Elastic Compute Cloud (EC2), the installer is already downloaded and can be found in the `/home/oracle/scripts` directory. Otherwise, you can download the file from the Cloud Computing Center link found on the Oracle Technical Network (OTN) website. Oracle recommends that users include any of the command-line options in a file and secure the file with appropriate operating system permissions. The S3 Backup Installer can then read the file, invoke the options, and prohibit unauthorized users from reading the file.

Hardware/Software	Version
Oracle Wallet Directory	The Oracle Wallet Directory stores your AWS identifiers and must exist before you can run the S3 Backup installer. If you have not set up a wallet directory then you must create one. Here are the suggested platform-specific locations for the wallet directory: ■ Linux: `$ORACLE_HOME/dbs/osbws_wallet` ■ Windows: `$ORACLE_HOME\database\osbws_wallet`
System Time	The authentication method used by S3 relies on the client's system time being similar to S3's time. In this case, the client is the computer where you are running the OSB Web Services library. S3 time is Coordinated Universal Time (UTC), so you must ensure that the system time on your client is within a few minutes of UTC.

Registering for An Oracle Technology Network (OTN) Account

You need an OTN account in order to download and run the S3 Backup installer for the OSB Cloud Module. If you do not have an OTN account, you may register for one at: `http://otn.oracle.com`. You can download the installer from the OTN Cloud Computing Center home page.

Signing Up For Amazon S3 - AWS Account

Before you can use the OSB Cloud Backup Module and access Amazon S3, you must create an AWS account.

You can open one at: `http://aws.amazon.com`, via the link: "*Create an AWS Account*".

The account requires that you provide a means of payment for Amazon to charge for your AWS S3 usage.

Getting Your AWS Identifiers

You will need the two AWS Identifiers that are assigned when you create your AWS account:

- Access Key ID
- Secret Access Key

You obtain these two credentials by:

- Going to the AWS web site at: `http://aws.amazon.com`
- Pointing to "*Your Web Services Account*" to display a list of options
- Clicking on "*AWS Access Identifiers*" link and logging in to your AWS account.

Note: It is a good idea to secure these credentials since they authorize charges for all Amazon Web Services and enable access to RMAN backups stored on Amazon S3.

Installing the OSB Cloud Module Library

If this is the first time you are running the S3 Backup installer, Oracle recommends that you run it initially without any parameters in order to get a listing and explanation of the mandatory and optional parameters. Analyzing and reviewing this information prior to executing the S3 Backup installer helps to ensure a successful installation.

To run the S3 Backup installer without parameters, type:

```
% java -jar osbws_install.jar
```

The explanation for the various parameters in the out put of the first run follows:

```
-AWSID:       AWS Access Key ID
-AWSKey:      AWS Secret Access Key
```

The credentials for Amazon Web Services. Mandatory.

```
-otnUser:     OTN Username
-otnPass:     OTN Password
```

The credentials for OTN, which the installer uses to identify the customer. Mandatory.

```
-walletDir:   Directory to store wallet
```

The location for the Oracle Wallet used to store S3 credentials and proxy information (if specified, see below). Mandatory.

Note: The Oracle wallet directory must exist prior to running the S3 Backup installer. Consult Hardware and Software Prerequisites for more information.

```
-configFile:   File name of config file
```

The location where the configuration file will reside. If omitted, the installer will create the configuration file and place it in a default system-dependent location. Optional.

Default Linux location: `$ORACLE_HOME/dbs/osbsws<ORACLE_SID>.ora`

Default Windows location: `$ORACLE_HOME\database\osbsws<ORACLE_SID>.ora`

```
-libDir:      Directory to store library
```

The location where the library will be placed. If this parameter is omitted, the installer does not download the library. Optional.

Suggested Linux location: `$ORACLE_HOME/lib/`

Suggested Windows location: `$$ORACLE_HOME\bin\`

```
 -libPlatform:   The desired platform for the library
```

The install tool will determine the platform automatically by examining the system where it is running. This parameter allows specifying it explicitly. Optional.

In this version the supported values for the parameter are: linux32, linux64, windows32.

```
-proxyHost:   HTTP proxy server
-proxyPort:   HTTP proxy server port
-proxyID: HTTP proxy server username
-proxyPass: HTTP proxy server password
```

The name and port of the HTTP proxy server, if required. If the proxy server is specified, the username and password, if required. Optional.

Steps 1 through 4 help you identify all the mandatory parameters and needed set up information for the installer. Make sure to check and collect the relevant information for any optional parameters that you want to include. For example, you may need to know the proxy server name, port and credentials of your installation.

At this point, you are ready to execute the installer.

Running the S3 Backup Installer

Oracle recommends running the Java installer in a secure mode. This means that you should avoid running it directly from the command line. A preferred method is to include the command-line options in a file and secure access to the file with the appropriate operating system permissions:

```
% java -jar osbws_install.jar -ARGFILE filename
```

Another method is to embed the run command, parameters and values in a file that can be executed as either a shell script or a Windows batch file.

Follow these steps:

- Create a file
- Set file permissions to grant owner of the file exclusive access

> **Note:** Setting the file permissions to restrict access is critical since the file contains both AWS and OTN credentials.

- Edit the file to contain a single line with the installer's run command and the mandatory parameters. You can compose a one-line invocation by populating the parameters with the information you obtained in the previous section.
- Execute the file as a shell script or Windows batch file

Note: To make the following example easier to read, `/orclhome` is set to `/orclhome`. In your installation, the value of `$ORACLE_HOME` will be something like: `/usr/oracle/product/9.2.0` (Linux).

Example C–1 Running the S3 Backup Installer

The following shows a sample run of the S3 Backup installer under Linux.

The first thing to do is to verify that the correct version of Java is present on the computer and that `$ORACLE_HOME` is defined.

Enter the following commands and review the output:

```
% java -version
java version "1.5.0_11"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_11-b03)
Java HotSpot(TM) Client VM (build 1.5.0_11-b03, mixed mode)


% echo $ORACLE_HOME
/orclhome
```

Create a file that contains the installer's invocation line and make sure that the file permissions restrict access except to the file owner.

```
% touch osbws.sh
```

```
% chmod 700 osbws.sh
```

Edit the file and add a line to invoke the installer.

```
java -jar osbws_install.jar -AWSID access key ID -AWSKey secret key -otnUser
jane.doe@example.com  -otnPass password for OTN -walletDir
$ORACLE_HOME/dbs/osbws_wallet  -libDir $ORACLE_HOME/lib/ -proxyHost
www-proxy.example.com
```

Execute the file.

```
% ./osbws.sh
```

Here is the start of the S3 Backup installer output:

```
OTN userid is valid.
AWS credentials are valid.
Creating new registration for this S3 user.
Created new log bucket.
Registration ID: 0f0a8aac-dad0-6254-7d70-be4ac4f112c4
S3 Logging Bucket: oracle-log-jane-doe-1
Create credential oracle.security.client.connect_string1
OSB web-services wallet created in directory /orclhome/dbs/osbws_wallet.OSB
web-services initialization file /orclhome/dbs/osbwst1.ora created.
Downloading OSB Web Services Software Library.
Downloaded 13165919 bytes in 204 seconds. Transfer rate was 64538 bytes/second.
Download complete.
Extracted file /orclhome/lib/libosbws11.so
```

When the installer completes, you should have these three files on your system:

1. The OSB Web Services Library
2. The Configuration file
3. The OSB Web Services Wallet

Note: The installer requires both the OTN and AWS credentials to create the wallet and perform other installation operations. Only your AWS credentials are retained when the installer has finished and they are stored in the Oracle Wallet. The AWS credentials are used solely to authenticate the library's interactions with S3 and are not used or sent anywhere else.

Storing Configuration in RMAN Repository - *optional*

To avoid having to provide the configuration information each time a backup is invoked, it is a good idea to store the OSB Cloud Module configuration information in the RMAN repository.

This example is for a pre-11g Release 2 database:

```
RMAN> configure channel device type sbt parms
"SBT_LIBRARY=/orclhome/lib/libosbwsll.so
ENV=(OSB_WS_PFILE=/orclhome/dbs/osbwst1.ora)';


using target database control file instead of recovery catalog
new RMAN configuration parameters:

"SBT_LIBRARY=/orclhome/lib/libosbwsll.so
ENV=(OSB_WS_PFILE=/orclhome/dbs/osbwst1.ora)';
```

```
new RMAN configuration parameters are successfully stored
```

When this example completes, the system is configured for OSB Cloud Module backups and you can use your usual RMAN backup and restore commands.

Note: For 11*g* Release 2 databases and later, you must use the `SBT_PARMS` parameter to specify environment variables.

Using the OSB Web Services Library and First Backup

You are now ready to connect to your target database and configure an RMAN channel. You need to specify both the library and the configuration file in the command.

This example configures an RMAN channel for an Oracle 11*g* Release 2 database:

```
RMAN> run {
            allocate channel dev1 type
            sbt parms='SBT_LIBRARY=/orclhome/lib/libosbws11.so,
            SBT_PARMS=(OSB_WS_PFILE=/orclhome/dbs/osbwst1.ora)';
          }
```

At this point, you can issue your usual RMAN backup and restore commands.

Note: For Oracle 11*g* Release 2 databases and later, you must use the `SBT_PARMS` parameter for specifying environment variables. For pre-Oracle 11*g* Release 2 databases, you can still use the `ENV` parameter of the `PARMS` option to specify environment variables.

> **See Also:** *Database Backup in the Cloud* technical white paper and the *Backup Up Database* Demonstration on the OTN: Cloud Computing Center website

Securing OSB Cloud Module Backups

To ensure that your data is properly secured, Oracle recommends that you make RMAN backup encryption a standard part of your backup processes. This recommendation is even more important to implement when you are storing critical backup data off-premises. Encrypting RMAN backups on Amazon S3 can also assist you in meeting key audit and regulatory compliance requirements for your organization's data.

> **See Also:** Encryption of Backup Sets for a discussion of RMAN encryption options.

Helpful Links

For more information on database backup in the cloud, see Oracle in the Cloud Frequently Asked Questions (FAQ) on the OTN website:
http://www.oracle.com/technology/tech/cloud/faq.html#S3

Troubleshooting the OSB Cloud Module

This section lists potential issues that may affect the installation or the operation of the OSB Cloud Module:

Symptoms	Error Messages	Resolution
The S3 Backup installation cannot create the license file on Amazon S3.	`Time-out waiting for license file to be created.`	The first time you run the S3 Backup installer for a set of AWS identifiers, the installer creates a license file on Amazon S3. If there are problems preventing its creation the Time-out error message is displayed in the installation output. Please contact Oracle support in order to resolve the issue.

Index

Symbols

%U format, 4-3
? symbol in quoted strings, 1-3
@ command, 2-2 to 2-3
@ parameter, RMAN command, 3-61
@ symbol in quoted strings, 1-3
@@ command, 2-4

A

ABORT parameter, SHUTDOWN command, 3-85
ACTIVE DATABASE parameter, DUPLICATE command, 2-133
Advanced Compression option, 2-70
Advanced Security option, 2-21
ADVISE FAILURE command, 2-6 to 2-8
ADVISE OPTION parameter, REPAIR FAILURE command, 3-22
AFTER parameter, completedTimeSpec subclause, 4-11
AL16UTF16 character set, CLOB storage, 2-90
ALGORITHM parameter, SET command, 3-73
ALL parameter
  BACKUP command, 2-43
  SHOW command, 3-83
ALL parameter, archivelogRecordSpecifier subclause, 4-6
ALL PRIVILEGES parameter, REVOKE command, 3-57
ALL SCRIPT NAMES parameter, LIST command, 2-169
ALLOCATE CHANNEL command, 2-10 to 2-11
  allocOperandList subclause, 4-2
ALLOCATE CHANNEL FOR MAINTENANCE command, 2-13 to 2-14
  shared server, 2-13
allocOperandList subclause
  parameter descriptions, 4-2
  syntax diagram, 4-2
ALLOW CORRUPTION parameter, RECOVER command, 3-9
ALTER DATABASE BACKUP command (SQL), 2-38
ALTER DATABASE command, 2-17
analyzing RMAN repositories, 3-28
APPEND parameter, RMAN command, 3-60
archived log backups, 2-176
archived log copies, 2-178
archived log failover, 2-37
archived logs
  in backup sets, 2-172
  validation report, 3-113
archived redo files, optimization algorithms, 2-75
archived redo logs, 4-6
  adding to repository, 2-53
  backing up, 2-28, 2-37, 4-32
  backing up *n* times, 2-46
  cataloging in RMAN repository, 2-52
  days for recovery, 3-30
  deleting, 2-40, 2-76, 2-123, 3-10
  deletion policies, 2-72
  displaying information, 2-169
  excluding from backup, 2-35, 2-46
  excluding from validation, 3-112
  foreign (LogMiner), 4-21
  image copies, 2-42
  limiting number of backups, 2-34
  missing or inaccessible, 3-2
  number of backups, 3-83
  obsolete, 3-29
  previewing, 3-43
  recovering, 3-2
  report output, 3-32
  restoring, 3-37
  restoring a range, 3-45
  setting the destination, 3-74
  specifying a range, 4-6
  using for recovery (example), 3-12
  validating, 3-109, 3-111
ARCHIVELOG BACKUP COPIES parameter
  CONFIGURE command, 2-75
  SHOW command, 3-83
ARCHIVELOG DELETION POLICY parameter
  CONFIGURE command, 2-72
  SHOW command, 3-83
ARCHIVELOG DESTINATION parameter, SET command, 3-74
ARCHIVELOG mode, 2-19, 2-128, 2-150
ARCHIVELOG parameter
  CATALOG command, 2-53
  VALIDATE command, 3-109

recordSpec subclause, 4-37
ARCHIVELOG TAG parameter, RECOVER command, 3-10
archivelogRecordSpecifier subclause, 4-6, 4-21
parameter descriptions, 4-6
syntax diagram, 4-6
archiving redo logs, 2-17
archlogRange subclause
parameter descriptions, 4-6
syntax diagram, 4-6, 4-21
arguments to stored scripts, 2-105
AS BACKUPSET parameter, BACKUP command, 2-41
AS COPY parameter, BACKUP command, 2-42
ASM
See Automatic Storage Management (ASM)
AT SCN parameter, REPORT command, 3-31
AT SEQUENCE parameter, REPORT command, 3-31
at sign command (@), 2-2 to 2-3
at sign parameter, RMAN command, 3-61
AT TIME parameter, REPORT command, 3-31
atClause subclause
parameter descriptions, 3-31
syntax diagram, 3-29
AUTOBACKUP parameter, RESTORE command, 3-48
autoBackupOptList subclause
parameter descriptions, 3-48
syntax diagram, 3-42
autobackups, control file, 2-78
Automated Diagnostic Repository, logging failures, 3-107
automated repair, 3-21
automatic diagnostic repository
changing failure status, 2-63
changing logged failures, 2-57
See diagnostic repository
Automatic Storage Management (ASM)
channel allocation, 2-10
converting format to, 2-96
converting format to (example), 2-98
converting formats, 2-89
disk groups, 3-39
duplicating databases, 2-128
specifying disk groups, 2-79
AUXILIARY CHANNEL parameter
CONFIGURE command, 2-80
SHOW command, 3-83
auxiliary databases
connecting to, 2-86, 4-13
initialization files, 2-135
AUXILIARY DESTINATION parameter
RECOVER command, 3-10
TRANSPORT TABLESPACE command, 3-98
auxiliary filename settings, 2-69
AUXILIARY FORMAT parameter
allocOperandList subclause, 4-3
BACKUP command, 2-31
AUXILIARY INSTANCE PARAMETER FILE TO parameter, SET command, 3-78
auxiliary instances, specifying channels, 2-80
AUXILIARY parameter
ALLOCATE CHANNEL command, 2-11
CONNECT command, 2-87
RMAN command, 3-60
AUXNAME parameter
CONFIGURE command, 2-69
SHOW command, 3-83
availability, changing status for backup or copy, 2-59, 2-60
AVAILABLE parameter, CHANGE command, 2-59
AVAILABLE status
definition, 2-110
effect of DELETE command, 2-113

B

BACKED UP parameter, maintQualifier subclause, 4-32
backing up, excluded files, 2-20
backup clause
parameter descriptions, 2-27
syntax diagram, 2-23
BACKUP command, 2-19 to 2-47
backup clause, 2-23, 2-27
backupOperand subclause, 2-24, 2-28
backupSpec subclause, 2-25, 2-36
backupSpecOperand subclause, 2-26, 2-39
backupTypeSpec subclause, 2-26, 2-40
copyOfSpec subclause, 2-26, 2-42
datafileCopySpec subclause, 2-27, 2-43
duration subclause, 2-27, 2-43
fileNameConversionSpec parameter, 4-17
forRecoveryOfSpec subclause, 2-27, 2-44
limiting processing time, 2-44
notBackedUpSpec subclause, 2-27, 2-45
sizeSpec subclause, 2-27, 2-46
skipSpec subclause, 2-27, 2-46
backup commands, 3-65
BACKUP COPIES parameter
SET command, 3-74
SHOW command, 3-83
backup levels, 2-20
backup media, 2-19
BACKUP OPTIMIZATION parameter
SHOW command, 3-83
BACKUP OPTIMIZATION parameter, CONFIGURE command, 2-76
BACKUP parameter
LIST command, 2-169
maintSpec subclause, 4-34
backup pieces
adding to repository, 2-52, 2-54
available information, 2-171, 2-175
crosschecking or deleting, 2-13
definition, 2-41
displaying information, 2-169
format, 4-3
maximum size, 4-3

obsolete, 3-33
overriding default copies, 3-74
restoring, 3-38
validating, 3-110
backup retention policies, 2-116
backup sections, sizing, 2-35
backup set failover, 2-37
backup sets
about archived logs, 2-172
available information, 2-171
backing up, 2-19
backing up encrypted, 2-37
backup pieces, available information, 2-173
binary compression, 2-21, 2-42
cataloging in RMAN repository, 2-52
date ranges, 2-111
deleting, 2-40, 2-123
displaying information, 2-169
encryption, 2-21, 3-73
information about copies, 2-173
information about datafiles, 2-172
maximum size, 2-34
maximum size per channel, 2-76
obsolete
report output, 3-33
optimization algorithms, 2-75
recovering, 3-10
restoring only from, 3-43
retention policies, 2-77
status, 2-109
summary information, 2-174
tag names, 2-36
unused block compression, 2-41
validating, 3-110, 3-111
BACKUP_TAPE_IO_SLAVES initialization parameter, 2-12, 2-75
backupCommands subclause, syntax diagram, 3-65
backupConf subclause
parameter descriptions, 2-74
syntax diagram, 2-67
backupOperand subclause
parameter descriptions, 2-28
syntax diagram, 2-24
BACKUPPIECE parameter
CATALOG command, 2-54
recordSpec subclause, 4-37
backups
applying incremental, 3-2
binary compression, 2-21, 2-42
changing availability, 2-59, 2-60
configuration, 2-65
corruption tolerance (example), 2-50
creating multiple copies, 2-75
creating transportable tablespaces, 3-96
criteria for obsolescence, 4-36
cumulative, 2-20
datafiles, 2-41
datafiles requiring new, 3-30
deleting, 2-113
deleting obsolete, 2-116
duplexing example, 2-82
duplicate copies, 2-12
duplicating, 2-127
failures, 2-63
inconsistent, 2-19
incremental, 2-20, 2-44, 3-31
limiting duration, 2-44
listing, 2-164, 3-31
minimum required, 3-31
multisection, 2-35
obsolete, 3-29
overriding defaults in restore, 3-48
overwriting, 2-35
partial, 2-44
previewing, 3-43
redundant, 3-33
removing from recovery catalog, 3-103
restarting, 2-46
retaining multiple full (level 0), 2-77
retention policies, 2-75
summary list, 2-170
using incremental (example), 3-14
using incremental updates (example), 3-13
validating files, 3-12
validating in repository, 2-109
BACKUPS parameter, UNREGISTER command, 3-103
BACKUPSET parameter
BACKUP command, 2-37, 2-41
RECOVER command, 3-10
RESTORE command, 3-43
VALIDATE command, 3-110
recordSpec subclause, 4-37
backupSpec subclause
parameter descriptions, 2-36
syntax diagram, 2-25
backupSpecOperand subclause
parameter descriptions, 2-39
syntax diagram, 2-26
backupTypeSpec subclause
parameter descriptions, 2-40
syntax diagram, 2-26
Backus Naur Form syntax diagrams, 1-2
base recovery catalog
See recovery catalog
batch files
See command files
BEFORE parameter, completedTimeSpec subclause, 4-11
BEFORE RESETLOGS parameter, FLASHBACK DATABASE command, 2-153
BEFORE SCN parameter, FLASHBACK DATABASE command, 2-153
BEFORE SEQUENCE parameter, FLASHBACK DATABASE command, 2-153
BEFORE TIME parameter, FLASHBACK DATABASE command, 2-153
BETWEEN parameter, completedTimeSpec subclause, 4-11
BFILE data type, excluded from backup, 2-20

binary compression, 2-21, 2-42
block change tracking files
  excluded from backup, 2-20
  improving performance, 2-21
block checksums, 2-34
block corruptions
  checking for, 3-107
  flashback, 2-151
block media recovery, 3-2
block sizes of backup media, 2-19
blockObject subclause
  parameter descriptions, 3-8
  syntax diagram, 3-5, 3-108
blocks
  corrupt, 3-107
  creating, 3-65
  See also data blocks
BNF syntax diagrams, 1-2
braces {}, 3-65
BY FILE parameter, LIST command, 2-170

C

CATALOG command, 2-52 to 2-55
CATALOG FOR DATABASE parameter
  GRANT command, 2-157
  REVOKE command, 3-57
CATALOG parameter
  CONNECT command, 2-87
  RMAN command, 3-60
CATALOG privilege, 2-102
catalogs
  importing, 2-161
  virtual private, 2-103
CDs as backup media, 2-19
cfauConf subclause
  parameter descriptions, 2-77
  syntax diagram, 2-67
CHANGE command, 2-57 to 2-63
  changeFailure subclause, 2-58, 2-62
  forDbUniqueNameOption subclause, 2-58
  maintSpec subclause, 2-58, 4-34
  recordSpec subclause, 4-37
  resetDbUniqueNameOption subclause, 2-58, 2-61
changeFailure subclause
  parameter descriptions, 2-62
  syntax diagram, 2-58
changes, reversing, 2-150
channel names, 2-11
CHANNEL parameter
  ALLOCATE CHANNEL command, 2-11
  BACKUP command, 2-28
  CONFIGURE command, 2-80
  RESTORE command, 3-42, 3-47
  SEND command, 3-68
  SHOW command, 3-83
  SQL command, 3-88
channel parameters, 3-68
channels
  allocating for duplication, 2-133
  allocating for restore, 3-43
  allocating for validation, 3-111
  allocating manually, 2-10, 2-13
  allocating to shared server sessions, 2-13
  configuring for RAC (example), 2-83
  configuring type for recovery, 3-6
  device type default, 2-81
  identifying database server sessions, 3-78
  maximum number, 2-10
  naming conventions, 2-79
  releasing, 3-18
  restoring in Data Guard, 3-46
  setting for restore (example), 3-12
  with proxy capabilities, 2-35
character sets, CLOB storage, 2-90
CHECK LOGICAL parameter
  BACKUP command, 2-29, 2-50
  RECOVER command, 3-10
  RESTORE command, 3-42
  VALIDATE command, 3-111
checksums, 2-34
CHECKSYNTAX parameter, RMAN command, 3-60
client compatibility, B-1
client, opening RMAN, 3-59
CLOB data type, transporting, 2-90
closing RMAN, 2-149, 2-187
Cloud computing, C-1
CMDFILE parameter, RMAN command, 3-61
cmdLine clause
  parameter descriptions, 3-60
  syntax diagram, 3-60
cmdLine command
  See RMAN command
command entry description, 1-1
command files, 2-2, 2-4, 3-61
command format, 1-3
COMMAND ID TO parameter
  SET command, 3-78
command line arguments, 3-59
command subclause summary, 1-8
command subclauses
  See specific subclause name
command summary, 1-6 to 1-8
command terminator, 1-3
commands
  creating blocks, 3-65
  deprecated, A-1
  executing host, 2-159
  executing SQL, 3-88
  RMAN, 3-59
  sending to media manager, 3-61
  sending to media managers, 3-68
  See specific command name
comment character (#), 1-3
COMMENT parameter
  CREATE SCRIPT command, 2-106
  DUPLICATE command, 2-137
  REPLACE SCRIPT command, 3-26
compatibility, B-1
COMPATIBLE initialization parameter, 2-125

completedTimeSpec subclause, 4-11
compression, 2-42
COMPRESSION ALGORITHM parameter
  BASIC, 2-70
  CONFIGURE command, 2-70
  HIGH, 2-70
  LOW, 2-70
  MEDIUM, 2-70
  SHOW command, 3-83
compression levels, 2-70
configuration
  displaying, 2-65
  displaying settings, 3-82
  overriding, 3-65, 3-70
  restoring default settings, 2-65
  setting, 2-65
configuration files, excluded from backup, 2-20
CONFIGURE BACKUP OPTIMIZATION
    setting, 2-39
configure clause
  parameter descriptions, 2-68
  syntax diagram, 2-66
CONFIGURE command, 2-65 to 2-82
  backupConf subclause, 2-67, 2-74
  cfauConf subclause, 2-67, 2-77
  configure clause, 2-66, 2-68
  delalConf subclause, 2-66, 2-72
  deviceConf subclause, 2-67, 2-79
  encryption settings, 2-22
  forDbUniqueNameOption subclause, 2-68
  sizeSpec subclause, 2-67
CONFIGURE commands, 3-82
CONFIGURE CONTROLFILE AUTOBACKUP
    parameter, 2-38
CONNECT command, 2-86 to 2-87
CONNECT IDENTIFIER parameter, CONFIGURE
    command, 2-68
CONNECT parameter, allocOperandList
    subclause, 4-2
connect strings, 4-2
  to auxiliary database, 3-60
  to recovery catalog database, 3-60
  to target database, 3-62
connections, changing, 2-86
connectStringSpec subclause, 4-13
consistent shutdowns, 2-19
control file backups, available information, 2-177
control file copies
  available information, 2-178
  tag names, 2-36
control files
  adding copies to repository, 2-54
  autobackup validation, 3-111
  automatic backups, 2-78
  backing up copies, 2-43
  copies
    cataloging in RMAN repository, 2-52
    validating, 3-110
  filenames for control file autobackup, 3-78
  including in database backup, 2-38
  limiting search, 3-48
  re-creating, 3-54
  restoring, 3-37, 3-40, 3-45
  restoring (example), 3-14
  restoring autobackups, 3-48
  restoring for primary database, 3-46
  resynchronization, 3-53
  snapshots, 2-40, 2-71
  validating, 3-109
  validation report, 3-113
CONTROL_FILES initialization parameter, 2-125
CONTROLFILE AUTOBACKUP FORMAT
    parameter, SET command, 3-78
CONTROLFILE AUTOBACKUP parameter
  CONFIGURE command, 2-77
  SHOW command, 3-83
CONTROLFILE parameter
  RESTORE command, 3-45, 3-47
  VALIDATE command, 3-109, 3-110
CONTROLFILE parameter, listObjList
    subclause, 4-29
CONTROLFILECOPY parameter
  BACKUP command, 2-38
  CATALOG command, 2-54
  RESYNC CATALOG command, 3-54
  VALIDATE command, 3-110
  recordSpec subclause, 4-37
convert clause
  parameter descriptions, 2-92
  syntax diagram, 2-91
CONVERT command, 2-89 to 2-96
  convert clause, 2-91, 2-92
  convertOptionList subclause, 2-91, 2-95
  fileNameConversionSpec parameter, 4-17
  formatSpec subclause, 2-92
  skipSpec subclause, 2-91, 2-95
  transportOptionList subclause, 2-91, 2-94
convertOptionList subclause
  parameter descriptions, 2-95
  syntax diagram, 2-91
copies
  changing availability, 2-59, 2-60
  creating multiple backups, 2-75
  deleting, 2-113, 2-123
  deleting obsolete, 2-116
  expired, 2-170
  minimum required, 3-31
  of backup sets, available information, 2-173
  validating in repository, 2-109
  viewing status, 2-109
COPIES parameter, BACKUP command, 2-29
copy location, 2-42
COPY OF parameter
  BACKUP command, 2-43
  RECOVER command, 3-6
COPY parameter
  BACKUP command, 2-42
  LIST command, 2-169
  maintSpec subclause, 4-35
copying files, 2-42

copyOfSpec subclause
parameter descriptions, 2-42
syntax diagram, 2-26, 3-108
corrupt blocks
checking for, 3-107
restoring, 3-38
tolerance for, 3-75
tolerance for (example), 2-50
CORRUPTION LIST parameter, RECOVER command, 3-9
CORRUPTION parameter
RECOVER command, 3-9
CPU_COUNT initialization parameter, 3-11
CREATE CATALOG command, 2-102 to 2-103
CREATE SCRIPT command, 2-105 to 2-106
CREATE TYPE privilege, 3-105
CROSSCHECK command, 2-109 to 2-111
maintSpec subclause, 2-111, 4-34
recordSpec subclause, 4-37
cross-platform compatibility, 2-20
cumulative backups, 2-20
CUMULATIVE parameter
BACKUP command, 2-29
CURRENT CONTROLFILE parameter
BACKUP command, 2-38
VALIDATE command, 3-109, 3-110

D

data block address (DBA), 3-9
data blocks
checking for corrupted, 2-36
recovering corrupt, 3-2
recovering individual, 3-9
recovering logically corrupt, 3-10
recovering physically corrupt, 3-9
data compression, 2-42
data encryption, 2-125
Data Guard
changing backup associations, 2-61
configuring databases, 2-65
creating physical standby databases, 2-65
creating standby databases (example), 2-144
deleting backups and copies, 2-114
restoring backups, 3-40
RMAN configuration in recovery catalog, 2-72
specifying databases, 4-20
updating backup status, 2-110
Data Guard backups
accessibility by databases, 2-22
changing associated database, 2-57
database association, 2-22
Data Pump, 2-90, 2-152, 3-96, B-3
Data Recovery Advisor, 2-6, 3-21
data types, endian conversion, 2-90
database changes, reversing, 2-150
database configuration
See configuration
database duplication
See duplication
database failure, repairing, 3-21
database identifiers
changing, 2-20
duplicates, 3-16
database incarnations, 2-179
DATABASE parameter
BACKUP command, 2-38
RECOVER clause, 3-7
REPORT command, 3-31
RESTORE command, 3-45
SWITCH command, 3-93
UNREGISTER command, 3-102
VALIDATE command, 3-110
DATABASE parameter, DUPLICATE command, 2-131
database point-in-time recovery (DBPITR), 3-2
database registration, 3-57
database schemas
report output, 3-32
database server sessions
identifying channels, 3-78
DATABASE SKIP TABLESPACE parameter, listObjList subclause, 4-29
databases, 2-125
available information, 2-180
backing up, 2-19
changing connections, 2-86
connecting to, 2-86
converting platform format, 2-89
copying, 2-124
deleting, 2-123
encryption settings, 3-83
mounting, 2-17
opening, 2-17
platform conversion scripts, 2-94
recovering, 3-2, 3-7
recovering with incremental backups (example), 3-14
recovering with incremental updates, 3-13
registering, 2-157, 3-16
removing from recovery catalog, 3-102
resetting incarnation, 3-35
resynchronization, 3-53
revoking registration privileges, 3-57
shutting down target, 3-85
specifying in Data Guard environment, 4-20
standby, restoring control files, 3-47
starting target from RMAN, 3-90
transparent encryption, 2-71
transporting re-created files, 2-92
updating filenames, 3-92
validating, 3-110
DATAFILE BACKUP COPIES parameter
CONFIGURE command, 2-75
SHOW command, 3-83
datafile backup sets, available information, 2-171
datafile backups
available information, 2-175
expired, 2-170
DATAFILE BLOCK parameter

RECOVER command, 3-9
datafile copies
available information, 2-177
deleting, 2-40
displaying information, 2-169
listing, 3-31
obsolete, 3-29
report output, 3-33
restoring only from, 3-43
tag names, 2-36
validating, 3-111
DATAFILE parameter
BACKUP command, 2-38
CONVERT command, 2-93
RECOVER command, 3-8
REPORT command, 3-31
RESTORE command, 3-46
SWITCH command, 3-93
VALIDATE command, 3-111
DATAFILECOPY parameter
CATALOG command, 2-54
RECOVER command, 3-6, 3-10
RESTORE command, 3-43
recordSpec subclause, 4-38
datafileCopySpec subclause
parameter descriptions, 2-43
syntax diagram, 2-27
datafiles
auxiliary filenames, 2-69
backing up, 2-19, 2-41
backing up copies, 2-43
backing up large, 2-35
backing up new only, 2-45
backups
deleting obsolete, 2-116
converting platform format, 2-89
copies
adding to repository, 2-54
cataloging in RMAN repository, 2-52
incremental backups, 2-54
tag names, 2-54
validating, 3-111
days of archived redo logs, 3-30
excluding from backup, 2-35, 2-46
excluding from conversion, 2-95
excluding from validation, 3-112
flashback status changes, 2-151
forcing restore, 3-43
image copies, 2-42
in backup sets
available information, 2-172
listing within a tablespace, 3-31
minimum number of backups, 3-31
offline, 2-46
optimization algorithms, 2-75
point-in-time lists, 3-30
readonly, 2-46
read-only, omitting from restore, 3-44
recovering, 3-2, 3-8
recovering corrupt data blocks, 3-2
renaming, 3-75, 3-76
report output, 3-32, 3-33
requiring backup
report output, 3-33
requiring new backups, 3-30
restored location, 3-38
restoring, 3-37
all online, 3-45
from image copies, 3-10
individual, 3-46
to new locations, 3-39
reversing changes, 2-150
unrecoverable, 3-30
updating filenames, 3-92
validating, 3-110, 3-111
validation report, 3-112
datafileSpec subclause, 4-15
DATAPUMP DIRECTORY parameter, TRANSPORT TABLESPACE command, 3-98
date ranges of backup sets, 2-111
dates in RMAN commands, 4-43
DAYS parameter, REPORT command, 3-30
DB_BLOCK_CHECKSUM initialization parameter, 2-34
DB_CREATE_FILE_DEST initialization parameter, 4-3
DB_FILE_NAME_CONVERT initialization parameter, 2-129, 2-133, 2-137, 3-98, 4-17
DB_FLASHBACK_RETENTION_TARGET initialization parameter, 2-151
DB_NAME parameter
IMPORT CATALOG command, 2-162
RESTORE command, 3-48
DB_NAME, changing, 2-20
DB_RECOVERY_FILE_DEST initialization parameter, 3-45
DB_RECOVERY_FILE_DEST parameter
BACKUP command, 2-39
CATALOG command, 2-55
RESTORE command, 3-48
VALIDATE command, 3-111
DB_UNIQUE_NAME initialization parameter, 3-48
DB_UNIQUE_NAME parameter
CHANGE command, 2-61
CONFIGURE command, 2-68, 2-72
DELETE command, 2-115
LIST command, 2-166
RESTORE command, 3-46
RESYNC CATALOG command, 3-55
SHOW command, 3-83
UNREGISTER command, 3-103
DB_UNIQUE_NAME parameter, forDbUniqueNameOption, 4-20
DB_UNIQUE_NAME setting, 2-54, 2-57, 2-180, 3-37
DBA parameter, STARTUP command, 3-90
DBID parameter
IMPORT CATALOG command, 2-162
SET command, 3-78, 3-79
DBIDs
available information, 2-180

changing, 2-20
duplicates, 3-16
of standby databases, 2-124
DBNEWID utility, 3-16
dbObject subclause
parameter descriptions, 3-7, 4-30
syntax diagram, 3-4, 4-29
DECRYPTION parameter, SET command, 3-72
DEFAULT DEVICE TYPE parameter
CONFIGURE command, 2-81
SHOW command, 3-83
delalConf subclause
command parameters, 2-72
syntax diagram, 2-66
DELETE ARCHIVELOG parameter, RECOVER command, 3-10
delete clause, syntax diagram, 2-114
DELETE command, 2-113 to 2-116
delete clause, 2-114
forDbUniqueNameOption subclause, 2-115
maintSpec subclause, 2-114, 4-34
recordSpec subclause, 4-37
DELETE INPUT parameter, BACKUP command, 2-40
DELETE SCRIPT command, 2-119
deprecated commands, A-1
DESTINATION parameter, RECOVER command, 3-10
DESTINATION PLATFORM parameter, CONVERT command, 2-94
destinations for restoring datafiles, 3-39
DEVICE TYPE parameter
ALLOCATE CHANNEL command, 2-11
ALLOCATE CHANNEL FOR MAINTENANCE command, 2-14
BACKUP command, 2-29
CHANGE command, 2-60
CONFIGURE command, 2-81
DELETE command, 2-116
DUPLICATE command, 2-133
FLASHBACK DATABASE command, 2-153
RECOVER command, 3-6
REPORT command, 3-30
RESTORE command, 3-43
SEND command, 3-68
SHOW command, 3-83
VALIDATE command, 3-111
maintSpec subclause, 4-35
DEVICE TYPE parameter, maintQualifier subclause, 4-32
device types, specifying, 2-80
deviceConf subclause
parameter descriptions, 2-79
syntax diagram, 2-67
deviceSpecifier subclause, 4-16
diagnostic repository, 2-7
diagrams of syntax, 1-2
disk device type setting, 2-81
disk drives, using several for backup, 2-11
disk files, creating from stored scripts, 2-185
DISK parameter, deviceSpecifier subclause, 4-16
DISKRATIO parameter, BACKUP command, 2-30
displaying scripts, 2-185
DORECOVER parameter, DUPLICATE command, 2-133
double at sign command (@@), 2-4
double quotes, 1-3
DROP CATALOG command, 2-121 to 2-122
DROP DATABASE command, 2-123
dual-mode encryption, 2-21
duplexing, 2-75, 2-82, 3-74, 3-83
duplicate clause
parameter descriptions, 2-131
syntax diagram, 2-130
DUPLICATE command, 2-124 to 2-138
duplicate clause, 2-130, 2-131
dupOptionList subclause, 2-130, 2-132
fileNameConversionSpec parameter, 4-17
logSpec subclause, 2-131, 2-137
setParameter subclause, 2-131, 2-136
sizeSpec subclause, 2-131
duplicate databases
creating
using CONFIGURE AUXNAME, 2-143
duplicate databases, opening, 2-134, 2-135
duplicating files, 2-42
duplication
backup based, 2-128
configuration, 2-65
example using auxiliary channels, 2-12
dupOptionList subclause
parameter descriptions, 2-132
syntax diagram, 2-130
DURATION parameter, BACKUP command, 2-44
duration subclause
parameter descriptions, 2-43
syntax diagram, 2-27
DVDs as backup media, 2-19
dynamic performance views
summary list, 5-1
V$BACKUP_PIECE, 2-171
V$BACKUP_SET, 2-171
V$DATABASE, 3-79
V$DATABASE_BLOCK_CORRUPTION, 2-36, 3-2, 3-9, 3-10, 3-14, 3-75, 3-107
V$DATABASE_FLASHBACK_ON, 2-150
V$DATABASE_INCARNATION, 3-3
V$DATABASE.CURRENT_SCN, 2-151
V$DATAFILE_COPY, 2-177
V$FLASHBACK_DATABASE_LOG, 2-153 to 2-154
V$RMAN_COMPRESSION_ALGORITHM, 2-70, 3-72
V$RMAN_ENCRYPTION_ALGORITHMS, 2-21, 2-71, 3-73
V$RMAN_OUTPUT, 2-20, 3-61
V$RMAN_STATUS, 3-61
V$SESSION, 3-78
V$TABLESPACE, 2-152
V$TRANSPORTABLE_PLATFORM, 2-89

E

ECHO parameter, SET command, 3-73
encryption
  backup sets, 2-21, 2-37, 3-38
  of duplicate databases, 2-125
  recovering tablespaces, 3-2
  specifying decryption passwords, 3-72
ENCRYPTION ALGORITHM parameter, CONFIGURATION command, 2-71
encryption algorithms, 2-21, 3-73
ENCRYPTION parameter
  CONFIGURE command, 2-70
  SET command, 3-73
  SHOW command, 3-83
encryption settings
  overriding, 2-70, 2-71
  transparent mode, 2-70
  V$RMAN_ENCRYPTION_ALGORITHMS view, 2-71
encryption wallets
  See wallets
end times
  for restore, 3-78
endian formats
  converting, 2-90
  transportable tablespaces, 3-96
  transporting across platforms, 2-89
environment variables
  in RMAN strings, 1-3
  NLS_DATE_FORMAT, 3-31
  NLS_LANG, 3-31
error messages
  ORA-1152, 2-133
  ORA-1578, 3-9
  ORA-19504, 2-134
  ORA-19624, 2-134
  ORA-19916, 2-21
  ORA-27086, 2-134
  RMAN-0558, 3-60
  RMAN-06004, 2-37, 2-146, 2-185
  RMAN-06445, 2-87, 2-88
  RMAN-06496, 3-40
  RMAN-06710, 2-119
  RMAN-10031, 2-134
  RMAN-10035, 2-134
  RMAN-20401, 2-105
  RMAN-6758, 3-46
  too many open files, 4-3
EXCLUDE FLASHBACK LOG parameter, RECOVER command, 3-10
EXCLUDE FOR TABLESPACE parameter, CONFIGURE command, 2-76
EXCLUDE parameter, SHOW command, 3-83
EXECUTE SCRIPT command, 2-146 to 2-147
EXIT command, 2-149
exiting RMAN, 2-187
EXPIRED parameter
  DELETE command, 2-115
  LIST command, 2-166
EXPIRED status, 2-110, 2-114
EXPORT LOG parameter, TRANSPORT TABLESPACE command, 3-98
export utilities, 2-90
external tables, excluded from backup, 2-20

F

failed backups, restarting, 2-46
FAILURE parameter
  CHANGE command, 2-63
  LIST command, 2-167
failure summary, 2-6
failures, available information, 2-180
fast recovery, 2-39
fast recovery areas
  cataloging contents, 2-55
  defining for duplicate databases, 2-127
  location of autobackup, 2-79
  specifying for restore, 3-48
  validating, 3-111
file copies, 2-42, 3-48
  See also backups
FILE parameter
  CREATE SCRIPT command, 2-106
  LIST command, 2-170
  PRINT SCRIPT command, 2-185
  REPLACE SCRIPT command, 3-26
file recovery, 3-11
file sections, sizing, 2-35
filename formats, 3-78
fileNameConversionSpec subclause, 4-17
filenames
  changing during platform conversion, 2-95
  datafile image copies, 2-38
  generating new, 4-17
  setting auxiliary, 2-69
  updating, 3-92
files
  backing up new only, 2-45
  backing up with proxy copy, 2-35
  changing availability, 2-59, 2-60
  creating from stored scripts, 2-185
  excluded from backup, 2-20
  listing those needing backup, 3-29
  naming backups, 2-36
  overwriting backups, 2-35
  sources for duplication, 2-127
  unavailable, 2-60
  validating, 2-36
FILESPERSET parameter, BACKUP command, 2-30
flashback, 2-153
FLASHBACK DATABASE command, 2-150 to 2-154
FLASHBACK LOG parameter, RECOVER command, 3-10
flashback logs, 2-150, 3-37
flashback NOLOGGING operations, 2-151
FOR DB_UNIQUE_NAME parameter, forDbUniqueNameOption subclause, 4-20
FOR DB_UNIQUE_NAME parameter, RESTORE command, 3-46

FOR RECOVER OF COPY parameter, BACKUP command, 2-44
FOR RECOVER OF TAG parameter, BACKUP command, 2-45
FORCE parameter
BACKUP command, 2-31
DELETE command, 2-115
RESTORE command, 3-43
STARTUP command, 3-90
forDbUniqueNameOption subclause
parameter descriptions, 4-20
syntax, 4-20
syntax diagram, 2-58, 2-68, 2-115
foreignlogRecordSpecifier subclause
syntax diagram, 4-21
FOREVER parameter, RECOVER command, 3-7
FORMAT parameter
ALLOCATE CHANNEL command, 4-3
BACKUP command, 2-32
CONVERT command, 2-96
FORMAT parameter, allocOperandList subclause, 4-3
formatSpec subclause, syntax diagram, 2-92
forRecoveryOfSpec subclause
parameter descriptions, 2-44
syntax diagram, 2-27
FROM ACTIVE DATABASE parameter
DUPLICATE command, 2-133
FROM AUTOBACKUP parameter, RESTORE command, 3-48
FROM BACKUPSET parameter
RECOVER command, 3-10
RESTORE command, 3-43
FROM CONTROLFILECOPY parameter, RESYNC CATALOG command, 3-54
FROM DATAFILECOPY parameter
RECOVER command, 3-10
RESTORE command, 3-43
FROM DB_UNIQUE_NAME parameter, RESYNC CATALOG command, 3-55
FROM FILE parameter, REPLACE SCRIPT command, 3-26
FROM SCN parameter, archlogRange subclause, 4-8
FROM SEQUENCE parameter, archlogRange subclause, 4-8
FROM TAG parameter
BACKUP command, 2-40, 2-43
RECOVER command, 3-10
RESTORE command, 3-43, 3-48
FROM TIME parameter, archlogRange subclause, 4-8
FTP, moving backups, 2-23
FULL parameter, BACKUP command, 2-32

G

GLOBAL parameter
CREATE SCRIPT command, 2-106
DELETE SCRIPT command, 2-119
EXECUTE SCRIPT command, 2-146
PRINT SCRIPT command, 2-185
REPLACE SCRIPT command, 3-26
GLOBAL SCRIPT NAMES parameter, LIST command, 2-169
global scripts
executing, 2-146
importing, 2-161
listing names, 2-169
printing, 2-185
See also scripts
GRANT command, 2-156 to 2-157

H

hardware error recovery (example), 3-12
header blocks, 3-9
HEADER parameter
RECOVER command, 3-12
RESTORE command, 3-44
HOST command, 2-159

I

image copies
adding to repository, 2-52
applying incremental backups, 3-2
backing up, 2-43
creating, 2-42
deleting, 2-123
displaying information, 2-169
format, 4-3
restoring, 3-10, 3-38
retention policies, 2-77
IMMEDIATE parameter, SHUTDOWN command, 3-85
IMPORT CATALOG command, 2-161 to 2-162
IMPORT SCRIPT parameter, TRANSPORT TABLESPACE command, 3-98
import utilities, 2-90
INACCESSIBLE parameter
CONVERT command, 2-95
VALIDATE command, 3-112
INCARNATION parameter
Archived redo log file ranges, 4-8
DUPLICATE command
orphan, 2-132
LIST command, 2-168
RESET DATABASE TO INCARNATION, 3-35
SET command, 3-79
incarnations, 2-179, 3-35
INCLUDE CURRENT CONTROLFILE parameter
BACKUP command, 2-38, 2-40
VALIDATE command, 3-109
INCLUDING BACKUPS parameter
DROP DATABASE command, 2-123
UNREGISTER command, 3-103
inconsistent backups, 2-19
inconsistent shutdowns, 2-19
incremental backups, 2-20, 2-44
adding datafile copies, 2-54

bitmap limit, 2-21
report output, 3-32
sample script, 2-48
using datafile copy, 2-52
INCREMENTAL parameter
BACKUP command, 2-33
REPORT command, 3-31
index blocks, logical corruption, 3-10
initialization files
for auxiliary databases, 2-135
for database startup, 3-91
initialization parameters
BACKUP_TAPE_IO_SLAVES, 2-75
COMPATIBLE, 2-125
CONTROL_FILES, 2-125
CPU_COUNT, 3-11
DB_BLOCK_CHECKSUM, 2-34
DB_CREATE_FILE_DEST, 4-3
DB_FILE_NAME_CONVERT, 2-129, 2-133, 3-98, 4-17
DB_FLASHBACK_RETENTION_TARGET, 2-151
DB_RECOVERY_FILE_DEST, 3-45
DB_UNIQUE_NAME, 3-48
LOG_ARCHIVE_DEST_n, 3-45, 3-74
LOG_ARCHIVE_FORMAT, 3-45
LOG_FILE_NAME_CONVERT, 2-129, 3-98
RECOVERY_PARALLELISM, 3-11
replacing values, 2-136
setting for duplicate database, 2-137
input files, deleting, 2-40

J

job commands, 3-65

K

KEEP parameter
BACKUP command, 2-33
KEEP parameter, keepOption subclause, 4-26
keepOption subclause, 4-26
keywords in syntax diagrams, 1-2

L

LIKE parameter
archivelogRecordSpecifier subclause, 4-6
maintQualifier subclause, 4-32
LIKE parameter, BACKUP command, 2-43
Linux commands, 2-159
Linux-to-Windows conversion (example), 2-99
list clause
parameter descriptions, 2-166
syntax diagram, 2-165
LIST command, 2-164 to 2-180
list clause, 2-165, 2-166
listBackupOption subclause, 2-166, 2-170
listObjectSpec subclause, 2-166, 2-169
maintQualifier subclause, 2-165
recordSpec subclause, 4-37
recoverableClause subclause, 2-166, 2-169
listBackupOption subclause
parameter descriptions, 2-170
syntax diagram, 2-166
listObjectSpec subclause
parameter descriptions, 2-169
syntax diagram, 2-166
listObjList subclause, 4-29
parameter descriptions, 4-29
syntax diagram, 4-29
LOB data types, transporting, 2-90
local scripts
executing, 2-146
listing names, 2-169
printing, 2-185
See also scripts
LOCATION parameter, 3-3
log files
appending output, 3-60
creating for RMAN sessions, 3-61
directing output to, 3-87
LOG parameter, RMAN command, 3-61
log sequence number, 3-78
log sequence numbers
point-in-time recovery, 2-136
undoing changes, 2-150
log sequence, resetting, 2-17
LOG_ARCHIVE_DEST_n initialization parameter, 3-3, 3-45, 3-74
LOG_ARCHIVE_FORMAT initialization parameter, 3-45
LOG_FILE_NAME_CONVERT initialization parameter, 2-129, 3-98
LOG_FILE_NAME_CONVERT parameter, DUPLICATE command, 2-137
LOGFILE parameter
DUPLICATE command, 2-133, 2-137
logging in, 4-13
logical corruption, checking for, 3-111
LogMiner utility, 2-53, 2-55, 2-59, 2-72, 4-21
logSpec subclause
parameter descriptions, 2-137
syntax diagram, 2-131

M

maintenance channels
allocating, 2-13
naming conventions, 2-13
releasing, 3-18
maintenance commands, 3-65
maintenance job configuration, 2-65
maintenanceCommands subclause, syntax diagram, 3-65
maintQualifier subclause
parameter descriptions, 4-32
syntax diagram, 2-165, 4-32
maintSpec subclause, 4-34
syntax diagram, 2-58, 2-111, 2-114
MAXCORRUPT parameter
SET command, 3-75

MAXDAYS parameter, RESTORE command, 3-48
MAXOPENFILES parameter, allocOperandList subclause, 4-3
MAXPIECESIZE parameter, allocOperandList subclause, 4-3
MAXSEQ parameter, RESTORE command, 3-48
MAXSETSIZE parameter
  BACKUP command, 2-34
  CONFIGURE command, 2-76
  SHOW command, 3-83
media failure, restoring backups, 3-37
media libraries for sbt channels, 4-4
media managers, 2-41
  compressing data, 2-42
  controlling data transfer, 2-35
  sending commands, 3-61, 3-68
  specifying environment variables, 4-4
  status of offsite backups, 3-43
  validating metadata, 2-109
  See also Oracle Secure Backup
media pool, backup storage, 2-35
media recovery
  parallel processing, 3-11
  single processing, 3-11
message logs
  appending output, 3-60
  controlling content, 3-73
message numbers, 3-61
metadata
  changing catalog schemas, 2-161
  resynchronization, 3-53
  validating in repository, 2-109
miscellaneous commands, 3-66
missing files
  checking for, 3-107
  deleting from repository, 2-115
MOUNT parameter
  ALTER DATABASE command, 2-17
  STARTUP command, 3-90
mounting databases, 2-17
moving backups, 2-23
MSGNO parameter, RMAN command, 3-61
multisection backups, 2-35

N

NEED BACKUP parameter, REPORT command, 3-30
needBackupOption subclause
  parameter descriptions, 3-30
  syntax diagram, 3-28
nested command files, 2-4
net service names, 2-68, 4-13
NEW DATABASE parameter, CONVERT command, 2-94
new files, backing up only, 2-45
NEWNAME FOR DATAFILE parameter, SET command, 3-75, 3-76
NEWNAME FOR TEMPFILE parameter, SET command, 3-76
NEWNAME setting, persistent alternative, 2-69
NLS_DATE_FORMAT configuration setting, 2-171
NLS_DATE_FORMAT environment variable, 3-31
NLS_LANG environment variable, 3-31
NLS_LANG environment variables, 2-171
NO UNREGISTER parameter, IMPORT CATALOG command, 2-162
NOARCHIVELOG mode, 2-19, 2-128
NOCATALOG parameter, RMAN command, 3-61
NOCHECKSUM parameter, BACKUP command, 2-34
NOEXCLUDE parameter
  BACKUP command, 2-34
  VALIDATE command, 3-111
NOFILENAMECHECK parameter, DUPLICATE command, 2-134, 2-137
NOKEEP parameter, keepOption subclause, 4-27
NOMOUNT parameter, STARTUP command, 3-90
nonquoted strings, 1-3
NOPARALLEL parameter, RECOVER command, 3-11
NOPROMPT parameter
  DELETE command, 2-115
  DROP DATABASE command, 2-123
NOREDO parameter
  DUPLICATE command, 2-136
  RECOVER command, 3-11
NORMAL parameter, SHUTDOWN command, 3-86
normal resynchronization, 3-55
NOT BACKED UP parameter, BACKUP command, 2-45
notBackedUpSpec subclause
  parameter descriptions, 2-45
  syntax diagram, 2-27
numbers, displaying message, 3-61

O

OBSOLETE parameter
  DELETE command, 2-116
  REPORT command, 3-29
obsOperandList subclause, 4-36
OFF parameter, SPOOL command, 3-87
OFFLINE parameter
  CONVERT command, 2-95
  VALIDATE command, 3-112
offsite backups, 3-43
OMF format, 2-129
online redo logs
  for duplicate database, 2-137
  for RAC, 2-134
OPEN parameter, ALTER DATABASE command, 2-17
OPEN RESTRICTED parameter, DUPLICATE command, 2-134, 2-135
opening databases, 2-17
operating system commands, 2-159
operating systems
  compatibility, 2-20
  duplicating databases across, 2-125

transporting across, 2-89
optimization algorithms, 2-75
optimization, overriding backup, 2-76
ORA$RMAN_RPI_IN pipe, 3-61
ORA$RMAN_RPI_OUT pipe, 3-61
ORA-1152 error message, 2-133
ORA-1578 error message, 3-9
ORA-19504 error message, 2-134
ORA-19624 error message, 2-134
ORA-19916 error message, 2-21
ORA-27086 error message, 2-134
Oracle Advanced Compression, 2-70
Oracle home, excluded from backup, 2-20
Oracle Managed Files (OMF), 2-129, 4-17
duplicating databases (example), 2-144
Oracle Net connections, 3-55
Oracle RAC
recovering from failure, 3-21
See RAC
Oracle Secure Backup, 2-41
creating encrypted backups, 2-21
sending commands to (example), 3-68
undo data, 2-76
Oracle Secure Backup (OSB) Cloud Module
configuring RMAN channel, C-7
creating an AWS account, C-3
environment variables
ENV parameter of PARMS option, C-7
SBT_PARMS parameter, C-7
getting AWS Identifiers, C-3
prerequisites, C-2
registering for OTN account, C-3
running the S3 Backup installer
first time, C-3
Linux example, C-5
Securing backups, C-7
storing configuration in RMAN repository, C-6
Troubleshooting, C-7
Oracle wallets, 3-38, 3-96
Oracle10g compatibility, B-1
Oracle8i compatibiity, B-1
Oracle9i compatibiity, B-1
output
displaying message numbers, 3-61
storing in files, 3-61
overwriting backup files, 2-35

P

PARALLEL parameter, RECOVER command, 3-11
parallelism
allocating channels, 2-10
backup using channels (example), 2-49
channel allocation, 2-10
specifying for recovery, 3-11
validation, 3-112
PARALLELISM parameter, CONVERT command, 2-96
parallelism settings, 3-83
parameter files
backing up, 2-19, 2-39
restoring, 3-40, 3-46
See also initialization parameter files, server parameter files, 3-46
specifying path, 3-78
PARAMETER_VALUE_CONVERT parameter
DUPLICATE command, 2-136
overriding, 2-137
parameters
deprecated, A-1
for sbt channels, 4-4
in syntax diagrams, 1-2
parameters in stored scripts, 2-105
PARMS parameter
allocOperandList subclause, 4-4
SEND command, 3-68
partial backups, 2-44
password encryption, 2-71
PASSWORD FILE parameter, DUPLICATE command, 2-135
password files
excluded from backup, 2-20
for duplicate databases, 2-126
password-based encryption, 2-21
passwords, 4-13
encrypted, 2-71
encrypted backup sets, 3-38
for decryption, 3-72
RAC SYSDBA, 2-10
performance
allocating channels, 2-10
incremental backups, 2-21
PFILE parameter
DUPLICATE command, 2-135
STARTUP command, 3-91
PIPE parameter, RMAN command, 3-61
pipes
opening input and output, 3-61
timing out, 3-62
placeholders in syntax diagrams, 1-2
platform compatibility, 2-20
PLATFORM parameter, CONVERT command, 2-96
platforms
duplicating databases across, 2-125
transporting across, 2-89
PL/SQL code, converting formats, 2-125
PL/SQL stored procedures, 3-88
PLUS ARCHIVELOG parameter, VALIDATE command, 3-109
point-in-time recovery, 2-136, 3-2, 3-37, 4-9
POOL parameter, BACKUP command, 2-35
PREVIEW parameter, RESTORE command, 3-43
previewing backups, 3-37
PRIMARY CONTROLFILE parameter, RESTORE command, 3-46
primary databases
backing up, 2-19
copying, 2-124
PRINT SCRIPT command, 2-185
private catalog schema, assigning privileges, 2-156

privileges
assigning, 2-156
CATALOG, 2-102
CREATE TYPE, 3-105
REGISTER, 2-102
revoking, 3-57
SYSDBA, 2-86
processing time for backup, 2-44
proxy backups, restoring, 3-2
proxy copies
available information, 2-173
crosschecking, 2-13
definition, 2-41
deleting, 2-13, 2-123
obsolete
report output, 3-33
restoring, 3-38
tag names, 2-36
PROXY parameter
BACKUP command, 2-35
recordSpec subclause, 4-37

Q

QUIT command, 2-187
quoted strings, 1-3

R

RAC
automatic channels (example), 2-83
creating online redo logs, 2-134
crosschecking backups, 2-16
crosschecking or deleting, 2-13
spreading work, 4-2
SYSDBA passwords, 2-10
RATE parameter, allocOperandList subclause, 4-4
RC_ARCHIVED_LOG view, 5-4
RC_BACKUP_ARCHIVELOG_DETAILS view, 5-6
RC_BACKUP_ARCHIVELOG_SUMMARY view, 5-7
RC_BACKUP_CONTROLFILE view, 5-8
RC_BACKUP_CONTROLFILE_DETAILS view, 5-10
RC_BACKUP_CONTROLFILE_SUMMARY view, 5-11
RC_BACKUP_COPY_DETAILS view, 5-12
RC_BACKUP_COPY_SUMMARY view, 5-13
RC_BACKUP_CORRUPTION view, 5-14
RC_BACKUP_DATAFILE view, 5-15
RC_BACKUP_DATAFILE_DETAILS view, 5-17
RC_BACKUP_DATAFILE_SUMMARY view, 5-18
RC_BACKUP_FILES view, 5-19
RC_BACKUP_PIECE view, 5-22
RC_BACKUP_PIECE_DETAILS, 5-24
RC_BACKUP_REDOLOG view, 5-26
RC_BACKUP_SET view, 5-28
RC_BACKUP_SET_DETAILS view, 5-30
RC_BACKUP_SET_SUMMARY view, 5-32
RC_BACKUP_SPFILE view, 5-33
RC_BACKUP_SPFILE_DETAILS view, 5-34
RC_BACKUP_SPFILE_SUMMARY view, 5-35
RC_CHECKPOINT view
See RC_RESYNC view
RC_CONTROLFILE_COPY view, 5-37
RC_COPY_CORRUPTION view, 5-39
RC_DATABASE view, 3-79, 5-40
RC_DATABASE_BLOCK_CORRUPTION view, 5-41
RC_DATABASE_INCARNATION view, 3-79, 5-42
RC_DATAFILE view, 5-43
RC_DATAFILE_COPY view, 5-45
RC_LOG_HISTORY view, 5-47
RC_OFFLINE_RANGE view, 5-48
RC_PROXY_ARCHIVEDLOG view, 5-49
RC_PROXY_ARCHIVELOG_DETAILS view, 5-51
RC_PROXY_CONTROLFILE view, 5-53
RC_PROXY_COPY_DETAILS view, 5-55
RC_PROXY_COPY_SUMMARY view, 5-56
RC_PROXY_DATAFILE view, 5-57
RC_REDO_LOG view, 5-59
RC_REDO_THREAD view, 5-60
RC_RESTORE_POINT view, 5-61
RC_RESYNC view, 5-62
RC_RMAN_BACKUP_JOB_DETAILS view, 5-63
RC_RMAN_BACKUP_SUBJOB_DETAILS view, 5-65
RC_RMAN_BACKUP_TYPE view, 5-67
RC_RMAN_CONFIGURATION view, 5-68
RC_RMAN_OUTPUT view, 5-69
RC_RMAN_STATUS view, 5-70
RC_SITE view, 5-72
RC_STORED_SCRIPT view, 5-73
RC_STORED_SCRIPT_LINE view, 5-74
RC_TABLESPACE view, 5-75
RC_TEMPFILE view, 5-76
RC_UNUSABLE_BACKUPFILE_DETAILS view, 5-78
READONLY parameter
CONVERT command, 2-95
RECOVER command, 3-11
VALIDATE command, 3-112
Real Application Clusters
See RAC
recordSpec subclause, 4-37
recover clause
parameter descriptions, 3-6
syntax diagram, 3-4
RECOVER command, 3-2 to 3-12
blockObject subclause, 3-5, 3-8
dbObject subclause, 3-4, 3-7
recover clause, 3-4, 3-6
recoverObject subclause, 3-4, 3-6
recoverOptionList subclause, 3-5, 3-9
recoverSpec subclause, 3-4, 3-6
sizeSpec subclause, 3-5
RECOVER OF COPY parameter, BACKUP command, 2-44
RECOVER OF TAG parameter, BACKUP command, 2-45
RECOVERABLE parameter, LIST command, 2-170

recoverableClause subclause
parameter descriptions, 2-169
syntax diagram, 2-166
recoverObject subclause
parameter descriptions, 3-6
syntax diagram, 3-4
recoverOptionList subclause
parameter descriptions, 3-9
syntax diagram, 3-5
recoverSpec subclause
parameter descriptions, 3-6
syntax diagram, 3-4
recovery
allowing corrupt blocks, 3-9
database in NOARCHIVELOG mode (example), 3-14
incremental, 3-10
point-in-time of duplicate databases, 2-136
search order for backups, 3-3
RECOVERY AREA parameter
BACKUP command, 2-39
CATALOG command, 2-55
RESTORE command, 3-48
VALIDATE command, 3-111
recovery catalog
connecting to, 4-13
DB_UNIQUE_NAME, 3-83
granting access, 2-157
registering databases, 3-16
removing databases, 3-102
removing references, 2-60
removing repository records, 2-113
replacing scripts, 3-25
updating DB_UNIQUE_NAME, 2-57
upgrading versions, 3-105
recovery catalog databases, connecting to, 2-86
recovery catalog views, 5-1
RC_DATABASE, 3-79
RC_DATABASE_INCARNATION, 3-79
summary list, 5-1
See also specific view names
recovery catalogs
creating, 2-102
creating stored scripts, 2-105
deleting, 2-121
removing references, 2-60
resynchronization, 3-53
revoking privileges, 3-57
running RMAN without, 3-61
stored scripts, available information, 2-179
unregistering, 2-123
recovery catalogs, moving metadata, 2-161
recovery catalogs.NOCATALOG mode, 2-22
RECOVERY FILES parameter
BACKUP command, 2-39
VALIDATE command, 3-111
recovery files, backing up, 2-39
Recovery Manager
compatibility, B-1
dates in commands, 4-43
Recovery Manager client, opening, 3-59
Recovery Manager, control file autobackups, 2-78
RECOVERY WINDOW OF parameter, obsOperandList subclause, 4-36
RECOVERY WINDOW parameter
CONFIGURE command, 2-77
REPORT command, 3-31
recovery windows
report output, 3-32
recovery windows, report output, 3-32
RECOVERY_CATALOG_OWNER role, 2-102, 2-156, 3-105
RECOVERY_PARALLELISM initialization parameter, 3-11
redo logs
archiving, 2-17
backing up, 2-19
for duplicate databases, 2-133
for RAC, 2-134
omitting from recovery, 3-11
sequence numbers, 2-153
redo threads, 4-42
REDUNDANCY parameter
CONFIGURE command, 2-77
REPORT command, 3-31
REDUNDANCY parameter, obsOperandList subclause, 4-36
redundant backups, 3-33
REGISTER DATABASE command, 3-16 to 3-17
REGISTER DATABASE parameter
GRANT command, 2-157
REVOKE command, 3-57
REGISTER privilege, 2-102
RELEASE CHANNEL command, 3-18
release clause, 3-18
releaseForMaint clause, 3-18
release clause, syntax diagram, 3-18
releaseForMaint clause
syntax diagram, 3-18
repair clause
parameter description, 3-22
syntax diagram, 3-22
REPAIR FAILURE command, 3-21 to 3-22
repair clause, 3-22
repair options, 2-6, 2-7
repair status, 2-180
repair strategies, 2-6
REPLACE SCRIPT command, 3-25 to 3-26
replaceScript clause, 3-25
replaceScript clause
syntax diagram, 3-25
report clause
parameter descriptions, 3-29
syntax diagram, 3-28
REPORT command, 3-28 to 3-33
atClause subclause, 3-29, 3-31
needBackupOption subclause, 3-28, 3-30
report clause, 3-28, 3-29
reportObject subclause, 3-29, 3-31
reportObject subclause

parameter descriptions, 3-31
syntax diagram, 3-29
repositories
analyzing, 3-28
listing contents, 2-164
updating, 2-113
updating availability status, 2-57
updating filenames, 3-92
validating contents, 2-109
reserved words, 1-4 to 1-5
reset clause, syntax diagram, 3-35
RESET DATABASE command, 3-35
reset clause, 3-35
RESET DB_UNIQUE_NAME parameter, CHANGE command, 2-62
resetDbUniqueNameOption subclause
parameter descriptions, 2-61
syntax diagram, 2-58
RESETLOGS parameter, FLASHBACK DATABASE command, 2-153
restoration configuration, 2-65
restore clause
parameter descriptions, 3-42
syntax diagram, 3-41
RESTORE command, 3-37 to 3-49
autoBackupOptList subclause, 3-42, 3-48
restore clause, 3-41, 3-42
restoreObject subclause, 3-41, 3-44
restoreSpecOperand subclause, 3-42
restore commands, 3-66
restore failover, 3-38
RESTORE POINT parameter
DUPLICATE command, 2-136
FLASHBACK DATABASE command, 2-154
LIST command, 2-168
RECOVER command, 3-7
RESTORE command, 3-44
SET command, 3-77
TRANSPORT TABLESPACE command, 3-98
restore points, 2-41, 2-179
restoreCommands subclause, syntax diagram, 3-66
restoreObject subclause
parameter descriptions, 3-44
syntax diagram, 3-41
restoreSpecOperand subclause, syntax diagram, 3-42
RESTRICTED SESSION privilege, 3-90
RESYNC CATALOG command, 3-53 to 3-55
RESYNC CATALOG, resync clause, 3-54
resync clause, syntax diagram, 3-54
resynchronization, 3-55
retention policies, 4-26
exemptions (example), 2-50
overriding, 2-33, 2-59
window-based, 3-31
RETENTION POLICY parameter
CONFIGURE command, 2-77
SHOW command, 3-83
REUSE parameter, BACKUP command, 2-35
reverse resynchronization, 3-55
revoke clause, syntax diagram, 3-57
REVOKE command, 3-57 to 3-58
revoke clause, 3-57
RMAN command, 3-59
cmdLine clause, 3-60
RMAN command (operating system), 3-59 to 3-62
RMAN commands
See specific command names
executing from files, 3-61
RMAN-0558 error message, 3-60
RMAN-06004 error message, 2-37, 2-146, 2-185
RMAN-06445 error message, 2-87, 2-88
RMAN-06496 error message, 3-40
RMAN-06710 error message, 2-119
RMAN-10031 error message, 2-134
RMAN-10035 error message, 2-134
RMAN-20401 error message, 2-105
RMAN-6758 error message, 3-46
roles
RECOVER_CATALOG_OWNER, 2-102
RECOVERY_CATALOG_OWNER, 2-156, 3-105
rollback segments, resynchronization, 3-53
run clause, syntax diagram, 3-65
RUN command, 3-65 to 3-66
backupCommands subclause, 3-65
maintenanceCommands subclause, 3-65
restoreCommands subclause, 3-66
run clause, 3-65

S

saving scripts, 2-185
sbt channels, specifying parameters, 4-4
SCHEMA parameter, REPORT command, 3-30
SCN
for restore, 3-78
setting restore point, 3-77
See also system change numbers
SCN BETWEEN parameter, archlogRange subclause, 4-8
SCN parameter
FLASHBACK DATABASE command, 2-153
REPORT command, 3-31
scopes, creating for scripts, 3-65
SCRIPT NAMES parameter, LIST command, 2-169
SCRIPT parameter
DELETE SCRIPT command, 2-119
EXECUTE SCRIPT command, 2-146
REPLACE SCRIPT command, 3-26
RMAN command, 3-62
scripts
available information, 2-179
creating from files, 2-106
creating in recovery catalog, 2-105
listing names, 2-169
printing, 2-185
replacing, 3-25
See command files
substitution variables, 2-105
SECTION SIZE parameter

BACKUP command, 2-35
VALIDATE command, 3-112
Secure Backup, 2-21
SEND command, 3-68
SEND parameter
RMAN command, 3-61
SEND parameter, allocOperandList subclause, 4-4
SEQUENCE BETWEEN parameter, archlogRange subclause, 4-8
SEQUENCE parameter
FLASHBACK DATABASE command, 2-153, 2-154
REPORT command, 3-31
SEQUENCE parameter, archivelogRecordSpecifier subclause, 4-8
server parameter files
backing up, 2-19, 2-39
for auxiliary databases, 2-135
restoring, 3-40, 3-46
restoring lost, 3-90
validating, 3-111
validation report, 3-113
sessions
ending RMAN, 2-149
identifying channels, 3-78
set clause, syntax diagram, 3-70
SET command, 3-70 to 3-79
encryption settings, 2-22
set clause, 3-70
setRmanOption subclause, 3-70, 3-71
setRmanOrRunOption subclause, 3-71, 3-78
setRunOption subclause, 3-71, 3-74
SET MAXCORRUPT command, 2-50
SET NEWNAME command, 3-39
SET parameter, DUPLICATE command, 2-137
setParameter subclause
parameter descriptions, 2-136
syntax diagram, 2-131
setRmanOption subclause
parameter descriptions, 3-71
syntax diagram, 3-70
setRmanOrRunOption subclause
parameter descriptions, 3-78
syntax diagram, 3-71
setRunOption subclause
parameter descriptions, 3-74
syntax diagram, 3-71
shared servers, allocating channels, 2-13
shell commands, 2-159
show clause, syntax diagram, 3-82
SHOW command, 3-82 to 3-84
show clause, 3-82
SHUTDOWN command, 3-85 to 3-86
shutdowns, 2-19
single quotes, 1-3
SITE_KEY column null values, 2-22
sizeSpec subclause
parameter descriptions, 2-46, 4-4
syntax diagram, 2-27, 2-67, 2-131, 3-5, 3-109, 4-2
SKIP keyword, CONVERT command, 2-95
SKIP parameter
BACKUP command, 2-46
RECOVER command, 3-7
RESTORE command, 3-45
VALIDATE command, 3-112
SKIP READONLY parameter
DUPLICATE command, 2-135
RECOVER command, 3-11
RESTORE command, 3-44
SKIP TABLESPACE parameter, DUPLICATE command, 2-135
SKIP TABLESPACE parameter, listObjList subclause, 4-29
skipSpec subclause
parameter descriptions, 2-46, 2-95, 3-112
syntax diagram, 2-27, 2-91, 3-109
SNAPSHOT CONTROLFILE NAME parameter, CONFIGURE command, 2-71
SNAPSHOT CONTROLFILE parameter, SHOW command, 3-83
snapshots
control file, 2-40
Volume Shadow Copy Service (VSS), 2-33
source databases
copying, 2-124
used for duplication, 2-127
SPFILE parameter
BACKUP command, 2-39
DUPLICATE command, 2-135
RESTORE command, 3-46
VALIDATE command, 3-111
listObjList subclause, 4-29
SPFILE SET, overriding, 2-137
SPOOL command, 3-87
SQL command, 3-88
standard output
printing scripts, 2-185
redirecting to files, 3-61
STANDBY CONTROLFILE parameter, RESTORE command, 3-47
standby databases
backing up, 2-19
copying, 2-124
fast recovery areas, 2-127
password files, 2-126
updating (example), 2-49
STANDBY parameter, DUPLICATE command, 2-132
STARTUP command, 3-90 to 3-91
stdout, redirecting, 3-87
stored scripts
available information, 2-179
deleting, 2-119
executing, 2-146, 3-62
printing, 2-185
See also scripts
strings
valid characters in RMAN commands, 1-3
striped files, allocating channels, 2-10
striping backup sets, 2-41

subclause summary, 1-8
subclauses
See specific subclause name
substitution variables, 2-79
complete list, 4-24
example of %D, 2-47
example of %F, 2-47
example of %U, 2-47
in scripts, 2-3, 2-105, 3-25
passing to stored scripts, 2-146
providing values, 3-62
SUMMARY parameter, LIST command, 2-170
switch clause
parameter descriptions, 3-93
syntax diagram, 3-92
SWITCH command, 3-92 to 3-94
effect on RESTORE, 3-39
switch clause, 3-92, 3-93
switchFile subclause, 3-93
switchFile subclause
parameter descriptions, 3-93
syntax diagram, 3-93
syntax conventions, 1-2
syntax diagrams
keywords, 1-2
parameters, 1-2
placeholders, 1-2
syntax errors, checking for, 3-60
syntax, deprecated, A-1
SYSAUX tablespaces, 3-96
SYSDBA passwords
for active duplication, 2-126
RAC environment, 2-10
SYSDBA privilege, 2-86, 4-13
system change numbers
See also SCN
system change numbers (SCN), 2-136, 2-150, 2-153

T

TABLESPACE DBA parameter, RECOVER command, 3-9
TABLESPACE DESTINATION parameter, TRANSPORT TABLESPACE command, 3-98
TABLESPACE parameter
BACKUP command, 2-39
CONVERT command, 2-94
DUPLICATE command, 2-136
RECOVER command, 3-7, 3-8
REPORT command, 3-31
RESTORE command, 3-47
SWITCH command, 3-93
VALIDATE command, 3-111
tablespace point-in-time recovery (TSPITR), 3-2, 3-12
tablespace recovery (example), 3-12
tablespaces
backing up, 2-19
converting platform format, 2-89
duplicating, 2-135
encryption settings, 3-83
excluding from database backup, 2-76
excluding from duplicate database, 2-135
including in duplicate database, 2-136
listing datafiles, 3-31
point-in-time lists, 3-30
recovering individual, 3-8
recovering undo segments, 3-12
renamed, 2-39
show excluded, 3-83
transparent encryption, 2-71
updating filenames, 3-92
validating individual, 3-111
See also transportable tablespaces
TAG parameter
BACKUP command, 2-36
RECOVER command, 3-6, 3-10
RESTORE command, 3-43
RESTORE commands, 3-48
maintQualifier subclause, 4-33
tape backups, allocating channels, 2-10
tape device type setting, 2-81
target control files, 2-113
TARGET DATABASE parameter, DUPLICATE command, 2-131
target databases
connecting to, 2-86, 4-13
deleting, 2-123
maintaining single metadata catalog, 2-161
registering, 3-16
starting from RMAN, 3-90
TARGET parameter
CONNECT command, 2-87
RMAN command, 3-62
target times, undoing changes, 2-150
tee command (Linux), 3-61
TEMPFILE parameter, SWITCH command, 3-93
tempfiles
re-creating, 2-17, 2-129
renaming, 3-76
tempfileSpec subclause, 4-39
temporary tablespaces, 2-39
TEST parameter, RECOVER command, 3-11
THREAD parameter
FLASHBACK DATABASE command, 2-154
REPORT command, 3-31
THREAD parameter, archlogRange subclause, 4-8
TIME BETWEEN parameter, archlogRange clause, 4-8
time constraints
partial backups, 2-44
recovery period, 2-77
time limits, 4-42
TIME parameter
FLASHBACK DATABASE command, 2-153, 2-154
REPORT command, 3-31
time ranges, 4-11
time, backup completion, 4-11
time-based backups, 2-46
TIMEOUT parameter, RMAN command, 3-62

TO BEFORE RESETLOGS parameter, FLASHBACK DATABASE command, 2-153
TO BEFORE SCN parameter, FLASHBACK DATABASE command, 2-153
TO BEFORE SEQUENCE parameter, FLASHBACK DATABASE command, 2-153
TO BEFORE TIME parameter, FLASHBACK DATABASE command, 2-153
TO FILE parameter, PRINT SCRIPT command, 2-185
TO parameter, SPOOL command, 3-87
TO RESTORE POINT parameter
  DUPLICATE command, 2-136
  FLASHBACK DATABASE command, 2-154
  RECOVER command, 3-7
  RESTORE command, 3-44
  SET command, 3-77
  TRANSPORT TABLESPACE command, 3-98
TO SEQUENCE parameter, FLASHBACK DATABASE command, 2-154
TO TIME parameter, FLASHBACK DATABASE command, 2-154
TRANSACTIONAL parameter
  SHUTDOWN command, 3-86
transferring backups, 2-23
transparent encryption, 2-21, 2-125
TRANSPORT SCRIPT parameter, CONVERT command, 2-95
TRANSPORT TABLESPACE command, 3-96 to 3-99
  transpt_tbs clause, 3-97, 3-98
  transpt_tbs_optlist subclause, 3-97, 3-98
transportable tablespaces
  backing up read-only, 2-39
  creating from backups, 3-96
  duplicating, 2-125
transportOptionList subclause
  parameter descriptions, 2-94
  syntax diagram, 2-91
transpt_tbs clause
  parameter descriptions, 3-98
  syntax diagram, 3-97
transpt_tbs_optlist subclause
  parameter descriptions, 3-98
  syntax diagram, 3-97
trial recovery, 3-11

U

UNAVAILABLE parameter, CHANGE command, 2-60
UNAVAILABLE status, DELETE command, 2-114
UNCATALOG parameter, CHANGE command, 2-60
undo data, 2-76
undo segments, 2-89
UNDO TABLESPACE parameter, RECOVER command, 3-12
UNIX commands, 2-159
unrecoverable operations
  report output, 3-33
UNRECOVERABLE parameter, REPORT command, 3-30
UNREGISTER DATABASE command, 3-102 to 3-103
UNTIL SCN parameter
  archlogRange subclause, 4-8
  untilClause subclause, 4-42
UNTIL SEQUENCE parameter
  archlogRange subclause, 4-8
  untilClause subclause, 4-42
UNTIL TIME parameter
  archlogRange subclause, 4-8
  untilClause subclause, 4-43
untilClause subclause, 4-42
unused block compression, 2-41
UPGRADE CATALOG command, 3-105
USE_DB_RECOVERY_FILE_DEST parameter, 3-3
user names, 4-13
user privileges, 2-86
user roles, 2-102
user-managed backups, 2-52
USING ADVISE OPTION parameter, REPAIR FAILURE command, 3-22
USING parameter, RMAN command, 3-62
UTF character set, CLOB storage, 2-90
utlirp script, converting PL/SQL code, 2-125

V

V$ARCHIVED_LOG view, 2-178, 4-7, 4-8
V$BACKUP_DEVICE view, 2-11, 2-14
V$BACKUP_PIECE view, 2-171, 2-173, 2-175
V$BACKUP_SET view, 2-171, 2-174, 2-175, 2-176
V$DATABASE view, 3-79
V$DATABASE_BLOCK_CORRUPTION view, 2-36, 3-2, 3-8, 3-9, 3-10, 3-14, 3-42, 3-75, 3-107, 3-111
V$DATABASE_INCARNATION view, 3-3
V$DATABASE.CURRENT_SCN view, 2-151
V$DATABASE.FLASHBACK_ON view, 2-150
V$DATAFILE view, 4-15, 4-39
V$DATAFILE_COPY view, 2-177, 2-178, 4-15
V$DATAFILE_HEADER view, 4-15
V$FLASHBACK_DATABASE_LOG view, 2-153 to 2-154
V$PROXY_DATAFILE view, 2-173
V$RMAN_COMPRESSION_ALGORITHM view, 2-70, 3-72
V$RMAN_ENCRYPTION_ALGORITHMS view, 2-21, 2-71, 3-73
V$RMAN_OUTPUT view, 2-20, 3-61
V$RMAN_STATUS view, 3-61
V$SESSION view, 3-78
V$TABLESPACE view, 2-152
V$TRANSPORTABLE_PLATFORM view, 2-89
validate clause
  parameter descriptions, 3-109
  syntax diagram, 3-107
VALIDATE command, 3-107 to 3-113
  blockObject subclause, 3-108
  copyOfSpec subclause, 3-108

sizeSpec subclause, 3-109
skipSpec subclause, 3-109, 3-112
validate clause, 3-107, 3-109
validateObject subclause, 3-108, 3-110
validateOperand subclause, 3-109, 3-111
VALIDATE HEADER parameter, RECOVER command, 3-12
VALIDATE parameter
BACKUP command, 2-36
RESTORE command, 3-44
validateObject subclause
parameter descriptions, 3-110
syntax diagram, 3-108
validateOperand subclause
parameter descriptions, 3-111
syntax diagram, 3-109
validating backups, 3-37
variables in command files, 2-3
views, recovery catalog, 5-1
VIRTUAL parameter, CREATE CATALOG command, 2-103
virtual private catalog schema, assigning privileges, 2-156
virtual private catalogs
access to global scripts, 2-119
creating, 2-103
deleting, 2-121
virtual recovery catalog
See virtual private catalog
Volume Shadow Copy Service (VSS) snapshots, 2-33

W

wallet-based encryption, 2-21
wallets, 2-125, 3-96
windows, 4-42
recovery, 2-44, 2-77
Windows commands, 2-159
WITH TAG parameter
RECOVER command, 3-6

X

XMLType datatype in transportable tablespaces, 3-96