

Oracle® Database

Backup and Recovery Reference

11g Release 1 (11.1)

B28273-01

July 2007

This book provides complete reference information on the Recovery Manager client, including command syntax, a compatibility matrix, and recovery catalog views.

Oracle Database Backup and Recovery Reference, 11g Release 1 (11.1)

B28273-01

Copyright © 2004, 2007, Oracle. All rights reserved.

Primary Author: Lance Ashdown

Contributing Author: Antonio Romero

Contributors: Tammy Bednar, Anand Beldalker, Timothy Chien, Mark Dilman, Senad Dizdar, Raymond Guzman, Wei Hu, Alex Hwang, Ashok Joshi, J. William Lee, Reem Munakash, Muthu Olagappan, Cris Pedregal-Martin, Samitha Samaranayake, Francisco Sanchez, Vivian Schupmann, Mike Stewart, Steven Wertheimer, Wanli Yang

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States 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, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Preface	vii
Audience.....	vii
Documentation Accessibility.....	vii
Related Documentation.....	viii
Conventions.....	viii
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	
@.....	2-2
@@.....	2-4
ADVISE FAILURE	2-6
ALLOCATE CHANNEL	2-10
ALLOCATE CHANNEL FOR MAINTENANCE	2-13
ALTER DATABASE	2-17
BACKUP	2-19
CATALOG	2-51
CHANGE	2-56
CONFIGURE	2-64
CONNECT	2-83
CONVERT	2-86
CREATE CATALOG	2-98
CREATE SCRIPT	2-101
CROSSCHECK	2-104
DELETE	2-108
DELETE SCRIPT	2-114

DROP CATALOG	2-116
DROP DATABASE.....	2-118
DUPLICATE	2-119
EXECUTE SCRIPT.....	2-136
EXIT	2-139
FLASHBACK DATABASE	2-140
GRANT.....	2-146
HOST	2-149
IMPORT CATALOG	2-151
LIST	2-154
PRINT SCRIPT	2-175
QUIT	2-177
RECOVER.....	2-178
REGISTER DATABASE	2-191
RELEASE CHANNEL.....	2-193
REPAIR FAILURE	2-195
REPLACE SCRIPT.....	2-199
REPORT	2-202
RESET DATABASE.....	2-209
RESTORE.....	2-211
RESYNC CATALOG	2-226
REVOKE.....	2-230
RMAN.....	2-232
RUN.....	2-238
SEND	2-241
SET	2-243
SHOW	2-252
SHUTDOWN	2-255
SPOOL.....	2-257
SQL.....	2-258
STARTUP	2-260
SWITCH.....	2-262
TRANSPORT TABLESPACE	2-266
UNREGISTER.....	2-271
UPGRADE CATALOG	2-274
VALIDATE	2-276

3 RMAN Subclauses

allocOperandList	3-2
archivelogRecordSpecifier	3-6
completedTimeSpec.....	3-10
connectStringSpec	3-12
datafileSpec	3-14
deviceSpecifier.....	3-15
fileNameConversionSpec	3-16
forDbUniqueNameOption	3-19
foreignlogRecordSpecifier	3-20

formatSpec.....	3-22
keepOption.....	3-25
listObjList.....	3-28
maintQualifier.....	3-31
maintSpec.....	3-33
obsOperandList.....	3-35
recordSpec.....	3-36
tempfileSpec.....	3-38
untilClause.....	3-39

4 Recovery Catalog Views

Summary of RMAN Recovery Catalog Views.....	4-1
RC_ARCHIVED_LOG.....	4-4
RC_BACKUP_ARCHIVELOG_DETAILS.....	4-6
RC_BACKUP_ARCHIVELOG_SUMMARY.....	4-7
RC_BACKUP_CONTROLFILE.....	4-8
RC_BACKUP_CONTROLFILE_DETAILS.....	4-10
RC_BACKUP_CONTROLFILE_SUMMARY.....	4-11
RC_BACKUP_COPY_DETAILS.....	4-12
RC_BACKUP_COPY_SUMMARY.....	4-13
RC_BACKUP_CORRUPTION.....	4-14
RC_BACKUP_DATAFILE.....	4-15
RC_BACKUP_DATAFILE_DETAILS.....	4-17
RC_BACKUP_DATAFILE_SUMMARY.....	4-18
RC_BACKUP_FILES.....	4-19
RC_BACKUP_PIECE.....	4-22
RC_BACKUP_PIECE_DETAILS.....	4-24
RC_BACKUP_REDOLOG.....	4-26
RC_BACKUP_SET.....	4-28
RC_BACKUP_SET_DETAILS.....	4-30
RC_BACKUP_SET_SUMMARY.....	4-32
RC_BACKUP_SPFILE.....	4-33
RC_BACKUP_SPFILE_DETAILS.....	4-34
RC_BACKUP_SPFILE_SUMMARY.....	4-35
RC_CHECKPOINT.....	4-36
RC_CONTROLFILE_COPY.....	4-37
RC_COPY_CORRUPTION.....	4-39
RC_DATABASE.....	4-40
RC_DATABASE_BLOCK_CORRUPTION.....	4-41
RC_DATABASE_INCARNATION.....	4-42
RC_DATAFILE.....	4-43
RC_DATAFILE_COPY.....	4-45
RC_LOG_HISTORY.....	4-47
RC_OFFLINE_RANGE.....	4-48
RC_PROXY_ARCHIVEDLOG.....	4-49
RC_PROXY_ARCHIVELOG_DETAILS.....	4-51
RC_PROXY_ARCHIVELOG_SUMMARY.....	4-52

RC_PROXY_CONTROLFILE.....	4-53
RC_PROXY_COPY_DETAILS	4-55
RC_PROXY_COPY_SUMMARY.....	4-56
RC_PROXY_DATAFILE.....	4-57
RC_REDO_LOG	4-59
RC_REDO_THREAD	4-60
RC_RESTORE_POINT.....	4-61
RC_RESYNC	4-62
RC_RMAN_BACKUP_JOB_DETAILS	4-63
RC_RMAN_BACKUP_SUBJOB_DETAILS	4-65
RC_RMAN_BACKUP_TYPE	4-67
RC_RMAN_CONFIGURATION	4-68
RC_RMAN_OUTPUT	4-69
RC_RMAN_STATUS	4-70
RC_SITE.....	4-72
RC_STORED_SCRIPT	4-73
RC_STORED_SCRIPT_LINE.....	4-74
RC_TABLESPACE	4-75
RC_TEMPFILE.....	4-76
RC_UNUSABLE_BACKUPFILE_DETAILS.....	4-78

A Deprecated RMAN Commands

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

Index

Preface

This preface contains these topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documentation](#)
- [Conventions](#)

Audience

Oracle Database 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 need to know the following:

- Relational database concepts and basic database administration as described in *Oracle Database Concepts* and the *Oracle Database Administrator's Guide*
- 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, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility/>

Accessibility of Code Examples in Documentation

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

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

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

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"](#) 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, parameter, 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 viii.

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

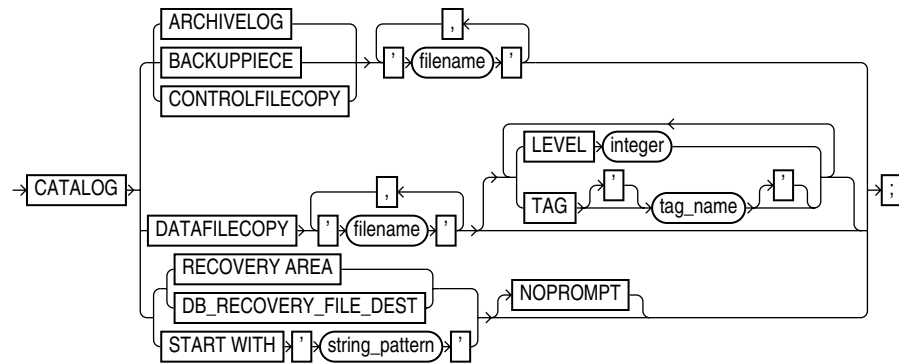
Optional sections following the examples provide more information on how and when to use the statement.

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 datatype 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, legal 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 <code>'filename'</code> , <code>'tablespace_name'</code> , <code>'channel_name'</code> , <code>'channel_parms'</code>	A string of characters contained in either single or double quotes. A quoted string may contain white space, punctuation, and RMAN and SQL keywords.	<code>"?/dbs/cf.f"</code> <code>'dev1'</code>
Nonquoted strings such as <code>channel_id</code> , <code>tag_name</code> , <code>date_string</code>	A sequence of characters containing no white space and no punctuation characters and starting with an alphabetic character.	<code>ch1</code>
<code>integer</code>	Any sequence of only number characters.	<code>67843</code>

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 legal in RMAN syntax. For example, you may specify either `'filename'` or `"filename"`. For the SQL command, it is recommended that you use double quotes because the SQL statement itself may also contain a quote, and the most common type of quote in a SQL statement is a single quote. Single and double quotes do not mean the same in SQL as they do in RMAN.

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"](#), 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. [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	PARALLELMEDIAR ESTORE	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	MAXPIECEWISE	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
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

Table 1–1 (Cont.) RMAN Reserved Words

A-C	C-D	D-L	L-P	P-S	S-W
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	

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

```

In general, avoid using reserved words in ways that conflict with their primary meaning in the RMAN command language.

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 Commands"). For command-line options for the RMAN client, refer to RMAN on page 2-232.

Table 1–2 Recovery Manager Commands

Command	Purpose
@ on page 2-2	Run a command file.
@@ 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 2-51	Add information about file copies and user-managed backups to the repository.
CHANGE on page 2-56	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 2-64	Configure persistent RMAN settings. These settings apply to all RMAN sessions until explicitly changed or disabled.
CONNECT on page 2-83	Establish a connection between RMAN and a target, auxiliary, or recovery catalog database.
CONVERT on page 2-86	Convert datafile formats for transporting tablespaces and databases across platforms.
CREATE CATALOG on page 2-98	Create the schema for the recovery catalog.
CREATE SCRIPT on page 2-101	Create a stored script and store it in the recovery catalog.
CROSSCHECK on page 2-104	Determine whether files managed by RMAN, such as archived logs, datafile copies, and backup pieces, still exist on disk or tape.
DELETE on page 2-108	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 2-114	Delete a stored script from the recovery catalog.
DROP CATALOG on page 2-116	Remove the schema from the recovery catalog.
DROP DATABASE on page 2-118	Delete the target database from disk and unregisters it.
DUPLICATE on page 2-119	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 2-136	Run an RMAN stored script.
EXIT on page 2-139	Quit the RMAN executable.

Table 1–2 (Cont.) Recovery Manager Commands

Command	Purpose
FLASHBACK DATABASE on page 2-140	Returns the database to its state at a previous time or SCN.
GRANT on page 2-146	Grants privileges to a recovery catalog user.
HOST on page 2-149	Invoke an operating system command-line subshell from within RMAN or run a specific operating system command.
IMPORT CATALOG on page 2-151	Imports the metadata from one recovery catalog into a different recovery catalog.
LIST on page 2-154	Produce a detailed listing of backup sets or copies.
PRINT SCRIPT on page 2-175	Display a stored script.
QUIT on page 2-177	Exit the RMAN executable.
RECOVER on page 2-178	Apply redo logs and incremental backups to datafiles or data blocks restored from backup or datafile copies, in order to update them to a specified time.
REGISTER DATABASE on page 2-191	Register the target database in the recovery catalog.
RELEASE CHANNEL on page 2-193	Release a channel that was allocated with an ALLOCATE CHANNEL command or ALLOCATE CHANNEL FOR MAINTENANCE command.
REPAIR FAILURE on page 2-195	Repair one or more failures recorded in the automated diagnostic repository.
REPLACE SCRIPT on page 2-199	Replace an existing script stored in the recovery catalog. If the script does not exist, then REPLACE SCRIPT creates it.
REPORT on page 2-202	Perform detailed analyses of the content of the recovery catalog.
RESET DATABASE on page 2-209	Inform RMAN that the SQL statement <code>ALTER DATABASE OPEN RESETLOGS</code> 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 2-211	Restore files from backup sets or from disk copies to the default or a new location.
RESYNC CATALOG on page 2-226	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 2-230	Revoke privileges from a recovery catalog user.
RMAN on page 2-232	Start RMAN from the operating system command line.
RUN on page 2-238	Execute a sequence of one or more RMAN commands, which are one or more statements executed within the braces of <code>RUN</code> .
SEND on page 2-241	Send a vendor-specific quoted string to one or more specific channels.
SET on page 2-243	Set the value of various attributes that affect RMAN behavior for the duration of a <code>RUN</code> block or a session.
SHOW on page 2-252	Display the current <code>CONFIGURE</code> settings.
SHUTDOWN on page 2-255	Shut down the target database. This command is equivalent to the <code>SQL*Plus SHUTDOWN</code> command.
SPOOL on page 2-257	Write RMAN output to a log file.
SQL on page 2-258	Execute a SQL statement from within Recovery Manager.
STARTUP on page 2-260	Start up the target database. This command is equivalent to the <code>SQL*Plus STARTUP</code> command.

Table 1–2 (Cont.) Recovery Manager Commands

Command	Purpose
SWITCH on page 2-262	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 <code>ALTER DATABASE RENAME FILE</code> as it applies to datafiles.
TRANSPORT TABLESPACE on page 2-266	Create transportable tablespace sets from backup for one or more tablespaces.
UNREGISTER on page 2-271	Unregister a database from the recovery catalog.
UPGRADE CATALOG on page 2-274	Upgrade the recovery catalog schema from an older version to the version required by the RMAN executable.
VALIDATE on page 2-276	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 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 3, "RMAN Subclauses"](#). [Table 1–3](#) summarizes the RMAN subclauses.

Table 1–3 Recovery Manager Subclauses

Subclause	Specifies . . .
allocOperandList on page 3-2	Channel control options such as <code>PARMS</code> and <code>FORMAT</code> .
archiveLogRecordSpecifier on page 3-6	A range of archived redo log files.
completedTimeSpec on page 3-10	A time range during which the backup or copy completed.
connectStringSpec on page 3-12	The username, password, and net service name for connecting to a target, recovery catalog, or auxiliary database. The connection is necessary to authenticate the user and identify the database.
datafileSpec on page 3-14	A datafile by filename or absolute file number.
deviceSpecifier on page 3-15	The type of storage device for a backup or copy.
fileNameConversionSpec on page 3-16	Patterns to transform source to target filenames during <code>BACKUP AS COPY</code> , <code>CONVERT</code> and <code>DUPLICATE</code> .
forDbUniqueNameOption on page 3-19	All databases in a Data Guard environment or a database with the specified <code>DB_UNIQUE_NAME</code> .
foreignLogRecordSpecifier on page 3-20	A range of foreign archived redo log files.
formatSpec on page 3-22	A filename format for a backup or copy.
keepOption on page 3-25	A backup or copy should or should not be exempt from the current retention policy.
listObjList on page 3-28	Items that will be displayed by the <code>LIST</code> command.
maintQualifier on page 3-31	Additional options for maintenance commands such as <code>DELETE</code> and <code>CHANGE</code> .
maintSpec on page 3-33	Files operated on by maintenance commands such as <code>CHANGE</code> , <code>CROSSCHECK</code> , and <code>DELETE</code> .
obsOperandList on page 3-35	Backups that are obsolete according to specified criteria.

Table 1–3 (Cont.) Recovery Manager Subclauses

Subclause	Specifies . . .
<i>recordSpec</i> on page 3-36	Objects that the maintenance commands should operate on.
<i>tempfileSpec</i> on page 3-38	A tempfile by path or by file number.
<i>untilClause</i> on page 3-39	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.

RMAN Commands

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.

@

@

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 2-232 to learn more about using substitution variables in RMAN

Syntax

at::=

→ @ filename →

Semantics

Syntax Element	Description
<i>filename</i>	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 legal. 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 a 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.cmd` with the following contents:

```
# name: whole_db.cmd
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 SYS/password@prod1
RMAN> @/tmp/whole_db.cmd Q106 2
```

@@

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 @ 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 @/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

atat::=

→ @@ → filename →

Semantics

Syntax Element	Description
<i>filename</i>	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. 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 SYS/password@prod1
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, which 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. 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

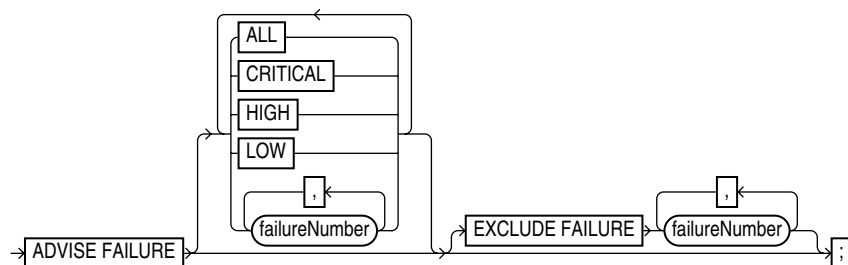
Each automated repair option is either a single repair or a set of repair steps (see [Table 2-1, "Automated Repair Options"](#) for a description of command output). When a repair option has a script that contains multiple repair steps, `ADVISE FAILURE` generates the script so that the repair steps are 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 then 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	Excludes the specified failures from the list.
failureNumber	

ADVISE FAILURE Command Output

The ADVISE FAILURE output includes the LIST FAILURE output, which is described in [Table 2–26, "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	<p>A strategy to fix the failure with the <code>REPAIR FAILURE</code> command.</p> <p>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:</p> <ul style="list-style-type: none"> ■ Repair with no data loss ■ Repair with data loss, for example, Flashback Database <p>Note: The <code>ADVISE</code> command maps a set of failures to a 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.</p>
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

Use the `ALLOCATE CHANNEL` command to manually allocate a **channel**, which is a connection between RMAN and a database instance. `ALLOCATE CHANNEL` must be issued within a `REPAIR FAILURE` block and applies only to the block in which it is issued.

Prerequisites

The target instance must be started.

Usage Notes

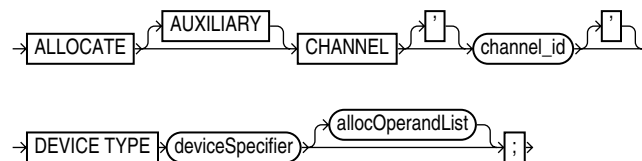
Manually allocated channels are distinct from automatically allocated channels specified with `CONFIGURE`. Automatic channels apply to any RMAN job in which you do *not* manually allocate channels. You can override automatic channel configurations by manually allocating channels within a `RUN` command,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

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 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).

Syntax

allocate::=



(*deviceSpecifier::=* on page 3-15, *allocOperandList::=* on page 3-2)

Semantics

Syntax Element	Description
AUXILIARY	<p>Specifies a connection between RMAN and an auxiliary database instance.</p> <p>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.</p> <p>See Also: DUPLICATE on page 2-119 to learn how to duplicate a database, and CONFIGURE on page 2-83 to learn how to connect to a duplicate database</p>
CHANNEL ' <i>channel_id</i> '	<p>Specifies a connection between RMAN and the target database instance. The <i>channel_id</i> is the case-sensitive name of the channel. The database uses the <i>channel_id</i> to report I/O errors.</p> <p>Each connection initiates a 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.</p> <p>Whether <code>ALLOCATE CHANNEL</code> 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.</p> <p>Each channel operates on one backup set or image copy at a time. RMAN automatically releases the channel at the end of the job.</p> <p>Note: You cannot prefix <code>ORA_</code> to a channel name. RMAN reserves channel names beginning with the <code>ORA_</code> prefix for its own use.</p>
DEVICE TYPE <i>deviceSpecifier</i>	<p>Specifies the type of storage for a backup. Query the <code>V\$BACKUP_DEVICE</code> view for information about available device types and names.</p> <p>Note: When you specify <code>DEVICE TYPE DISK</code>, no operating system resources are allocated other than for the creation of the server session.</p> <p>See Also: deviceSpecifier on page 3-15</p>
<i>allocOperandList</i>	<p>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).</p> <p>See Also: allocOperandList on page 3-2</p>

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. 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/';
}

```

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 or proxy copies, 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 or proxy copy exists only on one node (see [Example 2-12](#) on page 2-15)

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::=

```
→ ALLOCATE CHANNEL FOR MAINTENANCE }
```

```
→ [DEVICE TYPE] {deviceSpecifier} {allocOperandList} ;
```

(*deviceSpecifier::=* on page 3-15, *allocOperandList::=* on page 3-2)

Semantics

allocateForMaint

Syntax Element	Description
DEVICE TYPE <i>deviceSpecifier</i>	Specifies the type of storage for a backup. Query the V\$BACKUP_DEVICE view for information about available device types and names. See Also: <i>deviceSpecifier</i> on page 3-15
<i>allocOperandList</i>	Specifies control options for the allocated channel. Note that the channel parameters for sequential I/O devices are platform-specific. See Also: <i>allocOperandList</i> on page 3-2

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 logs 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

Oracle recommends that all nodes in an Oracle RAC configuration have the same access to all backups on all storage devices, but this is not a requirement. In this example, you 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.

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK
```

ALLOCATE CHANNEL FOR MAINTENANCE

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

ALTER DATABASE

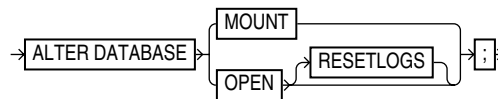
Purpose

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 <code>ALTER DATABASE MOUNT</code> .
OPEN	Opens the database (see Example 2-13). When you open the database after <code>RECOVER DATABASE</code> , 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 logs (or up to the last redo record before redo corruption if corruption is found), clears the contents of the online redo logs, and resets the online redo logs to log sequence 1. The RMAN command <code>ALTER DATABASE OPEN RESETLOGS</code> is equivalent to the SQL statement <code>ALTER DATABASE OPEN RESETLOGS</code> . If you use a recovery catalog, then RMAN issues an implicit <code>RESET DATABASE</code> after the database is opened to make this new incarnation the current one in the catalog. If you execute the SQL statement <code>ALTER DATABASE OPEN RESETLOGS</code> rather than the RMAN command of the same name, then you must manually run the <code>RESET DATABASE</code> 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 logs.

```
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 database (primary or standby), 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.

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 logs 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 Database home. Likewise, some features of Oracle Database, such as external tables or the `BFILE` datatype, 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.

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. Note that 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 key management infrastructure 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 Encryption Wallet, or a password the DBA supplies at decryption time.

The `CONFIGURE` and `SET` commands are used to manage the encryption settings for database backups. See the reference entries for those commands for more details. Backup sets containing archived logs are encrypted if any of the following are 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 the different modes of encryption

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 it 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.

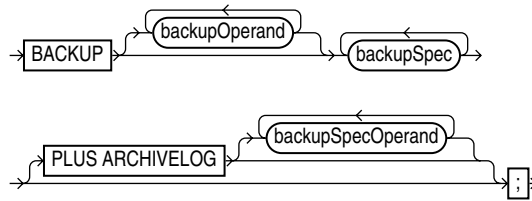
Note: You can FTP a backup from a standby host to a primary host or vice versa and then [CATALOG](#) it. After a file is cataloged by the target database, the file is associated with the target database.

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

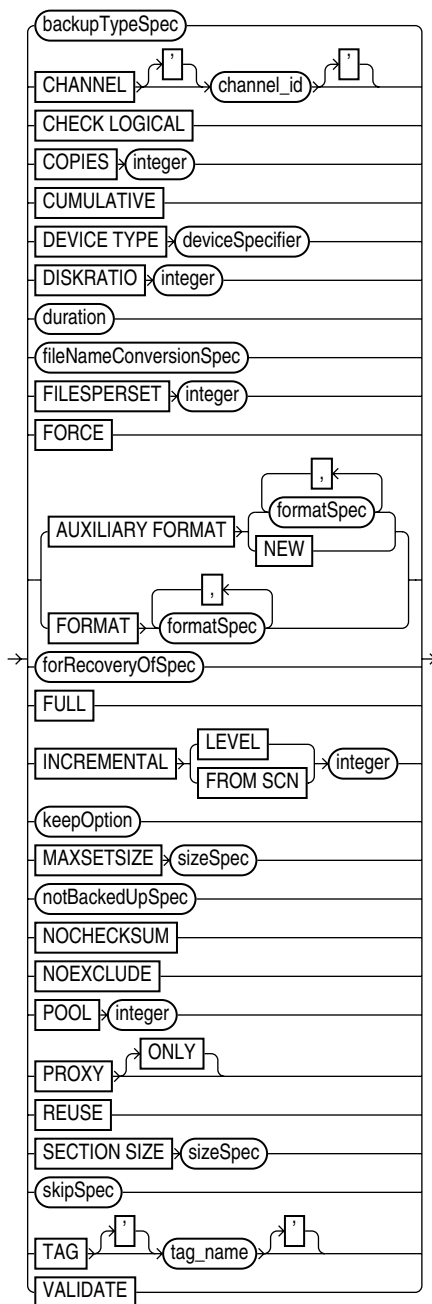
Syntax

backup::=



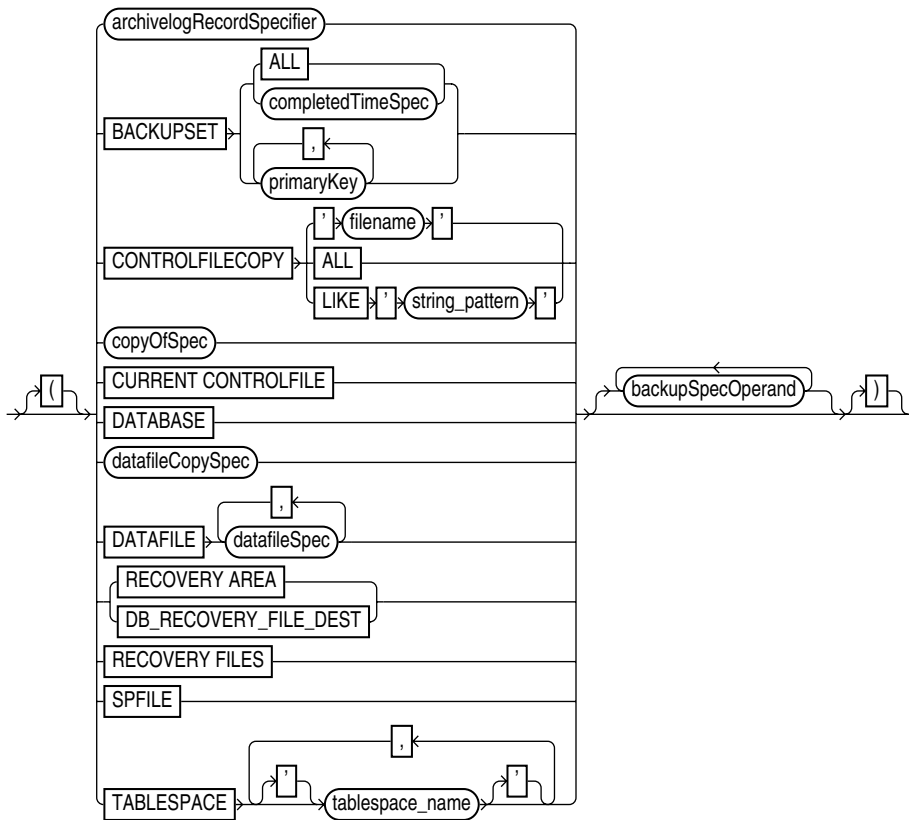
(*backupOperand::=* on page 2-23, *backupSpec::=* on page 2-24, *backupSpecOperand::=* on page 2-25)

backupOperand::=



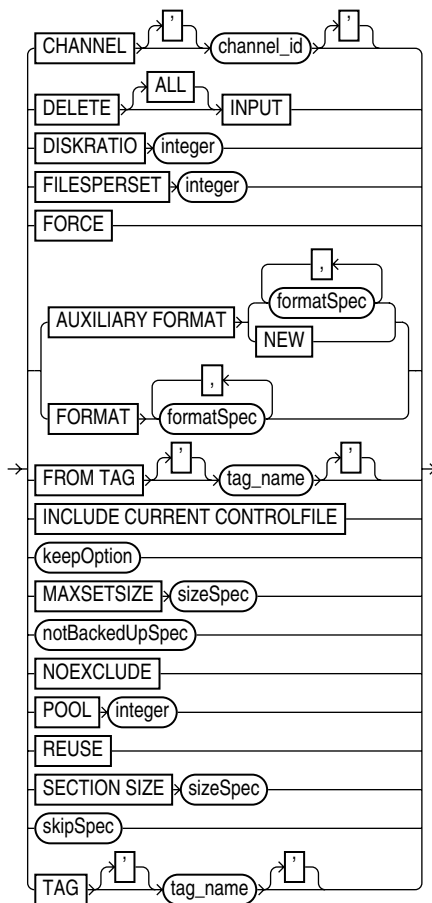
(*deviceSpecifier::=* on page 3-15, *fileNameConversionSpec::=* on page 3-17, *formatSpec::=* on page 3-22, *forRecoveryOfSpec::=* on page 2-26, *keepOption::=* on page 3-25, *notBackedUpSpec::=* on page 2-26, *sizeSpec::=* on page 2-26, *skipSpec::=* on page 2-26)

backupSpec::=



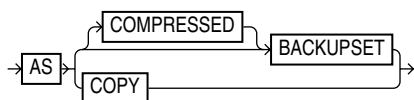
(*archivelogRecordSpecifier::=* on page 3-6, *completedTimeSpec::=* on page 3-10, *copyOfSpec::=* on page 2-25, *datafileCopySpec::=* on page 2-26, *datafileSpec::=* on page 3-14, *backupSpecOperand::=* on page 2-25)

backupSpecOperand::=

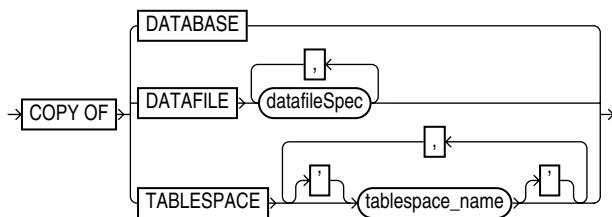


(*formatSpec::=* on page 3-22, *keepOption::=* on page 3-25, *notBackedUpSpec::=* on page 2-26, *sizeSpec::=* on page 2-26, *skipSpec::=* on page 2-26)

backupTypeSpec::=

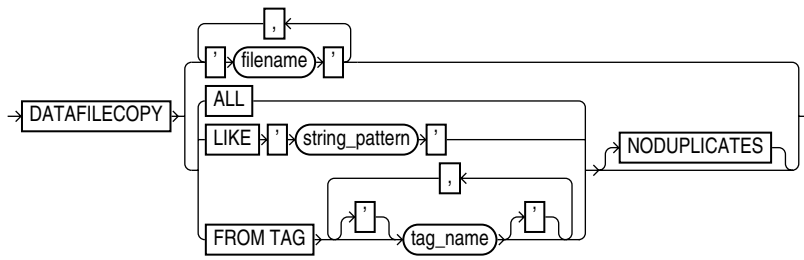


copyOfSpec::=

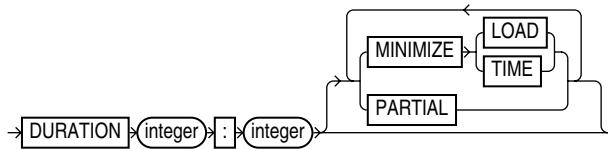


(*datafileSpec::=* on page 3-14)

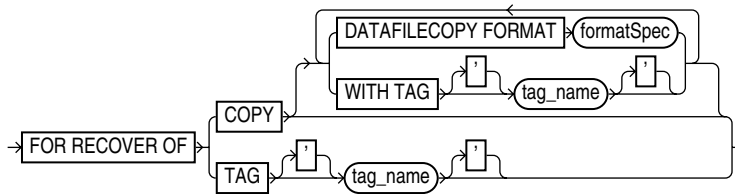
datafileCopySpec::=



duration::=

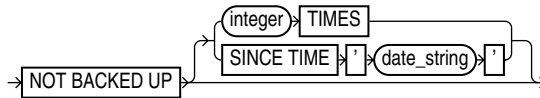


forRecoveryOfSpec::=

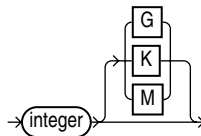


(*formatSpec::=* on page 3-22)

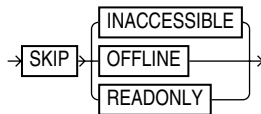
notBackedUpSpec::=



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 2-22 for the syntax diagram.

Syntax Element	Description
<i>backupOperand</i>	Specifies various options for the BACKUP command.
<i>backupSpec</i>	Specifies one or more objects to be backed up. Each <i>backupSpec</i> clause generates one or more backup sets (AS BACKUPSET) or image copies (AS COPY). For AS BACKUPSET, the <i>backupSpec</i> 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 logs in the backup (see Example 2-15 on page 2-46). Causes RMAN to perform the following steps: <ol style="list-style-type: none"> 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 logs. If backup optimization is not enabled, then RMAN backs up the logs generated in step 1 plus all the logs generated during the backup. <p>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.</p> <p>Note: Unless the online redo log is archived at the end of the backup, DUPLICATE will not be possible with this backup.</p>
<i>backupSpecOperand</i>	Specifies a variety of options and parameters that affect the <i>backupSpec</i> clause.

backupOperand

This subclause specifies options such as the device type and output format. Refer to *backupOperand::=* on page 2-23 for the syntax diagram.

Syntax Element	Description
<i>backupTypeSpec</i>	Specifies the type of backup being created, either backup sets (AS BACKUPSET) or image copies (AS COPY). See Also: <i>backupTypeSpec</i> on page 2-40 for details
CHANNEL <i>channel_id</i>	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 2-48, you can use CHANNEL to specify which channels back up which files. Note: You can also specify this parameter in the <i>backupSpec</i> clause.

Syntax Element	Description
CHECK LOGICAL	<p>Tests data and index blocks that pass physical corruption checks for logical corruption (see Example 2-25 on page 2-49).</p> <p>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 <code>SET MAXCORRUPT</code> command specifies the total number of physical and logical corruptions permitted on a datafile.</p> <p>By default, the <code>BACKUP</code> command computes a checksum for each block and stores it in the backup. If you specify the <code>NOCHECKSUM</code> option, then RMAN does not perform a checksum of the blocks when writing the backup.</p> <p>If <code>SET MAXCORRUPT</code> and <code>NOCHECKSUM</code> are not set, then <code>CHECK LOGICAL</code> detects all types of corruption that are possible to detect during a backup.</p>
COPIES <i>integer</i>	<p>Sets the number of identical backups (1 - 4) that RMAN creates. The default value is 1.</p> <p>You can use multiple format strings to specify different names and locations for the copies. Example 2-22 on page 2-48 illustrates a duplexed backup to different locations on disk.</p> <p>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 <code>COPIES</code> is greater than 1, then the <code>BACKUP_TAPE_IO_SLAVES</code> initialization parameter must be enabled on the target database.</p> <p>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:</p> <ol style="list-style-type: none"> 1. <code>BACKUP COPIES</code> 2. <code>SET BACKUP COPIES</code> 3. <code>CONFIGURE . . . BACKUP COPIES</code> <p>Note: This option does not apply with <code>AS COPY</code> and results in an error message.</p> <p>Note: Duplexing cannot be used when creating files in the flash recovery area.</p>
CUMULATIVE	<p>Copies the data blocks used since the most recent level 0 backup (see Example 2-16 on page 2-46).</p> <p>Note: This option does not apply with <code>AS COPY</code> and results in an error message.</p>
DEVICE TYPE <i>deviceSpecifier</i>	<p>Allocates automatic channels for the specified device type only. For example, if you configure disk and tape channels, then configure <code>sbt</code> as the default device type, then the following command allocates disk channels only:</p> <pre>BACKUP DEVICE TYPE DISK DATABASE;</pre> <p>The <code>DEVICE TYPE</code> option is valid only for automatic channels and is not valid for manually allocated channels. You cannot use the <code>DEVICE TYPE</code> option for a device other than <code>DISK</code> if you have not already run <code>CONFIGURE DEVICE TYPE</code> for this device.</p> <p>Note: You cannot specify <code>DEVICE TYPE DISK</code> when running the <code>BACKUP RECOVERY AREA</code> command.</p> <p>See Also: deviceSpecifier on page 3-15</p>

Syntax Element	Description
<code>DISKRATIO</code> <i>integer</i>	<p>Directs RMAN to populate each backup set with datafiles from at least <i>integer</i> disks.</p> <p>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 <code>DISKRATIO</code> to 0.</p> <p>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 <code>DISKRATIO</code> to 5 to direct RMAN to include datafiles from at least 5 disks in each backup set.</p> <p>If you set <code>FILESPERSET</code> but not <code>DISKRATIO</code>, then <code>DISKRATIO</code> defaults to the same value as <code>FILESPERSET</code>. If you specify neither parameter, then <code>DISKRATIO</code> defaults to 4. RMAN compares the <code>DISKRATIO</code> value to the actual number of devices involved in the backup and uses the lowest value. For example, if <code>DISKRATIO</code> is 4 and the datafiles are located on three disks, then RMAN attempts to include datafiles from three disks in each backup set.</p> <p>The <code>DISKRATIO</code> parameter is easier for datafile backups when the datafiles are striped or reside on separate disk spindles and you either:</p> <ul style="list-style-type: none"> ■ 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. <p>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.</p>
<i>duration</i>	<p>Specifies options related to the maximum time for a backup command to run.</p> <p>See Also: <i>duration</i> on page 2-43</p>
<i>fileNameConversionSpec</i>	<p>This option is valid only when <code>BACKUP</code> 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 <code>FORMAT</code> to name the output image copies. If no <code>FORMAT</code> is specified, then RMAN uses the default format <code>%U</code>.</p> <p>See Also: <i>fileNameConversionSpec</i> on page 3-16 for file renaming patterns</p>
<code>FILESPERSET</code> <i>integer</i>	<p>Specifies the maximum number of input files to include in each output backup set. This parameter is only relevant when <code>BACKUP</code> generates backup sets.</p> <p>RMAN backs up the files in each <i>backupSpec</i> as one or more backup sets. When the number of files in each <i>backupSpec</i> exceeds the <code>FILESPERSET</code> setting, then RMAN splits the files into multiple backup sets accordingly. The default value for <code>FILESPERSET</code> is 64.</p> <p>The RMAN behavior is illustrated by the following <code>BACKUP</code> commands:</p> <pre>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;</pre> <p>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.</p> <p>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 <code>FILESPERSET</code> value. Otherwise, <code>FILESPERSET</code> defaults to 64.</p> <p>Note: You cannot specify the number of backup pieces that should be in a backup set.</p>

Syntax Element	Description
FORCE	<p>Forces RMAN to ignore backup optimization. That is, even if CONFIGURE BACKUP OPTIMIZATION is set to ON, RMAN backs up all specified files and undo data for active transactions.</p> <p>Note: You can also specify this option in the <i>backupSpecOperand</i> clause.</p>
AUXILIARY FORMAT	<p>Copies the files on the target database to the specified location on the auxiliary instance. RMAN can only generate image copies when <code>AUXILIARY FORMAT</code> is specified. RMAN must be connected to both <code>TARGET</code> and <code>AUXILIARY</code> instances and have access to auxiliary channels.</p> <p>You can use the <code>BACKUP AUXILIARY FORMAT</code> command to copy datafiles 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 <code>TARGET</code> and the primary database as <code>AUXILIARY</code>, and copy an intact datafile from the standby host to the primary host.</p>
<i>formatSpec</i>	<p>Specifies a pattern for naming the output image copies on an auxiliary instance. The path must be valid on the auxiliary host.</p> <p>See Also: <i>formatSpec</i> on page 3-22 for legal substitution variables</p>
NEW	<p>Creates an image copy in the directory specified in the <code>DB_CREATE_FILE_DEST</code> initialization parameter of the auxiliary instance. The image copy is an Oracle-managed file.</p>

Syntax Element	Description
<p>FORMAT <i>formatSpec</i></p>	<p>Specifies a pattern for naming the output backup pieces or image copies (see Example 2-17 on page 2-46). For <code>AS COPY</code>, if one or more of the directories mentioned in the specified format does not exist, then RMAN signals an error.</p> <p>The default location for disk backups depends on whether a flash recovery area is enabled and whether <code>FORMAT</code> is specified:</p> <ul style="list-style-type: none"> ■ If a flash recovery area is enabled, and if you <i>do</i> specify <code>FORMAT</code>, then RMAN names the output files according to the <code>FORMAT</code> setting. If no location is specified in <code>FORMAT</code>, then RMAN creates the backup in a platform-specific location—not in the recovery area. ■ If a flash recovery area is enabled, and if you do <i>not</i> specify <code>FORMAT</code>, then RMAN creates the backup in the recovery area and uses the substitution variable <code>%U</code> to name the backup. ■ If a flash recovery area is <i>not</i> enabled, and if you do <i>not</i> specify <code>FORMAT</code>, then RMAN creates the backup in a platform-specific location and uses <code>%U</code> to name the backup. <p>To create RMAN backups in the flash recovery area with names in Oracle Managed Files format, do not specify the <code>FORMAT</code> clause on the <code>BACKUP</code> command or channel.</p> <p>Note: You cannot specify an Oracle Managed Files filename as the format for a backup. For example, if <code>+DISK1/datafile/system.732.609791431</code> is an OMF filename, then you cannot specify this filename in the <code>FORMAT</code> parameter.</p> <p>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.</p> <p>You cannot specify multiple, identical <code>FORMAT</code> strings within a single <i>backupSpec</i> (for example, <code>BACKUP DATAFILE 3 TO '/tmp/df3.f', DATAFILE 4 TO '/tmp/df4.f'</code>). However, RMAN permits a single <code>FORMAT</code> string to exist in multiple <i>backupSpec</i> clauses.</p> <p>Note: If you are making an archival backup with the <code>KEEP</code> option (see Example 2-26 on page 2-49), then the format string must contain <code>%U</code>. The <code>autobackup</code> also uses this format string.</p> <p>See Also: <i>formatSpec</i> on page 3-22 for legal substitution variables</p>
<p><i>forRecoveryOfSpec</i></p>	<p>Identifies the backup being created as an incremental backup to be used in rolling forward an image copy.</p> <p>See Also: <i>forRecoveryOfSpec</i> on page 2-44</p>
<p>FULL</p>	<p>Creates a backup of all blocks of datafiles included in the backup. <code>FULL</code> is the opposite of <code>INCREMENTAL</code>. RMAN makes full backups by default if neither <code>FULL</code> nor <code>INCREMENTAL</code> is specified.</p> <p>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 <code>RECOVER</code> command.</p> <p>Note: Unused block compression, which is described in the entry for <code>BACKUP AS BACKUPSET</code>, causes some datafile blocks to be skipped during full backups.</p>

Syntax Element	Description
INCREMENTAL LEVEL <i>integer</i>	<p>Copies only those data blocks that have changed since the last incremental <i>integer</i> backup, where <i>integer</i> is 0 or 1 (see Example 2-16 on page 2-46).</p> <p>If you specify <code>INCREMENTAL</code>, then the <i>backupSpec</i> clause must include one of the following parameters: <code>DATAFILE</code>, <code>DATAFILECOPY</code>, <code>TABLESPACE</code>, or <code>DATABASE</code>. RMAN does not support incremental backups of control files, archived redo logs, or backup sets.</p> <p>Note: You cannot make incremental backups when a <code>NOARCHIVELOG</code> database is open and in use, although you can make incremental backups when the database is mounted after a consistent shutdown.</p> <p>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 <code>FULL</code> backup, but unlike a full backup the level 0 backup is considered a part of the incremental backup strategy.</p> <p>A level 1 backup copies only changed blocks. A level 1 incremental backup is either differential or <code>CUMULATIVE</code>. If cumulative, RMAN backs up all blocks changed since the most recent level 0 backup. If differential, RMAN backs up blocks updated since the last 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.</p> <p>Incremental backups at level 0 can be either backup sets or image copies, but incremental backups at level 1 can only be backup sets.</p> <p>The database performs checks when attempting to create a level 1 incremental backup to ensure that the incremental backup is usable by a subsequent <code>RECOVER</code> command. Among the checks performed are:</p> <ul style="list-style-type: none"> ■ A level 0 backup must exist for each datafile in the <code>BACKUP</code> command as the base backup for an incremental strategy. Level 0 backups must not have status <code>UNAVAILABLE</code>. If no level 0 backup exists when you run a level 1 backup, then RMAN makes a level 0 backup automatically. ■ Sufficient incremental backups taken since the level 0 must exist and be available such that the incremental backup to be created is usable. <p>Note: When creating an incremental backup, then RMAN considers backups from parent incarnations as valid. For example, assume you make a level 0 backup and then <code>OPEN RESETLOGS</code>. If you make a level 1 incremental backup, then RMAN backs up all blocks changed since the pre-<code>RESETLOGS</code> level 0 backup.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>

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

Syntax Element	Description
INCREMENTAL FROM SCN <i>integer</i>	<p>Creates an incremental backup of all specified datafiles that includes all datafile blocks changed at SCNs greater than or equal to the specified SCN.</p> <p>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). 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.</p> <p>If you are <i>not</i> making incremental backups based on Volume Shadow Copy Service (VSS) snapshots, then you should specify <i>formatSpec</i> when you specify INCREMENTAL FROM SCN. The <code>FORMAT</code> string should include substitution variables such as <code>%U</code> because RMAN generates a control file backup.</p> <p>If you specify FROM SCN with the NOKEEP option, and if you do <i>not</i> specify <i>formatSpec</i> when you specify INCREMENTAL FROM SCN, then RMAN creates incremental backups in the flash 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 flash recovery area during recovery transparently.</p> <p>Note: You cannot use PLUS ARCHIVELOG when also specifying INCREMENTAL FROM SCN.</p> <p>See Also: <i>Oracle Database Platform Guide for Microsoft Windows</i> to learn about making backups with VSS</p>
<i>keepOption</i>	<p>Overrides any configured retention policy for this backup so that the backup is not considered obsolete, as shown in Example 2-26 on page 2-49.</p> <p>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.</p> <p>Note: You cannot use KEEP with BACKUP BACKUPSET.</p> <p>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 2-49, you must be connected to a recovery catalog when you specify KEEP FOREVER.</p> <p>Note: You can use CHANGE to alter the status of a backup generated with KEEP.</p> <p>See Also: <i>keepOption</i> on page 3-25 for more information about backups made with the KEEP option</p>

Syntax Element	Description
MAXSETSIZE <i>sizeSpec</i>	<p>Specifies a maximum size for a backup set (as shown in Example 2-17 on page 2-46). RMAN limits all backup sets to this size.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>
<i>notBackedUpSpec</i>	<p>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.</p> <p>See Also: notBackedUpSpec on page 2-45</p>
NOCHECKSUM	<p>Suppresses block checksums during the backup.</p> <p>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 <i>typical</i>, 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.</p> <p>Note: You cannot disable checksums for datafiles in the SYSTEM tablespace even if DB_BLOCK_CHECKSUM=<i>false</i>.</p> <p>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.</p> <p>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 <i>false</i>. If set to <i>typical</i>, then RMAN verifies the checksum when restoring from the backup and writing to the datafile.</p> <p>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.</p> <p>See Also: <i>Oracle Database Reference</i> for more information about the DB_BLOCK_CHECKSUM initialization parameter</p>
NOEXCLUDE	<p>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.</p>

Syntax Element	Description
<code>POOL <i>integer</i></code>	<p>Specifies the media pool in which the backup should be stored. Consult your media management documentation to see whether <code>POOL</code> is supported.</p> <p>Note: This option does not work with <code>AS COPY</code> and results in an error.</p>
<code>PROXY</code>	<p>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.</p> <p>When you run <code>BACKUP</code> with the <code>PROXY</code> option, RMAN performs these steps:</p> <ol style="list-style-type: none"> 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 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. <p>Note: If you specify <code>PROXY</code>, then the <code>%p</code> variable must be included in the <code>FORMAT</code> string either explicitly or implicitly within <code>%U</code>.</p> <p>Note: This option does not work with <code>AS COPY</code> and results in an error.</p>
<code>ONLY</code>	<p>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 <code>ONLY</code> option</p>
<code>REUSE</code>	<p>Enables RMAN to overwrite an already existing backup or copy with the same filename as the file that <code>BACKUP</code> is currently creating.</p>
<code>SECTION SIZE <i>sizeSpec</i></code>	<p>Specifies the size of each backup section produced during a datafile or datafile copy backup.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>Note: You cannot use <code>SECTION SIZE</code> in conjunction with <code>MAXPIECESIZE</code>.</p>
<code><i>skipSpec</i></code>	<p>Excludes datafiles or archived redo logs from the backup if they are inaccessible, offline, or read-only.</p> <p>See Also: <code><i>skipSpec</i></code> on page 2-46 for details.</p>

Syntax Element	Description
TAG <i>tag_name</i>	<p>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.</p> <p>The tag name is not case-sensitive. The name must be 30 characters or less. The characters are limited to the characters that are legal 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 <i>weekly-incremental</i> is not a legal tag name for backups in ASM disk groups.</p> <p>Typically, a tag name is a meaningful name such as <code>MON_PM_BKUP</code> or <code>WEEKLY_FULLL_BKUP</code>. Tags are reusable, so that backup set 100 can have the tag <code>MON_PM_BKUP</code> one week while backup set 105 has the same tag the next week.</p> <p>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 <code>TAGYYYYMMDDTHHMMSS</code>, where <i>YYYY</i> is the year, <i>MM</i> is the month, <i>DD</i> is the day, <i>HH</i> is the hour (in 24-hour format), <i>MM</i> is the minutes, and <i>SS</i> is the seconds. For example, a backup of datafile 1 might receive the tag <code>TAG20070208T133437</code>. The date and time refer to when RMAN started the backup. If multiple backup sets are created by one <code>BACKUP AS BACKUPSET</code> command, then each backup piece is assigned the same default tag.</p> <p>You can also specify the tag at the <i>backupSpec</i> level. If you specify the tag at:</p> <ul style="list-style-type: none"> ■ The command level, then all backup sets created by the command have the tag. ■ The <i>backupSpec</i> level, then backup sets created as a result of different backup specifications can have different tags. ■ Both levels, then the tag in the <i>backupSpec</i> takes precedence. <p>Note: A tag is an attribute of each backup piece in a given copy of a backup set (for <code>AS BACKUPSET</code>) or each image copy (for <code>AS COPY</code>). For example, if you run <code>BACKUP AS BACKUPSET COPIES 1 DATABASE TAG TUE_PM</code>, then only one copy of the backup set exists and each backup piece has tag <code>TUE_PM</code>. Assume that this backup set has primary key 1234. If you then run <code>BACKUP BACKUPSET 1234 TAG WED_PM</code>, then the first copy of the backup set has tag <code>TUE_PM</code> and the second copy of the backup set has tag <code>WED_PM</code>.</p>
VALIDATE	<p>Scans the specified files and verifies their contents, testing whether this file can be backed up and whether the data blocks are corrupt.</p> <p>RMAN creates no output files. This option is equivalent to using the <code>VALIDATE</code> command on the database files specified in the backup.</p> <p>If you do not specify <code>CHECK LOGICAL</code>, then <code>BACKUP VALIDATE</code> checks for physical corruption only. If you specify <code>CHECK LOGICAL</code>, then <code>BACKUP VALIDATE</code> checks for both physical and logical corruption. RMAN populates the <code>V\$DATABASE_BLOCK_CORRUPTION</code> view with any corruptions that it finds.</p> <p>You can use the <code>SET MAXCORRUPT</code> command to set a limit for the number of corrupt blocks tolerated during the backup validation. The default is zero.</p> <p>If you execute <code>BACKUP INCREMENTAL</code> with <code>VALIDATE</code>, then the behavior depends on whether block change tracking is enabled. If change tracking is enabled, then RMAN validates only changed blocks. If change tracking is not enabled, then RMAN validates all blocks in the files included in the backup.</p> <p>Note: You cannot validate backups of backup sets.</p>

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 logs in each backup set. Refer to *backupSpec::=* on page 2-24 for the syntax diagram.

Syntax Element	Description
<i>archivelogRecordSpecifier</i>	<p>Specifies a range of archived redo logs to be backed up.</p> <p>When backing up archived redo logs, 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 logs. RMAN does not signal an error if the command finds no logs to back up, because this situation probably exists because no new logs were generated after the previous <code>BACKUP ARCHIVELOG ALL DELETE INPUT</code> command.</p> <p>If you specify <code>BACKUP ARCHIVELOG ALL</code>, then RMAN backs up exactly one copy of each distinct log sequence number. For example, if you archive to multiple destinations, RMAN backs up <i>one</i> copy of each log sequence number—not each copy of each log sequence number. For other commands, such as <code>DELETE ALL</code> does refer to every log, even duplicate log sequences.</p> <p>If the database is open when you run <code>BACKUP ARCHIVELOG</code>, and if the <code>UNTIL</code> clause or <code>SEQUENCE</code> parameter is not specified, then RMAN runs <code>ALTER SYSTEM ARCHIVE LOG CURRENT</code>.</p> <p>Note: If you run <code>BACKUP ARCHIVELOG ALL</code>, 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 <code>OPEN RESETLOGS</code>.</p> <p>See Also: <i>archivelogRecordSpecifier</i> on page 3-6 for syntax, and <i>Oracle Database Backup and Recovery User's Guide</i> explanations of backup failover for logs and automatic log switching</p>
BACKUPSET	<p>Specifies a backup of backup sets. Use this parameter in conjunction with the <code>DEVICE TYPE sbt</code> 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.</p> <p>If you specify the <code>DELETE INPUT</code> option on the <code>BACKUP BACKUPSET</code> 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 <code>ALL</code> option does not add any functionality.</p> <p>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 logs that exist in multiple archiving destinations.</p> <p>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.</p> <p>Note: You can duplex backups of backup sets with <code>BACKUP COPIES</code> and <code>SET BACKUP COPIES</code>.</p> <p>Note: When you use <code>BACKUP BACKUPSET</code> command with encrypted backup sets, the backup sets are backed up in their encrypted form. Because <code>BACKUP BACKUPSET</code> just copies an already-encrypted backup set to disk or tape, no decryption key is needed during a <code>BACKUP BACKUPSET</code> operation. The data is never decrypted during any part of the operation. The <code>BACKUP BACKUPSET</code> command can neither encrypt nor decrypt backup sets.</p>
ALL	Specifies all backup sets.
<i>completedTimeSpec</i>	<p>Identifies backup sets according to completion time.</p> <p>See Also: <i>completedTimeSpec</i> on page 3-10</p>
<i>integer</i>	Specifies backup sets according to primary key. You can obtain the primary keys for backup sets from the output of the <code>LIST BACKUP</code> command.

Syntax Element	Description
CONTROLFILECOPY	<p>Specifies one or more control file copies for backups.</p> <p>A control file copy can be created with the <code>BACKUP AS COPY CURRENT CONTROLFILE</code> command or the SQL statement <code>ALTER DATABASE BACKUP CONTROLFILE TO '...'</code>.</p> <p>Note: A control file autobackup is not a control file copy.</p>
'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 or more characters; an underscore (_) is a wildcard meaning 1 character.
<i>copyOfSpec</i>	<p>Makes a backup of previous image copies of datafiles and possibly control files.</p> <p>See Also: <i>copyOfSpec</i> on page 2-42</p>
CURRENT CONTROLFILE	<p>Specifies the current control file.</p> <p>Note: You cannot assign a tag to a backup of the current control file.</p>
DATABASE	<p>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 logs.</p> <p>If the <i>backupSpec</i> includes datafile 1, and if <code>CONFIGURE CONTROLFILE AUTOBACKUP</code> 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.</p> <p>If the <i>backupSpec</i> includes datafile 1, and if <code>CONFIGURE CONTROLFILE AUTOBACKUP</code> is ON, then RMAN does <i>not</i> 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.</p> <p>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.</p> <p>Note: To force RMAN to include the current control file in the backup when <code>CONTROLFILE AUTOBACKUP</code> is ON, specify the <code>INCLUDE CURRENT CONTROLFILE</code> clause.</p> <p>See Also: The <code>TABLESPACE</code> description to learn about backup behavior when the database includes bigfile tablespaces</p>
<i>datafileCopySpec</i>	<p>Specifies the filenames of one or more datafile image copies.</p> <p>See Also: <i>datafileCopySpec</i> on page 2-43 for details</p>
DATAFILE <i>datafileSpec</i>	<p>Specifies a list of one or more datafiles. Refer to description of <code>BACKUP DATABASE</code> for RMAN behavior when datafile 1 is backed up.</p> <p>See Also: <i>datafileSpec</i> on page 3-14</p>

Syntax Element	Description
RECOVERY AREA	<p>Backs up recovery files created in the current and all previous flash recovery area destinations. The backup must go to SBT.</p> <p>Recovery files are full and incremental backup sets, control file autobackups, datafile copies, and archived redo logs. 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 logs are <i>not</i> backed up.</p> <p>By default, backup optimization is enabled for this command even if the CONFIGURE BACKUP OPTIMIZATION setting is OFF. You can disable backup optimization for <code>BACKUP RECOVERY AREA</code> by specifying <code>FORCE</code>.</p> <p>Note: If the flash recovery area is not enabled but has been enabled in the past, then files created in the previous flash recovery area location are backed up.</p>
DB_RECOVERY_FILE_DEST	<p><code>RECOVERY AREA</code> and <code>DB_RECOVERY_FILE_DEST</code> are synonyms.</p>
RECOVERY FILES	<p>Backs up <i>all</i> recovery files on disk, whether they are stored in the flash recovery area or other locations on disk. The backup must go to SBT.</p> <p>Recovery files include full and incremental backup sets, control file autobackups, archived redo log files, and datafile copies.</p> <p>By default, backup optimization is enabled for this command even if the CONFIGURE BACKUP OPTIMIZATION setting is OFF. You can disable backup optimization for <code>RECOVERY FILES</code> by specifying <code>FORCE</code>.</p>
SPFILE	<p>Specifies that the server parameter file should be included in a backup set. The <code>AS COPY</code> option is not supported for server parameter file backups.</p> <p>RMAN backs 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 <code>SPFILE</code>.</p>
TABLESPACE <i>tablespace_name</i>	<p>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 <code>SYSTEM</code> 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.</p> <p>You cannot back up locally-managed temporary tablespaces, although you can back up dictionary-managed tablespaces.</p> <p>If the following conditions are met, then RMAN can back up transportable tablespaces that have <i>not</i> made read/write after being transported:</p> <ul style="list-style-type: none"> ■ The <code>COMPATIBLE</code> initialization parameter is set to 11.0.0 or higher. ■ You are using an Oracle Database 11g RMAN client. <p>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.</p> <p>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.</p>
<i>backupSpecOperand</i>	<p>The <i>backupSpecOperand</i> that follows a <i>backupSpec</i> specifies options that apply to the <i>backupSpec</i>.</p>

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 2-24 for the syntax diagram.

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

backupTypeSpec

This subclause specifies the form of the BACKUP command output: backup set or image copy. Refer to [backupTypeSpec::=](#) on page 2-25 for the syntax diagram.

Syntax Element	Description
AS BACKUPSET	<p>Creates backup sets on the specified device. This is the default backup type.</p> <p>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.</p> <p>The <code>FILESPPERSET</code> parameter of the <code>BACKUP</code> command determines the maximum number of files in each backup set. Archived logs and datafiles are never combined into a single backup set.</p> <p>When using encrypted backups, datafiles from tablespaces with different encryption settings are never written into the same backup set.</p> <p>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.</p> <p>RMAN backup sets automatically use unused block compression. Skipping unused data blocks where possible enables RMAN to back up datafiles using less space, and can make I/O more efficient. When backing up datafiles into backup sets, RMAN does not include data blocks that have never been allocated. RMAN also skips datafile blocks that do not currently contain data, but only if all of the following conditions apply:</p> <ul style="list-style-type: none"> ■ The <code>COMPATIBLE</code> initialization parameter is set to 10.2 or higher. <p>Note that if <code>COMPATIBLE</code> is 10.2, then only tablespaces <i>created</i> with 10.2 compatibility will be optimized to exclude blocks that do not currently contain data. If <code>COMPATIBLE</code> is 11.0.0 or higher, however, then the first backup that produces backup sets after <code>COMPATIBLE</code> 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.</p> ■ There are currently no guaranteed restore points defined for the database. ■ The datafile is locally managed. ■ The datafile is being backed up to a backup set as part of a full backup or a level 0 incremental backup. ■ The backup set is created on disk or Oracle Secure Backup is the media manager. Thus, when backing up to a media manager other than Oracle Secure Backup, RMAN does <i>not</i> skip datafile blocks that do not currently contain data. <p>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 <code>BACKUP</code> command to generate a proxy copy, which is a backup in which the entire data transfer is conducted by a media manager.</p> <p>RMAN only records complete backup sets in the RMAN repository. There are no partial backup sets. When a <code>BACKUP</code> command creates backup pieces but does not produce a complete backup set, the backup pieces are discarded.</p> <p>Note: You cannot stripe a single backup set across multiple channels. You also cannot stripe a single input file across multiple backup sets.</p> <p>See Also: <i>Oracle Secure Backup Administrator's Guide</i> to learn how to use Oracle Secure Backup with RMAN</p>

Syntax Element	Description
COMPRESSED	<p>Enables binary compression.</p> <p>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.</p> <p>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 <i>locally attached</i> tape device, compression provided by the media management vendor is usually preferable to the binary compression provided by <code>BACKUP AS COMPRESSED BACKUPSET</code>. 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.</p> <p>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.</p>
AS COPY	<p>Creates image copies (rather than backup sets).</p> <p>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.</p> <p>By default, <code>BACKUP</code> generates backup sets. You can change the default backup type for disk backups to image copies using the <code>CONFIGURE DEVICE TYPE ... BACKUP TYPE TO COPY</code> command.</p> <p>RMAN chooses a location for the copy according to the following rules, listed in order of precedence:</p> <ol style="list-style-type: none"> 1. <code>FORMAT</code> specified on <code>BACKUP</code> command for the object being backed up 2. <code>FORMAT</code> specified for the <code>BACKUP</code> command 3. <code>fileNameConversionSpec</code> setting for <code>BACKUP</code> command 4. <code>CONFIGURE CHANNEL integer ... FORMAT</code> 5. <code>CONFIGURE CHANNEL DEVICE TYPE ... FORMAT</code> 6. Platform-specific default <code>FORMAT</code> (which includes a <code>%U</code> for generating a unique filename) <p>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 <code>CATALOG</code> command to add the user-managed copies to the RMAN repository so that RMAN can use them.</p> <p>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 <code>BACKUP BACKUPSET</code>.</p>

copyOfSpec

This subclause specifies the form of the `BACKUP` command output: backup set or image copy. Refer to `copyOfSpec::=` on page 2-25 for the syntax diagram.

Syntax Element	Description
COPY OF DATABASE	<p>Makes a backup of previous image copies of all datafiles and control files in the database. All datafiles that would normally be included by <code>BACKUP DATABASE</code> 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 <code>BACKUP</code> command. If multiple copies exist of datafile, then RMAN backs up the latest. Optionally, specify the copies by tag name (for example, <code>FULL_COLD_COPY</code>).</p> <p>Note: The output of this command can be image copies or backup sets.</p>
COPY OF DATAFILE <i>datafileSpec</i>	<p>Makes a backup of a previous image copy of one or more datafiles. Specify the datafile by file number (<code>DATAFILE 3</code>) or filename (<code>DATAFILE ' ?/oradata/trgt/users01.dbf '</code>). You specify the datafile filename and <i>not</i> 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.</p> <p>Note: It is not necessary for the image copies that you are backing up to have been created by a single <code>BACKUP</code> command.</p> <p>Note: The output of this command can be image copies or backup sets.</p> <p>See Also: <i>datafileSpec</i> on page 3-14</p>
COPY OF TABLESPACE <i>tablespace_name</i>	<p>Makes a backup of previous image copies of the datafiles in one or more specified tablespaces. All datafiles that would normally be included by <code>BACKUP TABLESPACE</code> should have copies: if not, then RMAN signals an error. It is not necessary for all copies to have been produced by a single <code>BACKUP</code> command. If multiple copies exist of datafile, then RMAN backs up the latest.</p> <p>Specify the tablespaces in the list by tablespace name (for example, <code>users</code>) or specify a particular copy by tag name (for example, <code>0403_CPY_OF_USERS</code>). If you do not specify <code>TAG</code>, then RMAN backs up the most recent datafile copy for each datafile in the tablespace.</p> <p>Note: The output of this command can be image copies or backup sets.</p>

datafileCopySpec

This subclause specifies datafile copies. Refer to *datafileCopySpec::=* on page 2-26 for the syntax diagram.

Syntax Element	Description
' <i>filename</i> '	Specifies the filenames of one or more datafile image copies.
ALL	Specifies that all datafile image copies should be backed up.
LIKE ' <i>string_pattern</i> '	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 <i>tag_name</i>	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 2-50). 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 2-26 for the syntax diagram.

Syntax Element	Description
DURATION <i>hh:mm</i>	<p>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.</p> <p>Without the <code>PARTIAL</code> 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 <code>RUN</code> block, then subsequent commands in the <code>RUN</code> block do not execute.</p>
MINIMIZE { <code>LOAD</code> <code>TIME</code> }	<p>With disk backups, you can use <code>MINIMIZE TIME</code> run the backup at maximum speed (default), or <code>MINIMIZE LOAD</code> to slow the rate of backup to lessen the load on the system. With <code>MINIMIZE LOAD</code> the backup will take the full specified duration.</p> <p>If you specify <code>TIME</code>, 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.</p>
PARTIAL	<p>With the <code>PARTIAL</code> 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.</p> <p>Without the <code>PARTIAL</code> 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 <code>RUN</code> block, then subsequent commands in the <code>RUN</code> block do not execute.</p> <p>Whether <code>PARTIAL</code> is used or not, all backup sets completed before the backup is interrupted are retained and can be used in <code>RESTORE</code> and <code>RECOVER</code> operations.</p>

forRecoveryOfSpec

This subclause specifies that a backup should be used in an incrementally updated backup strategy. Refer to *forRecoveryOfSpec::=* on page 2-26 for the syntax diagram.

Syntax Element	Description
FOR RECOVER OF COPY	<p>Specifies that this incremental backup should contain all changes since the SCN of a specified datafile copy (level 0 incremental backup) of the target database (see Example 2-20 on page 2-48).</p> <p>A <code>BACKUP FOR RECOVER OF COPY</code> command can create backup sets or image copies. RMAN creates backup sets if an incremental level 0 image copy already exists, but otherwise creates image copies.</p> <p>The datafile copies should be identified with either a <code>DATAFILE COPY</code> or <code>WITH TAG</code> clause, to separate the incremental backup strategy for which this backup will be used from the rest of your backup strategies. Otherwise, the most recent copy of each datafile will be used as the basis for the incremental backup.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to make incrementally updated backups</p>
WITH TAG <i>tag_name</i>	<p>Specifies the tag of the level 0 incremental backup serving as the basis of the incremental backup.</p> <p>If no level 0 with the tag specified in the <code>WITH TAG</code> option is found, then <code>FOR RECOVER OF COPY</code> option creates a level 0 datafile copy tagged with the value specified in the <code>WITH TAG</code> option.</p> <p>By using the <code>BACKUP INCREMENTAL ... FOR RECOVER OF COPY WITH TAG</code> syntax, you can create level 1 incremental backups suitable for rolling forward a level 0 incremental image copy backup of your database. You can then use <code>RECOVER COPY OF ... WITH TAG</code> to perform incremental updates of a backup. This technique is used in the Enterprise Manager Oracle-suggested strategy for backups to disk.</p>

Syntax Element	Description
DATAFILECOPY FORMAT <i>formatSpec</i>	Specifies a pattern for naming the output image copies.
FOR RECOVER OF TAG <i>tag_name</i>	Backs up the archived redo log files or incremental backups that are intended to recover the level 0 incremental backup specified by <i>tag_name</i> . For example, you can back up all archived redo log files needed to recover the level 0 incremental backup tagged <code>wholedb</code> .

notBackedUpSpec

This subclause specifies that RMAN should only back up files that have not yet been backed up. Refer to *notBackedUpSpec::=* on page 2-26 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 (see Example 2-28 on page 2-50). This option 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. Using BACKUP with this clause does manually what an archived redo log deletion policy can do automatically. If you specify the <code>CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP integer TIMES</code> command, then a <code>BACKUP ARCHIVELOG ALL</code> command copies all logs unless <i>integer</i> backups already exist on the specified device type. If <i>integer</i> backups of the logs exist, then the BACKUP command skips the logs. In this way, the archived log deletion policy functions as a default <code>NOT BACKED UP integer TIMES</code> clause on the BACKUP command. Note: This clause overrides backup optimization (<code>CONFIGURE BACKUP OPTIMIZATION</code>) and archived log deletion policies (<code>CONFIGURE ARCHIVELOG DELETION POLICY</code>).
<i>integer</i> TIMES	Backs up only those archived redo log files that have not been backed up at least <i>integer</i> times. Note: You can only specify <code>NOT BACKED UP integer TIMES</code> 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 logs on a specified media. For example, you want to keep at least three copies of each log on tape.
SINCE TIME ' <i>date_string</i> '	Specifies the date after which RMAN should back up files that have no backups. The <i>date_string</i> is either a date in the current <code>NLS_DATE_FORMAT</code> or a SQL date expression such as <code>'SYSDATE-1'</code> . 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 <code>NOT BACKED UP SINCE TIME</code> clause and avoid backing up those files that you already backed up. If <code>AS BACKUPSET</code> 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 <code>BACKUP AS BACKUPSET</code> , 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 2-26 for the syntax diagram.

Syntax Element	Description
<i>integer</i> [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 2-26 for the syntax diagram.

Syntax Element	Description
SKIP	Excludes datafiles or archived redo logs from the backup according to the criteria specified by the following keywords. Note: You can also specify this option in the <i>backupSpec</i> clause.
INACCESSIBLE	Specifies that datafiles or archived redo logs 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 backs up all datafiles to tape, as well as the current control file, the server parameter file, and archived redo log files:

```
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 Logs

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_2z45s13g_.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 time required for media recovery of your database. For example, if you run a daily backup script, then you never have more than one day of redo to apply for media recovery.

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 backup of the database. On every subsequent execution, RMAN applies the level 1 incremental 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
  PARS 'ENV=(OB_DEVICE_1=stape1)';
  ALLOCATE CHANNEL ch2 DEVICE TYPE sbt
  PARS 'ENV=(OB_DEVICE_1=stape2)';
```

```

BACKUP
  (DATABASE          # channel ch1 backs up database to tape drive stapel
    CHANNEL ch1)
  (ARCHIVELOG ALL
    CHANNEL ch2); # channel ch2 backs up archived redo logs to tape drive stape2
}

```

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

This example creates an incremental backup at a primary database that can be applied at a standby database to update it with changes since the specified SCN.

```

CONNECT TARGET SYS/password@prod
CONNECT CATALOG rman/password@catdb
BACKUP DEVICE TYPE DISK
  INCREMENTAL FROM SCN 404128 DATABASE
  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 that 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 logs 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`.

```

CONNECT TARGET SYS/password@prod1
CONNECT CATALOG rman/password@catdb
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 logs 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
```

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. Note that in a Data Guard environment it is strongly recommended (although not required) that RMAN be connected to a recovery catalog.

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.

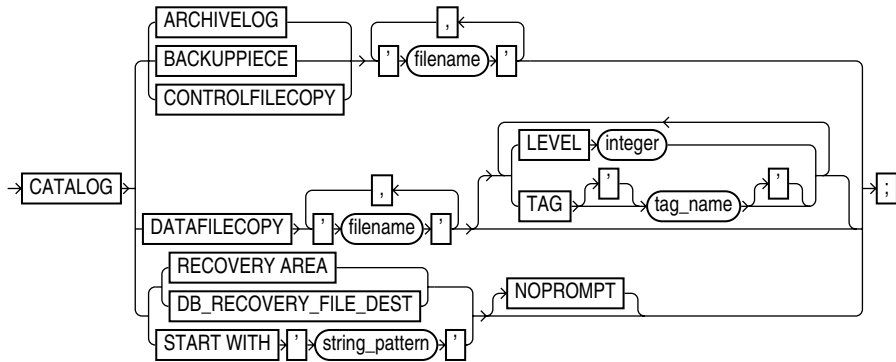
A recovery catalog is required when using RMAN in a Data Guard environment. The recovery catalog supports a unified file namespace for all primary and standby databases with the same DBID but different `DB_UNIQUE_NAME` values. Thus, the recovery catalog keeps track of database file names for all primary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

"[RMAN Backups in a Data Guard Environment](#)" on page 2-21 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 logs, which are redo logs received by a logical standby database for a LogMiner session. Unlike normal archived logs, foreign archived logs have a different DBID.

Syntax Element	Description
<code>BACKUPPIECE 'filename'</code>	<p>Specifies the name of a backup piece to be added to the RMAN repository (see Example 2-30 on page 2-54).</p> <p>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.</p> <p>You may choose to catalog backup pieces in the following situations:</p> <ul style="list-style-type: none"> ■ 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 <code>DELETE</code> command on a backup piece that was only temporarily unavailable. ■ You make a <code>NOCATALOG</code> 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 logs. You can restore and mount the control file, but must catalog the backup pieces containing the archived redo logs backed up after the control file. <p>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 <code>V\$BACKUP_PIECE</code>. A backup set is only usable when <i>all</i> backup pieces are cataloged because otherwise it is only in a partially available state.</p> <p>Note: If RMAN creates a server parameter file backup when the <code>COMPATIBLE</code> 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 <code>DB_UNIQUE_NAME</code> associated with this backup does not change. For example, if RMAN backs up the server parameter file of the database with <code>DB_UNIQUE_NAME 'NEWYORK'</code> when <code>COMPATIBLE</code> is 11.0.0, then RMAN cannot use the server parameter file backup created at database <code>NEWYORK</code> to restore the server parameter file on database <code>BOSTON</code>.</p>
<code>CONTROLFILECOPY 'filename'</code>	<p>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</p> <ul style="list-style-type: none"> ■ The RMAN command <code>BACKUP AS COPY CURRENT CONTROLFILE</code> ■ The SQL statement <code>ALTER DATABASE BACKUP CONTROLFILE</code> ■ The SQL statement <code>ALTER DATABASE CREATE STANDBY CONTROLFILE</code> <p>Note: RMAN can automatically convert a primary database control file backup to a standby control file during a restore operation.</p>
<code>DATAFILECOPY 'filename'</code>	<p>Specifies the filename of a datafile copy to be added to the RMAN repository (see Example 2-30 on page 2-54). You can create datafile copies with the RMAN <code>BACKUP AS COPY</code> command or with operating system utilities used in conjunction with <code>ALTER TABLESPACE BEGIN/END BACKUP</code>.</p>
<code>LEVEL integer</code>	<p>Specifies that the datafile copy should be recorded as a level 0 incremental backup (0 is the only legal value of <code>LEVEL</code>).</p> <p>You can perform incremental backups by using this datafile copy as the base level 0 backup.</p>
<code>TAG tagname</code>	<p>Specifies a tag for the datafile copy.</p>

Syntax Element	Description
RECOVERY AREA	<p>Catalogs all valid backup sets, datafile copies, and archived redo logs in the flash recovery area (see Example 2-32 on page 2-55).</p> <p>RMAN must be connected to the target database and the target database must be mounted or open. The keywords <code>RECOVERY AREA</code> and <code>DB_RECOVERY_FILE_DEST</code> are exact synonyms.</p> <p>Note: This command also catalogs foreign archived logs, which are archived redo logs received by logical standby for a LogMiner session, if they exist in the flash recovery area.</p>
DB_RECOVERY_FILE_DEST	<p>The keywords <code>RECOVERY AREA</code> and <code>DB_RECOVERY_FILE_DEST</code> are exact synonyms.</p>
START WITH ' <i>string_pattern</i> '	<p>Catalogs all valid backup sets, datafile and control file copies, and archived redo logs whose name start with <i>string_pattern</i>. The string pattern can be an ASM disk group, Oracle-managed files directory, or part of a file name (see Example 2-31 on page 2-54).</p> <p>RMAN reports any files in the disk location that it cannot catalog. RMAN must be connected a mounted target database.</p> <p>If the string pattern specifies a filename, then it matches the left part of the filename pattern. For example, <code>/tmp/arc</code> matches everything in directory <code>/tmp/arc_dest</code> and <code>/tmp/archive/january</code> as well as file <code>/tmp/arc.cpy</code>.</p> <p>Note: You cannot use wildcard characters in the string pattern, only a strict prefix.</p>
NOPROMPT	<p>Suppresses the confirmation prompt. By default, RMAN prompts after every match.</p>

Examples

Example 2-30 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 2-31 Cataloging Multiple Copies in a Directory

This example catalogs a directory full of archived redo logs 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 2-32 Cataloging Files in the Flash Recovery Area

This example catalogs all files in the currently enabled flash 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 2-33 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):

```
RMAN> 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

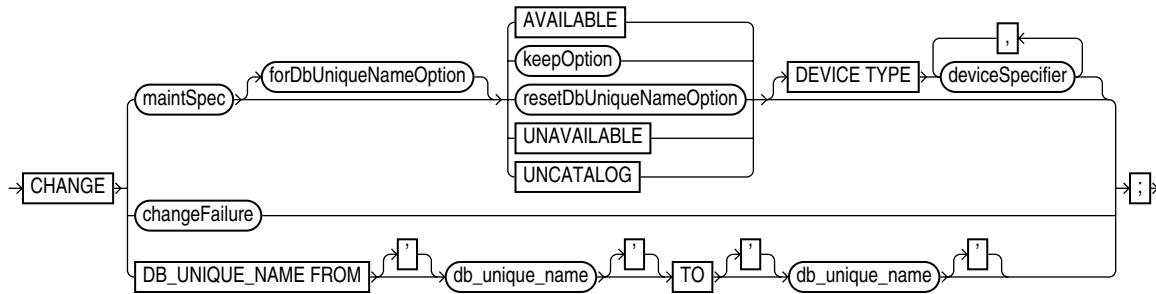
"[RMAN Backups in a Data Guard Environment](#)" on page 2-21 explains the difference between the association and accessibility of a backup. In a Data Guard environment, the database that creates a backup or copy is associated with the file. You can use maintenance commands such as `CHANGE`, `DELETE`, 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, 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`.

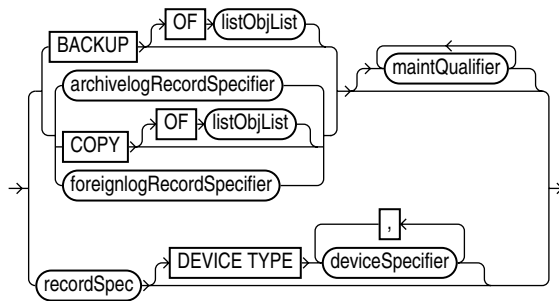
Syntax

change::=



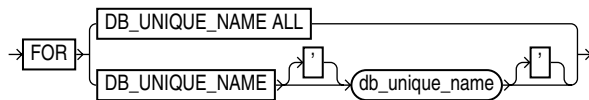
(*maintSpec::=* on page 2-57, *forDbUniqueNameOption::=* on page 3-19, *keepOption::=* on page 3-25, *deviceSpecifier::=* on page 3-15)

maintSpec::=

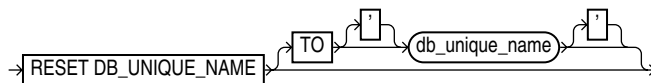


(*listObjList::=* on page 3-28, *archivelogRecordSpecifier::=* on page 3-6, *maintQualifier::=* on page 3-31, *recordSpec::=* on page 3-36, *deviceSpecifier::=* on page 3-15)

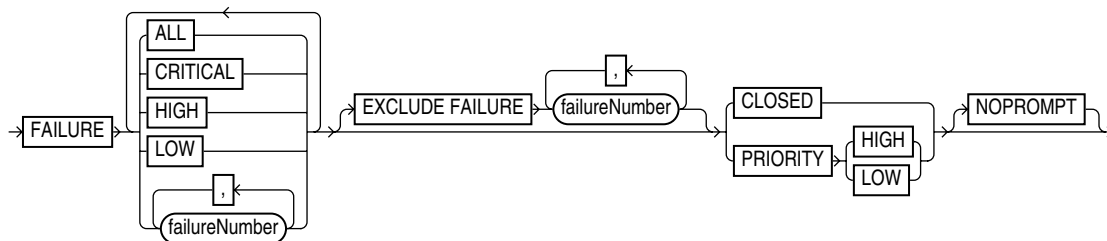
forDbUniqueNameOption::=



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
<i>maintSpec</i>	<p>Specifies which files you want to <code>CHANGE</code>.</p> <p>See Also: <i>maintSpec</i> on page 3-33 for descriptions of the options in this clause.</p>
<i>forDbUniqueNameOption</i>	<p>Changes the metadata for objects that are exclusively associated with the specified <code>DB_UNIQUE_NAME</code> in a Data Guard environment.</p> <p>See Also: <i>forDbUniqueNameOption</i> on page 3-19 for descriptions of the options in this clause.</p>
AVAILABLE	<p>Changes the status of a backup or copy to <code>AVAILABLE</code> in the repository. RMAN searches for the file and verifies that it exists.</p> <p>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.</p> <p>This is the only <code>CHANGE</code> option that requires either a manual or automatic maintenance channel. A maintenance channel is not required, however, when <code>CHANGE . . . AVAILABLE</code> is used with a file that is disk only (that is, an <code>ARCHIVELOG</code>, <code>DATAFILECOPY</code>, or <code>CONTROLFILECOPY</code>). If you use <code>CHANGE . . . AVAILABLE</code> 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 <code>sbt</code> channel, but have only a <code>DISK</code> channel automatically configured, then you must manually allocate an <code>sbt</code> channel before <code>CHANGE . . . AVAILABLE</code> can operate on the backup.</p> <p>If you execute <code>CHANGE . . . AVAILABLE</code> for a file in a Data Guard environment, then RMAN attempts to crosscheck the file before updating its status to <code>AVAILABLE</code>. If the file is inaccessible, then RMAN prompts you to perform the same operation when connected as <code>TARGET</code> to the database associated with the file. If the file is accessible, then RMAN updates the status as requested.</p> <p>Note: You can view the status of backups in the <code>LIST</code> output or recovery catalog views.</p> <p>Note: <code>CHANGE . . . AVAILABLE</code> is not valid for foreign archived redo logs, which are received by a logical standby database for a LogMiner session. Unlike normal archived logs, foreign archived logs have a different DBID.</p>
<i>keepOption</i>	<p>Changes the exemption status of a backup or copy in relation to the configured retention policy. For example, specify <code>CHANGE . . . NOKEEP</code> to remove the <code>KEEP</code> attributes for a backup, making it subject to the backup retention policy.</p> <p>The <code>KEEP FOREVER</code> clause requires use of a recovery catalog (see Example 2-36 on page 2-62). The <code>RESTORE POINT</code> option is not valid with <code>CHANGE</code>. You cannot use <code>CHANGE . . . UNAVAILABLE</code> or <code>KEEP</code> attributes for files stored in the flash recovery area.</p> <p>See Also: <i>keepOption</i> on page 3-25</p>
<i>resetDbUniqueNameOption</i>	<p>Associates the files in <i>maintSpec</i> with a different database in a Data Guard environment.</p> <p>See Also: <i>resetDbUniqueNameOption</i> on page 2-60</p>

Syntax Element	Description
UNAVAILABLE	<p>Changes the status of a backup or copy to UNAVAILABLE in the repository (see Example 2-34 on page 2-62). View the status in the LIST output or recovery catalog views.</p> <p>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.</p> <p>CHANGE . . . UNAVAILABLE is not valid for files in the flash recovery area. This command is also not valid for foreign archived redo logs, which are received by a logical standby database for a LogMiner session. Unlike normal archived logs, foreign archived logs have a different DBID.</p> <p>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.</p>
UNCATALOG	<p>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 2-35 on page 2-62). 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.</p> <p>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.</p> <p>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.</p>
DEVICE TYPE <i>deviceSpecifier</i>	<p>Executes the CHANGE for the specified device type only (see deviceSpecifier on page 3-15). 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.</p>
<i>changeFailure</i>	<p>Specifies changes for failures recorded by the Data Recovery Advisor.</p>

Syntax Element	Description
<pre>DB_UNIQUE_NAME FROM db_unique_name TO db_unique_name</pre>	<p>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.</p> <p>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 <i>db_unique_name</i> parameter; otherwise, RMAN signals an error.</p> <p>Typically, you use this command after you have already changed the DB_UNIQUE_NAME initialization parameter of a database and need to 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 <code>LIST DB_UNIQUE_NAME</code> before <code>CHANGE DB_UNIQUE_NAME</code>.</p> <p>Assume that you have changed the DB_UNIQUE_NAME initialization parameter for a standby database from <i>standby_old</i> to <i>standby_new</i>. Typically, you execute <code>CHANGE DB_UNIQUE_NAME</code> in the following scenarios:</p> <ul style="list-style-type: none"> ■ A <code>LIST DB_UNIQUE_NAME</code> command shows the old DB_UNIQUE_NAME value but does <i>not</i> show the new one (see Example 2-39 on page 2-63). ■ A <code>LIST DB_UNIQUE_NAME</code> command shows both the old <i>and</i> 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, <code>LIST DB_UNIQUE_NAME</code> command can show both the old and new names (in this example, <i>standby_old</i> and <i>standby_new</i>) for a database whose DB_UNIQUE_NAME initialization parameter has been changed. <p>In the scenario in which only the old name is listed, execute <code>CHANGE DB_UNIQUE_NAME FROM standby_old TO standby_new</code> so that RMAN changes the DB_UNIQUE_NAME for <i>standby_old</i> to <i>standby_new</i> in the recovery catalog.</p> <p>In the scenario in which both the old and new names are listed, RMAN automatically executes the following commands when you run <code>CHANGE DB_UNIQUE_NAME FROM standby_old TO standby_new</code>:</p> <ol style="list-style-type: none"> 1. <code>CHANGE ARCHIVELOG ALL FOR DB_UNIQUE_NAME standby_old RESET DB_UNIQUE_NAME</code> 2. <code>CHANGE BACKUP FOR DB_UNIQUE_NAME standby_old RESET DB_UNIQUE_NAME</code> 3. <code>UNREGISTER DB_UNIQUE_NAME standby_old</code> <p>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.</p>

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 <i>db_unique_name</i>	FOR DB_UNIQUE_NAME	RMAN Behavior
No	No	RMAN associates the <i>maintSpec</i> 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 11g recovery catalog schema, so that you can associate the backups with the target database.
Yes	No	RMAN associates the <i>maintSpec</i> files with the database indicated by the TO <i>db_unique_name</i> . RMAN also changes the association of all backups that are not associated with any database.
No	Yes	RMAN restricts its operations to <i>maintSpec</i> 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 <i>maintSpec</i> 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 <i>db_unique_name</i> .

Syntax Element	Description
RESET DB_UNIQUE_NAME	<p>Associates the files in <i>maintSpec</i> with the target database (see Example 2–38 on page 2-63). Table 2–2 explains the RMAN behavior when different options are specified.</p> <p>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 <i>standby1</i> to database <i>prod</i>, then the recovery catalog has only one record for this file rather than two.</p> <p>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 <i>standby1</i> to database <i>prod</i>, the recovery catalog does not retain historical metadata about the database with which these files were previously associated. However, you can unregister database <i>standby1</i> and connect RMAN again to <i>standby1</i>, but in this case the recovery catalog will be updated with all metadata from the <i>standby1</i> control file.</p>
TO <i>db_unique_name</i>	Associates the files in <i>maintSpec</i> 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	<p>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.</p> <p>The database to which RMAN is connected must be a single-instance database and must not be a physical standby database.</p>
ALL	Changes only open failures.
CRITICAL	Changes only critical failures.

Syntax Element	Description
HIGH	Changes only failures with HIGH priority.
LOW	Changes only failures with LOW priority.
<i>failnum</i>	Changes only the specified failure.
EXCLUDE FAILURE <i>failnum</i>	Excludes the specified failures from the change.

Examples

Example 2-34 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 uncatlog the backup because you plan to move it back to its original location when space has been freed up on the disk. 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 2-35 Uncataloging and Recataloging Archived Redo Logs

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

```

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

```

Example 2-36 Changing a Database Backup into an Archival Backup

This example changes a consistent backup into an archival backup, which you plan to store offsite. Because the database is consistent and therefore requires no recovery, you do not need to save archived redo logs with the backup. The example specifies KEEP FOREVER to indicate that the backup should never become obsolete.

```

CONNECT TARGET /
CONNECT CATALOG rman/password@catdb
CHANGE BACKUP TAG 'consistent_db_bkup'
KEEP FOREVER;

```


Example 2-37 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. 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 2-38 Associating Backups with a New Database in a Data Guard Environment

Assume that `standby1`, `standby2`, and `standby3` are standby databases in a Data Guard environment. You are planning to remove `standby1` from your environment, so you want to association the `standby1` backups with your primary database, which is `prod`. You are also planning to remove `standby3` from your environment, so you want to associated the `standby3` backups with `standby2`. You execute the following commands:

```
CONNECT TARGET SYS/password@prod
CONNECT CATALOG rman/password@catdb
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 2-39 Updating a DB_UNIQUE_NAME in the Recovery Catalog

Assume that a standby database has the `DB_UNIQUE_NAME` initialization parameter setting of `dgrdbms4`. You shut down the standby instance, change the `DB_UNIQUE_NAME` initialization parameter to `sfrdbms4`, and restart the instance. Later, to update the recovery catalog to reflect the changed unique name, you connect RMAN to the primary database as `TARGET`, connect to the recovery catalog, and execute the `CHANGE` command as follows:

```
CONNECT TARGET SYS/password@prod
CONNECT CATALOG rman/password@catdb
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, a connection as `TARGET` 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, you can run `CONFIGURE CHANNEL DEVICE TYPE sbt CLEAR` but not `CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 5M CLEAR`.

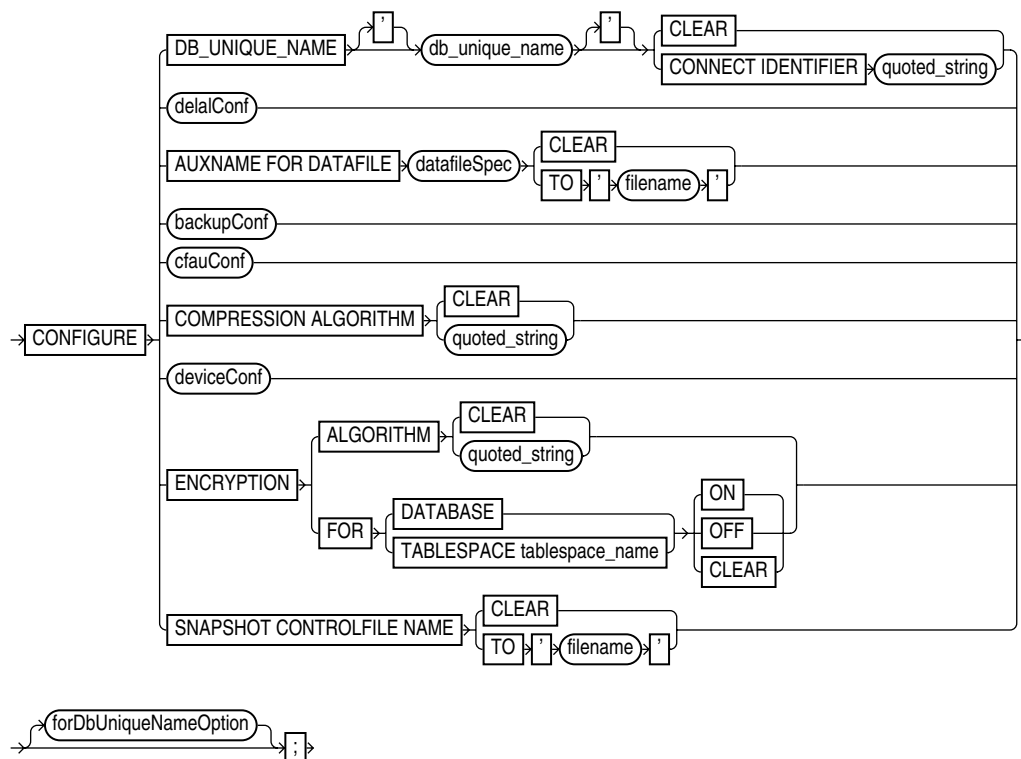
RMAN Configuration in a Data Guard Environment

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.

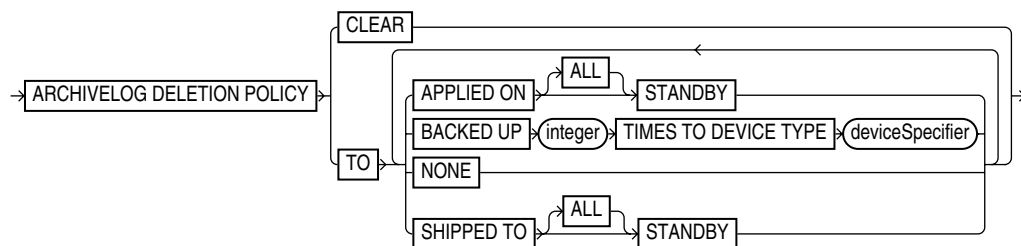
Syntax

configure::=



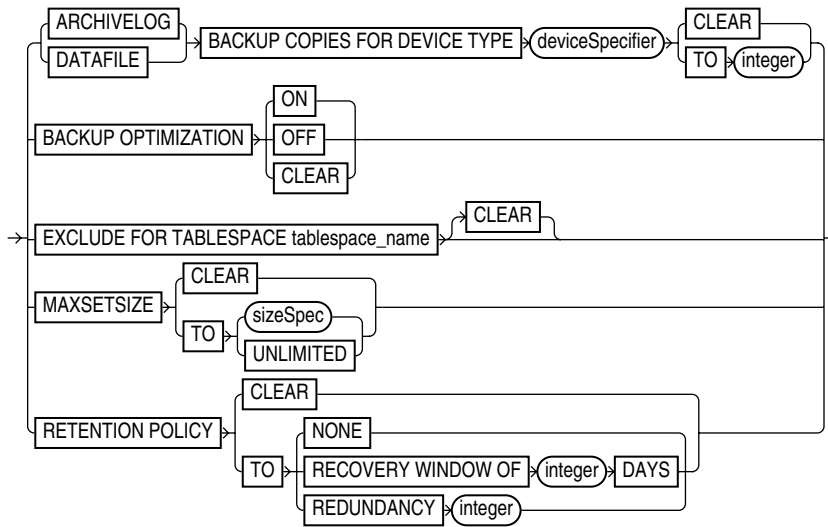
(*datafileSpec::=* on page 3-14, *backupConf::=* on page 2-66, *cfauConf::=* on page 2-66, *deviceConf::=* on page 2-66, *forDbUniqueNameOption::=* on page 3-19)

delalConf::=



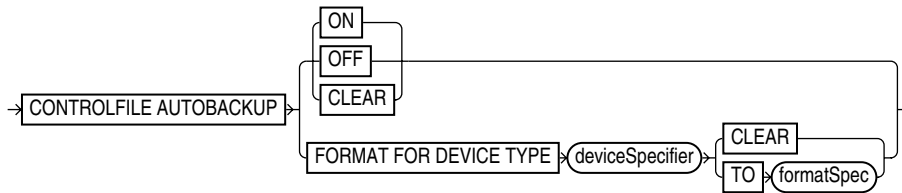
(*deviceSpecifier::=* on page 3-15)

backupConf::=



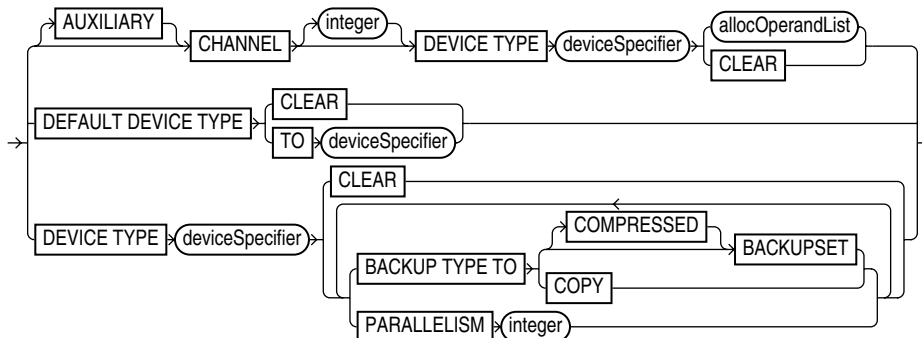
(*deviceSpecifier::=* on page 3-15, *sizeSpec::=* on page 2-66)

cfauConf::=



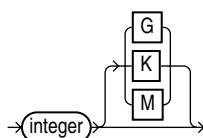
(*deviceSpecifier::=* on page 3-15, *formatSpec::=* on page 3-22)

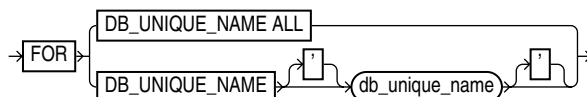
deviceConf::=



(*deviceSpecifier::=* on page 3-15, *allocOperandList::=* on page 3-2)

sizeSpec::=



forDbUniqueNameOption::=**Semantics****configure**

Syntax Element	Description
DB_UNIQUE_NAME	Specifies the net service name for the physical standby database specified by DB_UNIQUE_NAME. The connection identifier must not include the database username and password.
db_unique_name	
{CLEAR	RMAN must also be connected to the primary database as TARGET. RMAN must be connected to a recovery catalog.
CONNECT IDENTIFIER	
'connect_string' }	<p>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 <i>db_unique_name</i> database. For example, assume that a standby database has the unique name <i>standby1</i> and the net service name <i>sby1</i>. 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 <i>sby1</i> when it needs to make an Oracle Net connection to <i>standby1</i>.</p> <p>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.</p> <p>For a sample use case, 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 <i>standby_new</i>. However, you have not yet connected RMAN as TARGET to <i>standby_new</i>. 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.</p> <p>Note: If the database specified by CONFIGURE DB_UNIQUE_NAME is not registered in the recovery catalog, then RMAN implicitly registers it.</p>
<i>delalConf</i>	Configures an archived redo log deletion policy.

Syntax Element	Description
AUXNAME FOR DATAFILE <i>datafileSpec</i> TO ' <i>filename</i> '	<p>Configures the auxiliary filename for the specified target datafile to '<i>filename</i>' (see Example 2-44 on page 2-81). Specify CLEAR to unspecify the auxiliary filename.</p> <p>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.</p> <p>For example, use this command during TSPITR if the datafiles are on raw disk and you need to 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.</p> <p>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.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to perform RMAN TSPITR, and <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to duplicate a database with RMAN</p>
<i>backupConf</i>	Configures default backup options such as duplexing, optimization, excluding tablespaces, backup set sizes, and retention policies.
<i>cfauConf</i>	Configures control file autobackup settings.
COMPRESSION ALGORITHM ' <i>algorithm_name</i> '	<p>Specifies the algorithm that RMAN uses to create compressed backup sets. The default compression algorithm is BZIP2.</p> <p>The supported algorithms are ZLIB and BZIP2. ZLIB is optimized for CPU efficiency. BZIP2 is optimized for maximum compression. BZIP2 consumes more CPU resource than ZLIB, but will usually produce more compact backups. The COMPATIBLE initialization parameter must be set to 11.0.0 or higher for ZLIB compression.</p> <p>Note: The V\$RMAN_COMPRESSION_ALGORITHM view describes supported algorithms.</p>
<i>deviceConf</i>	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	<p>Specifies transparent-mode encryption settings for the database or tablespaces.</p> <p>This configuration applies unless overridden with the SET command. Options specified for an individual tablespace take precedence over options specified for the whole database.</p> <p>Note: To create encrypted RMAN backups on disk, the database must have the Advanced Security Option. To create encrypted backups directly on tape, RMAN must use Oracle Secure Backup as its media manager, but the Advanced Security Option is not required. Note that RMAN issues an ORA-19916 error if you attempt to create encrypted RMAN backups using an SBT library other than Oracle Secure Backup.</p>
ALGORITHM ' <i>algorithm_name</i> '	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.

Syntax Element	Description
FOR DATABASE [ON OFF CLEAR]	<p>Specifies whether to enable transparent encryption for the entire database. The options are as follows:</p> <ul style="list-style-type: none"> ■ ON enables encryption for all database files. ■ OFF turns off encryption for all database files. ■ CLEAR restores the default setting of OFF. <p>Note: You must use the <code>SET ENCRYPTION IDENTIFIED BY</code> command to enable password encryption.</p>
FOR TABLESPACE <i>tablespace_name</i> [ON OFF CLEAR]	<p>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:</p> <ul style="list-style-type: none"> ■ ON enables encryption for the specified tablespace unless <code>SET ENCRYPTION OFF FOR ALL TABLESPACES</code> is used. ■ OFF disables encryption for the specified tablespace unless <code>SET ENCRYPTION ON FOR ALL TABLESPACES</code> is used. ■ CLEAR means that encryption for the specified tablespace is determined by the current default for the whole database. <p>Note: You must use the <code>SET ENCRYPTION IDENTIFIED BY</code> command to enable password encryption.</p>
SNAPSHOT CONTROLFILE NAME TO ' <i>filename</i> '	<p>Configures the snapshot control file name and location to '<i>filename</i>'. If you run <code>CONFIGURE SNAPSHOT CONTROLFILE NAME CLEAR</code>, then RMAN sets the snapshot control file name to its default.</p> <p>The default value for the snapshot control file name is platform-specific and dependent on the Oracle home. For example, the default on some UNIX system is <code>?/dbs/snapcf_@.f</code>. If you clear the control file name, and if you change the Oracle home, then the default location of the snapshot control file changes as well.</p> <p>Note that 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 <code>DUPLICATE</code> 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.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> for more information about snapshot control files</p>

Syntax Element	Description
<i>forDbUniqueNameOption</i>	<p>Creates an RMAN configuration in the recovery catalog for the database in a Data Guard environment specified by <code>DB_UNIQUE_NAME</code>. You can specify a single database with <code>db_unique_name</code> or use <code>ALL</code> for all databases in the recovery catalog that share the DBID of the target database (or DBID specified by the <code>SET DBID</code> command).</p> <p>Oracle recommends that you connect RMAN to a recovery catalog when performing operations in a Data Guard environment. RMAN must be connected as <code>TARGET</code> to a mounted or open database (which can be a primary or standby database), <i>or</i> you must identify the target database with the <code>SET DBID</code> command. Thus, you can use this clause to create a persistent configuration for a standby database <i>without</i> connecting as <code>TARGET</code> 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 2-46 on page 2-81).</p> <p>When you specify <code>FOR DB_UNIQUE_NAME</code>, RMAN directly updates the configuration metadata in the recovery catalog. When RMAN connects as <code>TARGET</code> to a database whose configurations were changed with <code>FOR DB_UNIQUE_NAME</code>, RMAN updates the mounted control file with the configuration metadata from the recovery catalog.</p> <p>Note: It is possible to run <code>CONFIGURE</code> locally on a standby database and then run <code>CONFIGURE FOR DB_UNIQUE_NAME</code> for the same database while not connected to it as <code>TARGET</code>. In this case, the configuration in the recovery catalog overrides the configuration in the control file for the specific database.</p>

delalConf

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

Syntax Element	Description
<code>ARCHIVELOG DELETION POLICY</code>	<p>Determines when archived redo log files are eligible for deletion.</p> <p>The archived log deletion policy applies to all log archiving destinations, including the flash recovery area. Note that the policy does not apply to archived logs in backup sets.</p> <p>Only archived redo logs in the flash recovery area are automatically deleted by the database. You can execute the <code>BACKUP . . . DELETE INPUT</code>, <code>DELETE ARCHIVELOG</code>, or <code>DELETE OBSOLETE</code> commands to delete logs manually from log archiving destinations, including the recovery area. If <code>FORCE</code> is not specified on the deletion commands, then these deletion commands obey the archived log deletion policy. If <code>FORCE</code> is specified, then the deletion commands ignore the archived log deletion policy.</p> <p>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 logs needed by Oracle Streams.</p> <p>Note: The deletion policy does not apply to foreign archived redo logs, 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 logs they have a different DBID. Foreign archived redo logs cannot be backed up or restored on a logical standby database.</p>

Syntax Element	Description
TO APPLIED ON [ALL] STANDBY	<p>Specifies that archived redo logs are eligible for deletion if <i>both</i> of the following conditions are met:</p> <ul style="list-style-type: none"> ■ The archived redo logs 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. <p>Which remote destinations are considered depends on the following criteria:</p> <ul style="list-style-type: none"> ■ If you do not specify ALL, then archived logs are eligible for deletion after being applied to all mandatory remote destinations. ■ If you specify ALL, then archived logs are eligible after being applied or consumed on all remote destinations, whether mandatory or not. <p>For example, standby database <i>sby1</i> may be the only remote destination receiving logs, but other remote destinations may apply logs by referring to the same location on <i>sby1</i>. With ALL, <i>sby1</i> marks the log on the primary database as consumed as soon as it is not required at <i>sby1</i>, 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.</p> <p>Note: It is invalid to specify the TO APPLIED clause in combination with either NONE or the TO SHIPPED clause.</p> <p>See Also: <i>Oracle Data Guard Concepts and Administration</i> for details</p>
BACKED UP <i>integer</i> TIMES TO DEVICE TYPE <i>deviceSpecifier</i>	<p>Specifies that archived redo logs are eligible for deletion if <i>both</i> of the following conditions are met:</p> <ul style="list-style-type: none"> ■ 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. <p>If you configure the deletion policy with this clause, then a BACKUP ARCHIVELOG command copies the logs unless <i>integer</i> backups already exist on the specified device type. If <i>integer</i> 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 <i>integer</i> TIMES clause on the BACKUP ARCHIVELOG command. You can override this deletion policy by specifying FORCE option on the BACKUP command.</p> <p>See Also: <i>deviceSpecifier</i> on page 3-15</p>

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

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 be potentially 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.	<p>If a recovery window-based retention policy is enabled, then whether RMAN skips a datafile depends on the backup media.</p> <p>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.</p> <p>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.</p> <p>If a retention policy is enabled with <code>CONFIGURE RETENTION POLICY TO REDUNDANCY <i>r</i></code>, then RMAN skips backups only if at least <i>n</i> backups of an identical file exist on the specified device, where $n=r+1$.</p> <p>If no retention policy is enabled, then RMAN skips a backup only if at least <i>n</i> backups of an identical file exist on the specified device. RMAN searches for values of <i>n</i> in this order of precedence (that is, values higher on the list override values lower on the list):</p> <ol style="list-style-type: none"> 1. <code>BACKUP . . . COPIES <i>n</i></code> 2. <code>SET BACKUP COPIES <i>n</i></code> 3. <code>CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE . . . TO <i>n</i></code> 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.	<p>RMAN skips a backup only if at least <i>n</i> backups of an identical file exist on the specified device. RMAN searches for values of <i>n</i> in this order of precedence (that is, values higher on the list override values lower on the list):</p> <ol style="list-style-type: none"> 1. <code>BACKUP . . . COPIES <i>n</i></code> 2. <code>SET BACKUP COPIES <i>n</i></code> 3. <code>CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE . . . TO <i>n</i></code> 4. $n=1$
Backup set	The backup set must have the same record ID and stamp as an existing backup set.	<p>RMAN skips a backup only if at least <i>n</i> backups of an identical file exist on the specified device. By default, $n=1$. RMAN searches for other values of <i>n</i> in this order of precedence (that is, values higher on the list override values lower on the list):</p> <ol style="list-style-type: none"> 1. <code>BACKUP . . . COPIES <i>n</i></code> 2. <code>SET BACKUP COPIES <i>n</i></code> 3. $n=1$

Syntax Element	Description
<code>{ARCHIVELOG DATAFILE} BACKUP COPIES FOR DEVICE TYPE <i>deviceSpecifier</i> TO <i>integer</i></code>	<p>Specifies the number of copies of each backup set for <code>DATAFILE</code> (both datafiles and control files) or <code>ARCHIVELOG</code> files on the specified device type (see Example 2–41 on page 2-80). You can create from 1 (default) to 4 copies.</p> <p>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 <code>COPIES</code> is greater than 1, then the <code>BACKUP_TAPE_IO_SLAVES</code> initialization parameter must be enabled on the target database.</p> <p>Control file autobackups are never duplexed. Also, duplexing is not permitted in the flash recovery area.</p> <p>If duplexing is specified in the <code>BACKUP</code> command or in a <code>SET BACKUP COPIES</code> command, then the <code>CONFIGURE</code> setting is overridden.</p>

Syntax Element	Description
<p>BACKUP OPTIMIZATION [ON OFF CLEAR]</p>	<p>Toggles backup optimization ON or OFF (default). Specify CLEAR to return optimization to its default value of OFF.</p> <p>Backup optimization is enabled when all of the following conditions are met:</p> <ul style="list-style-type: none"> ■ 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. <p>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.</p> <p>For two files to be identical, their content must be exactly the same. See Table 2-3 on page 2-73 for an exhaustive explanation of when a file is considered identical. 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.</p> <p>Note: BACKUP . . . DELETE INPUT deletes all specified archived redo log whether or not optimization would skip these files during a backup.</p> <p>Note: You can override backup optimization with the FORCE option of the BACKUP command.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> for a description of how RMAN determines that it can skip the backup of a file</p>
<p>EXCLUDE FOR TABLESPACE <i>tablespace_name</i> [CLEAR]</p>	<p>Excludes the specified tablespace from BACKUP DATABASE commands (see Example 2-43 on page 2-81). Note that you cannot exclude the SYSTEM tablespace.</p> <p>By default, each tablespace is not excluded, that is, the exclude functionality is disabled. The exclusion is stored as an attribute of the tablespace, not the individual datafiles, so the exclusion applies 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.</p> <p>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.</p>
<p>MAXSETSIZE</p>	<p>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.</p> <p>Note: This option is ignored by BACKUP AS COPY.</p>
<p>TO <i>sizeSpec</i></p>	<p>Specifies the maximum size of each backup set as <i>integer</i> gigabytes, kilobytes, or megabytes.</p>
<p>TO UNLIMITED</p>	<p>Specifies no size limit for backup sets.</p>

Syntax Element	Description
RETENTION POLICY	<p>Specifies a persistent, ongoing policy for backup sets and copies that RMAN marks as obsolete, that is, not needed and eligible for deletion.</p> <p>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 flash recovery area when space is needed. RMAN does not automatically delete obsolete files outside the flash recovery area: you must manually execute <code>DELETE OBSOLETE</code> to remove them.</p> <p>For backups, the basic unit of the retention policy is a backup set (not a backup piece) or image copy. For example, <code>BACKUP AS BACKUPSET COPIES 4 TABLESPACE users</code> creates a single backup set that is duplexed into four identical backup pieces. The retention policy considers this as <i>one</i> backup, not four separate backups.</p> <p>Note: Use the <code>CLEAR</code> option to return <code>RETENTION POLICY</code> to its default of <code>REDUNDANCY 1</code>.</p>
TO NONE	<p>Disables the retention policy feature. RMAN does not consider any backup sets or copies as obsolete.</p>
TO RECOVERY WINDOW OF <i>integer</i> DAYS	<p>Specifies a time window in which RMAN should be able to recover the database.</p> <p>The window stretches from the current time (<code>SYSDATE</code>) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is <code>SYSDATE - integer</code> days in the past.</p> <p>Note: The <code>REDUNDANCY</code> and <code>RECOVERY WINDOW</code> options are mutually exclusive. Only one type of retention policy can be in effect at any time.</p>
TO REDUNDANCY <i>integer</i>	<p>Specifies that RMAN should retain <i>integer</i> backups of each datafile and control file. The default retention policy setting is <code>REDUNDANCY 1</code>.</p> <p>If more than <i>integer</i> 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 logs and log backups older than this backup as obsolete. The <code>DELETE OBSOLETE</code> command removes obsolete datafile backups (full or incremental), control file backups, and archived log backups or image copies.</p> <p>Note: The <code>REDUNDANCY</code> and <code>RECOVERY WINDOW</code> options are mutually exclusive. Only one type of retention policy can be in effect at any time.</p>

cfauConf

This subclause creates persistent configurations relating to control file autobackups.

Syntax Element	Description
CONTROLFILE AUTOBACKUP	<p>Controls the control file autobackup feature.</p> <p>Note: Oracle recommends that you enable control file autobackup feature when using RMAN without a recovery catalog.</p>

Syntax Element	Description
ON	<p>Specifies that RMAN should perform a control file autobackup in the following circumstances:</p> <ul style="list-style-type: none"> ■ After every <code>BACKUP</code> or <code>CREATE CATALOG</code> command issued at the RMAN prompt. ■ Whenever a <code>BACKUP</code> command within a <code>RUN</code> block is followed by a command that is not <code>BACKUP</code>. ■ At the end of every <code>RUN</code> block if the last command in the block was <code>BACKUP</code>. ■ After structural changes for databases in <code>ARCHIVELOG</code> mode. The autobackup after structural changes does not occur for databases in <code>NOARCHIVELOG</code> mode. <p>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 <code>CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK</code> to set a nondefault disk location.</p> <p>If you do not use RMAN with a recovery catalog, then control file autobackups are recommended.</p> <p>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.</p> <p>The default location for the autobackup on disk is the flash 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 <code>%F</code>. You can change the location and filename format with the <code>CONFIGURE CONTROLFILE AUTOBACKUP FORMAT</code> and <code>SET CONTROLFILE AUTOBACKUP FORMAT</code> commands.</p> <p>You cannot configure RMAN to duplex the control file autobackup, that is, write the autobackup to multiple locations. To create multiple control file backups, you can make the last command in your backup job a <code>BACKUP CURRENT CONTROLFILE FORMAT</code> command, which will back up the control file to the specified <code>FORMAT</code> location and then execute an autobackup.</p> <p>Note: The <code>SET CONTROLFILE AUTOBACKUP FORMAT</code> command, which you can specify either within a <code>RUN</code> block or at the RMAN prompt, overrides the configured autobackup format in the session only. The order of precedence is:</p> <ol style="list-style-type: none"> 1. <code>SET</code> within a <code>RUN</code> block 2. <code>SET</code> at RMAN prompt 3. <code>CONFIGURE CONTROLFILE AUTOBACKUP FORMAT</code> <p>You can configure the autobackup format even when <code>CONFIGURE CONTROLFILE AUTOBACKUP</code> is set to <code>OFF</code>, but RMAN does not generate autobackups in this case. For RMAN to make autobackups, you must set <code>CONFIGURE CONTROLFILE AUTOBACKUP</code> to <code>ON</code>.</p>
OFF	<p>Disables the autobackup feature (default).</p> <p>When this command is <code>OFF</code>, any <code>BACKUP</code> 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.</p>
CLEAR	<p>Returns this configuration to its default setting of <code>OFF</code>.</p>

Syntax Element	Description
FORMAT FOR DEVICE TYPE <i>deviceSpecifier</i> TO <i>formatSpec</i>	<p>Configures the default location and filename format for the control file autobackup on the specified device type (see Example 2-45 on page 2-81).</p> <p>By default, the format is <code>%F</code> for all devices. Any default format string specified with <code>CONFIGURE</code> must include the <code>%F</code> substitution variable. Use of any other substitution variable is an error. Specify <code>CLEAR</code> to return the format to the default <code>%F</code>.</p> <p>If a flash recovery area is enabled, and if the format is the default '<code>%F</code>', then RMAN creates the autobackup in the recovery area in a directory named <code>autobackup</code>. Otherwise, the default autobackup location is an operating system-specific location (<code>?/dbs</code> on UNIX, Linux, and Windows).</p> <p>The string <code># default</code> in the output of the <code>SHOW</code> command indicates when RMAN is using the default format. If you manually configure the disk format to '<code>%F</code>', 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 <code>CONFIGURE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR</code>.</p> <p>The <i>formatSpec</i> can specify an Automatic Storage Management disk group. The following example configures a channel for an ASM disk group:</p> <pre>CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK TO '+dgroup1';</pre> <p>See Also: <i>formatSpec</i> on page 3-22 for the semantics of the <code>%F</code> substitution variable</p>

deviceConf

This subclause creates persistent configurations relating to channels and devices.

Note that 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 numerical order, starting with 1 and ending with the parallelism setting (`CONFIGURE DEVICE TYPE . . . PARALLELISM n`).

To run `BACKUP` or `RESTORE` 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.

Syntax Element	Description
[AUXILIARY] CHANNEL [integer] DEVICE TYPE <i>deviceSpecifier</i>	<p>Specifies the standard or AUXILIARY channel that you are configuring or clearing, as well as the device type of the channel.</p> <p>Note: Channels allocated with <code>ALLOCATE CHANNEL</code> within a <code>RUN</code> command override configured automatic channels.</p> <p>Either configure a generic channel or specify a channel number, where <i>integer</i> is less than 255. See Example 2-43 on page 2-81 for an illustration of numbered channels.</p> <p>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.</p> <p>You must specify at least one channel option. For example, you cannot issue a command such as <code>CONFIGURE CHANNEL 2 DEVICE TYPE DISK</code>, but you can issue a command such as <code>CONFIGURE CHANNEL 2 DEVICE TYPE DISK MAXPIECESIZE 2500K</code>.</p> <p>For generic channels of a specified device type, a new command erases previous settings for this device type. Assume that you run these commands:</p> <pre>CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 1G; CONFIGURE CHANNEL DEVICE TYPE sbt FORMAT 'bkup_%U';</pre> <p>The second command erases the <code>MAXPIECESIZE</code> setting of the first command.</p> <p>Note: RMAN does not simultaneously allocate automatic channels for multiple device types in the <code>BACKUP</code> command.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to configure automatic channels specified by channel number</p>
<i>allocOperandList</i>	<p>Specifies control options for the configured channel.</p> <p>If you configure channels by using the nondefault <code>CONNECT</code> or <code>PARMS</code> 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.</p> <p>Note that the <code>FORMAT</code> parameter can specify an Automatic Storage Management disk group. The following example configures a channel for an ASM disk group:</p> <pre>CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '+dgroup1';</pre> <p>See Also: <i>allocOperandList</i> on page 3-2</p>
CLEAR	<p>Clears the specified channel. For example, <code>CONFIGURE CHANNEL 1 DEVICE TYPE DISK CLEAR</code> returns only channel 1 to its default, whereas <code>CONFIGURE CHANNEL DEVICE TYPE DISK CLEAR</code> returns the generic disk channel to its default. Note that you cannot specify any other channel options (for example, <code>PARMS</code>) when you specify <code>CLEAR</code>.</p>
DEFAULT DEVICE TYPE TO <i>deviceSpecifier</i>	<p>Specifies the default device type for automatic channels. By default, <code>DISK</code> is the default device type. <code>CLEAR</code> returns the default device type to <code>DISK</code>.</p> <p>By default, the <code>BACKUP</code> command only allocates channels of the default device type. For example, if you configure automatic channels for <code>DISK</code> and <code>sbt</code> and set the default device type to <code>sbt</code>, then RMAN only allocates tape channels when you run the <code>BACKUP DATABASE</code> command. You can override this behavior either by manually allocating channels in a <code>RUN</code> command, or by specifying <code>DEVICE TYPE</code> on the <code>BACKUP</code> command itself (see Example 2-41 on page 2-80).</p> <p>The <code>RESTORE</code> command allocates automatic channels of all configured device types, regardless of the default device type. The <code>RESTORE</code> command obeys the <code>PARALLELISM</code> setting for each configured device type.</p>

Syntax Element	Description
DEVICE TYPE <i>deviceSpecifier</i>	<p>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.</p> <p>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:</p> <pre>CONFIGURE DEVICE TYPE sbt PARALLELISM 1; BACKUP DEVICE TYPE sbt DATABASE;</pre> <p>In effect, RMAN does the following when executing this backup:</p> <pre>RUN { ALLOCATE CHANNEL ORA_SBT_TAPE_1 DEVICE TYPE sbt; BACKUP DATABASE; }</pre>
BACKUP TYPE TO [[COMPRESSED] BACKUPSET COPY]	<p>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.</p> <p>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.</p> <p>The default location for disk backups is the flash recovery area, if one is configured; otherwise, RMAN stores backups in a platform-specific location. The default format for backup filenames is %U.</p>
PARALLELISM <i>integer</i>	<p>Configures the number of automatic channels of the specified device type allocated for RMAN jobs. By default, PARALLELISM is set to 1.</p> <p>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.</p> <p>Suppose you set PARALLELISM for disk backups to 2 (see Example 2-42 on page 2-80). 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.</p> <p>Note: If you configure <i>n</i> manually numbered channels, then the PARALLELISM setting can be greater than or less than <i>n</i>. For example, you can manually number 10 automatic channels and configure PARALLELISM to 2 or 12.</p> <p>To change the parallelism for a device type to <i>n</i>, run a new CONFIGURE DEVICE TYPE . . . PARALLELISM <i>n</i> command. For example, you can change configure PARALLELISM to 3 for sbt and then change it to 2 as follows:</p> <pre>CONFIGURE DEVICE TYPE sbt PARALLELISM 3; CONFIGURE DEVICE TYPE sbt PARALLELISM 2;</pre>

Examples

Example 2-40 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 2-41 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 logs 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 2-42 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 2-43 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.

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

Example 2-44 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 2-45 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 2-46 Creating Configurations for Standby Databases

Assume that database `prod` is associated with two standby databases with the `DB_UNIQUE_NAME` names `dgprod3` and `dgprod4`. The following commands connect to the target database `prod` and recovery catalog database `catdb`. The example configures the default device type for databases `dgprod3` and `dgprod4`.

```
CONNECT TARGET SYS/password@prod
CONNECT CATALOG rman/password@catdb
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 2-47 Optimizing Backups

This scenario illustrates the backup optimization behavior described in [Table 2-3](#) on page 2-73. Assume that backup optimization is disabled. At 9 a.m., you back up three copies of all existing archived logs 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.

CONNECT

Purpose

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

Note: When connecting from the command line, the password may be visible to other users on the system. Use the `CONNECT` command at the RMAN prompt to avoid this problem.

See Also: [RMAN](#) on page 2-232 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.

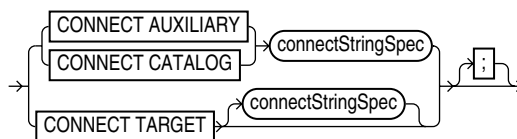
Usage Notes

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 a repository connection (as shown in [Example 2-49](#)).

Syntax

connect::=



(*connectStringSpec*::= on page 3-12)

Semantics

Syntax Element	Description
<code>CONNECT AUXILIARY</code>	Establishes a connection between RMAN and an auxiliary instance. Auxiliary instances are used with the <code>TRANSPORT TABLESPACE</code> and <code>DUPLICATE</code> commands, and during RMAN TSPITR.

Syntax Element	Description
CONNECT CATALOG	<p>Establishes a connection between RMAN and the recovery catalog database.</p> <p>If the recovery catalog is a virtual private catalog, then the RMAN client connecting to it must be at patch level 10.1.0.6 or 10.2.0.3. Oracle9i RMAN clients cannot connect to a virtual private catalog. This version restriction does not affect RMAN client connections to an Oracle Database 11g base recovery catalog, even if the base catalog has some virtual private catalog users.</p> <p>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 2-83).</p> <p>Note: Oracle recommends that you always use RMAN with a recovery catalog in a Data Guard environment.</p>
CONNECT TARGET	<p>Establishes a connection between RMAN and a target database.</p> <p>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.</p>
<i>connectStringSpec</i>	Specifies the connection information for the database.

Examples

Example 2-48 Connecting 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 NOCATALOG
RMAN> CONNECT TARGET SYS/password@prod1;
```

Example 2-49 Connecting in the Default NOCATALOG Mode

This example starts RMAN without specifying either CATALOG or NOCATALOG and then uses CONNECT TARGET to connect to database `prod1`. Because no CONNECT CATALOG 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/password@catdb

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

Example 2-50 Connecting to Target, Recovery Catalog, and Auxiliary Databases

This example connects to three databases. The example connects to the target database by means of operating system authentication and connects to the recovery catalog and auxiliary databases by means of password files.

```
% rman
```

```
RMAN> CONNECT TARGET;  
RMAN> CONNECT CATALOG rman/password@catdb;  
RMAN> CONNECT AUXILIARY SYS/password@dupdb;
```

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 10g 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.

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 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 reformatted 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.

Datatype Restrictions

`CONVERT` does not process user datatypes 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.

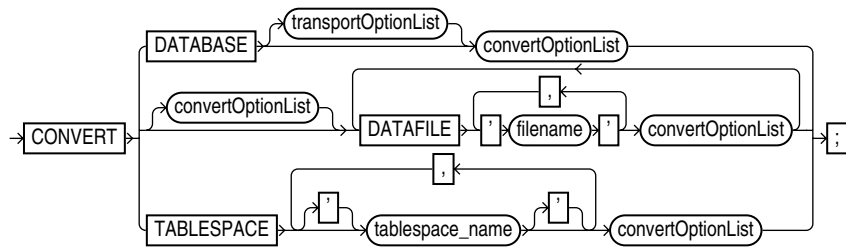
Before Oracle Database 10g, CLOBs in a variable-width character set such as `UTF8` were stored in an endian-dependent fixed width format. The `CONVERT` command does not perform conversions on these CLOBs. Instead, `RMAN` captures the endian format of each LOB column and propagates it to the target database. 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 10g 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

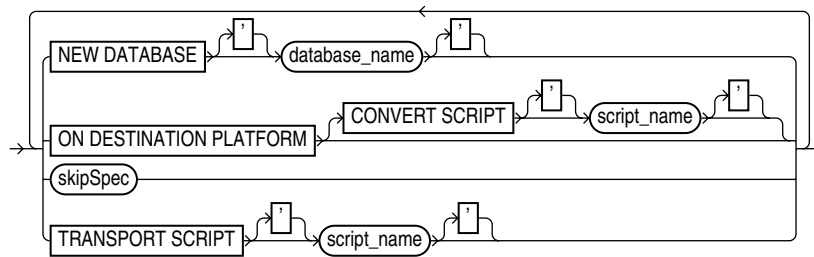
Syntax

convert::=



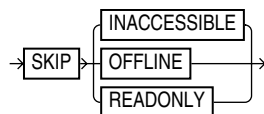
(*transportOptionList::=* on page 2-88, *convertOptionList::=* on page 2-88)

transportOptionList::=

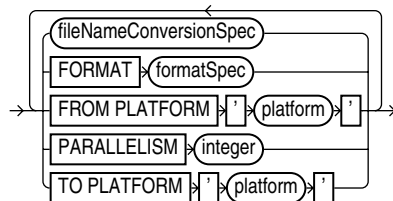


(*skipSpec::=* on page 2-88)

skipSpec::=

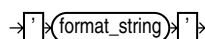


convertOptionList::=



(*fileNameConversionSpec::=* on page 3-17, *formatSpec::=* on page 2-88)

formatSpec::=



Semantics

convert

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

Syntax Element	Description
DATABASE	<p>Converts the datafiles to the format of the destination platform and ensures the creation of other required database files.</p> <p>You use <code>CONVERT DATABASE</code> to transport an entire database from a source platform to a destination platform. The source and destination platforms must have the same endian format.</p> <p>Depending on the situation, you can use <code>CONVERT DATABASE</code> on either the source or destination platform (see Example 2-54 on page 2-96). The following parts of the database are not transported directly:</p> <ul style="list-style-type: none"> ■ 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 <code>OPEN RESETLOGS</code> 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 <code>CONVERT DATABASE</code> output provides a list of objects that use the BFILE datatype, 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 <code>CONVERT DATABASE</code> output shows a list of affected objects, but you must redefine these objects on the target platform. See <i>Oracle Database Administrator's Guide</i> 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 <code>CONVERT DATABASE</code> includes a list of all user names and their associated privileges. Create a new password file on the target database with this information. See <i>Oracle Database Administrator's Guide</i> for more information on managing password files. <p>When using <code>CONVERT DATABASE</code>, RMAN detects the following problems and will not proceed until they are fixed:</p> <ul style="list-style-type: none"> ■ The database has active or in-doubt transactions. ■ The database has save undo segments. ■ The database <code>COMPATIBILITY</code> setting is below 10. ■ Some tablespaces have not been open read/write when the database <code>COMPATIBILITY</code> setting is 10 or higher. <p><i>transportOptionList</i> Specifies options that control the transport.</p> <p>See Also: <i>transportOptionList</i> on page 2-91</p>

Syntax Element	Description
<p>[<i>convertOptionList</i>]</p> <p>DATAFILE 'filename'</p> <p><i>convertOptionList</i></p>	<p>Specifies the name of a datafile to be transported into a destination database (see Example 2-52 on page 2-94).</p> <p>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 <i>Oracle Database Administrator's Guide</i> or from backups using the procedure described in <i>Oracle Database Backup and Recovery User's Guide</i>. Refer to that document before attempting to transport a tablespace across platforms.</p> <p>Use FROM PLATFORM in <i>convertOptionList</i> 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.</p> <p>You can use CONVERT DATAFILE without FROM PLATFORM or TO PLATFORM to move datafiles into and out of ASM (see Example 2-53 on page 2-95). 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:</p> <pre>SELECT NAME FROM V\$DATAFILE_COPY WHERE CONVERTED_FILE='YES' ;</pre> <p>The CONVERT DATAFILE syntax supports multiple format names, so that each datafile can have a separate format. The DATAFILE syntax supports <i>convertOptionList</i> both immediately following the CONVERT keyword and after each DATAFILE 'filename' clause. However, RMAN generates an error in the following situations:</p> <ul style="list-style-type: none"> ■ Any option in <i>convertOptionList</i> except FORMAT is specified more than once ■ Any option in <i>convertOptionList</i> except FORMAT is specified in the DATAFILE options list when multiple DATAFILE clauses are specified

Syntax Element	Description
<p>TABLESPACE <i>tablespace_name</i> <i>convertOptionList</i></p>	<p>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 2-51 on page 2-93).</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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 <i>Oracle Database Administrator's Guide</i> or from backups using the procedure described in <i>Oracle Database Backup and Recovery User's Guide</i>. Refer to that document before attempting to transport a tablespace across platforms.</p> <p>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.</p>
<p><i>convertOptionList</i></p>	<p>Specifies options that control the conversion.</p> <p>See Also: convertOptionList on page 2-92</p>

transportOptionList

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

Syntax Element	Description
<p>NEW DATABASE <i>database_name</i></p>	<p>Specifies the DB_NAME for the new database produced by the CONVERT DATABASE command.</p>
<p>ON DESTINATION PLATFORM</p>	<p>Generates a convert script of CONVERT DATAFILE commands (see CONVERT SCRIPT parameter) that you can run on the destination host to create the database.</p> <p>Note: When this option is specified, CONVERT generates a script but does not generate converted datafile copies.</p> <p>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.</p> <p>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.</p>

Syntax Element	Description
CONVERT SCRIPT <i>script_name</i>	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.
<i>skipSpec</i>	Specifies that CONVERT DATABASE should skip inaccessible, offline, or read-only datafiles during the conversion process.
TRANSPORT SCRIPT <i>script_name</i>	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
<i>fileNameConversionSpec</i>	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 2-123 to learn about restrictions related to ASM and Oracle Managed Files

Syntax Element	Description
FORMAT <i>formatSpec</i>	<p>Specifies the name template for the output files. See the <code>BACKUP AS COPY</code> command for the format values that are valid here.</p> <p>If the database to which RMAN is connected as <code>TARGET</code> uses a recovery area, then you must specify the <code>FORMAT</code> clause.</p> <p>You can use <code>CONVERT . . . FORMAT</code> without specifying <code>FROM PLATFORM</code> or <code>TO PLATFORM</code>. If you do not specify platforms, then running <code>CONVERT TABLESPACE</code> on the source database generates datafile copies that are not cataloged. If you run <code>CONVERT DATAFILE</code> on the destination database, and if the datafile copy already uses the same endianness, then the command generates another datafile copy.</p> <p>As shown in Example 2-53 on page 2-95, you can use <code>CONVERT DATAFILE . . . FORMAT</code> 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.</p>
FROM PLATFORM ' <i>platform</i> '	<p>Specifies the name of the source platform. If not specified, the default is the platform of the database to which RMAN is connected as <code>TARGET</code>.</p> <p>The specified platform must be one of the platforms listed in the <code>PLATFORM_NAME</code> column of <code>V\$TRANSPORTABLE_PLATFORM</code>. You must use the exact name of the source or target platform as a parameter to the <code>CONVERT</code> command. The following SQL statement queries supported Linux platforms:</p> <pre>SELECT PLATFORM_NAME, ENDIAN_FORMAT FROM V\$TRANSPORTABLE_PLATFORM WHERE UPPER(PLATFORM_NAME) LIKE 'LINUX%';</pre>
PARALLELISM <i>integer</i>	<p>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.</p>
TO PLATFORM ' <i>platform</i> '	<p>Specifies the name of the destination platform. If not specified, the default is the platform of the database to which RMAN is connected as <code>TARGET</code>.</p> <p>The specified platform must be one of the platforms listed in the <code>PLATFORM_NAME</code> column of <code>V\$TRANSPORTABLE_PLATFORM</code>. You must use the exact name of the source or target platform as a parameter to the <code>CONVERT</code> command. The following SQL statement queries supported Linux platforms:</p> <pre>SELECT PLATFORM_NAME, ENDIAN_FORMAT FROM V\$TRANSPORTABLE_PLATFORM WHERE UPPER(PLATFORM_NAME) LIKE 'LINUX%';</pre>

Examples

Example 2-51 Converting Tablespaces on the Source Platform

Suppose you need to 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/password@prodlin
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 2-52 Converting Datafiles on the Destination Platform

This example assumes that you want to convert the `finance` and `hr` tablespaces from a database on host `sun01` into a format usable 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)`.

Now use RMAN's `CONVERT` command to convert the datafiles to be transported to the destination host format and deposit the results in `/disk2/orahome/dbs`.

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.

```
CONNECT TARGET SYS/password@prodlin
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 2-53 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 2-54 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 entirety before attempting a database conversion.

Assume that you want to transport a database 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 Linux host `lin01` and uses CONVERT DATABASE NEW DATABASE to convert the datafiles and generate the transport script:

```
CONNECT TARGET SYS/password@lin01
CONVERT DATABASE
  NEW DATABASE 'newdb'
  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 Linux host `lin01` and executes CONVERT DATABASE ON DESTINATION PLATFORM:

```
CONNECT TARGET SYS/password@lin01
CONVERT DATABASE
  ON DESTINATION PLATFORM
  CONVERT SCRIPT '/tmp/convertdb/convertscript.rman'
  TRANSPORT SCRIPT '/tmp/convertdb/transportscript.sql'
  NEW DATABASE 'newdb'
  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 command also generates a transport script.

Example 2-55 Transporting a Database to a Different Platform and Storage Type

In this scenario, you have a database 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 executes CONVERT DATABASE to generate the necessary scripts:

```
CONNECT TARGET SYS/password@sun01
CONVERT DATABASE
  ON DESTINATION PLATFORM
  CONVERT SCRIPT '/tmp/convert_newdb.rman'
  TRANSPORT SCRIPT '/tmp/transport_newdb.sql'
  NEW DATABASE 'newdb'
  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_21g3905p_.dbf'
  FROM PLATFORM 'your_source_platform'
  FORMAT '+DATA/o1_mf_system_21g3905p_.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 need to 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_21g3905p_.dbf'
  FROM PLATFORM 'your_source_platform'
  FORMAT '+DATA/o1_mf_system_21g3905p_.dbf';
```

You then connect RMAN to the destination database, in this case aix01, and run the convert script to convert the datafiles. Afterward, you connect SQL*Plus to 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**, which is a database schema that contains RMAN metadata for a set of target databases. A **virtual recovery catalog** is a set of synonyms and views that enable user access to a subset of a base catalog.

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

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 the target database is not required.

The recovery catalog owner, whether the catalog is a base catalog or a virtual 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 recovery catalog, then the base recovery catalog owner must have used the `GRANT` command to grant either the `CATALOG` or `REGISTER` privilege (see [Example 2-57](#) on page 2-99).

See the `CONNECT CATALOG` description for restrictions for RMAN client connections to a virtual catalog when the RMAN client is from release Oracle Database 10g 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	<p>Creates a virtual private catalog in an existing recovery catalog.</p> <p>Run this command after connecting RMAN to the recovery catalog database as the virtual catalog user.</p> <p>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.</p>

Examples

Example 2–56 Creating a Recovery Catalog and Registering Databases

Assume that you start SQL*Plus and issue the following commands to create a user `catowner` in database `catdb` and grant the `catowner` user the `RECOVERY_CATALOG_OWNER` role:

```

SQL> CONNECT SYS/password@catdb AS SYSDBA
SQL> CREATE USER catowner IDENTIFIED BY oracle
      2 DEFAULT TABLESPACE cattbs
      3 QUOTA UNLIMITED ON cattbs;
SQL> GRANT recovery_catalog_owner TO catowner;
SQL> EXIT
  
```

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

```

RMAN> CONNECT CATALOG catowner/password@catdb
RMAN> CREATE CATALOG;
  
```

In the same RMAN session, you connect to the target database `prod1` and use the `REGISTER DATABASE` command to register `prod1` in the catalog and then exit RMAN:

```

RMAN> CONNECT TARGET SYS/password@prod1
RMAN> REGISTER DATABASE;
RMAN> EXIT
  
```

Example 2–57 Creating a Virtual Private Catalog

Assume that you created the recovery catalog and registered a database as shown in [Example 2–56](#). Now you want to create a virtual private catalog for database user `vpc1`. You create the `vpc1` user and grant recovery catalog ownership as follows:

```

SQL> CONNECT SYS/password@catdb AS SYSDBA
SQL> CREATE USER vpc1 IDENTIFIED BY apwd
      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 `vp1` for RMAN operations on database `prod1` (but not `prod2`):

```
RMAN> CONNECT CATALOG catowner/password@catdb
RMAN> GRANT CATALOG FOR DATABASE prod1 TO vp1;
RMAN> EXIT;
```

At this point the backup operator who will use virtual private catalog `vp1` is ready to create the virtual catalog. In the following example, the backup operator connects to the recovery catalog database as `vp1` and creates the virtual private catalog for `vp1`

```
RMAN> CONNECT CATALOG vp1/password@catdb
RMAN> CREATE VIRTUAL CATALOG;
RMAN> EXIT;
```

Because this operator eventually intends to use the virtual catalog with Oracle Database 10g 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 2-98). In the following example, the backup operator connects to the recovery catalog database as `vp1` and executes the PL/SQL procedure as follows:

```
SQL> CONNECT vp1/password@catdb
SQL> BEGIN
  2 catowner.DBMS_RVCAT.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 2-199 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

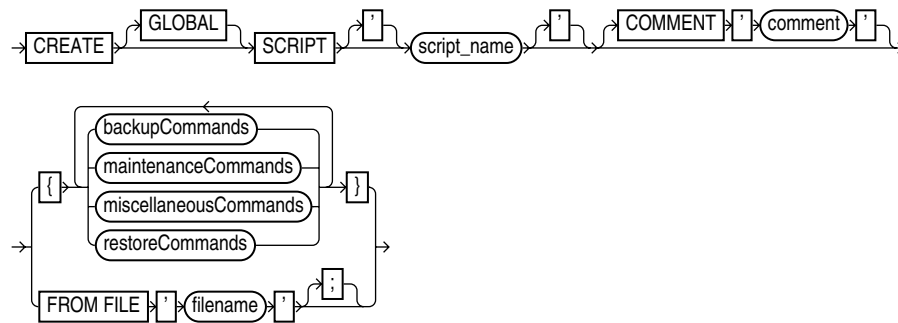
RMAN supports the use of substitution variables in a stored script. `&1` indicates where to place the first value, `&2` indicate where to place the second value, and so on. Special characters must be quoted.

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 2-232) or enter them when prompted (see [Example 2-60](#)).

Syntax

createScript::=



(*backupCommands::=* on page 2-238, *maintenanceCommands::=* on page 2-238, *miscellaneousCommands::=* on page 2-239, *restoreCommands::=* on page 2-239)

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 catalog.
SCRIPT <i>script_name</i>	Specifies the name of the script. Quotes must be used around the script name when the name contains either spaces or reserved words.
COMMENT ' <i>comment</i> '	Associates an explanatory comment with the stored script in the recovery catalog. The comment is used in the output of <code>LIST SCRIPT NAMES</code> .
<i>backupCommands</i>	Specifies commands to include in the stored script. The commands allowable within the brackets of the <code>CREATE SCRIPT 'script_name' {...}</code> command are the same commands supported within a <code>RUN</code> command. Any command that is legal within a <code>RUN</code> command is permitted in the stored script. The following commands are not legal within stored scripts: <code>RUN</code> , <code>@</code> , and <code>@@</code> .
<i>maintenanceCommands</i>	
<i>miscellaneousCommands</i>	
<i>restoreCommands</i>	
FROM FILE ' <i>filename</i> '	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 2-58 Creating a Local Stored Script

Assume that you want to create a local script for backing up database `prod`. This example connects to target database `prod` and a recovery catalog database as user `rman`. The example then creates a stored script called `backup_whole` and then uses `EXECUTE SCRIPT` to run it:

```

CONNECT TARGET SYS/password@prod
CONNECT CATALOG rman/password@catdb
CREATE SCRIPT backup_whole
COMMENT "backup whole database and archived redo logs"
{
    BACKUP
        INCREMENTAL LEVEL 0 TAG backup_whole
        FORMAT "/disk2/backup/%U"
}
  
```



```

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

```

Example 2–59 Creating a Global Stored Script

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

```

CONNECT TARGET SYS/password@prod
CONNECT CATALOG rco/password@catdb
CREATE GLOBAL SCRIPT global_backup_db
COMMENT "back up any database from the recovery catalog, with logs"
{
    BACKUP DATABASE PLUS ARCHIVELOG;
}
EXIT;

```

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

```

CONNECT TARGET SYS/password@prod2
CONNECT CATALOG rman/password@catdb
RUN { EXECUTE SCRIPT global_backup_db; }

```

Example 2–60 Creating a Stored Script That Uses Substitution Variables

The following example connects to the 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 in bold).

```

CONNECT TARGET /
CONNECT CATALOG rman/password@catdb
CREATE SCRIPT 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

```

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:

```

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 the 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	<p>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.</p> <p>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.</p> <p>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.</p>
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 2-21 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.

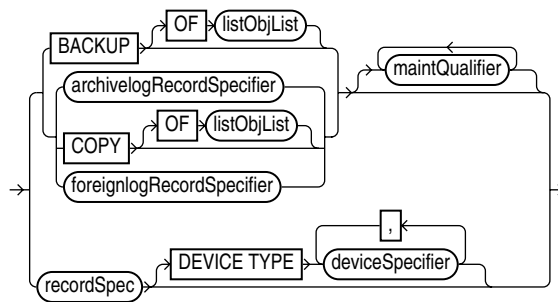
RMAN can only update the status of a backup from AVAILABLE to EXPIRED or DELETED when connected as TARGET to the database associated with the backup. If RMAN cannot delete a backup because it is not associated with the target database, then RMAN prompts you to perform the same CROSSCHECK operation for 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 ::=

```
→ CROSSCHECK (maintSpec) ;
```

maintSpec::=

(*listObjList::=* on page 3-28, *archivelogRecordSpecifier::=* on page 3-6, *foreignlogRecordSpecifier::=* on page 3-20, *maintQualifier::=* on page 3-31, *recordSpec::=* on page 3-36, *deviceSpecifier::=* on page 3-15)

Semantics

Syntax Element	Description
<i>maintSpec</i>	Crosschecks backups and copies. For <i>maintSpec</i> options, refer to the parameter descriptions in <i>maintSpec</i> on page 3-33.

Examples

Example 2-61 Crosschecking All Backups and Copies

This example, which assumes that the default configured channel is `DEVICE TYPE sbt`, crosschecks all backups and copies on tape 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 2-62 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 the 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.

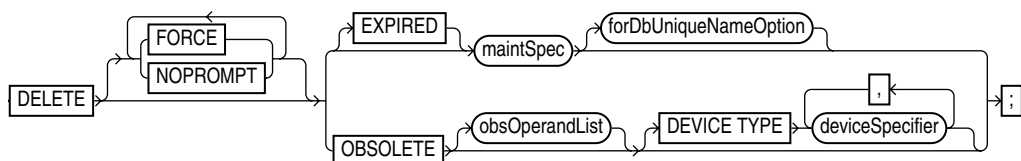
Backup Deletion in a Data Guard Environment

"[RMAN Backups in a Data Guard Environment](#)" on page 2-21 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.

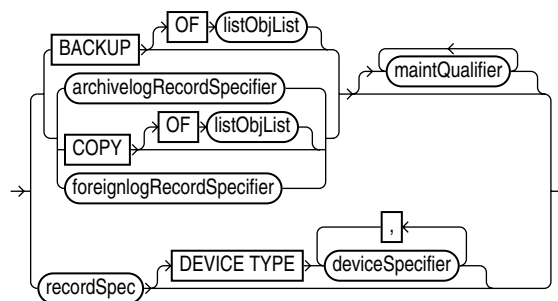
Syntax

delete::=

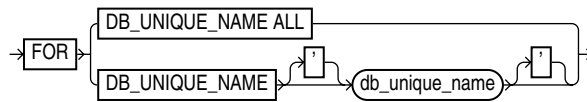


([maintSpec::=](#) on page 3-33, [obsOperandList::=](#) on page 3-35, [deviceSpecifier::=](#) on page 3-15)

maintSpec::=



([listObjList::=](#) on page 3-28, [archivelogRecordSpecifier::=](#) on page 3-6, [maintQualifier::=](#) on page 3-31, [deviceSpecifier::=](#) on page 3-15, [recordSpec::=](#) on page 3-36)

forDbUniqueNameOption::=**Semantics**

Syntax Element	Description
FORCE	<p>Deletes specified files—whether or not they exist on the media—and removes repository records (see Example 2–66 on page 2-112).</p> <p>RMAN ignores any I/O errors for the deleted objects. RMAN also ignores any <code>CONFIGURE ARCHIVELOG DELETION POLICY</code> settings. RMAN displays the number of deleted objects at the end of the job.</p>
NOPROMPT	<p>Deletes specified files without first listing the files or prompting for confirmation. The <code>DELETE NOPROMPT</code> command displays each item as it is deleted.</p>
EXPIRED	<p>Removes only files whose status in the repository is <code>EXPIRED</code> (see Example 2–63 on page 2-111). RMAN marks backups and copies as expired when you run a <code>CROSSCHECK</code> command and the files are absent or inaccessible. To determine which files are expired, run a <code>LIST EXPIRED</code> command.</p> <p>If for some reason a backup marked <code>EXPIRED</code> exists when you run the <code>DELETE EXPIRED</code> command, then RMAN does not delete the physical file.</p>
<i>maintSpec</i>	<p>Deletes backups and copies.</p> <p>You can set rules for the deletion with the <i>maintQualifier</i> clause. For example, you can delete archived logs that have already been backed up to tape (see Example 2–65 on page 2-111).</p> <p>Note: <code>DELETE ARCHIVELOG ALL</code> considers only the archived log deletion policy and does not consider the configured retention policy.</p> <p>See Also: <i>maintSpec</i> on page 3-33 and <i>maintQualifier</i> on page 3-31</p>
<i>forDbUniqueNameOption</i>	<p>Deletes the backups and copies in <i>maintSpec</i> that are exclusively associated with the specified <code>DB_UNIQUE_NAME</code> in a Data Guard environment.</p> <p>Note: The <code>FOR DB_UNIQUE_NAME</code> option is not allowed with the <code>DELETE OBSOLETE</code> command.</p> <p>If RMAN successfully deletes tape backups associated with the specified <code>DB_UNIQUE_NAME</code>, 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 <code>DELETE</code> operation for these files at the database that is associated with them.</p> <p>Note: You cannot use <code>FORCE</code> to override the default behavior and specify that 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 <code>CHANGE RESET DB_UNIQUE_NAME</code> command.</p> <p>See Also: <i>forDbUniqueNameOption</i> on page 3-19 for descriptions of the options in this clause</p>

Syntax Element	Description
OBSOLETE	<p>Deletes datafile backups and copies recorded in the RMAN repository that are obsolete, that is, no longer needed (see Example 2-64 on page 2-111). RMAN also deletes obsolete archived redo logs and log backups.</p> <p>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.</p> <p>RMAN first uses the options specified with <i>obsOperandList</i> to determine which files are obsolete. If you do not specify options in <i>obsOperandList</i>, then RMAN uses the options specified in <code>CONFIGURE RETENTION POLICY</code>.</p> <p>Note: <code>DELETE OBSOLETE</code> considers only the backup retention policy and does not use the configured archived log deletion policy to determine which logs are obsolete. In contrast, <code>DELETE ARCHIVELOG ALL</code> considers only the configured archived log deletion policy.</p> <p>Note: If you make a backup with the <code>KEEP UNTIL TIME</code> clause, then this backup becomes obsolete after the specified <code>KEEP</code> time passes and will be removed by <code>DELETE OBSOLETE</code>. RMAN does not consider the backup retention policy for archival backups whose <code>KEEP</code> time has expired.</p>
<i>obsOperandList</i>	<p>Specifies the criteria for determining which backups and copies are obsolete.</p> <p>See Also: <i>obsOperandList</i> on page 3-35</p>
DEVICE TYPE <i>deviceSpecifier</i>	<p>Restricts the deletion to obsolete backups and copies created on the specified device type only.</p> <p>See Also: <i>deviceSpecifier</i> on page 3-15</p>

Examples

Example 2-63 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 2-64 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 logs that are no longer needed.

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

Example 2-65 Deleting Archived Redo Logs 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 logs 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_.arc					
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 2-66 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.

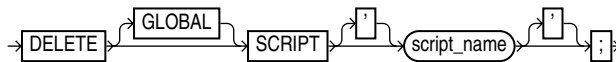
Usage Notes

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
<code>GLOBAL</code>	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 2-101 for an explanation of the difference between global and local scripts
<code>SCRIPT script_name</code>	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 2-67 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 logs

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 logs

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.

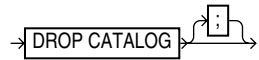
Note that you must use a different technique to drop a virtual catalog when using a 10.2 or earlier release of the RMAN client. Before dropping the virtual private catalog, the user must connect to the recovery catalog database as the virtual private catalog owner and execute the following PL/SQL procedure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

→ DROP CATALOG → 

Example

Example 2–68 Dropping a Virtual Private Catalog

Assume that you want to remove the virtual private catalog belonging to database user `vpw1`, but do not want to drop the base recovery catalog. This example connects to the recovery catalog database as `vpw1` 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 vpw1/password@catdb

connected to recovery catalog database

RMAN> DROP CATALOG;

recovery catalog owner is VPW1
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 the target database. The target database must be mounted exclusive and not open, and started in `RESTRICT` mode.

Syntax

dropDatabase::=



Semantics

Syntax Element	Description
<code>INCLUDING BACKUPS</code>	Deletes backup sets, proxy copies, image copies, and archived redo logs associated with the target database from all configured device types. Note: If you have been using a recovery catalog but run RMAN in <code>NOCATALOG</code> 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.
<code>NOPROMPT</code>	Does not prompt for confirmation before deleting the database.

Example

Example 2–69 Deleting a Database

In this example, you want to delete a test database called `test1` that is registered in the recovery catalog. You connect RMAN to the target and catalog databases, mount the database, and enable a restricted session. You then delete the database files, as well as all backups, copies, and archived logs associated with the database:

```

CONNECT TARGET SYS/password@test1
CONNECT CATALOG test1/password@catdb
STARTUP FORCE MOUNT
SQL 'ALTER SYSTEM ENABLE RESTRICTED SESSION';
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 logs from the primary database. A standby database does *not* get a new DBID.

RMAN can perform the duplication either directly from an open or mounted database or from pre-existing RMAN backups and copies.

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

RMAN must be connected as `TARGET` to the source database, which is the database that is being copied. The source database must be mounted or open. The source database must not be a standby database.

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 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) Little is considered the same platform as Linux IA (64-bit) Little. 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 configured or allocated auxiliary channels. These channels perform the work of the duplication on the auxiliary database instance. RMAN is connected as `TARGET` to the source database. 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 use the same channel configuration for the channels on the auxiliary instance. It is recommended that you manually allocate auxiliary channels instead.

The source host is the database on which the source database resides. The destination host is the database 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.

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.

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:

- If you are duplicating an encrypted tablespace, then you must manually copy the wallet to the duplicate database.
- If the duplicate database has an existing wallet, then you cannot copy the wallet from the source database to the duplicate 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 duplicate database.
- You cannot duplicate an encrypted tablespace across platforms with different endianness.

Prerequisites Specific to Backup-Based Duplication

When you execute `DUPLICATE` without `FROM ACTIVE DATABASE`, at least one auxiliary channel is required, but no normal channels are required in the source database.

When you duplicate the database from backups, all backups and archived redo logs used for creating and recovering the duplicate database must be accessible by the server session on the destination host. If the destination host is not the same as 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.

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 connecting 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.

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, and if it is not a standby database, then it must have been shut down consistently.

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.

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 2-6](#) shows which files from the source database are duplicated.

Table 2-6 Duplicated Files

Source Database Files	Active Database	Backup-Based
Control files	Copied from source database when <code>FOR STANDBY</code> specified; otherwise re-created	Restored from backups when <code>FOR STANDBY</code> specified; otherwise re-created
Datafiles	Copied from source database (unless excluded with a <code>SKIP</code> option)	Restored from backups (unless excluded with a <code>SKIP</code> option)
Tempfiles	Re-created (see " Tempfile Re-Creation " on page 2-123)	Re-created (see " Tempfile Re-Creation " on page 2-123)
Online redo log files	Re-created	Re-created
Standby redo log files	Re-created when <code>FOR STANDBY</code> specified and defined on primary database	Re-created when <code>FOR STANDBY</code> 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 <code>SPFILE</code> clause in dupOptionList on page 2-127)	Restored from backup if <code>SPFILE</code> clause is specified (see dupOptionList on page 2-127)
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 <code>PASSWORD FILE</code> option is specified	Not re-created
Backups and other files in flash recovery area	Not copied	Not copied

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

Note: The recovery catalog only knows about current read-only tablespaces. If a tablespace is currently read/write, but you use `UNTIL` to duplicate the database to a past SCN at which a tablespace was read-only, then this tablespace is *not* included in the duplicate database. Tablespaces that were read-only in the past are considered offline tablespaces and so are not included in the duplication.

The flash 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 [dupshbyOptionList](#) on page 2-126 or [dupOptionList](#) on page 2-127 for usage notes.

Backup-Based Duplication

In backup-based duplication of databases 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.

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.

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 flash 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 flash 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. OMF filenames are unique and generated by Oracle Database.

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`.

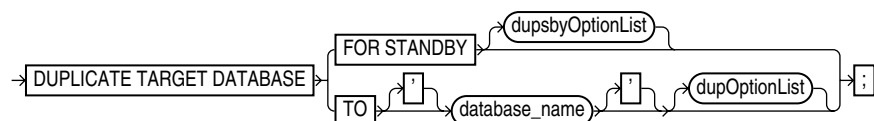
Supported options for naming online redo logs duplicated from Oracle-managed files are `DB_CREATE_FILE_DEST`, `DB_RECOVERY_FILE_DEST`, or `DB_CREATE_ONLINE_LOG_DEST_n`.

Tempfile Re-Creation

When using `DUPLICATE` with Oracle-managed files, RMAN re-creates tempfiles in the current `DB_CREATE_FILE_DEST`, either when the database is opened to become a primary or when it is opened read-only. When not using Oracle-managed files, RMAN uses `DB_FILE_NAME_CONVERT` to convert the tempfile names for the new database. When the standby or duplicate database is opened in read-only or read/write mode, Oracle automatically creates temporary files as needed, with the converted names based upon `DB_FILE_NAME_CONVERT`. To specify different filenames for the tempfiles, see the discussion of `SWITCH TEMPFILE` on page 2-262.

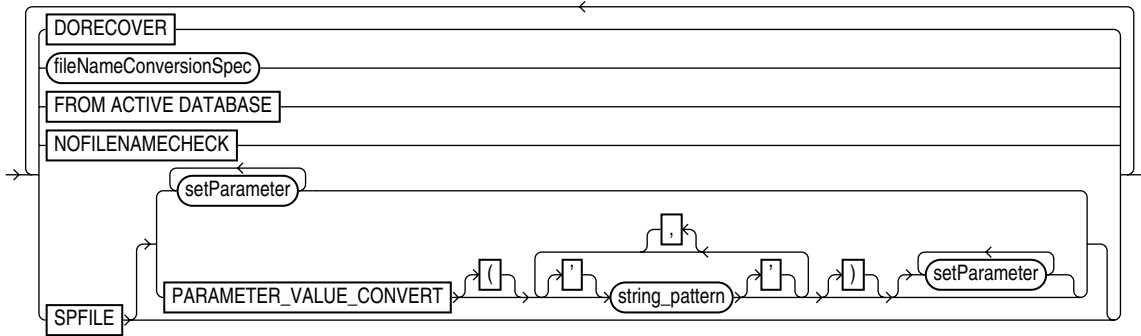
Syntax

`duplicate::=`



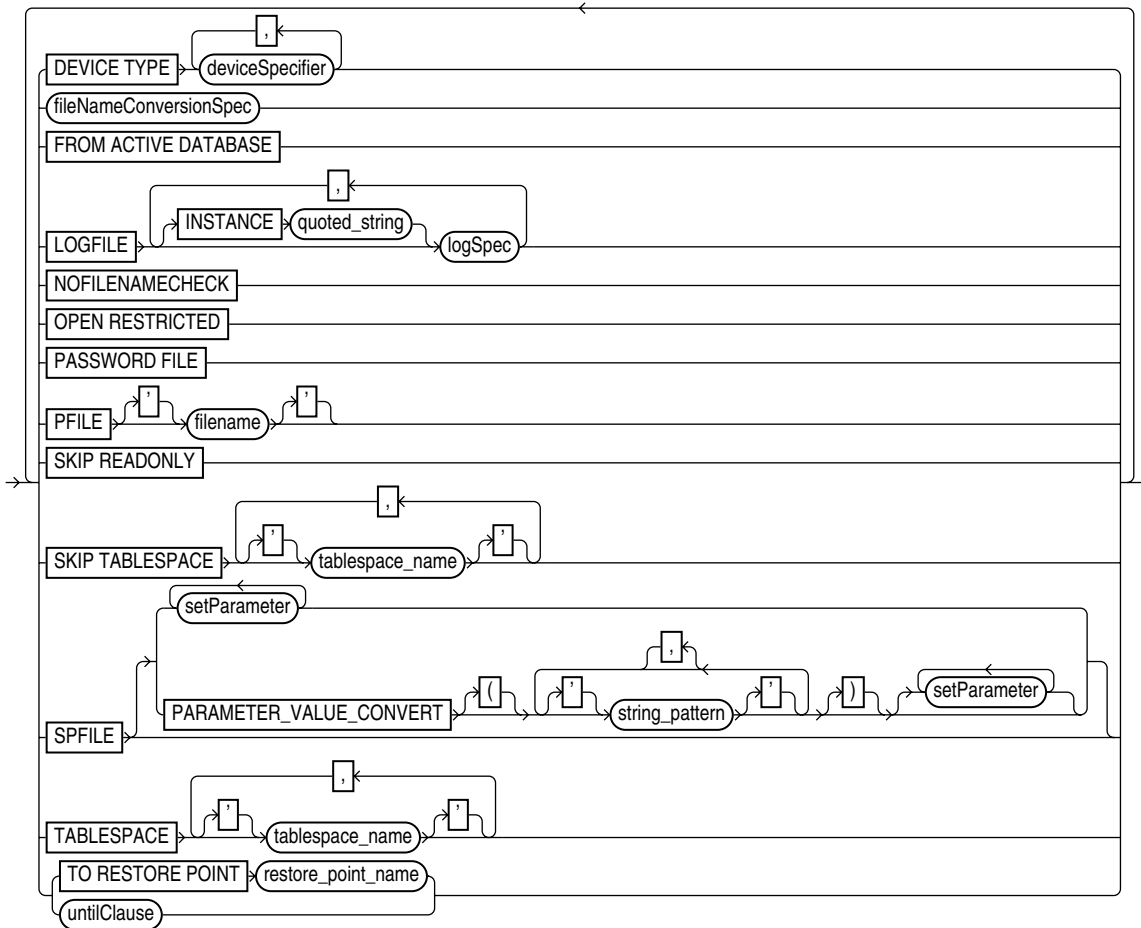
(*dupbyOptionList::=* on page 2-124, *dupOptionList::=* on page 2-124)

dupbyOptionList::=



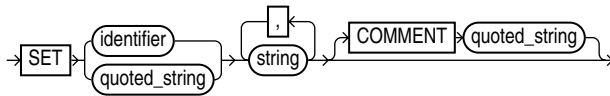
(*fileNameConversionSpec::=* on page 3-17, *setParameter::=* on page 2-125)

dupOptionList::=

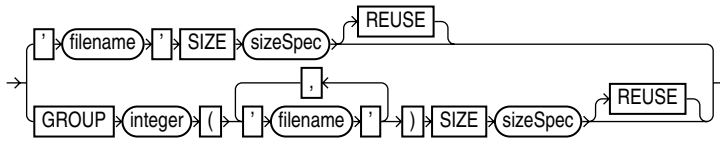


(*deviceSpecifier::=* on page 3-15, *fileNameConversionSpec::=* on page 3-17, *logSpec::=* on page 2-125, *setParameter::=* on page 2-125, *untilClause::=* on page 3-39)

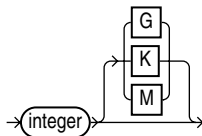
setParameter::=



logSpec::=



sizeSpec::=



Semantics

duplicate

This clause enables you to duplicate a database or tablespace. Refer to the [duplicate::=](#) diagram on page 2-123 for the syntax.

Syntax Element	Description
FOR STANDBY	<p>Specifies that database being duplicated is to be used as a standby database (see Example 2-75 on page 2-134).</p> <p>To create a standby database with the <code>DUPLICATE</code> command you must specify the <code>FOR STANDBY</code> option. The <code>DUPLICATE . . . FOR STANDBY</code> command creates the standby database by restoring a standby control file and mounting the standby control file. If you specify <code>FROM ACTIVE DATABASE</code>, 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 <code>SET UNTIL</code> is specified.</p> <p>If <code>DORECOVER</code> is specified, then RMAN also recovers database. The standby database is left mounted after duplication is complete.</p> <p>You cannot use <code>SET NEWNAME</code> or <code>CONFIGURE AUXNAME</code> to transform the filenames for the online redo logs on the standby database.</p> <p>You cannot connect to the standby database and then use <code>DUPLICATE . . . FOR STANDBY</code> to create an additional standby database. To create additional standby databases, connect to the original primary database and run <code>DUPLICATE . . . FOR STANDBY</code>.</p> <p>Note: Although you can use the <code>DUPLICATE</code> command to create a standby database, you cannot use this command to activate a standby database.</p> <p>When you connect 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 <code>REGISTER</code> command for the standby database.</p> <p>dupsbyOptionList Specifies options that only apply when creating a standby database. See dupsbyOptionList on page 2-126.</p>

Syntax Element	Description
TO <i>database_name</i>	<p>Specifies the name of the duplicate database. This duplicate database will not be a standby database.</p> <p>If you do not specify the <i>SPFILE</i> 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 <i>AUXILIARY</i>. Otherwise, the database signals an error.</p> <p>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.</p>
<i>dupOptionList</i>	<p>Specifies options that only apply when creating a duplicate database that is not a standby database. See <i>dupOptionList</i> on page 2-127.</p>

dupsbyOptionList

This subclause specifies options that only apply when creating a standby database. Refer to the *dupsbyOptionList::=* diagram on page 2-124 for the syntax.

Syntax Element	Description
DORECOVER	<p>Specifies that RMAN should recover the standby database after creating it. If you specify an <i>untilClause</i>, then RMAN recovers to the specified SCN or time and leaves the database mounted.</p> <p>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.</p> <p>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 <i>ORA-1152</i> error message because the standby control file backup was taken after the point in time.</p>
<i>fileNameConversionSpec</i>	<p>Specifies how to convert original datafile names to new datafile names in the standby database.</p> <p>See Also: <i>fileNameConversionSpec</i> on page 3-16</p>
FROM ACTIVE DATABASE	<p>Specifies that the files for the standby database should be provided directly from the source database and not from a backup of the source database (see <i>Example 2-71</i> on page 2-132).</p> <p>See Also: "Prerequisites Specific to Active Database Duplication" on page 2-120 for command prerequisites</p>
NOFILENAMECHECK	<p>Prevents RMAN from checking whether datafiles of the source database share the same names as the standby database files that are in use.</p> <p>The <i>NOFILENAMECHECK</i> option is required when the standby and primary datafiles and online redo logs have identical filenames (see <i>Example 2-74</i> on page 2-134). Thus, if you want the duplicate database filenames to be the same as the source database filenames, and if the databases are in different hosts, then you must specify <i>NOFILENAMECHECK</i>.</p> <p>See Also: The description of <i>NOFILENAMECHECK</i> in <i>dupOptionList</i> on page 2-127</p>

Syntax Element	Description
SPFILE	<p>Copies the server parameter file from the source database to the operating system-specific default location for this file on the standby database.</p> <p>RMAN uses the server parameter file to start the auxiliary instance for standby database creation. Any remaining options of the DUPLICATE command are processed after the database instance is started with the server parameter file.</p> <p>If you execute DUPLICATE with the SPFILE clause, then the auxiliary instance must already be started with a text-based initialization parameter file. In this case, the only required parameter in the temporary initialization parameter file is DB_NAME, which can be set to any arbitrary value. RMAN copies the binary server parameter file, modifies the parameters based on the settings in the SPFILE clause, and then restarts the standby instance with the server parameter file. When you specify SPFILE, RMAN never uses the temporary text-based initialization parameter file to start the instance.</p> <p>If FROM ACTIVE DATABASE is specified on DUPLICATE, then a server parameter file must be in use by the source database instance. If FROM ACTIVE DATABASE is not specified on DUPLICATE, then RMAN restores a backup of the server parameter file to the standby database.</p> <p>See Also: Example 2-70 on page 2-132 and Example 2-71 on page 2-132 for examples of SPFILE usage</p>
<i>setParameter</i>	<p>Sets the specified initialization parameters to the specified values. Refer to setParameter on page 2-131.</p>
PARAMETER_VALUE_CONVERT <i>string_pattern</i> [<i>setParameter</i>]	<p>Replaces the first string with the second string in all matching initialization parameter values. Note that DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT are exceptions to this rule and are not affected.</p> <p>You can use PARAMETER_VALUE_CONVERT to set a collection of initialization parameter values and avoid explicitly setting them all. For example, if the source database uses disk group +ALPHA while the standby database will use +BETA, then you could modify all parameters that refer to these disk groups by specifying SPFILE PARAMETER_VALUE_CONVERT ('+ALHPA', '+BETA').</p> <p>Note: Parameter values are case-sensitive in PARAMETER_VALUE_CONVERT even though the same values may not be case-sensitive when setting them directly in an initialization parameter file or server parameter file.</p>

dupOptionList

This subclause includes options that control aspects of the duplication such as naming the files and determining an end point for the duplication. Refer to the [dupOptionList::=](#) diagram on page 2-124 for the syntax.

Note: Several options in this clause are identical to options in the [dupbyOptionList](#) on page 2-126. Descriptions of these options are not repeated here.

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
DEVICE TYPE <i>deviceSpecifier</i> <i>fileNameConversionSpec</i>	<p>Allocates automatic channels for the specified device only (for example, DISK or sbt).</p> <p>This option is valid only if you have configured automatic channels and have <i>not</i> manually allocated channels. For example, if you CONFIGURE automatic disk and tape channels, and if you run <code>DUPLICATE . . . DEVICE TYPE DISK</code>, then RMAN allocates only disk channels.</p> <p>See Also: deviceSpecifier on page 3-15</p> <p>Specifies one or more patterns to map source database filenames to duplicate database filenames (see Example 2-72 on page 2-133).</p> <p>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 <code>DB_FILE_NAME_CONVERT=('disk1', 'disk2')</code>, but you execute <code>DUPLICATE . . . DB_FILE_NAME_CONVERT ('disk1', 'disk3')</code>, then RMAN does not convert the <code>disk1</code> substring to <code>disk2</code>. Instead, RMAN converts the <code>disk1</code> substring to <code>disk3</code>.</p> <p>If a file in the specification list is not affected by the conversion parameter in <code>DUPLICATE</code>, then you must rename it by other means, such as <code>SET NEWNAME</code>.</p> <p>Note: If you specify the SPFILE clause, then <code>DUPLICATE . . . DB_FILE_NAME_CONVERT</code> overrides any conversion parameter specified in the <code>SPFILE</code> syntax. For example, if you specify <code>DB_FILE_NAME_CONVERT</code> twice in the <code>DUPLICATE</code> command, both in the <code>SPFILE</code> clause and outside of the <code>SPFILE</code> clause, then the setting outside of the <code>SPFILE</code> clause takes precedence.</p> <p>See Also: fileNameConversionSpec on page 3-16</p>
FROM ACTIVE DATABASE	<p>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 2-70 on page 2-132). If you do not specify an UNTIL time, then RMAN chooses an end time for the duplication based on when the online datafiles are copied.</p> <p>See Also: "Prerequisites Specific to Active Database Duplication" on page 2-120 for command prerequisites</p>
LOGFILE	<p>Specifies options for creating online redo logs when creating a duplicate database that is not a standby database (see Example 2-72).</p>
INSTANCE ' <i>inst_name</i> ' <i>logSpec</i>	<p>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.</p> <p>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.</p> <p>This clause is relevant when you use <code>DUPLICATE TARGET DATABASE</code> to duplicate an Oracle RAC database to a single-instance database. Otherwise, you do not need to use <code>INSTANCE</code>. If you use the <code>LOGFILE</code> clause, then use <code>INSTANCE</code> 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).</p> <p>Specifies the filenames and groups for the online redo log files.</p> <p>See Also: logSpec on page 2-131 for the legal options</p>

Syntax Element	Description
NOFILENAMECHECK	<p>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 (see Example 2-73 on page 2-134). You are responsible for determining that the duplicate operation will not overwrite useful data.</p> <p>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 <code>/dbs</code> directory of <code>host1</code>:</p> <pre>/oracle/dbs/system_prod1.dbf /oracle/dbs/users_prod1.dbf /oracle/dbs/rbs_prod1.dbf</pre> <p>Assume that you want to duplicate this database to <code>host2</code>, which has the same file system <code>/oracle/dbs/*</code>, and you want to use the same filenames in the duplicate database as in the source database. In this case, specify the <code>NOFILENAMECHECK</code> 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.</p> <p>If duplicating a database on the same host as the source database, then make sure that <code>NOFILENAMECHECK</code> is not set. Otherwise, RMAN may signal the following error:</p> <pre>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</pre>
OPEN RESTRICTED	<p>Enables a restricted session in the duplicate database by issuing the following SQL statement: <code>ALTER SYSTEM ENABLE RESTRICTED SESSION</code>. RMAN issues this statement immediately before the duplicate database is opened.</p>
PASSWORD FILE	<p>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 2-70 on page 2-132). This option is only valid when <code>FROM ACTIVE DATABASE</code> is specified; otherwise, RMAN signals an error.</p> <p>If <code>FOR STANDBY</code> 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 <code>PASSWORD FILE</code> 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 <code>PASSWORD FILE</code> option.</p>
PFILE <i>filename</i>	<p>Specifies a text-based initialization parameter file used by the auxiliary instance (see Example 2-72 on page 2-133). 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.</p> <p>If the auxiliary instance uses a server parameter file in the default location, then you do not need to specify <code>PFILE</code>.</p>

Syntax Element	Description
SKIP READONLY	<p>Excludes datafiles in current read-only tablespaces from the duplicate database (see Example 2-73 on page 2-134). By default RMAN duplicates current read-only tablespaces.</p> <p>If a tablespace is currently read/write, but you use <i>untilClause</i> 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.</p> <p>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.</p>
SKIP TABLESPACE <i>tbs_name</i>	<p>Excludes the specified tablespace from the duplicate database (see Example 2-72 on page 2-133). Note that you cannot exclude the SYSTEM tablespace, SYSAUX tablespace, undo tablespaces, and tablespaces with rollback segments.</p> <p>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:</p> <ul style="list-style-type: none"> ■ All datafiles in online tablespaces, whether or not the datafiles are online. ■ All tablespaces taken offline with an option <i>other than</i> 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. ■ If no valid backups exist of any tablespace or datafile, then the DUPLICATE command fails. <p>RMAN does not check for completeness. For example, you can duplicate a data tablespace but not the tablespace containing the index for the data, or duplicate a tablespace that contains only one partition of a partitioned table.</p>
SPFILE	<p>Copies the server parameter file from the source database to the duplicate database. Refer to the description of SPFILE in dupsbyOptionList on page 2-126.</p>
<i>setParameter</i>	<p>Sets the specified initialization parameters to the specified values. Refer to setParameter on page 2-131.</p>
PARAMETER_VALUE_CONVERT <i>string_pattern</i> [<i>setParameter</i>]	<p>Replaces the first string with the second string in all matching initialization parameter values. Refer to the description of PARAMETER_VALUE_CONVERT in dupsbyOptionList on page 2-126.</p>
TABLESPACE <i>tablespace_name</i>	<p>Specifies which tablespaces should be included in the specified database. Unlike SKIP TABLESPACE, which specifies which tablespaces should be <i>excluded</i> from the duplicate database, this option specifies which tablespaces should be included and then skips the remaining tablespaces.</p> <p>Note: RMAN automatically includes the SYSTEM, SYTAUX, and undo tablespaces in the duplicate database: these tablespaces cannot be skipped.</p>
TO RESTORE POINT <i>restore_point_name</i>	<p>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 <i>and including</i> the corresponding SCN.</p> <p>Note: The same restrictions that apply to <i>untilClause</i> also apply to TO RESTORE POINT.</p>

Syntax Element	Description
<i>untilClause</i>	<p>Sets the end time, SCN, or log sequence number for point-in-time recovery in backup-based duplication (see Example 2-72 on page 2-133). The <code>UNTIL</code> clause is not supported in active database duplication.</p> <p>You can achieve the same result by running <code>SET UNTIL</code> before the <code>DUPLICATE</code> command. If you specify the <code>UNTIL</code> clause for duplication, then the following restrictions apply:</p> <ul style="list-style-type: none"> ■ RMAN determines whether to use <code>NOREDO</code> based on the current state of the database. If the database was in an archiving mode at the specified <code>UNTIL</code> time or SCN that is different from the current archiving mode, then RMAN does not use <code>NOREDO</code>. ■ If a tablespace was read-only at the time of the duplication, then RMAN does not include it even if <code>SKIP READONLY</code> was not used. ■ The end point for a <code>DUPLICATE</code> command cannot be before the SCN of the most recent <code>ALTER DATABASE OPEN RESETLOGS</code>. Duplication to previous database incarnations is not supported. ■ 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. <p>See Also: untilClause on page 3-39</p>

setParameter

This subclause specifies server parameter file values.

Syntax Element	Description
<code>SET identifier string</code>	<p>Sets the specified initialization parameters to the specified values (see Example 2-71 on page 2-132). You can use <code>SET</code> to adjust for differences in memory, turn off replication options, and set other options for the duplicate database.</p> <p>This <code>SET</code> functionality is equivalent to pausing the duplication after restoring the server parameter file and issuing <code>ALTER SYSTEM SET</code> statements to change the initialization parameter file.</p> <p>RMAN processes <code>SET</code> after <code>PARAMETER_VALUE_CONVERT</code>. If <code>PARAMETER_VALUE_CONVERT</code> sets the filename specified by a parameter, and if <code>SET</code> sets the filename specified by the same parameter, then the <code>SET</code> value overrides the <code>PARAMETER_VALUE_CONVERT</code> setting.</p> <p>Note: If <code>DB_FILE_NAME_CONVERT</code> is specified on the <code>DUPLICATE</code> command, then its filename settings override competing settings specified by <code>SPFILE SET</code>.</p>
<code>COMMENT 'string'</code>	Specifies an optional comment for the parameter setting.

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 2-125 for the syntax diagram.

If you do not specify `LOGFILE`, then RMAN uses `LOG_FILE_NAME_CONVERT` if it is set. If neither `LOGFILE` nor `LOG_FILE_NAME_CONVERT` is 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 2–70 Duplicating from an Active Database

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. You can connect to the source database and auxiliary instances as follows and duplicate the database:

```
CONNECT TARGET SYS/password@prod1 # source database
CONNECT AUXILIARY SYS/password@dup1 # duplicate database instance
DUPLICATE TARGET DATABASE TO dup1
FROM ACTIVE DATABASE
PASSWORD FILE
SPFILE;
```

Example 2–71 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.

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 duplicate the database as follows:

```
CONNECT TARGET SYS/password@prod1 # source database
CONNECT AUXILIARY SYS/password@dup1 # duplicate database instance
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 2–72 Setting New Filenames Manually for Duplication

Assume that you want to duplicate the source database on `host1` to `newdb` on `host2`.

In this scenario, your source database does not use a server parameter file. You create a text-based initialization parameter file on `host2` and start the instance.

When executing `DUPLICATE` on `host2`, you must use the `PFILE` parameter to specify the location of the initialization parameter file. Note that 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 `host1` in directory `/h1/oracle/dbs/trgt`. You want to duplicate the datafiles to the directory `/h2/oracle/oradata/newdb`, so you specify `DUPLICATE ... DB_FILE_NAME_CONVERT` generate the names for the duplicate datafiles. You use `DUPLICATE ... LOGFILE` to specify names for the online redo log files in the duplicate database.

You start the RMAN client on `host2` and run the following commands to duplicate the database:

```
CONNECT TARGET SYS/password@prod1 # source database
CONNECT AUXILIARY SYS/password@newdb # duplicate database instance
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 ('?/oradata/newdb/redo01_1.f',
           '?/oradata/newdb/redo01_2.f') SIZE 4M,
  GROUP 2 ('?/oradata/newdb/redo02_1.f',
           '?/oradata/newdb/redo02_2.f') SIZE 4M,
  GROUP 3 ('?/oradata/newdb/redo03_1.f',
           '?/oradata/newdb/redo03_2.f') SIZE 4M REUSE;
}
```


Example 2–73 Using the Source Database Filenames for the Duplicate Database

Assume that you want to use RMAN backups to create a duplicate database for testing. The following conditions apply:

- You are restoring to a destination host that is different from the source host.
- RMAN is not connected to a recovery catalog.
- You have configured automatic channels.
- The source host and destination host have the same file structure.
- You want to name the duplicate database files exactly like the source database files.
- You do not want to duplicate read-only tablespaces.
- You want to prevent RMAN from checking whether files on the source database are in use if these files have the same names as the duplicate database files.

```
CONNECT TARGET                               # source database
CONNECT AUXILIARY SYS/password@newdb        # duplicate databavvse instance
DUPLICATE TARGET DATABASE TO ndbnewh
LOGFILE
  '?/dbs/log_1.f' SIZE 4M,
  '?/dbs/log_2.f' SIZE 4M
SKIP READONLY
NOFILENAMECHECK;
```

Example 2–74 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, you connect to `prod1` as `TARGET` and `CONFIGURE` the default device type to `sbt` for a standby database with the `DB_UNIQUE_NAME` of `standby1`:

```
CONNECT TARGET SYS/password@prod1
CONNECT CATALOG rman/password@catdb
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 issue the following commands, specifying the `NOFILENAMECHECK` option because the standby and primary datafiles and online redo logs have the same names:

```
CONNECT TARGET SYS/password@prod1           # source database
CONNECT CATALOG rman/password@catdb        # recovery catalog database
CONNECT AUXILIARY SYS/password@standby1    # standby database instance
DUPLICATE TARGET DATABASE FOR STANDBY
NOFILENAMECHECK;
```

Example 2–75 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, you connect to `prod1` as `TARGET` and `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`.

```
CONNECT TARGET SYS/password@prod1
CONNECT CATALOG rman/password@catdb
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 standby database initialization parameter file:

- 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`.

After starting the standby instance in `NOMOUNT` mode, you issue the following commands in the RMAN client:

```
CONNECT TARGET SYS/password@prod1
CONNECT CATALOG rman/password@catdb
CONNECT AUXILIARY SYS/password@standby1
DUPLICATE TARGET DATABASE FOR STANDBY TO standby1;
```

RMAN automatically generates new OMF/ASM datafile names for the restored datafiles. The new database name and file names will be automatically resynchronized to the recovery catalog.

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 2-101 and [REPLACE SCRIPT](#) on page 2-199

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.

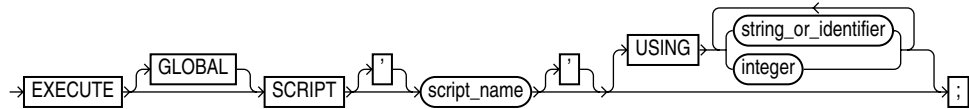
Usage Notes

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
<code>GLOBAL</code>	Specifies the execution of a global stored script instead of a local one. See Also: "Usage Notes" on page 2-101 for an explanation of the difference between global and local scripts
<code>SCRIPT script_name</code>	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.
<code>USING</code> <code>[string_or_identifier integer]</code>	Specifies one or more values for use in substitution variables in a stored script (see Example 2-77). See Also: CREATE SCRIPT to learn how to create a stored script with substitution variables, and RMAN on page 2-232 and @ on page 2-2 to learn how to use substitution variables with RMAN

Example

Example 2-76 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 2-59, "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 2-77 Creating and Executing a Stored Script That Uses Substitution Variables

The following example connects to the target database and recovery catalog and uses `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).

```
CONNECT TARGET /  
CONNECT CATALOG rman/password@catdb  
REPLACE SCRIPT 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 script with different values. The following example passes the values 3, `test_backup`, and `test` to the substitution variables in the stored script:

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

After the values are substituted, RMAN executes the script 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:=`

→ `EXIT` ↲

Example

Example 2-78 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. Refer to *Oracle Data Guard Concepts and Administration* for details.

See Also: *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 to the target database, which must be Oracle Database 10g or later. The 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 flash recovery area must be configured to enable flashback logging. Flashback logs are stored as Oracle-managed files in the flash recovery area and cannot be created if no flash 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 logs 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 2-7](#) for details.

Table 2-7 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 <code>FLASHBACK DATABASE</code> 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.
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

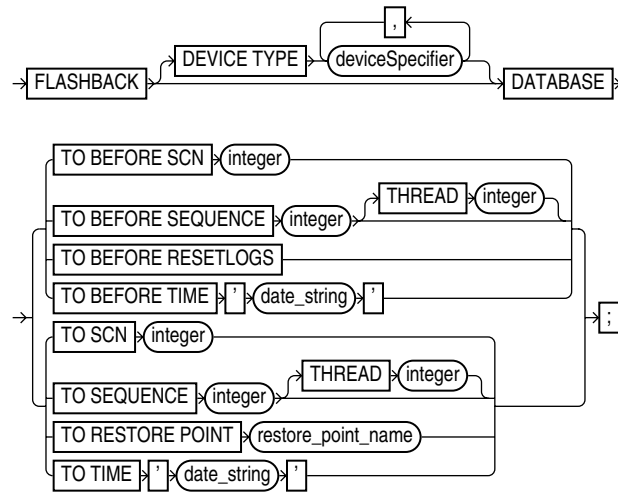
After running `FLASHBACK DATABASE`, the database may not be left at the SCN most immediately before the target time. Events other than transactions can cause the database SCN to be updated. If you use the `FLASHBACK DATABASE TO` form of the command, and if a transaction is associated with the target 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 DATABASE`.

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 3-15)

Semantics

Syntax Element	Description
DEVICE TYPE <i>deviceSpecifier</i>	Allocates automatic channels for the specified device type only. For example, if you configure automatic disk and tape channels, and issue <code>FLASHBACK . . . DEVICE TYPE DISK</code> , 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 <code>FLASHBACK DATABASE</code> operation fails. See Also: <i>deviceSpecifier</i> on page 3-15
TO BEFORE SCN <i>integer</i>	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 <code>RESET DATABASE INCARNATION</code> command. Query <code>OLDEST_FLASHBACK_SCN</code> in <code>V\$FLASHBACK_DATABASE_LOG</code> to see the approximate lowest SCN to which you can flash back.
TO BEFORE SEQUENCE <i>integer</i> [<code>THREAD <i>integer</i></code>]	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 <code>OPEN RESETLOGS</code> . Note: <code>FLASHBACK DATABASE</code> can only return the database to a point before the most recent <code>OPEN RESETLOGS</code> operation if your database has been upgraded to Oracle Database 10g Release 2 or later.
TO BEFORE TIME ' <i>date_string</i> '	Returns the database to its state including all changes up to but not including changes at the specified time. Query <code>OLDEST_FLASHBACK_TIME</code> in <code>V\$FLASHBACK_DATABASE_LOG</code> to see the approximate lowest time to which you can flash back.

Syntax Element	Description
TO SCN <i>integer</i>	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 <code>RESET DATABASE</code> command to set the recovery target incarnation. Query <code>OLDEST_FLASHBACK_SCN</code> in <code>V\$FLASHBACK_DATABASE_LOG</code> to see the approximate lowest SCN to which you can flash back.
TO SEQUENCE <i>integer</i> THREAD <i>integer</i>	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 <i>restore_point_name</i>	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 ' <i>date_string</i> '	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, <code>FLASHBACK DATABASE TO TIME 'SYSDATE-7'</code> . Query <code>OLDEST_FLASHBACK_TIME</code> in <code>V\$FLASHBACK_DATABASE_LOG</code> to see the approximate lowest time to which you can flash back.

Examples

Example 2-79 FLASHBACK DATABASE to a Specific SCN

Assume that you inserted corrupted database in many tables at 5:00 p.m. on February 14. The following statement queries 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 start an RMAN session and enter commands as follows (sample output for the `FLASHBACK DATABASE` is included):

```
RMAN> CONNECT TARGET /
RMAN> CONNECT CATALOG rman/password@catdb
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 2-80 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 updates:

```
SQL> CREATE RESTORE POINT before_update GUARANTEE FLASHBACK DATABASE;
```

The update fails, leaving the database with extensive corrupted data. You start an RMAN session and list the guaranteed restore points:

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

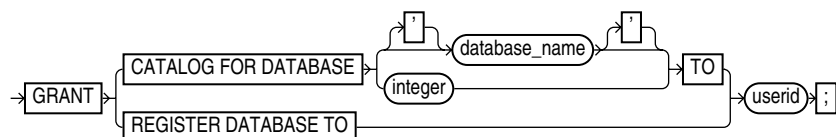
In this scenario, both `vpc1` and `vpc2` can access the metadata for backups of `prod1` made by the base recovery catalog owner. Both users can also access the metadata for backups of `prod1` made by each other. Neither `vpc1` nor `vpc2` can access backup metadata for database `prod2`.

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 [<i>database_name</i> <i>integer</i>] TO <i>userid</i>	<p>Grants recovery catalog access for the specified database to the specified user.</p> <p>Note: The catalog operations granted on the specified database include registering and unregistering this database.</p> <p>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.</p> <p>To grant access to databases that are already registered in the recovery catalog, you must use the <code>GRANT CATALOG</code> 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.</p>
REGISTER DATABASE TO <i>userid</i>	<p>Grants the ability to for the specified user to use <code>REGISTER DATABASE</code> to register databases that are currently unknown to the recovery catalog.</p> <p>When you grant <code>REGISTER DATABASE</code> to a user, RMAN implicitly grants recovery <code>CATALOG FOR DATABASE</code> privileges for any database registered by the user. The <code>CATALOG FOR DATABASE</code> privileges on the database include registering and unregistering this database.</p> <p>For example, assume that user <code>virtcat</code> is granted <code>REGISTER DATABASE</code> and registers database <code>prod</code> in the catalog. RMAN implicitly grants recovery <code>CATALOG FOR DATABASE</code> privileges for <code>prod</code> to <code>virtcat</code>.</p>

Examples

Example 2–81 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.

You want to create virtual private catalogs for two backup operators in the data center. You start SQL*Plus, create the `bckop2` and `bckop3` users on `catdb`, and grant recovery catalog ownership to these users as follows:

```
SQL> CONNECT SYS/password@catdb AS SYSDBA
SQL> CREATE USER bckop2 IDENTIFIED BY pwd2;
SQL> CREATE USER bckop3 IDENTIFIED BY pwd3;
SQL> GRANT recovery_catalog_owner TO bckop2, bckop3;
SQL> EXIT
```

You then start RMAN 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/password@catdb
```

```
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/password@catdb  
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/password@catdb  
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 target database prodb:

```
RMAN> CONNECT TARGET dba1/password@prodb  
RMAN> CONNECT CATALOG bckop3/password@catdb  
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 2-82 on page 2-149).
' <i>command</i> '	Runs the command in the specified string and then continues (see Example 2-83 on page 2-150).

Examples

Example 2-82 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.cpy
/disk2/df3.cpy
% exit
exit
host command complete

RMAN>
```

Example 2–83 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 2-98

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

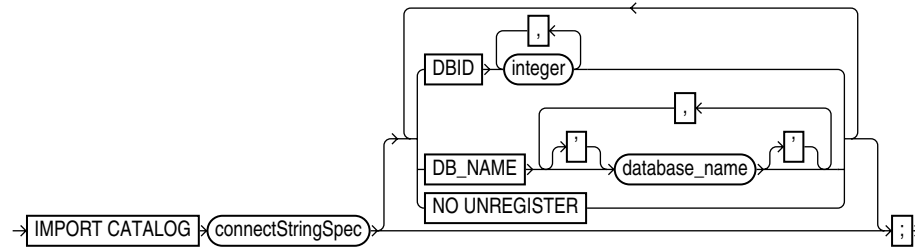
If the operation fails in the middle of the import, then the import is rolled back. Thus, a partial import is not permitted. The unregister operation is separate from the import. By default, the imported database IDs are unregistered from the source recovery catalog schema after a successful import.

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 3-12)

Semantics

Syntax Element	Description
<i>connectStringSpec</i>	Specifies the connection string for the source recovery catalog, which is the catalog whose metadata will be imported.
DBID <i>integer</i>	Specifies the list of DBIDs for the databases whose metadata should be imported from the source catalog schema (see Example 2-85). 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 <i>database_name</i>	Specifies the list of databases whose metadata should be imported from the source catalog schema (see Example 2-85). 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 2-84 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/password@catdb

connected to recovery catalog database

RMAN> IMPORT CATALOG rcat/oracle@inst1;

Starting import catalog at 15-FEB-07
connected to source recovery catalog database
import validation complete
database unregistered from the source recovery catalog
Finished import catalog at 15-FEB-07
```

Example 2–85 Importing Metadata for a Subset of Registered Databases

This example is a variation on [Example 2–84](#). Instead of importing the entire recovery catalog, it imports only the metadata for the database with DBID 1618984270.

```

RMAN> CONNECT CATALOG rco/password@catdb

connected to recovery catalog database

RMAN> IMPORT CATALOG rcat/oracle@inst1 DBID=1618984270;

Starting import catalog at 15-FEB-07
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 [RMAN](#) on page 2-232

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.

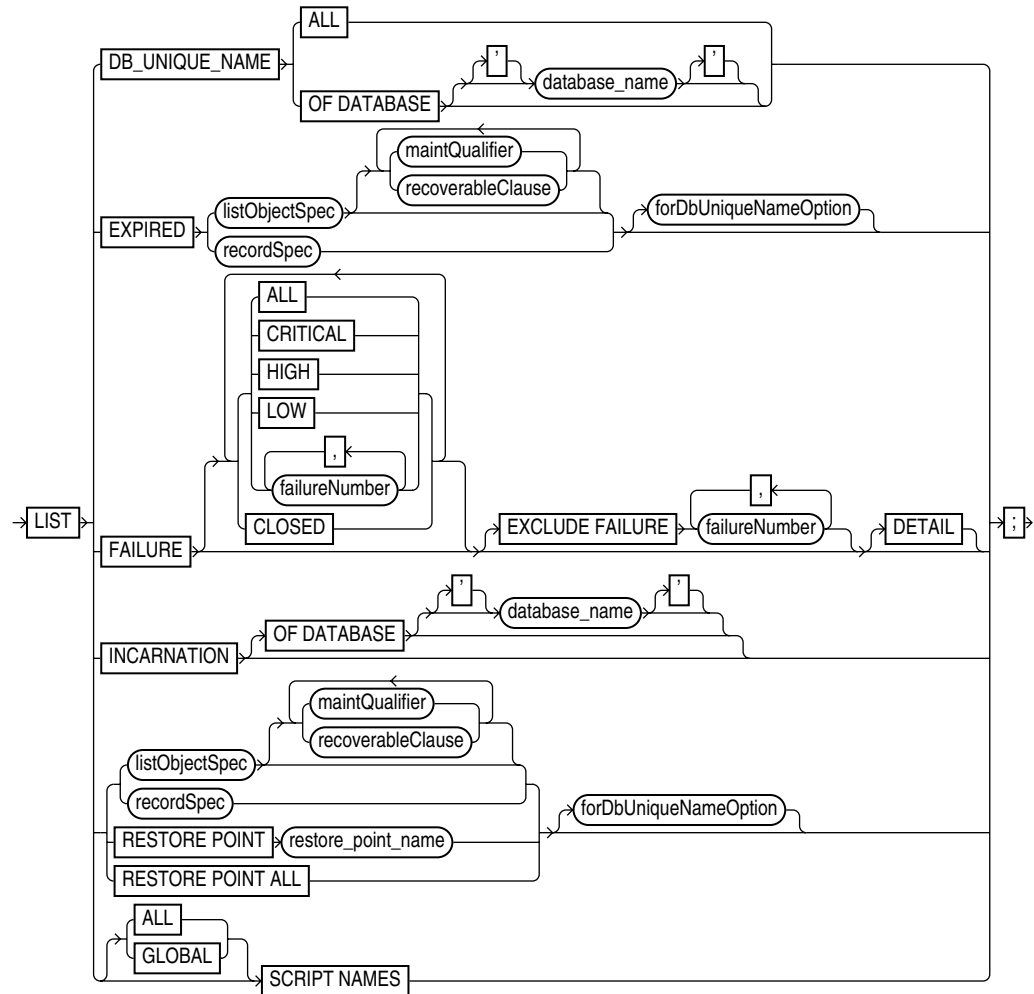
Usage Notes

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.

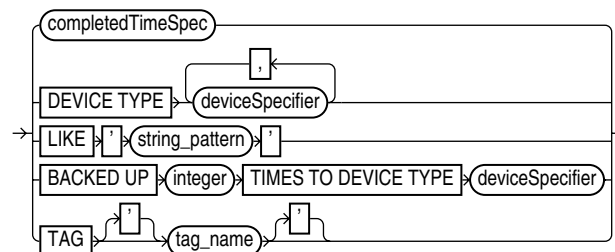
"[RMAN Backups in a Data Guard Environment](#)" on page 2-21 explains how RMAN handles backups in a Data Guard environment. In general, RMAN considers tape backups created on one database in the environment as accessible to all databases in the environment, whereas disk backups are accessible only to 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.

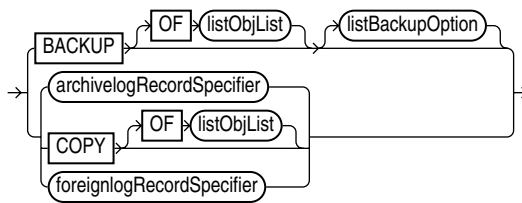
Syntax

list::=

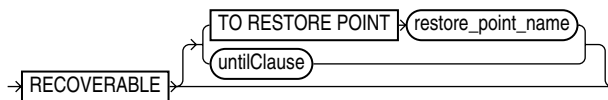
(*listObjectSpec::=* on page 2-156, *recordSpec::=* on page 3-36, *maintQualifier::=* on page 3-31, *forDbUniqueNameOption::=* on page 3-19, *untilClause::=* on page 3-39)

maintQualifier::=

(*completedTimeSpec::=* on page 3-10, *deviceSpecifier::=* on page 3-15)

listObjectSpec::=

(*listObjList::=* on page 3-28, *listBackupOption::=* on page 2-156,
archiveLogRecordSpecifier::= on page 3-6, *foreignLogRecordSpecifier::=* on page 3-20)

recoverableClause::=

(*untilClause::=* on page 3-39)

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 2-27 for a description of the output
ALL	Lists the DB_UNIQUE_NAME of every database registered in the RMAN repository.
OF DATABASE <i>database_name</i>	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 2-8, "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.
<i>listObjectSpec</i>	Specifies the type of expired object or objects that you are listing. See Also: listObjectSpec on page 2-159

Syntax Element	Description
<i>maintQualifier</i>	Restricts the range of the listing. See Also: <i>maintQualifier</i> on page 3-31
<i>recordSpec</i>	Specifies the expired object or objects that you are listing. See Also: <i>recordSpec</i> on page 3-36
<i>forDbUniqueNameOption</i>	Lists the expired files in <i>listObjectSpec</i> or <i>recordSpec</i> that are exclusively associated with the specified DB_UNIQUE_NAME in a Data Guard environment. You can specify a database with <i>db_unique_name</i> 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: <i>forDbUniqueNameOption</i> on page 3-19 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 2-26 for an explanation of the column headings of the LIST FAILURE output table and Example 2-91 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.
<i>failureNumber</i>	Specifies the failures by failure number.

Syntax Element	Description
CLOSED	Lists only closed failures.
EXCLUDE FAILURE <i>failureNumber</i>	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 <code>DETAIL</code> option would list each of the block corruptions.
INCARNATION	Displays information about the incarnations of a database. Whenever you open a database with the <code>RESETLOGS</code> option, you create a new incarnation of the database. If <code>LIST INCARNATION</code> displays <i>n</i> incarnations of a database, then you have reset the online redo logs for this database <i>n</i> -1 times. The <code>LIST</code> output includes the primary keys of all database incarnation records for the specified database name (in the column <code>Inc Key</code> , which contains the incarnation key). Use the key in a <code>RESET DATABASE</code> command to change the incarnation that RMAN considers to be current to a previous incarnation. See Also: Table 2-23 for an explanation of the column headings of the <code>LIST INCARNATION</code> output table and Example 2-90 for an illustration
OF DATABASE <i>database_name</i>	Specifies the name of the database. If you do not specify the <code>OF DATABASE</code> option, then <code>LIST</code> displays all databases registered in the recovery catalog.
<i>listObjectSpec</i>	Specifies the type of expired object or objects that you are listing. See Also: listObjectSpec on page 2-159
<i>maintQualifier</i>	Restricts the range of the listing. See Also: maintQualifier on page 3-31
<i>recoverableClause</i>	Restricts the list to datafile backups or copies whose status in the repository is <code>AVAILABLE</code> and which can be used for restore and recovery in the current incarnation of the target database. See Also: recoverableClause on page 2-159
<i>recordSpec</i>	Specifies the object or objects that you are listing. See Also: recordSpec on page 3-36
<i>untilClause</i>	Specifies an end time, SCN, or log sequence number. See Also: untilClause on page 3-39
RESTORE POINT	Displays restore points known to the RMAN repository. See Also: Table 2-25 for an explanation of the column headings of the <code>LIST RESTORE POINT</code> output table
<i>restore_point_name</i>	Displays the specified restore point.
ALL	Displays all restore points known to the RMAN repository.
<i>forDbUniqueNameOption</i>	Lists the backups and restore points that are exclusively associated with the specified <code>DB_UNIQUE_NAME</code> in a Data Guard environment. You can specify a database with <i>db_unique_name</i> or use <code>ALL</code> 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 <code>DB_UNIQUE_NAME</code> 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 3-19 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 2-24 for a description of the output and Example 2-107 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 2-8 for a description of LIST BACKUP output and Example 2-86 for an illustration
OF <i>listObjList</i>	Restricts the list of objects operated on to the object type specified in the <i>listObjList</i> 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 3-28
<i>listBackupOption</i>	Specifies whether to list summary information about backups or detailed information.
<i>archivelogRecordSpecifier</i>	Displays information about a range of archived redo logs.
COPY	Displays only information about datafile copies, archived redo logs, and image copies of archived redo logs. By default, LIST COPY displays copies of all database files and archived redo logs. Both usable and unusable image copies are included in the output, even those that cannot be restored or are expired or unavailable. See Also: Table 2-20 on page 2-167 and Table 2-22 on page 2-168 for an explanation of the column headings of the LIST COPY output tables
OF <i>listObjList</i>	Restricts the list of objects operated on to the object type specified in the <i>listObjList</i> clause. See Also: listObjList on page 3-28
<i>foreignlogRecordSpecifier</i>	Displays information about a range of foreign archived redo logs.

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 <i>restore_point_name</i>	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 <i>and including</i> the SCN corresponding to the restore point.
<i>untilClause</i>	Specifies an end time, SCN, or log sequence number. See Also: <i>untilClause</i> on page 3-39

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 2-17, Table 2-18, and Table 2-19 for a description of LIST BACKUP . . . BY FILE output
SUMMARY	Gives a one-line summary for each backup. See Also: Table 2-15 for a description of LIST BACKUP . . . SUMMARY output and Example 2-87 for sample output

LIST Command Output

The information that appears in the output is shown in the following tables:

- Table 2-8, " List of Backup Sets (for datafile backup sets)"
- Table 2-9, " List of Backup Pieces (for sets with only one piece)"
- Table 2-10, " List of Datafiles in backup set ..."
- Table 2-11, " List of Archived Logs in backup set ..."
- Table 2-12, " Backup Set Copy ... of backup set ... (only if multiple pieces)"
- Table 2-13, " List of Backup Pieces for backup set ... Copy ... (if multiple pieces)"
- Table 2-14, " List of Proxy Copies"
- Table 2-15, " List of Backup Sets (LIST BACKUP ... SUMMARY)"
- Table 2-16, " List of Backup Pieces (LIST BACKUPPIECE ...)"
- Table 2-17, " List of Datafile Backups (LIST BACKUP ... BY FILE)"
- Table 2-18, " List of Archived Log Backups (LIST BACKUP ... BY FILE)"
- Table 2-19, " List of Controlfile Backups (LIST BACKUP ... BY FILE)"
- Table 2-20, " List of Datafile Copies"
- Table 2-21, " List of Controlfile Copies"
- Table 2-22, " List of Archived Log Copies"
- Table 2-23, " List of Database Incarnations"

- [Table 2–24, " List of Stored Scripts in the Recovery Catalog \(LIST SCRIPT NAMES\)"](#)
- [Table 2–25, " List of Restore Points \(LIST RESTORE POINT\)"](#)
- [Table 2–26, " List of Failures"](#)
- [Table 2–27, " List of Databases \(LIST DB_UNIQUE_NAME\)"](#)

Table 2–8 List of Backup Sets (for datafile backup sets)

Column	Indicates
BS Key	A unique key identifying this backup set. If you are 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 you are 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 2–10

Table 2–9 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 you are 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 you are 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 2–9 (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 2–10 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 2-243

Table 2–11 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 2–12 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 2–13 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 you are 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 you are 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 2–14 List of Proxy Copies

Column	Indicates
PC Key	A unique key identifying this proxy copy. If you are 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 you are 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 2–14 (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 2-243
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 2–15 List of Backup Sets (LIST BACKUP ... SUMMARY)

Column	Indicates
Key	A unique key identifying this backup set. If you are 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 you are 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 logs, 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 2–16 List of Backup Pieces (LIST BACKUPPIECE ...)

Column	Indicates
BP Key	<p>A unique identifier for this backup piece in the recovery catalog or target database control file.</p> <p>If you are 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 you are connected in NOCATALOG mode, then BP Key displays the RECID from V\$BACKUP_PIECE.</p> <p>Note: The values for KEY in the recovery catalog and the control file are different.</p>
BS Key	<p>A unique key identifying this backup set.</p> <p>If you are 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 you are connected in NOCATALOG mode, then BS Key displays the RECID from V\$BACKUP_SET.</p>
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 2–17 List of Datafile Backups (LIST BACKUP ... BY FILE)

Column	Indicates
File	The absolute datafile number.
Key	<p>A unique key identifying this backup set.</p> <p>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.</p>
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 2–17 (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 2–18 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 you are 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 you are 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 2–19 List of Controlfile 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 you are 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 you are 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 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 you are 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 you are 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 2–21 List of Controlfile Copies

Column	Indicates
Key	<p>The unique identifier for the control file copy. Use this value in a CHANGE command to alter the status of the copy.</p> <p>If you are connected to a recovery catalog, then <i>Key</i> is the primary key of the control file copy in the catalog. It corresponds to <i>CCF_KEY</i> in the <i>RC_CONTROLFILE_COPY</i> view. If you are connected in NOCATALOG mode, then <i>Key</i> displays the <i>RECID</i> from <i>V\$DATAFILE_COPY</i>.</p> <p>Note: The values for <i>Key</i> in the recovery catalog and the control file are different.</p>
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 <i>NLS_LANG</i> and <i>NLS_DATE_FORMAT</i> 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 2–22 List of Archived Log Copies

Column	Indicates
Key	<p>The unique identifier for this archived redo log copy. Use this value in a CHANGE command to alter the status of the copy.</p> <p>If you are connected to a recovery catalog, then <i>Key</i> is the primary key of the backup set in the catalog. It corresponds to <i>AL_KEY</i> in the <i>RC_ARCHIVED_LOG</i> view. If you are connected in NOCATALOG mode, then <i>Key</i> displays the <i>RECID</i> from <i>V\$ARCHIVED_LOG</i>.</p> <p>Note: The values for <i>Key</i> in the recovery catalog and the control file are different.</p>
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 2–23 List of Database Incarnations

Column	Indicates
DB Key	When combined with the <i>Inc Key</i> , 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 <i>DB Key</i> , the unique key by which RMAN identifies the database incarnation in the recovery catalog. Use this key in <code>RESET DATABASE TO INCARNATION</code> when recovering the database to a time before the most recent <code>RESETLOGS</code> .
DB Name	The database name as listed in the <code>DB_NAME</code> parameter.
DB ID	The database identification number, which the database generates automatically at database creation.
STATUS	<code>CURRENT</code> for the current incarnation, <code>PARENT</code> for the parent incarnations of the current incarnation, and <code>ORPHAN</code> for orphaned incarnations.
Reset SCN	The SCN at which the incarnation was created.
Reset Time	The time at which the incarnation was created.

Table 2–24 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 2–25 List of Restore Points (LIST RESTORE POINT)

Column	Indicates
SCN	The SCN for the restore point.
RSP Time	The time specified in the <code>CREATE RESTORE POINT</code> statement; otherwise null.
Type	<code>GUARANTEED</code> 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 2–26 List of Failures

Column	Indicates
Failure ID	The unique identifier for a failure.
Priority	<p>The priority of the failure: CRITICAL, HIGH, or LOW.</p> <p>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).</p> <p>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.</p> <p>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.</p>
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 2–27 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 2–86 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/undots01.dbf
4          Full 421946    15-FEB-07 /disk1/oradata/prod/cwmlite01.dbf
5          Full 421946    15-FEB-07 /disk1/oradata/prod/drsyst01.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 2-87 Listing a Summary of Backups

This example summarizes the RMAN backups listed in [Example 2-86](#) on page 2-170.

```
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 2-88 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	1	NO	DF1
2	349	B	F	A 454997	16-FEB-07 1	1	1	NO	DF2
3	527	B	F	A 455218	16-FEB-07 1	1	1	NO	FRI_BKP
	368	B	F	A 455022	16-FEB-07 1	1	1	NO	DF3
4	387	B	F	X 455042	16-FEB-07 1	1	1	NO	DF4
5	407	B	F	A 455063	16-FEB-07 1	1	1	NO	DF5
6	428	B	F	A 455083	16-FEB-07 1	2	2	NO	*
7	450	B	F	X 455103	16-FEB-07 1	1	1	NO	DF7
8	473	B	F	A 455123	16-FEB-07 1	1	1	NO	DF8
9	497	B	F	A 455143	16-FEB-07 1	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	1	NO	DF1	
421968	15-FEB-07 201	A 1	1	1	NO	TAG20070215T171219	

List of SPFILE Backups

=====

Modification Time	BS Key	S	#Pieces	#Copies	Compressed	Tag
15-FEB-07	330	A 1	1	1	NO	DF1
15-FEB-07	201	A 1	1	1	NO	TAG20070215T171219

Example 2-89 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_2xdbg13_.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/ol_mf_TAG20070216T140706_2xdbbz5tb_.ctl
Tag: TAG20070216T140706

594     A 16-FEB-07         460650    16-FEB-07
Name: /disk2/PROD/controlfile/ol_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 2-90 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 2-91 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
=====

```

LIST

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.

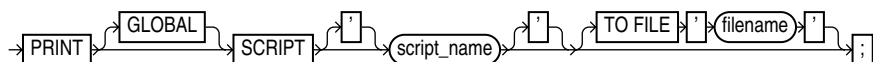
If the specified script is a local script, then RMAN must be connected to the target database that it was connected to when you created or replaced the script.

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
<code>GLOBAL</code>	Identifies the script as global. If <code>GLOBAL</code> 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 2-101 for an explanation of the difference between global and local scripts
<i>script_name</i>	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.
<code>TO FILE 'filename'</code>	Sends output to the specified file instead of standard output.

Examples

Example 2–92 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 2–93 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:=`

→ `QUIT` ↵

Example

Example 2-94 Quitting RMAN

This example terminates RMAN (sample output included):

```
RMAN> QUIT
```

```
Recovery Manager complete.
```

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 logs 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.

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.
- The target database must not be a standby database.
- You can only recover blocks marked media corrupt. The V\$DATABASE_BLOCK_CORRUPTION view indicates which blocks in a file were marked corrupt since the most recent BACKUP 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 logs 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.

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 that 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 logs for recovery. RMAN uses the following search order:

1. Incremental backup sets on disk or tape
2. Archived redo logs 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 logs, 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

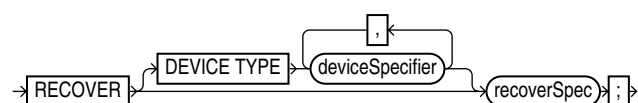
You must `RESTORE` datafiles before you can recover them. RMAN can recover through `RESETLOGS` operations transparently if the datafiles to be recovered are from a parent incarnation. If required, the `RECOVER` command can also restore and apply archived logs and incremental backups from previous database 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 logs from the missing incarnations.

See Also: `RESTORE` on page 2-211 command for explanation of the default location for restoring archived redo logs. Note that RMAN automatically specifies the `MAXSIZE` option when staging logs in the flash recovery area.

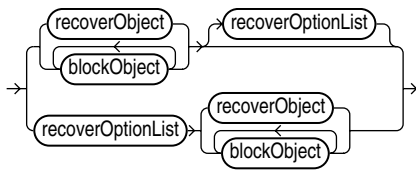
Syntax

`recover::=`



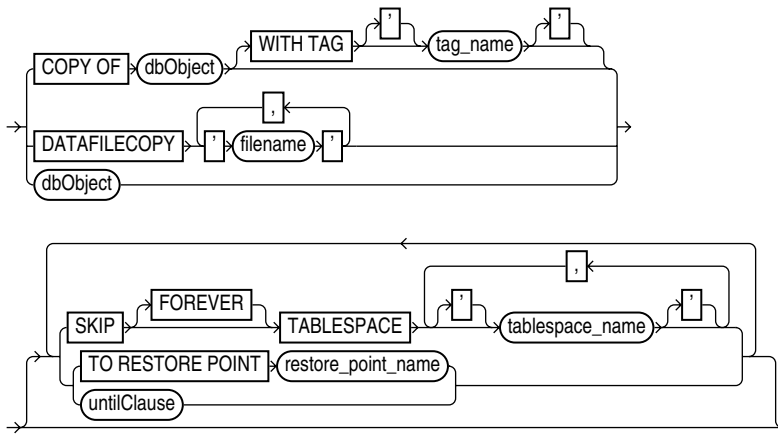
(*deviceSpecifier::=* on page 3-15, *recoverObject::=* on page 2-180, *recoverOptionList::=* on page 2-181)

recoverSpec::=



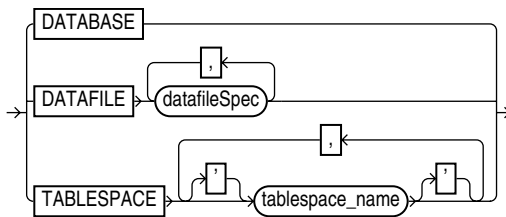
(*recoverObject::=* on page 2-180, *blockObject::=* on page 2-180, *recoverOptionList::=* on page 2-181)

recoverObject::=



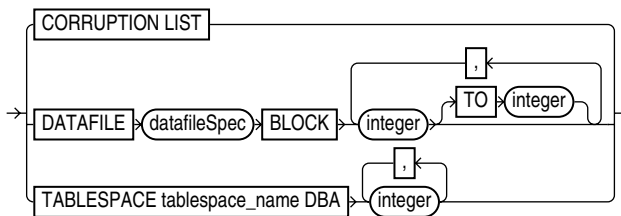
(*dbObject::=* on page 2-180, *blockObject::=* on page 2-180, *untilClause::=* on page 3-39)

dbObject::=

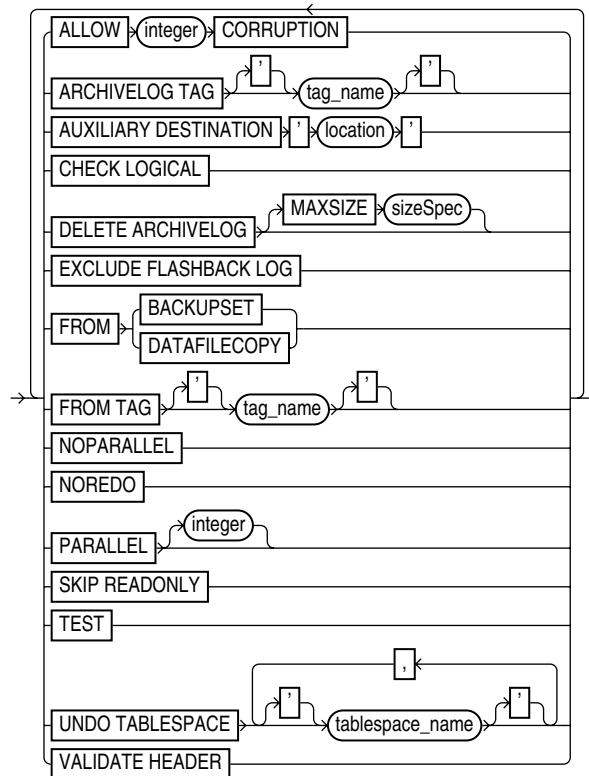
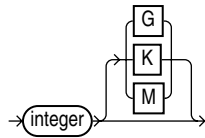


(*datafileSpec::=* on page 3-14)

blockObject::=



(*datafileSpec::=* on page 3-14)

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 3-15

recoverSpec

Specifies the type of object being recovered.

recoverSpec

Syntax Element	Description
<i>recoverObject</i>	Specifies the type of object being recovered.
<i>blockObject</i>	Specifies the blocks to be recovered with block media recovery.
<i>recoverOptionList</i>	Specifies recovery options.

recoverObject

This subclause specifies which files to recover. Refer to *recoverObject::=* on page 2-180 for the syntax diagram.

Syntax Element	Description
COPY OF <i>dbObject</i>	<p>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.</p> <p>This command updates a datafile copy and is <i>not</i> a media recovery of a current database file. This command is meant to be used in conjunction with the <code>BACKUP ... FOR RECOVER OF COPY</code> syntax to implement a strategy using incrementally updated backups.</p> <p>The following requirements must be met:</p> <ul style="list-style-type: none"> 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. <p>RMAN selects one suitable copy if there are multiple possible copies to which the incrementals can be applied to carry out the operation.</p> <p>Note: RMAN issues a warning (not an error) if it cannot recover to the specified time (or current time if none is specified) because no incrementals are available.</p>
WITH TAG <i>tag_name</i>	Specifies a tag name to identify the image copy to be rolled forward.
DATAFILECOPY ' <i>filename</i> '	Applies incremental backups to the specified datafile image copy (see Example 2-98). Refer to description of <code>RECOVER COPY OF</code> .
<i>dbObject</i>	<p>Specifies the data blocks that require recovery.</p> <p>See Also: <i>dbObject</i> on page 2-183</p>
SKIP	<p>Takes the datafiles in the specified tablespaces offline before starting media recovery. These files are left offline after the media recovery is complete.</p> <p>This option is useful for avoiding recovery of tablespaces containing only temporary data or for postponing recovery of some tablespaces.</p>
FOREVER	<p>Takes the datafiles offline with the DROP option (see Example 2-97 on page 2-188). Use <code>SKIP FOREVER TABLESPACE</code> when you intend to drop the specified tablespaces after opening the database with the <code>RESETLOGS</code> option.</p> <p>Note: If you perform incomplete recovery, then <code>SKIP</code> requires the <code>FOREVER</code> option.</p>
TABLESPACE <i>tablespace_name</i>	Specifies the name of the tablespace to take offline.
TO RESTORE POINT <i>restore_point_name</i>	<p>Specifies a restore point for termination of the <code>RECOVER</code> 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 <i>and including</i> the SCN corresponding to the restore point.</p>

Syntax Element	Description
<i>untilClause</i>	<p>Specifies a past time, SCN, or log sequence number for termination of the RECOVER command.</p> <p>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 2-178 for details). After database point-in-time recovery (DBPITR), you must open the database with the RESETLOGS option.</p> <p>See Also: <i>untilClause</i> on page 3-39</p>

dbObject

This subclass specifies whether to recover the database or a subset of the database. Refer to *dbObject::=* on page 2-180 for the syntax diagram.

Syntax Element	Description
DATABASE	<p>Specifies that the entire database is to be recovered (see Example 2-97 on page 2-188). The database must be mounted but not open.</p> <p>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.</p> <p>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 2-99 on page 2-189).</p> <p>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.</p>
DATAFILE <i>datafileSpec</i>	<p>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.</p> <p>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.</p> <p>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 <i>Oracle Database Backup and Recovery User's Guide</i>.</p> <p>See Also: <i>datafileSpec</i> on page 3-14</p>
TABLESPACE <i>tablespace_name</i>	<p>Specifies a list of one or more tablespaces to recover (see Example 2-95 on page 2-187 and Example 2-96 on page 2-187). The target database must be mounted or open. If the database is open, then the tablespaces to be recovered must be offline.</p> <p>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.</p>

blockObject

This subclause specifies the data blocks that require recovery. Refer to *blockObject::=* on page 2-180 for the syntax diagram. Refer to "Prerequisites" on page 2-178 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). You can only perform complete recovery of individual blocks.

Syntax Element	Description
CORRUPTION LIST	<p>Recovers all physically corrupt blocks listed in the V\$DATABASE_BLOCK_CORRUPTION view. Logical corruption cannot be repaired using block media recovery. To repair logical corruption, restore the datafile from backup and perform media recovery.</p> <p>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:</p> <ul style="list-style-type: none"> 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. <p>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.</p> <p>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.</p>
DATAFILE <i>datafileSpec</i> BLOCK <i>integer</i> TO <i>integer</i>	<p>Recovers an individual data block or set of data blocks within a datafile. Note that the TO range is inclusive, so that BLOCK 10 TO BLOCK 20 includes both block 10 and block 20.</p> <p>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:</p> <ul style="list-style-type: none"> I/O errors causing minor data loss Memory corruptions that get written to disk <p>If you do not specify an option from <i>recoverOptionList</i>, 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.</p> <p>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.</p> <p>See Also: <i>datafileSpec</i> on page 3-14</p>
TABLESPACE <i>tablespace_name</i> DBA <i>integer</i>	<p>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.</p> <p>Note: The datafile header block (block 1) cannot be recovered.</p>

recoverOptionList

This subclause specifies various recovery options. Refer to [recoverOptionList::=](#) on page 2-181 for the syntax diagram.

Syntax Element	Description
ALLOW <i>integer</i> CORRUPTION	<p>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.</p> <p>When you use this clause during trial recovery (that is, in conjunction with the TEST clause), <i>integer</i> can exceed 1. When using this clause during normal recovery, <i>integer</i> can only be 0 or 1.</p>
ARCHIVELOG TAG <i>tag_name</i>	<p>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 logs for recovery, then RMAN uses logs or incremental backups as needed from whatever is available.</p>
AUXILIARY DESTINATION ' <i>location</i> '	<p>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.</p> <p>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.</p> <p>See also: The chapter on TSPITR in <i>Oracle Database Backup and Recovery User's Guide</i> for more details about the auxiliary destination</p>
CHECK LOGICAL	<p>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.</p> <p>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.</p> <p>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.</p>
DELETE ARCHIVELOG	<p>Deletes archived logs restored from backups or copies that are no longer needed. RMAN does not delete archived logs that were already on disk before the RESTORE command started. If you do not specify MAXSIZE, then RMAN deletes restored archived logs as they are applied.</p> <p>Note: If archived redo logs are restored to the flash recovery area, then the DELETE ARCHIVELOG option is enabled by default.</p>
MAXSIZE <i>sizeSpec</i>	<p>Does not use more than <i>sizeSpec</i> amount of disk space for restored archived redo logs. 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.</p>
EXCLUDE FLASHBACK LOG	<p>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.</p>
FROM BACKUPSET	<p>Specifies that only backup sets should be restored.</p>
FROM DATAFILECOPY	<p>Specifies that only datafile image copies should be restored.</p>

Syntax Element	Description
FROM TAG ' <i>tag_name</i> '	<p>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.</p> <p>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</p>
NOPARALLEL	<p>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).</p>
NOREDO	<p>Suppresses the application of redo logs during recovery. Only incremental backups are applied.</p> <p>One use of this option is to use incremental backups to update full backups of NOARCHIVELOG databases (see Example 2-100 on page 2-189). 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.</p> <p>Note: Incremental backups of NOARCHIVELOG databases can only be taken after a consistent shutdown.</p> <p>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 <i>Oracle Data Guard Concepts and Administration</i>.</p>
PARALLEL	<p>Specifies parallel recovery (default).</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>Note: The RECOVERY_PARALLELISM initialization parameter controls instance or crash recovery only. Media recovery is not affected by the value used for RECOVERY_PARALLELISM.</p> <p>See Also: The description of the PARALLEL clause in the discussion of CREATE TABLE in <i>Oracle Database SQL Language Reference</i></p>
<i>integer</i>	<p>Specifies the <i>integer</i> degree of parallelism.</p> <p>Each parallel thread may use one or two parallel execution servers. Typically, it is not necessary for you to specify the degree of parallelism.</p>
SKIP READONLY	<p>Omits read-only files from the recovery.</p>

Syntax Element	Description
TEST	<p>Initiates a trial recovery.</p> <p>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.</p> <p>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.</p>
UNDO TABLESPACE 'tablespace_name'	<p>Specifies a list of tablespaces with undo segments at the target time. Only for use with RECOVER TABLESPACE.</p> <p>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.</p> <p>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.</p>
VALIDATE HEADER	<p>Reports and validates—but does not restore—the backups that RMAN could use to restore files needed for the recovery.</p> <p>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.</p> <p>See Also: The description of the RESTORE . . . PREVIEW option</p>

Examples

Example 2–95 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 2–96 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 2-97 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 logs. You do not need to restore tablespace `tools` because it has been read-only since before the most recent backup. After connecting to the target 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 2-98 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 execution, the script creates a level 1 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 2–99 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. The following commands restore a control file that is usable by the standby database, update the filenames to existing files on disk, and recover the database:

```

CONNECT TARGET SYS/password@dgprod3
CONNECT CATALOG rman/password@catdb
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
RECOVER DATABASE;

```

You can then start redo apply on the standby database.

Example 2–100 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 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 logs, there is no way to recover changes made after the incremental backup.

Example 2–101 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
.

```

RECOVER

```
.
.
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 2-98

Prerequisites

Execute this command only at the RMAN prompt. You 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 an 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* ::=**

→ REGISTER DATABASE ;

Example

Example 2–102 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/password@catdb

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 the connection to the 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.

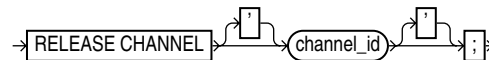
Usage Notes

Maintenance channels are unaffected by `ALLOCATE CHANNEL` and `RELEASE CHANNEL` commands issued within a `RUN` command.

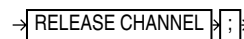
Using `RELEASE CHANNEL` to release channels within `RUN` is optional because RMAN automatically releases all normal channels when a `RUN` command terminates.

Syntax

release::=



releaseForMaint::=



Semantics

Syntax Element	Description
<code>channel_id</code>	Specifies the case-sensitive channel ID used in the <code>ALLOCATE CHANNEL</code> command (see Example 2-103).

Examples

Example 2-103 *Releasing a Channel Allocated in a RUN Command*

This example allocate 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 2-104 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 logs. If the files needed for recovery are not available, then the recovery will not be possible.

Usage Notes

Repairs are consolidated when possible so that a single repair can fix multiple failures. The command performs an implicit `ADVISE FAILURE` if this command has not yet been executed in the current session.

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 <i>integer</i>	Specifies a repair option by its option number (<i>not</i> 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 2-106 on page 2-198).

Examples

Example 2-105 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 2–106 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 2-101 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

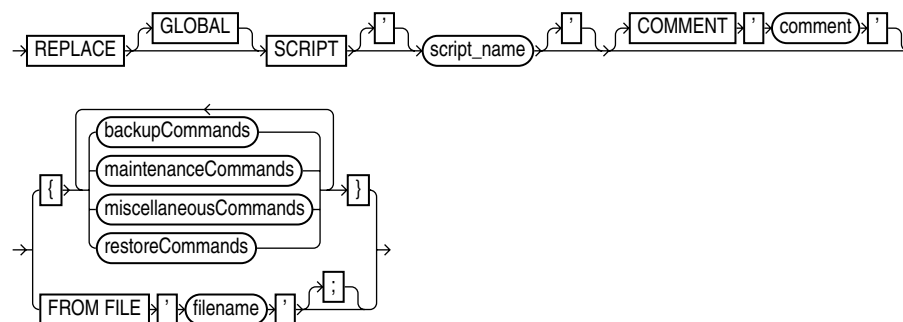
RMAN supports the use of substitution variables in a stored script. `&1` indicates where to place the first value, `&2` indicate where to place the second value, and so on. Special characters must be quoted.

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 2-232) or enter them when prompted (see [Example 2-60](#) on page 2-103).

Syntax

replaceScript::=



([backupCommands::=](#) on page 2-238, [maintenanceCommands::=](#) on page 2-238, [miscellaneousCommands::=](#) on page 2-239, [restoreCommands::=](#) on page 2-239)

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. See Also: "Usage Notes" on page 2-101 for an explanation of the difference between global and local scripts
SCRIPT <i>script_name</i>	Identifies the local or global script being replaced.
COMMENT ' <i>comment</i> '	Associates an explanatory comment with the stored script in the recovery catalog. The comment is shown in the output of LIST SCRIPT NAMES.
<i>backupCommands</i>	Specifies commands to include in the stored script. The commands allowable within the brackets of the REPLACE SCRIPT ' <i>script_name</i> ' {...} command are the same commands supported within a RUN block. Any command that is legal within a RUN command is permitted in the stored script. The following commands are not permitted within stored scripts: RUN, @, and @@.
<i>maintenanceCommands</i>	
<i>miscellaneousCommands</i>	
<i>restoreCommands</i>	
FROM FILE ' <i>filename</i> '	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 2-107 Replacing a Recovery Catalog Script

Assume that you use CREATE SCRIPT to create a global script named backup_db:

```
CONNECT TARGET SYS/password@prod
CONNECT CATALOG rco/password@catdb
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

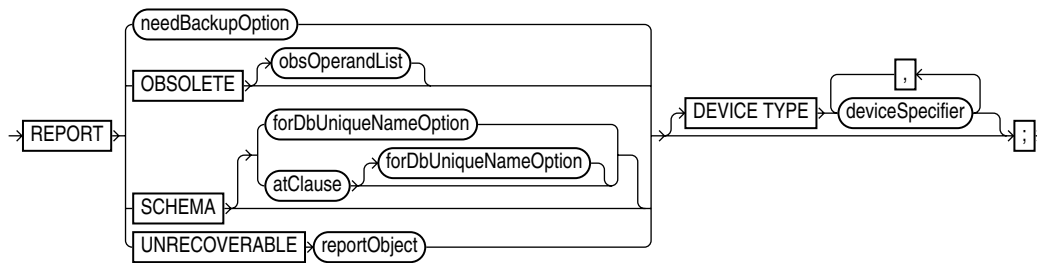
Prerequisites

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.

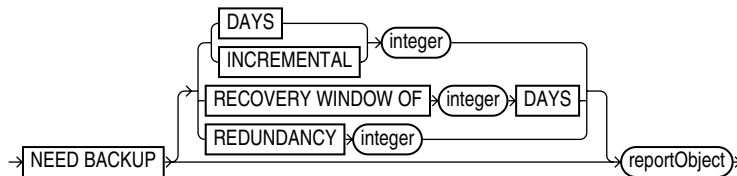
Syntax

report::=

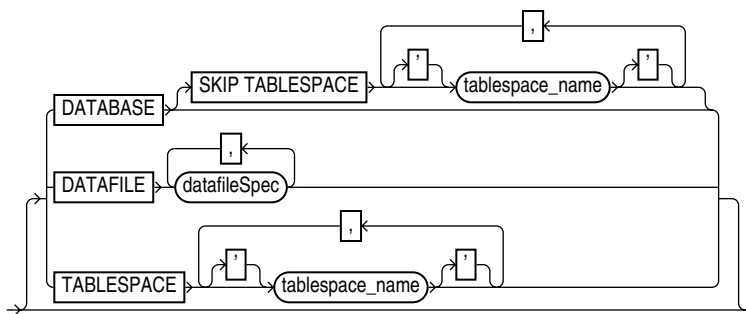


(*needBackupOption::=* on page 2-202, *atClause::=* on page 2-203, *reportObject::=* on page 2-203, *obsOperandList::=* on page 3-35, *deviceSpecifier::=* on page 3-15)

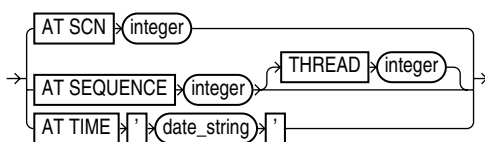
needBackupOption::=



(*reportObject::=* on page 2-203)

reportObject::=

(*datafileSpec::=* on page 3-14)

atClause::=**Semantics****report**

This clause specifies the type of report.

Syntax Element	Description
<i>needBackupOption</i>	Lists files that require backups. See Also: <i>needBackupOption</i> on page 2-204
OBSOLETE <i>obsOperandList</i>	Lists full backups, datafile copies, and archived redo logs recorded in the RMAN repository that can be deleted because they are no longer needed. See Table 2-33 for description of output. The command works in two steps: <ol style="list-style-type: none"> 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 logs 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 logs are not considered obsolete if they can be applied to nonobsolete level 0 or full backups. <p>The subclause <i>obsOperandList</i> describes the criteria that RMAN uses to determine what is obsolete. If you do not specify parameters in <i>obsOperandList</i>, then RMAN uses the options specified in CONFIGURE RETENTION POLICY (see Example 2-110 on page 2-208). 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 <code>REPORT OBSOLETE</code> with no other options and the retention policy is <code>NONE</code>.</p> <p>Note: A backup made with the <code>KEEP UNTIL TIME</code> clause is obsolete after the <code>KEEP</code> time passes, regardless of the configured retention policy settings.</p>

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 2-28 for description of output.
<i>forDbUniqueNameOption</i>	Reports the names of all datafiles and tablespaces for the database specified by its DB_UNIQUE_NAME. You can specify a database with <i>db_unique_name</i> 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: <i>forDbUniqueNameOption</i> on page 3-19 for descriptions of the options in this clause
<i>atClause</i>	Specifies an SCN, log sequence number, or time.
UNRECOVERABLE <i>reportObject</i>	Lists all unrecoverable datafiles. See Table 2-34 on page 2-207 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 <i>deviceSpecifier</i>	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 <i>reportObject</i> 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 <i>integer</i>	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 logs. See Table 2-29 for description of output. If the target database control file is mounted and current, then RMAN makes the following optimizations to this report: <ul style="list-style-type: none"> ■ 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 <i>integer</i>	Specifies a threshold number of incremental backups required for recovery (see Example 2-109 on page 2-208). If complete recovery of a datafile requires more than <i>integer</i> incremental backups, then the datafile requires a new full backup. See Table 2-30 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 <i>integer</i> 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 - <i>integer</i> . See Table 2-31 for description of output.
REDUNDANCY <i>integer</i>	Specifies the minimum number of backups or copies that must exist for a datafile to be considered <i>not</i> in need of a backup. In other words, a datafile needs a backup if there are fewer than <i>integer</i> 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 2-32 for description of output.
<i>reportObject</i>	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 <i>tablespace_name</i> to exclude the specified tablespace from the DATABASE specification.
DATAFILE <i>datafileSpec</i>	Lists the specified datafiles. RMAN reports on backups or datafile copies that contain at least one of the specified datafiles.
TABLESPACE <i>tablespace_name</i>	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 <i>integer</i>	Specifies an SCN.
AT SEQUENCE <i>integer</i>	Specifies a log sequence number. The integer indicates the time when the specified log was first opened.
THREAD <i>integer</i>	Specifies a redo THREAD number. The integer indicates the time when the thread was first opened.
AT TIME ' <i>date_string</i> '	Specifies a date (see Example 2-108 on page 2-207). 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 2–28, " Report of Database Schema"
- Table 2–29, " Report of Files Whose Recovery Needs More Than *n* Days of Archived Logs"
- Table 2–30, " Report of Files That Need More than *n* Incrementals During Recovery"
- Table 2–31, " Report of Files That Must Be Backed Up to Satisfy *n* Days Recovery Window"
- Table 2–32, " Report of Files with Fewer Than *n* Redundant Backups"
- Table 2–33, " Report of Obsolete Backups and Copies"
- Table 2–34, " Report of Files that Need Backup Due to Unrecoverable Operations"

Table 2–28 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 2–29 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 <i>n</i> days of archived redo logs for recovery.
Days	The number of days of archived redo data required for recovery.
Name	The name of the datafile.

Table 2–30 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 <i>n</i> incrementals for complete recovery.
Incrementals	The number of incremental backups required for complete recovery.
Name	The name of the datafile.

Table 2–31 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 <i>n</i> days.
Days	The number of days required for complete recovery.

Table 2–31 (Cont.) Report of Files That Must Be Backed Up to Satisfy *n* Days Recovery

Column	Indicates
Name	The name of the datafile that requires backup.

Table 2–32 Report of Files with Fewer Than *n* Redundant Backups

Column	Indicates
File	The absolute datafile number of a datafile with less than <i>n</i> redundant backups.
#bkps	The number of backups that exist for this file.
Name	The name of the file.

Table 2–33 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 2–34 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 2–108 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 2–109 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 2–110 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_2xnpmp01_.bkp
Datafile Copy	1073	19-FEB-07	/disk2/PROD/datafile/o1_mf_system_2xmz515m_.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. You must be connected to the 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 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::=

```
→ RESET DATABASE TO INCARNATION <primaryKey> ;
```

Semantics

Syntax Element	Description
<i>primary_key</i>	Changes the current incarnation to a noncurrent database incarnation specified by the primary key of the <code>DBINC</code> 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 <code>RESET DATABASE TO INCARNATION</code> , you can run RMAN commands such as FLASHBACK DATABASE , RESTORE , and RECOVER .

Examples

Example 2-111 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. if the current control file does not know about it, then
# you must restore an older control file
```

```
STARTUP NOMOUNT;
RESTORE CONTROLFILE UNTIL TIME 'SYSDATE-250';
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 then 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	154877	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 flash recovery area that is physically *different* from the recovery area used by the production database, then set `DB_RECOVERY_FILE_DEST` in the test database instance to the new location.
- If the test database will use a flash recovery area that is physically the *same* as the recovery area used by the production database, then set `DB_UNIQUE_NAME` in the test database instance to a different name from the production database.

If you do not perform either of the preceding actions, then RMAN assumes that you are restoring the production database and deletes flashback logs from flash recovery area because they are considered unusable.

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

Encrypted Backup Sets

How RMAN handles encrypted backup sets during restore operations depends upon 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`.

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 logs 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 2-20 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.

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 2-113](#) on page 2-222). If you do not issue `SWITCH` commands, then RMAN considers the restored files as valid copies for use in future restore

operations. [Table 2–35](#) describes the behavior of the RESTORE, SET NEWNAME, and SWITCH commands.

Table 2–35 SET NEWNAME, SWITCH, and RESTORE

SET NEWNAME Run	SWITCH Run	RESTORE Behavior
No	N/A	RMAN restores the files to their current path names.
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 2–36](#). Required commands and options for restoring autobackups are summarized in [Table 2–37](#).

Table 2–36 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 2–37 for required commands and options.	Must specify FROM AUTOBACKUP. Restores only to <i>filename</i> .	First run SET DBID. Restores from specified file (cannot restore from TAG). If TO ' <i>filename</i> ' not used, restores to all CONTROL_FILES locations.
No catalog, target database mounted or open	Error. Must use TO ' <i>filename</i> ', where <i>filename</i> is not in CONTROL_FILES list.	Error. Must use TO ' <i>filename</i> ', where <i>filename</i> is not in CONTROL_FILES list.	Restores only to <i>filename</i> , where <i>filename</i> is not in CONTROL_FILES list.	Restores from specified file. If TO ' <i>filename</i> ' not used, restores to all CONTROL_FILES locations.
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 <i>filename</i> , where <i>filename</i> is not in CONTROL_FILES list.	Restores from specified file. If TO ' <i>filename</i> ' not used, restores to all CONTROL_FILES locations.
Catalog, target database mounted or open	Error. Must use TO ' <i>filename</i> ', where <i>filename</i> is not in CONTROL_FILES list.	Do not use with recovery catalog.	Restores only to <i>filename</i> , where <i>filename</i> is not in CONTROL_FILES list.	RMAN issues error RMAN-06496. Use TO ' <i>filename</i> ' 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 2-21.

Control File and Server Parameter File Autobackup Options

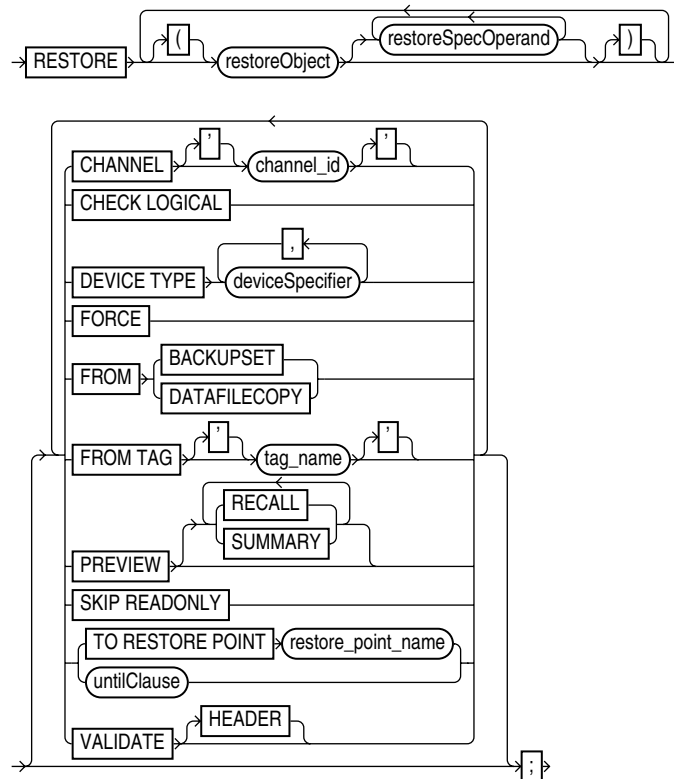
When restoring an autobackup, the commands and options that you use depend on the autobackup type (control file or server parameter file) and location (inside or outside flash recovery area). The options are summarized in [Table 2-37](#).

Table 2-37 RESTORE ... FROM AUTOBACKUP

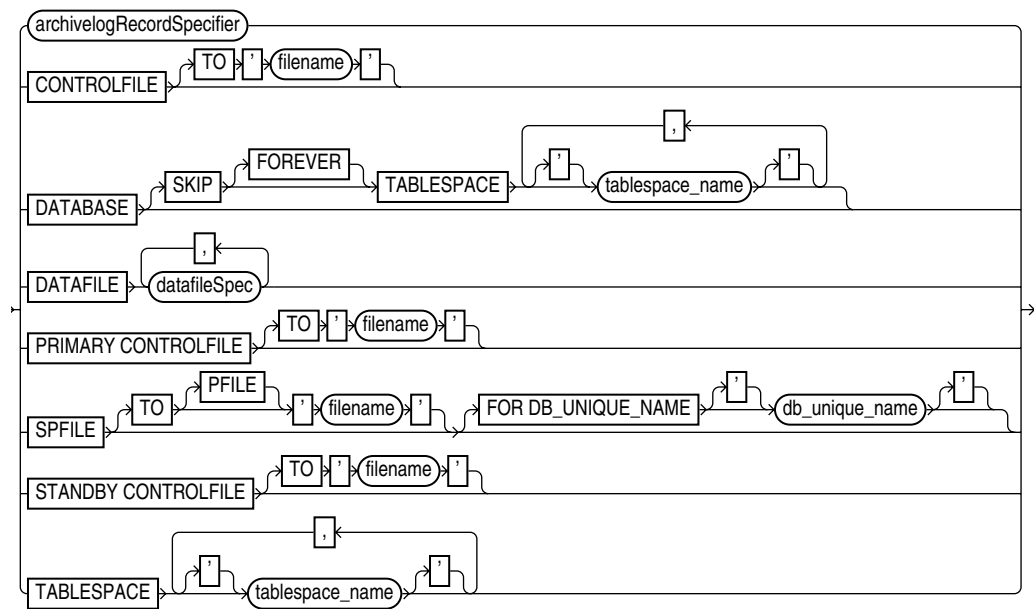
Restore Object	Autobackup Location	Run SET DBID?	Specify RECOVERY AREA on RESTORE?	Specify DB_NAME or DB_UNIQUE_NAME on RESTORE?	Run SET CONTROLFILE AUTOBACKUP FORMAT?
SPFILE	Recovery area	No	Yes	Yes	No
SPFILE	Outside recovery area	Yes	No	No	Only if autobackup is not in default location
Control file	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

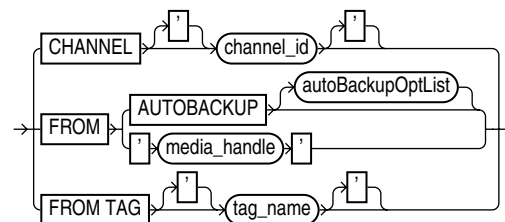
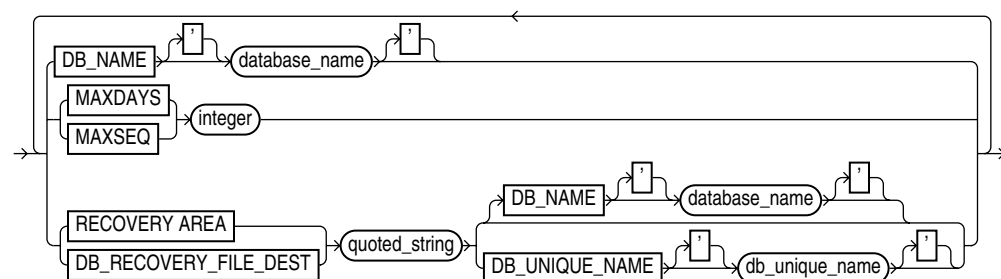
restore::=



(*restoreObject::=* on page 2-215, *restoreSpecOperand::=* on page 2-215, *deviceSpecifier::=* on page 3-15, *untilClause::=* on page 3-39)

restoreObject::=

(*archivelogRecordSpecifier::=* on page 3-6, *datafileSpec::=* on page 3-14)

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.

Syntax Element	Description
<i>restoreSpecOperand</i>	Specifies options for the <i>restoreObject</i> clause.
CHANNEL <i>channel_id</i>	Refer to the <i>restoreSpecOperand</i> clause.
CHECK LOGICAL	<p>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.</p> <p>If the total number of physical and logical corruptions detected in a file is less than its <code>SET MAXCORRUPT</code> setting, then the RMAN command completes and the database populates the <code>V\$DATABASE_BLOCK_CORRUPTION</code> view with corrupt block ranges. If <code>MAXCORRUPT</code> is exceeded, then the command terminates without populating the views.</p> <p>When restoring a backup datafile, RMAN honors the <code>DB_BLOCK_CHECKSUM</code> initialization parameter setting. RMAN clears the checksum if <code>DB_BLOCK_CHECKSUM</code> is set to <code>false</code>. If set to <code>typical</code>, then RMAN verifies the checksum when restoring from the backup and writing to the datafile. If the initialization parameter <code>DB_BLOCK_CHECKSUM=typical</code>, and if <code>MAXCORRUPT</code> is not set, then specifying <code>CHECK LOGICAL</code> detects all types of corruption that are possible to detect.</p> <p>Note: The <code>MAXCORRUPT</code> setting represents the total number of physical and logical corruptions permitted on a file.</p>
DEVICE TYPE <i>deviceSpecifier</i>	<p>Allocates automatic channels for the specified device type only. For example, if you configure automatic disk and tape channels, and issue <code>RESTORE . . . DEVICE TYPE DISK</code>, then RMAN allocates only disk channels. You must have already configured a device type by using <code>CONFIGURE</code> (except for <code>DISK</code>, which is preconfigured) before specifying the <code>DEVICE TYPE</code> option.</p> <p>Note: You cannot manually allocate channels within a <code>RUN</code> block and then run <code>RESTORE</code> with the <code>DEVICE TYPE</code> clause.</p> <p>See Also: <i>deviceSpecifier</i> on page 3-15</p>
FORCE	Overrides the restartable restore feature and restores all files regardless of whether they need to be restored. If you do not specify <code>FORCE</code> , then RMAN restores a file only if its header information does not match the information in the control file.
FROM BACKUPSET	<p>Specifies that RMAN should restore from backup sets only. By default <code>RESTORE</code> chooses the file copy or backup set that needs the least media recovery.</p> <p>If you use the <code>FROM BACKUPSET</code> 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 <code>sbt</code> channels have been allocated, then RMAN cannot find a candidate backup set to restore, and the <code>RESTORE</code> command fails.</p>
FROM DATAFILECOPY	Specifies that RMAN should restore datafile copies only. By default <code>RESTORE</code> chooses the file copy or backup set that needs the least media recovery. If you use the <code>FROM DATAFILECOPY</code> option, then the allocated channels must be of <code>DEVICE TYPE DISK</code> .
FROM TAG <i>tag_name</i>	Refer to the <i>restoreSpecOperand</i> clause.

Syntax Element	Description
PREVIEW	<p>Reports—but does not restore—the backups that RMAN could use to restore to the specified time. RMAN queries the metadata and does not actually read the backup files.</p> <p>The <code>RESTORE . . . PREVIEW</code> output is in the same format as the <code>LIST BACKUP</code> output (see Example 2-118 on page 2-224).</p> <p>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.</p> <p>Offsite backups are marked as <code>AVAILABLE</code> 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. <code>RESTORE . . . PREVIEW</code> can identify backups needed for a <code>RESTORE</code> operation that are stored on media that requires retrieval. The output indicates whether backups are stored offsite.</p> <p>If a needed backup is stored offsite, but the media manager does not support offsite backups, then your options are:</p> <ul style="list-style-type: none"> ■ Use <code>CHANGE . . . UNAVAILABLE</code> to prevent RMAN from selecting the needed offsite backups, and attempt the <code>RESTORE . . . PREVIEW</code> 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 <code>RESTORE . . . PREVIEW</code> with the <code>RECALL</code> option. <p>See Also: LIST on page 2-154, specifically the <code>BACKUPS</code> and <code>SUMMARY</code> options</p>
RECALL	<p>Instructs the media manager to retrieve the backup media needed for the specified restore operation from offsite storage (see Example 2-119 on page 2-224).</p> <p>Note: This option only works if your media manager supports this functionality. You can use <code>RESTORE . . . PREVIEW</code> periodically to monitor whether the needed backups are stored locally again.</p>
SUMMARY	<p>Summarizes the backups that RMAN would restore. The output is in the same format as the output of the <code>LIST BACKUPS . . . SUMMARY</code> command.</p>
SKIP READONLY	<p>Does not restore read-only files.</p>
TO RESTORE POINT <i>restore_point_name</i>	<p>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 <i>and including</i> the SCN corresponding to the restore point.</p>
<i>untilClause</i>	<p>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.</p> <p>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 <code>UNTIL</code> clause must fall within the current database incarnation.</p> <p>See Also: untilClause on page 3-39</p>
VALIDATE	<p>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 2-120 on page 2-225). No files are restored.</p> <p>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.</p> <p>Note: If you use <code>RESTORE</code> with the <code>VALIDATE</code> option, then the database can be open with datafiles online.</p> <p>See Also: VALIDATE on page 2-276</p>

Syntax Element	Description
HEADER	<p>Reports and validates—but does not restore—the backups that RMAN could use to restore to the specified time.</p> <p>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.</p> <p>See Also: The descriptions of the RESTORE PREVIEW option and the RECOVER ... VALIDATE HEADER option</p>

restoreObject

This subclause specifies the objects to be restored: control files, datafiles, archived redo logs, 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	<p>Restores the specified range of archived redo logs.</p> <p>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.</p> <p>Because the RECOVER command automatically restores archived redo logs as needed, you should seldom need to restore logs manually. Possible reasons for manually restoring archived redo logs 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 the FROM SCN or SCN BETWEEN ... AND ... clause.</p> <p>Note: The database can be started, mounted, or open for this operation.</p> <p>See Also: archivelogRecordSpecifier on page 3-6</p>
CONTROLFILE	<p>Restores either a standby or backup control file depending on the target database role.</p> <p>If the control file is lost, then restore the control file (see Table 2-36 on page 2-213) and restore the database after mounting the restored control file. You must always run the RECOVER command after mounting a restored control file and must open the database with the RESETLOGS option.</p> <p>Note: If the 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 database is mounted or open, then you must specify the TO <i>filename</i> clause with RESTORE CONTROLFILE.</p> <p>When you run RESTORE with a backup control file while connected to a recovery catalog (see Example 2-114 on page 2-222), RMAN automatically updates the control file to reflect the structure of the restored database based on the metadata in the catalog.</p>

Syntax Element	Description
TO 'filename'	Restores the control file to the specified filename. Table 2-36 on page 2-213 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 <i>not</i> 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. If you specify FOREVER, then RMAN specifies the DROP option of ALTER DATABASE DATAFILE . . . OFFLINE when taking the datafiles that belong to the tablespace offline before the restore. The DROP option indicates that RMAN does not intend to recover these files and intends to drop their tablespaces from the database after the database is opened again. Thus, FOREVER indicates that RMAN never intends to do anything with the skipped tablespaces again.
DATAFILE <i>datafileSpec</i>	Restores the datafiles specified by filename or absolute datafile number (see Example 2-113 on page 2-222). Note: Do not specify a datafile more than once in a restore job. For example, the following command is illegal because datafile 1 is both specified explicitly and implied by the SYSTEM tablespace: <pre>RESTORE TABLESPACE SYSTEM DATAFILE 1;</pre> See Also: datafileSpec on page 3-14
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 <code>dgny</code> to standby database <code>dgsf</code> , so that <code>dgsf</code> is the new primary database. You want to restore a control file on <code>dgsf</code> , but the recovery catalog was not resynchronized and still shows <code>dgsf</code> 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 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 with the restored server parameter file. Note: If the 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 database is started, mounted, or open, and if the database was started with a server parameter file, then you must specify the TO <i>filename</i> clause with RESTORE SPFILE .

Syntax Element	Description
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'	<p>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.</p> <p>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.</p> <p>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.</p> <p>Note: Using RESTORE SPFILE when the DB_NAME is not unique in the recovery catalog produces an RMAN-6758 error.</p> <p>See Also: <i>Oracle Data Guard Concepts and Administration</i> for a detailed procedure for restoring the server parameter file in a Data Guard environment</p>
TO 'filename'	Restores the standby control file to the specified filename. Table 2-36 explains the RMAN behavior when restoring the control file with the TO clause.
STANDBY CONTROLFILE	<p>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.</p> <p>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.</p> <p>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.</p> <p>See Also: Table 2-36 for restrictions and usage notes</p> <p>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.</p>
TABLESPACE 'tablespace_name'	<p>Restores all datafiles in the specified tablespaces (see Example 2-112 on page 2-222).</p> <p>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.</p> <p>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.</p>

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 <i>'channel_id'</i>	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.
FROM AUTOBACKUP	Restores a control file autobackup (see Example 2–115 on page 2-223). This option is only legal 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 2–36 for restrictions and usage notes.
<i>autoBackupOptList</i>	Specifies parameters that control the search for a control file autobackup.
<i>'media_handle'</i>	Specifies the name of the control file copy or backup piece containing a control file. The <i>media_handle</i> 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 2–36 for restrictions and usage notes.
FROM TAG <i>tag_name</i>	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 <i>database_name</i>	Provides a DB_NAME to be used in searching for control file autobackups. See Table 2–37 on page 2-214 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 <i>integer</i>	Limits the search for a control file autobackup to within the specified number of days in the past.
MAXSEQ <i>integer</i>	Specifies the highest sequence number for the control file autobackup search.
RECOVERY AREA <i>'pathname'</i>	Specifies a path to the flash recovery area to search for autobackups. RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms. See Table 2–37 on page 2-214 to determine when to set this parameter.

Syntax Element	Description
DB_RECOVERY_FILE_DEST ' <i>pathname</i> '	RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms.
DB_NAME <i>database_name</i>	Provides a DB_NAME to be used in searching for control file autobackups. See Table 2-37 on page 2-214 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 <i>db_unique_name</i>	Specifies the DB__UNIQUE_NAME of the database in the specified flash 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 2-112 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 2-113 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 2-114 Restoring the Control File When Using a Recovery Catalog

This example uses a tag to identify the control file to restore. RMAN restores the control file to its default location and replicates it automatically to all CONTROL_FILES locations. The example then 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.

```
CONNECT TARGET /
CONNECT CATALOG rman/password@catdb
RUN
{ # SET DBID is not necessary when 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

```

Example 2–115 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 2–116 Restoring a Control File Autobackup to a Nondefault Location

This example is a variation on [Example 2–115](#). 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 2–117 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 2–118 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 logs.

```
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 2–119 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 in the restore of archived logs 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 2-120 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_.bkp
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.
- You are connected as `TARGET` to a standby database and want to update the recovery catalog with metadata about RMAN operations performed on this database.
- You are connected as `TARGET` to a standby database and want to update the recovery catalog with metadata about a physical change on the primary database (see [Example 2-123](#) on page 2-228).

Prerequisites

RMAN must be connected as `TARGET` to a mounted or open database and connected as `CATALOG` to a recovery catalog database.

Usage Notes

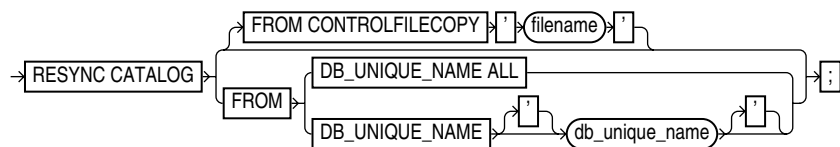
Resynchronizations are full or partial. If full, and if the target database has mounted the current control file (but not a newly created control file or a control file that is less current than a control file that was used previously), then RMAN updates all changed records for the physical schema: datafiles, 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 11g, 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 logs, 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	<p>Updates the recovery catalog with RMAN metadata in the current control file of the target database (default).</p> <p>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:</p> <ul style="list-style-type: none"> 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 logs created by archiving an online redo log, copying an existing archived log, or restoring backups of archived logs. 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'	<p>Updates the current control file and recovery catalog with RMAN metadata from a control file copy (see Example 2-122 on page 2-228). Use <i>filename</i> to specify the name of the control file copy to use for resynchronization.</p> <p>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.</p> <p>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).</p>

Syntax Element	Description
<pre>FROM DB_UNIQUE_NAME {ALL db_unique_name}</pre>	<p>Resynchronizes the recovery catalog with control file metadata in the specified database or databases (see Example 2–124 on page 2-229).</p> <p>You can specify a single database with <i>db_unique_name</i> 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.</p> <p>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.</p> <p>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.</p> <p>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.</p>

Examples

Example 2–121 Resynchronizing the Recovery Catalog in ARCHIVELOG Mode

This example performs a full resynchronization of the target database after archiving all unarchived redo logs.

```
CONNECT TARGET /
CONNECT CATALOG rman/password@catdb
SQL "ALTER SYSTEM ARCHIVE LOG CURRENT";
RESYNC CATALOG;
```

Example 2–122 Resynchronizing the Recovery Catalog from a Control File Copy

This example updates the RMAN repository in the current control file with metadata from a backup control file.

```
CONNECT TARGET /
CONNECT CATALOG rman/password@catdb
STARTUP FORCE MOUNT
RESYNC CATALOG FROM CONTROLFILECOPY '/disk1/cfile.dbf';
ALTER DATABASE OPEN;
```

Example 2–123 Resynchronizing the Recovery Catalog After a Structural Change

Suppose that you start SQL*Plus, connect to database prod, and add a datafile to tablespace users as follows:

```
CONNECT TARGET SYS/password@prod
ALTER TABLESPACE users ADD DATAFILE '"/oradata/prod/users03.dbf' '
SIZE 1M AUTOEXTEND ON
NEXT 10K MAXSIZE 10M";
```

After this change has propagated to standby database standby3, you start RMAN and connect to standby3 and the recovery catalog. You then resynchronize the catalog with the control file of the standby database. The recovery catalog is updated with metadata about the datafile added to prod.

```
CONNECT TARGET SYS/password@standby3
CONNECT CATALOG rman/password@catdb
RESYNC CATALOG;
```

Example 2–124 Resynchronizing the Recovery Catalog with a Standby Database

Suppose that primary database `prod` and standby database `dgprod3` exist in a Data Guard environment. You connect RMAN to `prod` and use `CONFIGURE` to update the persistent RMAN configuration for `dgprod3` in the recovery catalog as follows:

```
CONNECT TARGET SYS/password@prod
CONNECT CATALOG rman/password@catdb
CONFIGURE DEFAULT DEVICE TYPE TO sbt
  FOR DB_UNIQUE_NAME dgprod3;
CONFIGURE DB_UNIQUE_NAME dgprod CONNECT IDENTIFIER 'inst1';
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 configuration for `dgprod3` are not synchronized. In the same RMAN session, you synchronize the control file of `dgprod3` 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.

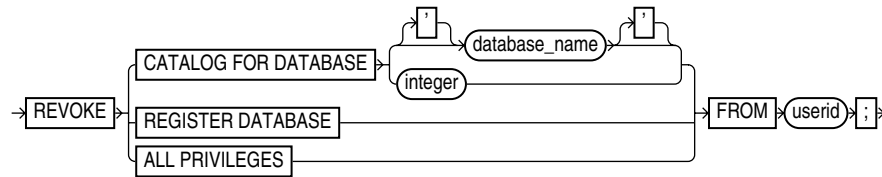
REVOKE

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
<code>CATALOG FOR DATABASE</code> { <i>datasenname</i> <i>integer</i> }	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.
<code>REGISTER DATABASE</code>	Revokes the ability to for the specified user to register new databases in this recovery catalog (see Example 2-125 on page 2-231).
<code>ALL PRIVILEGES</code>	Revokes all <code>CATALOG</code> and <code>REGISTER</code> privileges from the specified user.
<code>FROM userid</code>	Specifies the name of the user from which you are revoking privileges.

Examples

Example 2–125 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 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:

```
CONNECT CATALOG rco/password@catdb
GRANT REGISTER DATABASE TO bckop2;
GRANT CATALOG FOR DATABASE prod TO bckop3;
GRANT CATALOG FOR DATABASE prodb TO bckop3;
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.

```
CONNECT CATALOG rco/password@catdb
REVOKE REGISTER DATABASE FROM bckop2;
```

RMAN

Purpose

Use the `RMAN` command to start RMAN from the operating system command line.

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

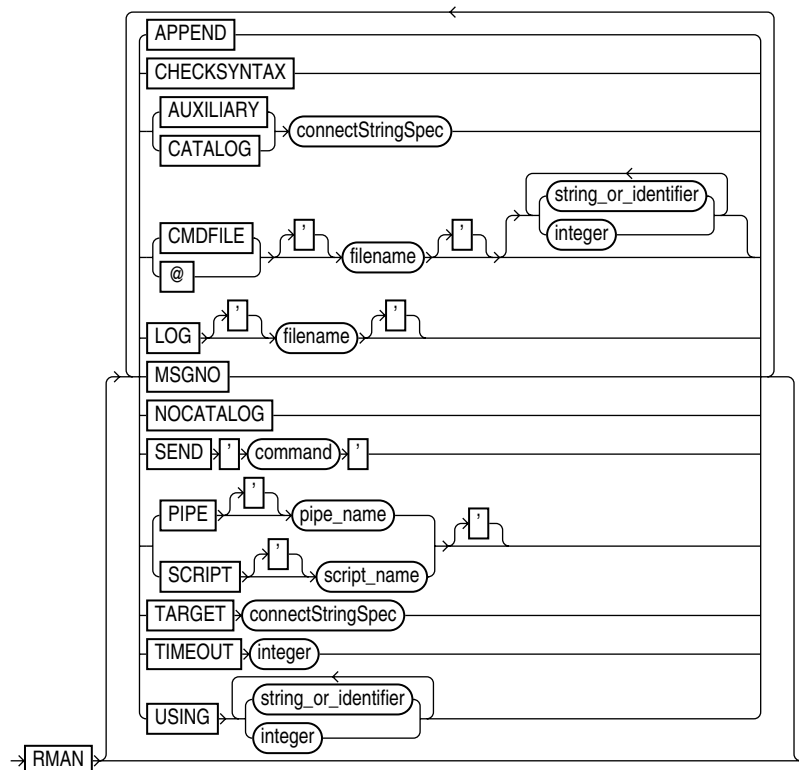
The command name that you enter at the operating system prompt is operating system-dependent. For example, enter `rman` in lowercase on Linux and UNIX systems.

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. If you maintain a recovery catalog, then the best practice is to connect to the recovery catalog before performing RMAN operations.

Caution: On some platforms, connecting to a database at the operating system command line means that credentials are visible to other users on the system. For example, on many UNIX systems the output of the `ps` command can show the complete command line used to start RMAN, including any credentials provided on the command line. The `CONNECT` command avoids this problem.

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 2-129). If used with a CMDFILE or @ argument, 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 <i>connectStringSpec</i>	Specifies a connect string to an auxiliary database, for example, AUXILIARY SYS/oracle@dupdb. See Also: connectStringSpec on page 3-12
CATALOG <i>connectStringSpec</i>	Specifies a connect string to the database containing the recovery catalog, for example, CATALOG rman/rman@inst2. See Also: SEND on page 2-241

Syntax Element	Description
<code>CMDFILE filename</code>	<p>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.</p> <p>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.</p> <p>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 <i>not</i> 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.</p>
<code>@filename</code>	Equivalent to <code>CMDFILE</code> .
<code>{string_or_identifier integer}</code>	Equivalent to options specified after <code>USING</code> syntax.
<code>LOG filename</code>	<p>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.</p> <p>RMAN output is also stored in the <code>V\$RMAN_OUTPUT</code> view, which is a memory-only view for jobs in progress, and in <code>V\$RMAN_STATUS</code>, which is a control file view for completed jobs and jobs in progress.</p> <p>The <code>LOG</code> parameter does not cause RMAN to terminate if the specified file cannot be opened. Instead, RMAN writes to standard output.</p> <p>Note: The easiest way to send RMAN output both to a log file and to standard output is to use the Linux <code>tee</code> command or its equivalent. For example:</p> <pre>% rman tee rman.log</pre>
<code>MSGNO</code>	Causes RMAN to print message numbers, that is, <code>RMAN-xxxx</code> , for the output of all commands. By default, RMAN does not print the <code>RMAN-xxxx</code> prefix.
<code>NOCATALOG</code>	Indicates that you are using RMAN without a recovery catalog.
<code>SEND 'command'</code>	<p>Sends a vendor-specific command string to all allocated channels.</p> <p>See Also: Your media management documentation to determine whether this feature is supported, and SEND on page 2-241</p>
<code>PIPE pipe_name</code>	<p>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 <code>PIPE</code> parameter. For example, you can invoke the RMAN pipe interface with the following options: <code>PIPE rpi TARGET SYS/pwd@tdb</code>.</p> <p>RMAN opens the following pipes in the target database:</p> <ul style="list-style-type: none"> ■ <code>ORA\$RMAN_RPI_IN</code>, which RMAN uses to receive user commands ■ <code>ORA\$RMAN_RPI_OUT</code>, which RMAN uses to send all output <p>All messages on both the input and output pipes are of type <code>VARCHAR2</code>.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to pass commands to RMAN through a pipe</p>

Syntax Element	Description
<code>SCRIPT <i>script_name</i></code>	<p>Specifies the name of a stored script.</p> <p>After connecting to the target database and recovery catalog (which must be specified with the <code>TARGET</code> and <code>CATALOG</code> 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 <i>script_name</i>, then RMAN runs the local script.</p> <p>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.</p> <p>See Also: CREATE SCRIPT on page 2-101 for more details about stored scripts</p>
<code>TARGET <i>connectStringSpec</i></code>	<p>Specifies a connect string to the target database, for example, <code>TARGET SYS/mypassword@inst1</code>.</p> <p>See Also: connectStringSpec on page 3-12</p>
<code>TIMEOUT <i>integer</i></code>	<p>Causes RMAN to exit automatically if it does not receive input from an input pipe within <i>integer</i> seconds. The <code>PIPE</code> parameter must be specified when using <code>TIMEOUT</code>.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to pass commands to RMAN through a pipe</p>
<code>USING {<i>string_or_identifier</i> <i>integer</i>}</code>	<p>Specifies one or more values for use in substitution variables in a command file. As in SQL*Plus, <code>&1</code> indicates where to place the first value, <code>&2</code> indicate where to place the second value, and so on. Example 2-128 on page 2-236 illustrates how to pass values specified in a <code>USING</code> clause to an RMAN command file.</p> <p>The substitution variable syntax is <code>&<i>integer</i></code> followed by an optional dot, for example, <code>&1.3</code>. 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 <code>mybackup</code> to a command file that contains the substitution variable <code>&1.3</code>, then the result of the substitution is <code>mybackup3</code>.</p> <p>See Also: EXECUTE SCRIPT on page 2-136 to learn how to specify the <code>USING</code> clause when executing a stored script</p>

Examples

Example 2-126 Connecting in Default NOCATALOG Mode

This example connects to the target database `prod` without specifying catalog options. Because `CONNECT CATALOG` is not run at the RMAN prompt, RMAN connects in default `NOCATALOG` mode when the first command requiring a repository connection is run.

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

Example 2-127 Connecting to an Auxiliary Instance

This example connects to target database `prod` and recovery catalog database `catdb` with net service names, and an auxiliary database instance with operating system authentication.

```
% rman TARGET SYS/password@prod
RMAN> CONNECT CATALOG rman/password@catdb
RMAN> CONNECT AUXILIARY /
```

Example 2–128 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 SYS/password@prod;
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 2–129 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 2–130 Running a Stored Script and Appending Output to a Message Log

This example connects to target database `prod1` and recovery catalog database `catdb` and then runs stored script `wdbb`. RMAN writes output to message log `/tmp/wdbb.log`.

```
% rman TARGET SYS/password@prod CATALOG rman/password@catdb SCRIPT wdbb LOG /tmp/wdbb.log
```

Example 2–131 Invoking the RMAN Pipe Interface

This example invokes the RMAN pipe `newpipe` with a 90 second timeout option.

```
% rman PIPE newpipe TARGET SYS/password@inst1 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 (`}`).

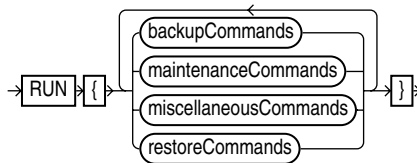
Usage Notes

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 2-132](#) on page 2-239). 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 2-133](#), you must use the `EXECUTE SCRIPT` command within a `RUN` block.

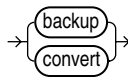
Syntax

run::=

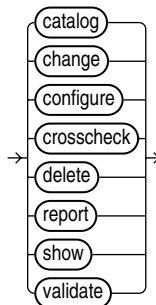


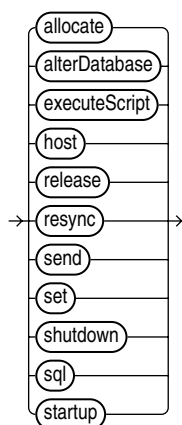
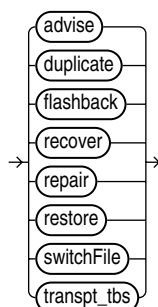
(*backupCommands::=* on page 2-238, *maintenanceCommands::=* on page 2-238, *miscellaneousCommands::=* on page 2-239, *restoreCommands::=* on page 2-239)

backupCommands::=



maintenanceCommands::=



miscellaneousCommands::=***restoreCommands::=*****Semantics**

Refer to individual command entries for information about commands that you can run from the RMAN prompt.

Examples***Example 2–132 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 2–133 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

```
RUN { EXECUTE SCRIPT backup_db; }
```

SEND

Purpose

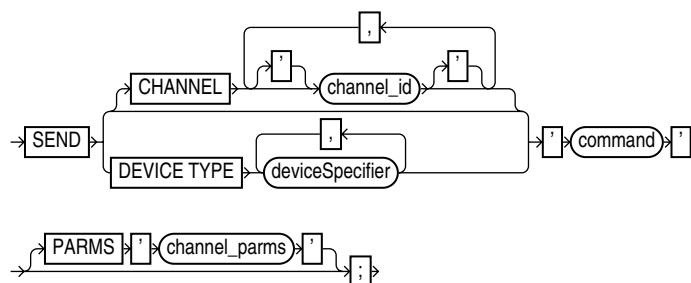
Use the `SEND` command to send a vendor-specific string to one or more channels supported by a media manager. Refer to your media management documentation to determine which commands are supported.

Usage Notes

Unless you specify `DEVICE TYPE` or `CHANNEL`, `RMAN` uses all allocated channels.

Syntax

send::=



(*deviceSpecifier::=* on page 3-15)

Semantics

Syntax Element	Description
<code>CHANNEL</code> <i>channel_id</i>	Specifies which channel to use. You must specify a case-sensitive channel ID, which is the name of the channel, after the <code>CHANNEL</code> keyword. The database uses the channel ID to report I/O errors.
<code>DEVICE TYPE</code> <i>deviceSpecifier</i>	Specifies the type of storage device and sends the command to all channels of the specified type. See Also: <i>deviceSpecifier</i> on page 3-15
<code>'command'</code>	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.
<code>PARMS</code> <code>'channel_parms'</code>	Specifies parameters for the channel communicating with the media manager.

Example

Example 2-134 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 stapel';  
  BACKUP TABLESPACE users;  
}
```

SET

Purpose

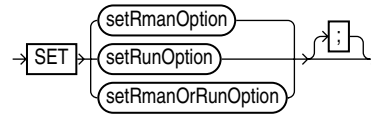
Use the SET command to control RMAN behavior within a job or session. Use [CONFIGURE](#) to configure options that persist across sessions.

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 2-244). 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 2-246).

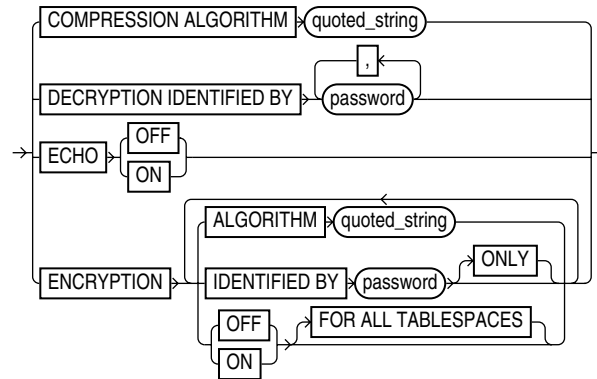
Syntax

set::=



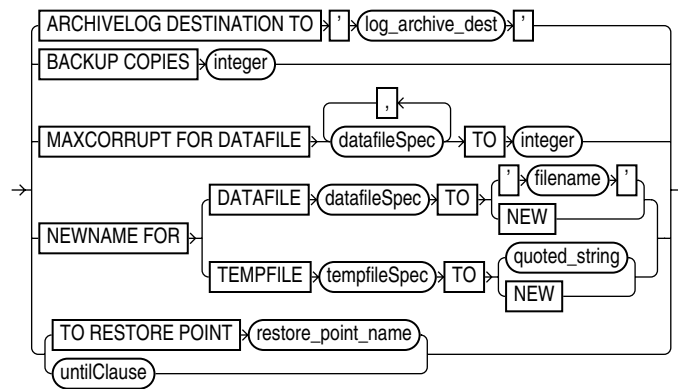
([setRmanOption::=](#) on page 2-243, [setRunOption::=](#) on page 2-244)

setRmanOption::=



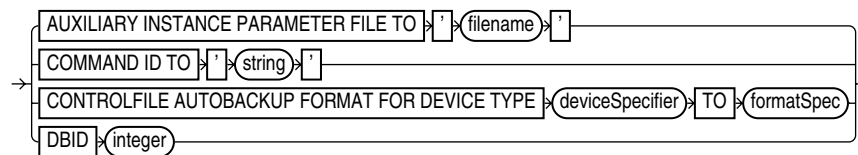
([deviceSpecifier::=](#) on page 3-15, [formatSpec::=](#) on page 3-22)

setRunOption::=



(*deviceSpecifier::=* on page 3-15, *formatSpec::=* on page 3-22, *datafileSpec::=* on page 3-14, *tempfileSpec::=* on page 3-38, *untilClause::=* on page 3-39)

setRmanOrRunOption::=



(*deviceSpecifier::=* on page 3-15, *formatSpec::=* on page 3-22)

Semantics

setRmanOption

This subclause specifies SET options that are usable outside of a RUN block.

Syntax Element	Description
COMPRESSION ALGORITHM 'algorithm_name'	<p>Specifies the compression algorithm for backup sets. This command overrides the current CONFIGURE COMPRESSION ALGORITHM setting for the current RMAN session. The default compression algorithm is BZIP2.</p> <p>Specify either ZLIB or BZIP2. By default, RMAN uses ZLIB compression, which is very fast but whose compression ratio is not as good as other algorithms. BZIP2 has a very good compression ratio, but is somewhat slow. In short, the ZLIB algorithm is faster than BZIP2, but this improvement in speed is obtained at the cost of a slightly worse compression ratio. The COMPATIBLE initialization parameter must be set to 11.0.0 or higher for ZLIB compression.</p> <p>Note: The view V\$RMAN_COMPRESSION_ALGORITHM describes supported algorithms.</p>

Syntax Element	Description
DECRYPTION IDENTIFIED BY <i>password</i>	<p>Specifies one or more decryption passwords to be used when reading dual-mode or password-encrypted backups.</p> <p>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.</p> <p>Note: If restoring from a group of backups created with different passwords, then specify all of the required passwords on the <code>SET DECRYPTION</code> command. RMAN will automatically use the correct password with each backup set.</p> <p>See Also: "Encryption of Backup Sets" on page 2-20 for more details on encryption of backup sets</p>
ECHO {OFF ON}	<p>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 <code>SET ECHO ON</code> command before running your command file. Run <code>SET ECHO OFF</code> to disable echoing to the command log.</p> <p>The command is useful when <code>stdin</code> and <code>stdout</code> have been redirected. For example, in UNIX you can redirect RMAN's input and output in this manner:</p> <pre>% rman TARGET sys/pwd@prod1 CATALOG rman/pwd@catdb < in_file > out_file</pre> <p>By including <code>SET ECHO ON</code> in the <code>in_file</code>, you enable the commands contained in <code>in_file</code> to be visible in <code>out_file</code>.</p>
ENCRYPTION	<p>Specifies encryption-related options that apply to <code>BACKUP</code> commands that create backup sets for the duration of the RMAN session.</p> <p>See Also: "Encryption of Backup Sets" on page 2-20 for more details on encryption of backup sets</p>
ALGORITHM ' <i>algorithm_name</i> '	<p>Specifies the algorithm used during this RMAN session. Overrides the configured default encryption algorithm specified by <code>CONFIGURE ALGORITHM</code>. Possible values are listed in <code>V\$RMAN_ENCRYPTION_ALGORITHMS</code>.</p>
IDENTIFIED BY <i>password</i> [ONLY]	<p>Specifies whether to employ a user-specified password in encryption according to the following rules:</p> <ul style="list-style-type: none"> ■ Omit <code>IDENTIFIED BY password</code> clause to specify transparent-mode encrypted backups. ■ Use <code>IDENTIFIED BY password ONLY</code> to specify password-mode encrypted backups. ■ Use <code>IDENTIFIED BY password</code> without <code>ONLY</code> to create dual-mode encrypted backups. <p>See Also: "Encryption of Backup Sets" on page 2-20 for details on the different encryption modes</p> <p>Note: If the password is not surrounded by quotes, then it is translated internally into upper case. Thus, the following clauses are all synonyms for <code>IDENTIFIED BY "SOMEPWD"</code>:</p> <ul style="list-style-type: none"> ■ <code>IDENTIFIED BY somepwd</code> ■ <code>IDENTIFIED BY Somepwd</code> ■ <code>IDENTIFIED BY sOmEpWd</code>

Syntax Element	Description
{OFF ON}	Specifies whether to encrypt backup sets. If ON, then the default is to encrypt backup sets. If OFF, then the default is to not 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 logs 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 logs to different locations while restoring a database. RMAN knows where to find the newly restored archived logs; 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 logs that are not already on disk. RMAN always looks for logs on disk first before restoring them from backups.

Syntax Element	Description
<code>BACKUP COPIES <i>integer</i></code>	<p>Specifies the number of copies of each backup piece that the channels should create: 1, 2, 3, or 4 (see Example 2-136 on page 2-250).</p> <p>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 <code>BACKUP COPIES</code> is greater than 1, then the <code>BACKUP_TAPE_IO_SLAVES</code> initialization parameter must be enabled on the target database.</p> <p>The <code>SET BACKUP COPIES</code> command affects all <code>BACKUP</code> commands in the <code>RUN</code> block issued after <code>SET BACKUP COPIES</code> (but not before) and is in effect until explicitly disabled or changed. The <code>SET BACKUP COPIES</code> command affects only <code>BACKUP</code> commands, but does not apply to the <code>BACKUP AS COPY</code> command.</p> <p>The <code>SET BACKUP COPIES</code> 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:</p> <ol style="list-style-type: none"> 1. <code>BACKUP COPIES</code> 2. <code>SET BACKUP COPIES</code> 3. <code>CONFIGURE . . . BACKUP COPIES</code> <p>The names of the backup pieces are dependent on the <code>FORMAT</code> clause in the <code>BACKUP</code> command. You can specify up to four <code>FORMAT</code> strings. RMAN uses the second, third, and fourth values only when <code>BACKUP COPIES</code>, <code>SET BACKUP COPIES</code>, or <code>CONFIGURE . . . BACKUP COPIES</code> 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.</p> <p>Note: <code>BACKUP COPIES</code> option is not valid when files are created in flash recovery area. Backups to the flash recovery area cannot be duplexed.</p> <p>Note: Control file autobackups on disk are a special case and are never duplexed: RMAN always writes one and only copy.</p>
<code>MAXCORRUPT FOR DATAFILE <i>datafileSpec</i> TO <i>integer</i></code>	<p>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.</p> <p>The <code>SET MAXCORRUPT</code> command specifies the total number of physical and logical corruptions permitted in a datafile during a backup or restore job. If the sum of physical and logical corruptions detected for a datafile is no more than its <code>MAXCORRUPT</code> setting, then the <code>BACKUP</code> or <code>RESTORE</code> command completes. If more than <code>MAXCORRUPT</code> corrupt blocks exist, then RMAN terminates without creating output files.</p> <p>Whether or not the <code>MAXCORRUPT</code> limit is exceeded, RMAN populates the <code>V\$DATABASE_BLOCK_CORRUPTION</code> view with any corrupt block ranges that it finds. However, a backup or restore job is terminated after <code>MAXCORRUPT+1</code> corrupt blocks are found, so in this case RMAN only records <code>MAXCORRUPT+1</code> corruptions. Any block corruptions beyond the point at which the backup or restore job terminated are not recorded.</p> <p>Note: If you specify <code>CHECK LOGICAL</code>, then the <code>MAXCORRUPT</code> limit applies to the sum of logical and physical corruptions detected. Otherwise, <code>MAXCORRUPT</code> only applies to the number of physical block corruptions.</p> <p>See Also: datafileSpec on page 3-14</p>

Syntax Element	Description
NEWNAME FOR DATAFILE <i>datafileSpec</i>	<p>Sets the default name for all subsequent RESTORE or SWITCH commands that affect the specified datafile. If you do not issue this command before the datafile restore operation, then RMAN restores the file to its default location.</p> <p>After you 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 the restored file is recorded as a datafile copy in the repository.</p> <p>You cannot use <code>SET NEWNAME TO NEW</code> when creating a duplicate or standby database or performing RMAN TSPITR.</p> <p>Note: The <code>SET NEWNAME</code> command supports ASM disk groups.</p> <p>See Also: <i>datafileSpec</i> on page 3-14</p>
NEWNAME FOR TEMPFILE <i>tempfileSpec</i>	<p>Sets the new tempfile name for a subsequent SWITCH command that renames the specified tempfile to the specified name.</p> <p>You cannot use <code>SET NEWNAME TO NEW</code> when creating a duplicate or standby database or performing RMAN TSPITR.</p> <p>Note: The <code>SET NEWNAME</code> command supports ASM disk groups.</p> <p>See Also: <i>tempfileSpec</i> on page 3-38</p>
TO 'filename'	<p>Specifies a user-defined filename or Automatic Storage Management disk group for the restored datafile or tempfile. If you set the NEWNAME to a disk group for a datafile and run a RESTORE, then RMAN restores the file to the disk group. If you specify a filename for a tempfile, then this will be the new name of the tempfile after the database is recovered and opened.</p>
TO NEW	<p>Creates an Oracle-managed file in the directory specified in <code>DB_CREATE_FILE_DEST</code>. If the original file is an Oracle-managed file or is on an Automatic Storage Management disk group, then RMAN attempts to delete the original file. If you specify <code>TO NEW</code> for a tempfile, then the tempfile is created in <code>DB_CREATE_FILE_DEST</code> when the database is opened.</p> <p>You cannot use this option when using the DUPLICATE command or performing RMAN TSPITR.</p> <p>See Also: <i>Oracle Database Administrator's Guide</i> for information about Oracle-managed files</p>
TO RESTORE POINT <i>restore_point_name</i>	<p>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 <i>and including</i> the SCN corresponding to the restore point.</p> <p>Note: You can only use <code>SET TO RESTORE POINT</code> when the database is mounted because the defined restore points are recorded in the control file. For example, you cannot use <code>SET TO RESTORE POINT</code> to specify the target SCN for a RESTORE CONTROLFILE operation.</p>
<i>untilClause</i>	<p>Specifies an end time, SCN, or log sequence number for a subsequent RESTORE or RECOVER command.</p> <p>See Also: <i>untilClause</i> on page 3-39</p>

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'	<p>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.</p> <p>Note: The <i>filename</i> is on the host running the RMAN client.</p> <p>See Also: <i>Oracle Database Reference</i> for more on <code>V\$SESSION.CLIENT_INFO</code></p>
COMMAND ID TO 'string'	<p>Enters the specified string into the <code>V\$SESSION.CLIENT_INFO</code> column of all channels. Use this information to determine which database server sessions correspond to which RMAN channels. The <code>SET COMMAND ID</code> command applies only to channels that are already allocated.</p> <p>The <code>V\$SESSION.CLIENT_INFO</code> column contains information for each RMAN server session. The data appears in one of the following formats:</p> <ul style="list-style-type: none"> ■ <code>id=string</code> ■ <code>id=string,ch=channel_id</code> <p>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 <code>V\$SESSION.CLIENT_INFO</code> column will be cleared.</p> <p>See Also: <i>Oracle Database Reference</i> for more on <code>V\$SESSION.CLIENT_INFO</code></p>
CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE <i>deviceSpecifier</i> TO <i>formatSpec</i>	<p>Overrides the default filename format for the control file autobackup on the specified device type. You can use this command either in <code>RUN</code> or at the RMAN prompt. The order of precedence is as follows:</p> <ol style="list-style-type: none"> 1. <code>SET CONTROLFILE AUTOBACKUP</code> executed within a <code>RUN</code> block 2. <code>SET CONTROLFILE AUTOBACKUP</code> executed at the RMAN prompt 3. <code>CONFIGURE CONTROLFILE AUTOBACKUP FORMAT</code> <p>The <code>%F</code> substitution variable is required to be in the new <i>formatSpec</i>. No other substitution variable is legal in a control file autobackup <i>formatSpec</i>.</p> <p>See Also: <i>formatSpec</i> on page 3-22 for the semantics of the <code>%F</code> substitution variable</p>
DBID <i>integer</i>	<p>Specifies the DBID, which is a unique 32-bit identification number computed when the database is created.</p> <p>RMAN displays the DBID upon connection to the target database. You can obtain the DBID by querying the <code>V\$DATABASE</code> view or the <code>RC_DATABASE</code> and <code>RC_DATABASE_INCARNATION</code> recovery catalog views.</p> <p>You should only run the <code>SET DBID</code> command in the following specialized circumstances:</p> <ul style="list-style-type: none"> ■ You are not connected to a recovery catalog and want to restore the control file (see Example 2-137 on page 2-250). The same restriction applies when you use the Data Recovery Advisor to restore a control file autobackup. <code>CONFIGURE</code> can locate an autobackup and restore it only if <code>SET DBID</code> is issued before <code>ADVISE FAILURE</code>. ■ You want to restore the server parameter file (see Example 2-138). ■ You are connected to the recovery catalog but not the target database and use the <code>FOR DB_UNIQUE_NAME</code> option on the <code>CONFIGURE</code>, <code>LIST</code>, <code>REPORT</code>, <code>SHOW</code>, or <code>UNREGISTER</code> commands.

Examples

Example 2-135 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 2–136 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 2–137 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 2–138 Restoring the Server Parameter File

Assume that the database is closed while maintenance is being performed on the database host. During this time, the server parameter file is accidentally deleted. This example restores a server parameter file from an autobackup on tape and then restarts the instance.

```

CONNECT TARGET /
CONNECT CATALOG rman/password@catdb
SET DBID 3257174182; # set dbid so RMAN knows the database name
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; # needed so that RMAN restarts database with restored server parameter file
```

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`.

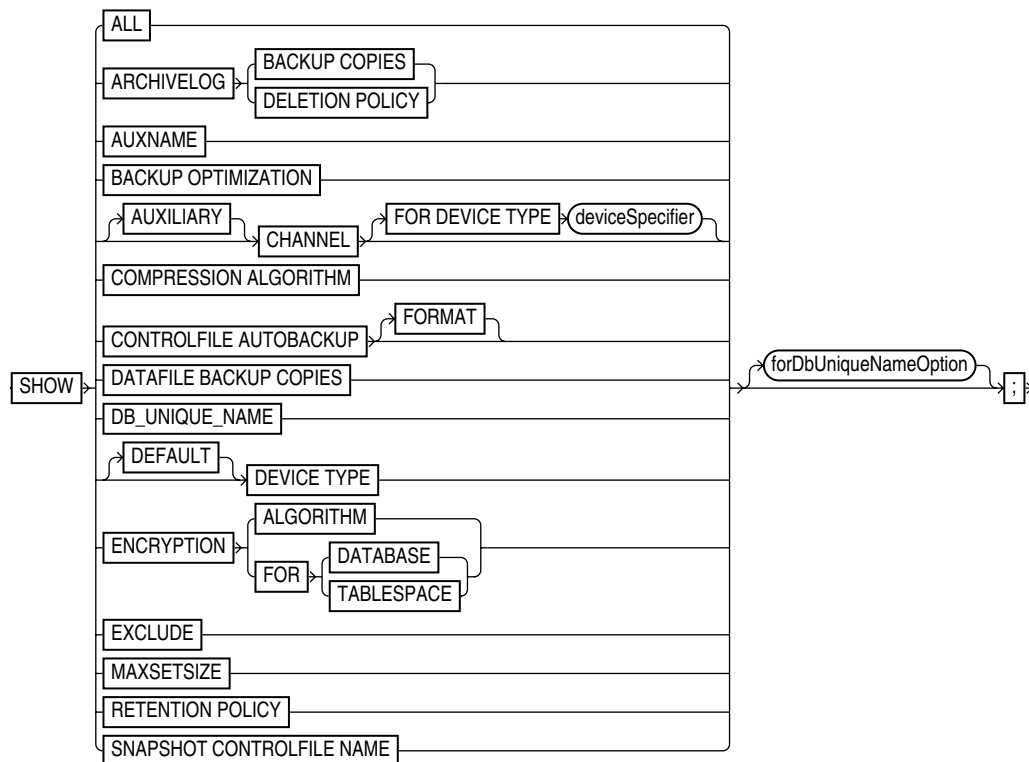
Prerequisites

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.

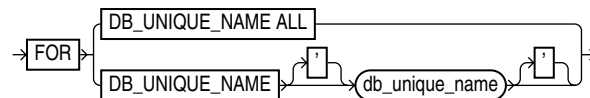
Syntax

show::=



(*deviceSpecifier::=* on page 3-15)

forDbUniqueNameOption::=



Semantics

Syntax Element	Description
ALL	Shows all user-entered <code>CONFIGURE</code> 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 currently configured archived redo log deletion policy.
AUXNAME	Shows the <code>CONFIGURE AUXNAME</code> settings.
BACKUP OPTIMIZATION	Shows the <code>CONFIGURE BACKUP OPTIMIZATION</code> settings: ON or OFF (default).
[AUXILIARY] CHANNEL FOR DEVICE TYPE <i>deviceSpecifier</i>	Shows the <code>CONFIGURE CHANNEL</code> settings. You can specify a normal channel or an AUXILIARY channel. Specifies the device type of the channel. For example, <code>SHOW CHANNEL FOR DEVICE TYPE DISK</code> Shows only channel settings for disk channels.
COMPRESSION ALGORITHM	Shows the configured backup compression algorithm.
CONTROLFILE AUTOBACKUP FORMAT	Shows the <code>CONFIGURE CONTROLFILE AUTOBACKUP</code> settings: ON or OFF. Shows the format for the control file autobackup file for configured devices.
DATAFILE BACKUP COPIES	Shows the <code>CONFIGURE . . . BACKUP COPIES</code> 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 ALGORITHM FOR DATABASE FOR TABLESPACE	Shows currently configured encryption settings for the database or tablespaces within the database, when used with ALGORITHM or FOR {DATABASE TABLESPACE}. Shows the configured default algorithm to use for encryption when writing encrypted backup sets. Possible values are listed in V\$RMAN_ENCRYPTION_ALGORITHMS. Shows current encryption settings for the database. Shows current encryption settings for each tablespace.
EXCLUDE	Shows only the tablespaces that you have specified should be excluded.
MAXSETSIZE	Shows the <code>CONFIGURE MAXSETSIZE</code> settings.
RETENTION POLICY	Shows the settings for <code>CONFIGURE RETENTION POLICY</code> for the current target database.
SNAPSHOT CONTROLFILE NAME	Shows the <code>CONFIGURE SNAPSHOT CONTROLFILE</code> settings.

Syntax Element	Description
<i>forDbUniqueNameOption</i>	<p>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 <i>db_unique_name</i> or use ALL for all uniquely named databases.</p> <p>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.</p> <p>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 2-46 on page 2-81).</p> <p>See Also: <i>forDbUniqueNameOption</i> on page 3-19 for descriptions of the options in this clause</p>

Examples

Example 2-139 Showing All Configurations for the Target Database

This example shows all persistent configurations for the target database `prod1` (and includes sample output):

```

RMAN> CONNECT TARGET SYS/password@prod
RMAN> CONNECT CATALOG rman/password@catdb
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'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DEVICE TYPE SBT_TAPE PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE 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

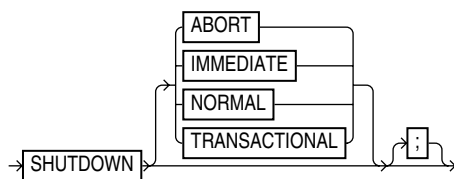
Usage Notes

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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none"> ■ 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: <ul style="list-style-type: none">■ 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 shutdown of the target database while minimizing interruption to clients, with the following consequences: <ul style="list-style-type: none">■ 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 2–140 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 2–141 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 2-232 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 <i>filename</i>	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 2-142 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.

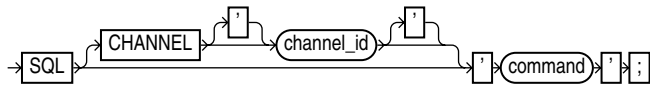
See Also: *Oracle Database SQL Language Reference*

Prerequisites

None.

Syntax

sql::=



Semantics

Syntax Element	Description
CHANNEL <i>channel_id</i>	<p>Specifies the case-sensitive name of a channel to use when executing an RMAN command within a RUN command.</p> <p>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.</p>
' <i>command</i> '	<p>Specifies a SQL statement for execution (see Example 2-143 on page 2-258). SELECT statements are not permitted.</p> <p>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 <i>single</i> quotes and the entire string following the SQL keyword must be enclosed in <i>double</i> quotes (see Example 2-144 on page 2-258).</p> <p>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 <code>rman.rman_purge</code> with the SQL command, issue the following command:</p> <pre>SQL 'BEGIN rman.rman_purge; END;';</pre>

Examples

Example 2-143 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 2-144 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

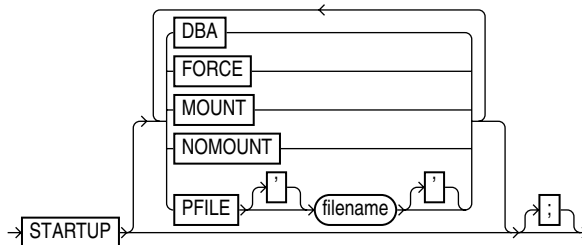
You must be connected to a target database. You can only use this command to start the target database.

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 2-146](#) on page 2-261).

Syntax

startup::=



Semantics

Syntax Element	Description
STARTUP	If you specify only <code>STARTUP</code> 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 <code>RESTRICTED SESSION</code> privilege.
FORCE	If the database is open, then <code>FORCE</code> shuts down the database with a <code>SHUTDOWN ABORT</code> statement before re-opening it. If the database is closed, then <code>FORCE</code> 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 <code>RESTORE SPFILE</code> to restore a backup server parameter file.

Syntax Element	Description
PFILE <i>filename</i>	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 2–145 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 SYS/password@prod
STARTUP FORCE MOUNT DBA PFILE=/tmp/initPROD.ora;
```

Example 2–146 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 flash recovery area, so SET DBID is not necessary.

```
CONNECT TARGET SYS/password@prod
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 with 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

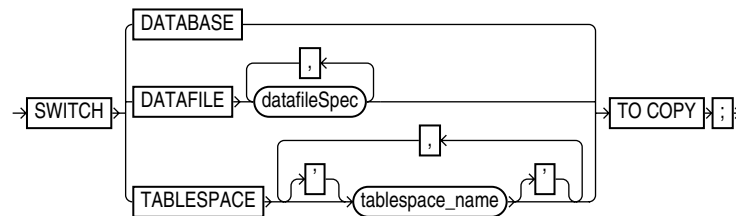
The `SWITCH` command deletes the RMAN repository records for the datafile copy from the recovery catalog and updates the control file records to status `DELETED`.

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.

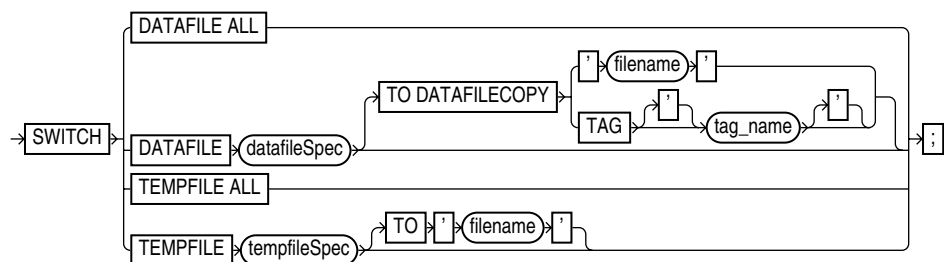
Execute `SWITCH . . . TO COPY` only at the RMAN prompt. Use `SWITCH` without `TO COPY` only within a `RUN` block.

Syntax

`switch::=`



(*datafileSpec::=* on page 3-14)

switchFile::=

(*datafileSpec*::= on page 3-14, *tempfileSpec*::= on page 3-38)

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 <i>datafileSpec</i>	Switches the specified datafiles to the latest image copies. After the switch, the control file no longer considers the specified datafile as current.
TABLESPACE <i>tablespace_name</i>	Switches all datafiles within the specified tablespace, as with <code>SWITCH DATAFILE . . . TO COPY</code> (see Example 2-147 on page 2-264).
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 <code>SET NEWNAME FOR DATAFILE</code> command has been issued in this job are switched to their new name (see Example 2-148 on page 2-264).
DATAFILE <i>datafileSpec</i>	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 <code>TO</code> option, then RMAN uses the filename specified on a prior <code>SET NEWNAME</code> command in the <code>RUN</code> block for this file as the switch target.
TO DATAFILECOPY { 'filename' TAG <i>tag_name</i> }	Specifies the input copy file for the switch, that is, the datafile copy that you intend to rename (see Example 2-150 on page 2-265).
TEMPFILE ALL	Specifies that all tempfiles for which a <code>SET NEWNAME FOR TEMPFILE</code> command has been issued in this job are switched to their new name.

Syntax Element	Description
TEMPFILE <i>tempfileSpec</i>	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 ' <i>filename</i> '	Renames the tempfile to the specified name (see Example 2-149 on page 2-264). The target database must be mounted but not open.

Examples

Example 2-147 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 flash recovery area. You can run `SWITCH` to point to the control file to the new datafiles and then run [RECOVER](#).

```
CONNECT TARGET SYS/password@prod
SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
SWITCH TABLESPACE users TO COPY;
RECOVER TABLESPACE users;
SQL "ALTER TABLESPACE users ONLINE";
```

Example 2-148 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. 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.

```
CONNECT TARGET SYS/password@prod
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 2-149 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 2-150 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.

`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 logs 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.

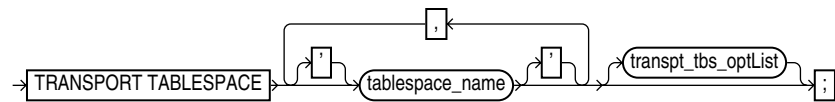
Usage Notes

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 logs are available on disk. Use the `CATALOG` command to record the datafile copies and archived logs 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`.

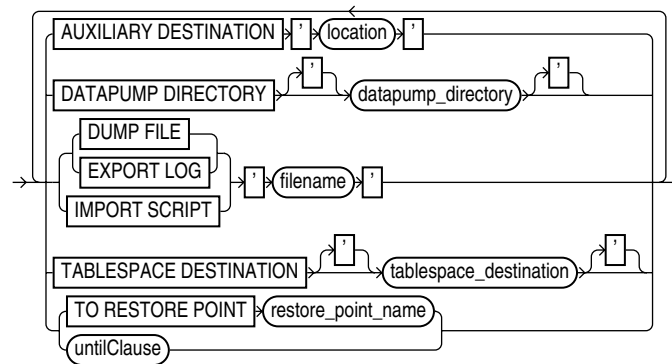
Syntax

transpt_tbs::=



(*transpt_tbs_optlist::=* on page 2-267)

transpt_tbs_optlist::=



(*untilClause::=* on page 3-39)

Semantics

transpt_tbs

Syntax Element	Description
<i>tablespace_name</i>	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 <code>TRANSPORT TABLESPACE</code> operation.

transpt_tbs_optlist

This subclause specifies optional parameters that affect the tablespace transport.

Syntax Element	Description
AUXILIARY DESTINATION <i>'location'</i>	<p>Specifies the location for files for the auxiliary instance.</p> <p>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.</p> <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> for details on the interactions among the different techniques for naming the auxiliary instance files</p>
DATAPUMP DIRECTORY <i>datapump_directory</i>	<p>Specifies a database directory object where Data Pump Export outputs are created (see Example 2-152). If not specified, then RMAN creates files in the location specified by TABLESPACE DESTINATION.</p> <p>See Also: <i>Oracle Database Utilities</i> for more details on Data Pump Export and database directory objects</p>
DUMP FILE <i>'filename'</i>	<p>Specifies where to create the Data Pump Export dump file. If not specified, the export dump file is named <code>dmpfile.dmp</code> and stored in the location specified by the DATAPUMP DIRECTORY clause or in the tablespace destination.</p> <p>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.</p>
EXPORT LOG <i>'filename'</i>	<p>Specifies the location of the log generated by Data Pump Export. If omitted, the export log is named <code>explog.log</code> and stored in the location specified by the DATAPUMP DIRECTORY clause or in the tablespace destination.</p>
IMPORT SCRIPT <i>'filename'</i>	<p>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 <code>impscript.sql</code>. The script is stored in the tablespace destination.</p>
TABLESPACE DESTINATION <i>tablespace_destination</i>	<p>Specifies the location of the datafiles for the transported tablespaces after the tablespace transport operation completes.</p>
TO RESTORE POINT <i>restore_point_name</i>	<p>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 <i>and including</i> the SCN corresponding to the restore point.</p>
<i>untilClause</i>	<p>Specifies a past time, SCN, or log sequence number (see Example 2-151 on page 2-269). If specified, RMAN restores and recovers the tablespaces at the auxiliary instance to their contents at that past point in time before export.</p> <p>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.</p> <p>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.</p>

Examples

Example 2-151 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 2-152 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.

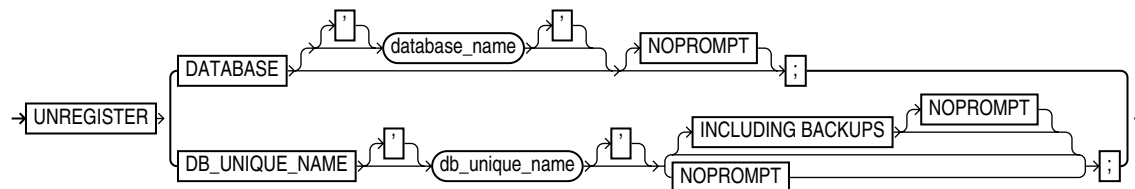
See Also: [DROP DATABASE](#) on page 2-118 to learn how to delete a database and unregister it with one command

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	<p>Specifies a primary database to be unregistered. RMAN unregisters the primary database as well as any of its associated standby databases (see Example 2-153 on page 2-272).</p> <p>If <i>database_name</i> is not specified, then RMAN must identify the database in one of the following ways:</p> <ul style="list-style-type: none"> ■ If RMAN is connected to a database with <code>TARGET</code>, then RMAN unregisters the target database and any associated standby databases. ■ If RMAN is not connected to a database with <code>TARGET</code>, or if the name of the <code>TARGET</code> database is not unique in the recovery catalog, then RMAN uses the value specified in the <code>SET DBID</code> command (see Example 2-154 on page 2-272). RMAN unregisters all databases with this DBID.
<i>database_name</i>	<p>Specifies the name of the primary database that you are unregistering.</p> <p>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 <code>SET DBID</code> command.</p>

Syntax Element	Description
DB_UNIQUE_NAME <i>db_unique_name</i>	<p>Specifies the removal of all metadata except backup metadata for the Data Guard database specified by <i>db_unique_name</i>.</p> <p>The specified database can be either a primary or standby database, although typically you use this clause to unregister a standby database (see Example 2-155 on page 2-273).</p> <p>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 <i>prod1</i> and use DB_UNIQUE_NAME to unregister <i>standby1</i>. Note that the DBID used in SET DBID is the same for the primary database and its associated standby databases.</p> <p>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.</p>
INCLUDING BACKUPS	<p>Specifies that backup metadata should also be removed for the database specified by <i>db_unique_name</i>.</p> <p>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.</p>
NOPROMPT	Does not prompt for confirmation before unregistering the database.

Example

Example 2-153 Unregistering a Primary Database and Its Standby Databases

Assume that primary database *rdbms* has associated standby databases *dgrdbms3* and *dgrdbms4*. In this example, you connect to the target database *prod*, whose database name is unique in the recovery catalog, and unregister it. RMAN removes all metadata about *prod*, *dgrdbms3*, and *dgrdbms4* from the catalog. Sample output is included.

```

RMAN> CONNECT TARGET SYS/password@inst1

connected to target database: RDBMS (DBID=1627709917)

RMAN> CONNECT CATALOG rman/password@inst2

connected to recovery catalog database

RMAN> UNREGISTER DATABASE NOPROMPT;

database name is "RDBMS" and DBID is 1627709917
database unregistered from the recovery catalog

RMAN> LIST DB_UNIQUE_NAME ALL;

RMAN>

```

Example 2-154 Unregistering a Database That is Not Unique in Catalog

Assume that two databases registered in a recovery catalog have the name *prod*. You want to unregister the *prod* database that has the DBID of 28014364. Because multiple databases called *prod* are registered in the recovery catalog, and because RMAN is not connected as TARGET to the *prod* 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 rcat/password@inst1

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 2-155 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/password@catdb

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 10gR1 version of the recovery catalog schema requires the `CREATE TYPE` privilege. If you created the recovery catalog owner in a release before 10gR1, 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 twice in a row 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::=

`UPGRADE CATALOG` 

Semantics

None.

Example

Example 2–156 Upgrading a Recovery Catalog

This example connects to recovery catalog database `catdb` at the operating system command line and then upgrades it to a more current version:

```

RMAN> CONNECT CATALOG rco/password@catdb

connected to recovery catalog database
PL/SQL package rman.DBMS_RVCAT version 10.01.00 in RVCAT 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_RVCAT 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.

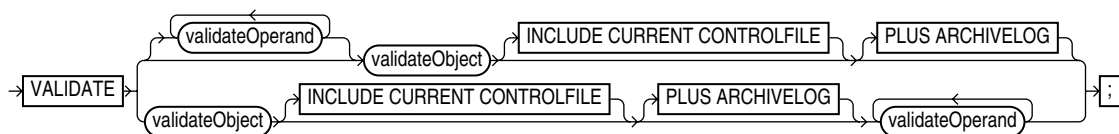
The `VALIDATE` command skips never-used blocks. RMAN also skips currently unused (as opposed to never used) blocks for locally managed tablespaces when the `COMPATIBLE` parameter is set to 10.2 or greater.

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.

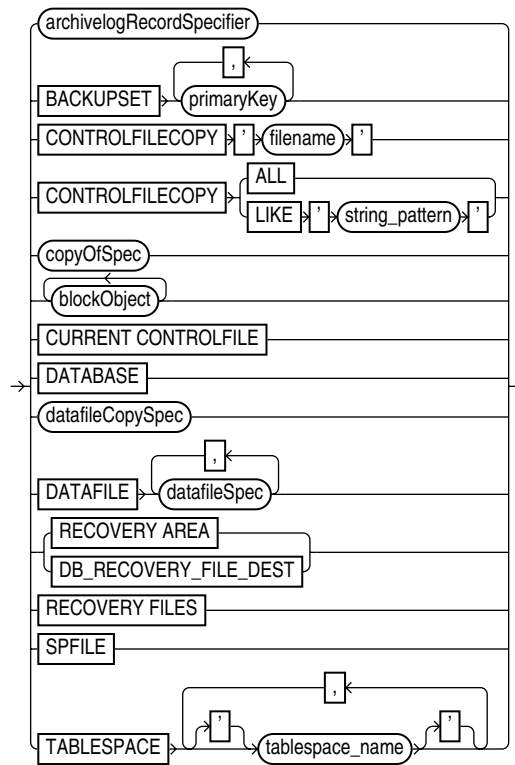
Syntax

`validate::=`



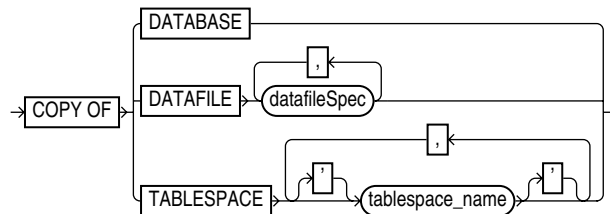
(*validateObject::=* on page 2-277, *validateOperand::=* on page 2-278)

validateObject::=



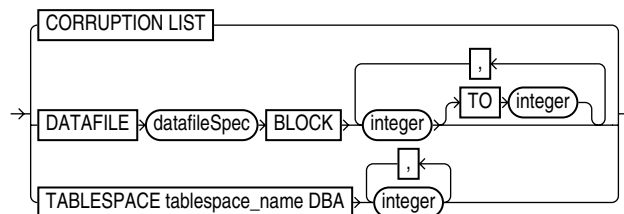
(*archivelogRecordSpecifier::=* on page 3-6, *copyOfSpec::=* on page 2-25, *blockObject::=* on page 2-180, *datafileCopySpec::=* on page 2-26, *datafileSpec::=* on page 3-14)

copyOfSpec::=



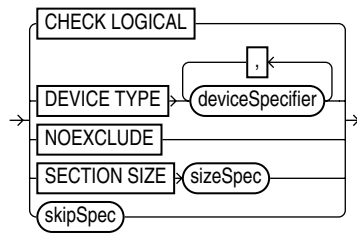
(*datafileSpec::=* on page 3-14)

blockObject::=



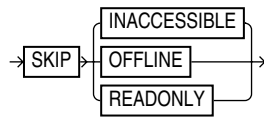
(*datafileSpec::=* on page 3-14)

validateOperand::=

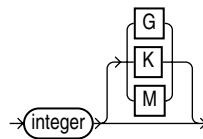


(*deviceSpecifier::=* on page 3-15, *sizeSpec::=* on page 2-278, *skipSpec::=* on page 2-278)

skipSpec::=



sizeSpec::=



Semantics

validate

This subclause specifies backup sets for validation. Refer to *validate::=* on page 2-276 for syntax.

Syntax Element	Description
<i>validateOperand</i>	Specifies options that control the validation. See Also: <i>validateOperand</i> on page 2-280
<i>validateObject</i>	Specifies the files to be validated. See Also: <i>validateObject</i> on page 2-279
INCLUDE CURRENT CONTROLFILE	Creates a snapshot of the current control file and validates it.
PLUS ARCHIVELOG	Includes archived redo logs in the validation. Causes RMAN to perform the following steps: <ol style="list-style-type: none"> 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 logs.

validateObject

This subclause specifies database files for validation. Refer to *validateObject::=* on page 2-277 for syntax.

Syntax Element	Description
<i>archiveLogRecordSpecifier</i>	Validates a range of archived redo logs. <code>VALIDATE ARCHIVELOG</code> is equivalent to <code>BACKUP VALIDATE ARCHIVELOG</code> .
<code>BACKUPSET primary_key</code>	<p>Checks that the backup sets specified by <i>primary_key</i> exist and can be restored.</p> <p>You can obtain the primary keys of backup sets by executing a <code>LIST</code> statement or, if you use a recovery catalog, by querying the <code>RC_BACKUP_SET</code> recovery catalog view.</p> <p>The <code>VALIDATE BACKUPSET</code> 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 <code>CROSSCHECK</code> command examines the headers of the specified files if they are on disk or queries the media management catalog if they are on tape.</p> <p>Use <code>VALIDATE BACKUPSET</code> when you suspect that one or more backup pieces in a backup set are missing or have been damaged. Note that <code>VALIDATE BACKUPSET</code> selects which backups to test, whereas the <code>VALIDATE</code> option of the <code>RESTORE</code> command lets RMAN choose which backups to validate. For validating image copies, run <code>RESTORE VALIDATE FROM DATAFILECOPY</code>.</p> <p>If you do not have automatic channels configured, then manually allocate at least one channel before executing <code>VALIDATE BACKUPSET</code>.</p> <p>Note: If multiple copies of a backup set exist, then RMAN validates only the most recent copy. The <code>VALIDATE</code> 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 <code>VALIDATE DEVICE TYPE</code> to validate the copy on the specified device. If both copies exist on the same device, then you can use <code>CHANGE</code> to make one copy temporarily <code>UNAVAILABLE</code> and then reissue <code>VALIDATE</code>.</p>
<code>CONTROLFILECOPY {'filename' ALL LIKE 'string_pattern'}</code>	<p>Validates control file copies. You can specify a control file copy in one of the following ways:</p> <ul style="list-style-type: none"> ▪ <code>'filename'</code> specifies a control file copy by filename ▪ <code>ALL</code> specifies that all control file copies should be backed up ▪ <code>LIKE 'pattern'</code> specifies a filename pattern. The percent sign (%) as a wildcard meaning 0 or more characters; an underscore (_) is a wildcard meaning 1 character. <p>The control file copy can be created with the <code>BACKUP AS COPY CURRENT CONTROLFILE</code> command or the SQL statement <code>ALTER DATABASE BACKUP CONTROLFILE TO '...'</code>.</p>
<i>copyOfSpec</i>	<p>Validates image copies of datafiles and control files.</p> <p>See Also: <i>copyOfSpec</i> on page 2-42 for details.</p>
<i>blockObject</i>	<p>Validates individual data blocks.</p> <p>See Also: <i>blockObject</i> on page 2-184</p>
<code>CURRENT CONTROLFILE</code>	Validates the current control file.
<code>DATABASE</code>	<p>Validates the database.</p> <p>RMAN validates all datafiles and control files. If the database is currently using a server parameter file, then RMAN validates the server parameter file.</p> <p>Note: The online redo log files and tempfiles are not validated.</p>

Syntax Element	Description
<i>datafileCopySpec</i>	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 <code>VALIDATE BACKUPSET</code> , RMAN proceeds and reports the number of blocks that are corrupted. See Also: <i>datafileCopySpec</i> on page 2-43 for details
DATAFILE <i>datafileSpec</i>	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: <i>datafileSpec</i> on page 3-14
RECOVERY AREA	Validates recovery files created in the current and all previous flash recovery area destinations. Recovery files are full and incremental backup sets, control file autobackups, archived redo logs, and datafile copies. Flashback logs, the current control file, and online redo logs are not validated.
DB_RECOVERY_FILE_DEST	Semantically equivalent to <code>RECOVERY AREA</code> .
RECOVERY FILES	Validates <i>all</i> recovery files on disk, whether they are stored in the flash 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 validate 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 <i>tablespace_name</i>	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 2-278 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 <code>V\$DATABASE_BLOCK_CORRUPTION</code> is populated with corrupt block ranges. Note: <code>VALIDATE</code> does not use <code>MAXCORRUPT</code> .
DEVICE TYPE <i>deviceSpecifier</i>	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 <code>VALIDATE . . . DEVICE TYPE DISK</code> , RMAN allocates only disk channels. See Also: <i>deviceSpecifier</i> on page 3-15
NOEXCLUDE	When specified on a <code>VALIDATE DATABASE</code> or <code>VALIDATE COPY OF DATABASE</code> command, RMAN validates all tablespaces, including any for which a <code>CONFIGURE EXCLUDE</code> command has been entered. This option does not override <code>SKIP OFFLINE</code> or <code>SKIP READONLY</code> .

Syntax Element	Description
SECTION SIZE <i>sizeSpec</i>	<p>Parallelizes the validation by dividing each file into the specified section size.</p> <p>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.</p> <p>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.</p> <p>See Also: BACKUP SECTION SIZE to learn how to make multisection backups</p>
<i>skipSpec</i>	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 logs from the validation if they are inaccessible, offline, or read-only.
INACCESSIBLE	<p>Specifies that datafiles and archived redo logs that cannot be read due to I/O errors should be excluded from the validation.</p> <p>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.</p>
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 2–38 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 2–39 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 2–40 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 2–157 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_TAG20070308T092426_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 2-158 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

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.

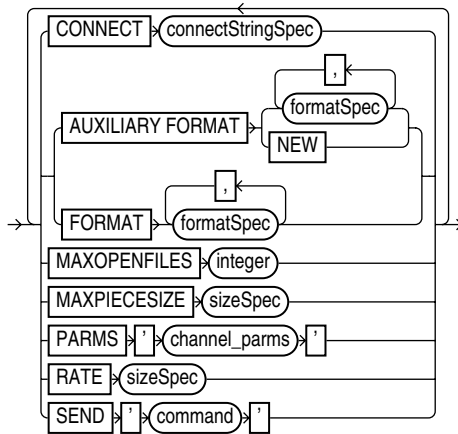
allocOperandList

Purpose

Use the *allocOperandList* subclause to control options on a **channel**, which is a connection between RMAN and a database instance.

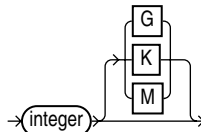
Syntax

allocOperandList::=



(*connectStringSpec::=* on page 3-12, *formatSpec::=* on page 3-22)

sizeSpec::=



Semantics

Syntax Element	Description
CONNECT <i>connectStringSpec</i>	<p>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.</p> <p>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.</p> <p>See Also: <i>connectStringSpec</i> on page 3-12 and RMAN on page 2-232</p>
AUXILIARY FORMAT	<p>Specifies the format of image copies created on an auxiliary instance.</p> <p>RMAN must be connected to the auxiliary instance with CONNECT AUXILIARY and have access to auxiliary channels.</p>

Syntax Element	Description
<i>formatSpec</i>	<p>Specifies a pattern for image copy names on an auxiliary instance.</p> <p>The path must be valid on the auxiliary host.</p> <p>See Also: <i>formatSpec</i> on page 3-22 for legal substitution variables</p>
NEW	<p>Creates an image copy in the directory specified by the DB_CREATE_FILE_DEST initialization parameter of the auxiliary instance.</p> <p>The image copy name is in an Oracle Managed Files format.</p>
FORMAT <i>formatSpec</i>	<p>Specifies the format to use for the names backup pieces or image copies created on this channel. Example 3-1 illustrates this technique.</p> <p>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.</p> <p>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 flash 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).</p> <p>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.</p> <p>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.</p> <p>See Also: <i>formatSpec</i> on page 3-22 for available FORMAT parameters</p>
MAXOPENFILES <i>integer</i>	<p>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.</p>
MAXPIECESIZE <i>sizeSpec</i>	<p>Specifies the maximum size of each backup piece created on this channel. Example 3-2 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.</p> <p>Note: You cannot use BACKUP . . . SECTION SIZE in conjunction with MAXPIECESIZE.</p>

Syntax Element	Description
PARMS ' <i>channel_parms</i> '	<p>Specifies parameters for the <code>sbt</code> channel. Example 3-3 illustrates this technique. Do not use this port-specific string if you have specified <code>DEVICE TYPE DISK</code>.</p> <p>You can use the following formats for the channel parameters:</p> <ul style="list-style-type: none"> ■ <code>'ENV=(var1=val1, var2=val2,...)'</code> 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 <code>ENV</code> parameter can be used to set server session-specific variables that perform operations on behalf of the RMAN client, for example, <code>PARMS 'ENV=(OB_DEVICE_1=tape1)'</code>. ■ <code>'SBT_LIBRARY=lib_name'</code> Specifies which media library should be used on this <code>sbt</code> channel, for example, <code>PARMS="SBT_LIBRARY=/oracle/lib/mmvs.so"</code>. Note that the default library is operating system-specific (for example, <code>libobk.so</code> on Linux and <code>ORASBT.DLL</code> on Windows). <p>See Also: <i>Oracle Database Backup and Recovery User's Guide</i> to learn how to integrate media management libraries</p>
RATE <i>sizeSpec</i>	<p>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.</p>
SEND ' <i>command</i> '	<p>Sends a vendor-specific command string to all allocated channels. For example, you could specify an Oracle Secure Backup media family with <code>SEND 'OB_MEDIA_FAMILY datafile_mf'</code>.</p> <p>See Also: Your media manager documentation to determine whether this feature is supported and when it should be used</p>

sizeSpec

This subclause specifies the size of data. Refer to [sizeSpec::=](#) on page 3-2 for the syntax diagram.

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

Examples

Example 3-1 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 3-2 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 3–3 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;
}
```

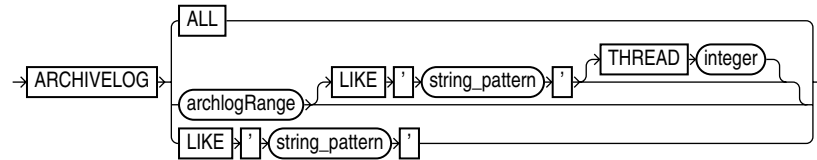
archivelogRecordSpecifier

Purpose

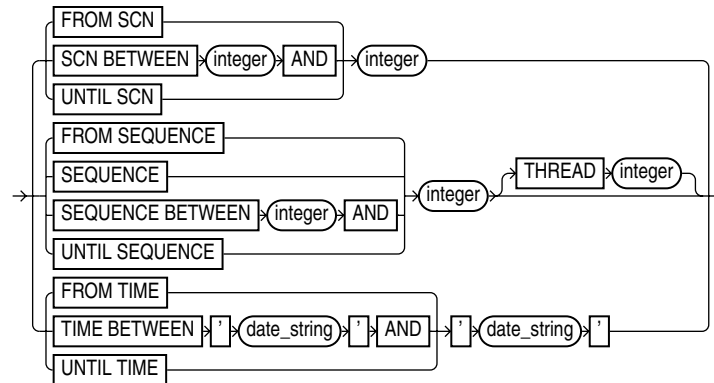
Use the *archivelogRecordSpecifier* subclause to specify a set of archived redo logs for use in RMAN operations.

Syntax

archivelogRecordSpecifier::=



archlogRange::=



Semantics

archivelogRecordSpecifier

This subclause specifies either all archived redo logs or logs whose filenames match a specified pattern.

Syntax Element	Description
ALL	Specifies all available archived logs.
LIKE ' <i>string_pattern</i> '	Specifies all archived logs that match the specified <i>string_pattern</i> . 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: <i>Oracle Real Application Clusters Administration and Deployment Guide</i> to learn about using RMAN in an Oracle RAC configuration

archlogRange

This subclause specifies a range of archived redo logs 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 logs, 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 3–1](#) explains how RMAN determines which logs are in the range. The columns `FIRST_TIME`, `NEXT_TIME`, and so on refer to columns in `V$ARCHIVED_LOG`. For example, if you specify `FROM SCN 1000 UNTIL SCN 2000`, then RMAN includes all logs whose `V$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 3–1 Determining Logs for Inclusion in the Range

Subclause	FIRST_TIME	NEXT_TIME	FIRST_SCN	NEXT_SCN	SEQUENCE#
FROM TIME t1	n/a	Later than t1	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
UNTIL TIME t2	Earlier than or the same as t2	n/a	n/a	n/a	n/a
FROM SCN s1	n/a	n/a	n/a	Greater than s1	n/a
FROM SCN s1 UNTIL SCN s2	n/a	n/a	Less than or equal to s2	Greater than s1	n/a
UNTIL SCN s2	n/a	n/a	Less than or equal to s2	n/a	n/a
FROM SEQUENCE q1	n/a	n/a	n/a	n/a	Between q1 and the maximum sequence (inclusive)
FROM SEQUENCE q1 UNTIL SEQUENCE q2	n/a	n/a	n/a	n/a	Between q1 and q2 (inclusive)
UNTIL SEQUENCE q2	n/a	n/a	n/a	n/a	Between 0 and q2 (inclusive)

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 logs 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 logs 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 <i>integer</i>	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 <i>integer</i> AND <i>integer</i>	Specifies the beginning and ending SCN for a range of logs. This syntax is semantically equivalent to FROM SCN <i>integer1</i> UNTIL SCN <i>integer2</i> .
UNTIL SCN <i>integer</i>	Specifies the ending SCN for a range of archived redo log files (see Table 3-1 for the rules determining the range).
FROM SEQUENCE <i>integer</i>	Specifies the beginning log sequence number for a sequence of archived redo log files (see Table 3-1 for the rules determining the range). Note: You can specify all log sequence numbers in a thread by using the syntax shown in Example 3-6 .
SEQUENCE <i>integer</i>	Specifies a single log sequence number.
SEQUENCE BETWEEN <i>integer</i> AND <i>integer</i>	Specifies a range of log sequence numbers. This syntax is semantically equivalent to FROM SEQUENCE <i>integer1</i> UNTIL SEQUENCE <i>integer2</i> .
UNTIL SEQUENCE <i>integer</i>	Specifies the terminating log sequence number for a sequence of archived redo log files (see Table 3-1 for the rules determining the range).
THREAD <i>integer</i>	Specifies the thread containing the archived redo log files you wish to include. This parameter is only necessary 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.
FROM TIME ' <i>date_string</i> '	Specifies the beginning time for a range of archived redo logs (see Table 3-1 for the rules determining the range). Example 3-5 shows FROM TIME in a LIST command.
TIME BETWEEN ' <i>date_string</i> ' AND ' <i>date_string</i> '	Specifies a range of times. This syntax is semantically equivalent to FROM TIME ' <i>date_string</i> ' UNTIL TIME ' <i>date_string</i> '.
UNTIL TIME ' <i>date_string</i> '	Specifies the end time for a range of archived redo logs (see Table 3-1 for the rules determining the range). Example 3-4 shows UNTIL TIME in a BACKUP command.

Examples

Example 3-4 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 logs 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 3-5 Listing Archived Log Backups by Time

As shown in [Table 3-1](#) on page 3-7, when you specify *date_string* for a range of archived redo logs, 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 3-6 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 logs 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

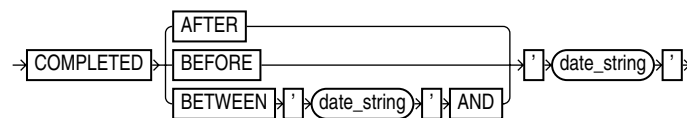
All date strings must be either:

- 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 3-7).
BEFORE 'date_string'	Specifies the time before which the backup was completed (see Example 3-9).
BETWEEN 'date_string' AND 'date_string'	Specifies a time range during which the backup was completed (see Example 3-8). Note that BETWEEN 'date1' AND 'date2' is exactly equivalent to AFTER 'date1' BEFORE 'date2'.

Examples

Example 3-7 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 3-8 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 3-9 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 username, password, and net service name for connecting to a target, recovery catalog, or auxiliary database. The connection is necessary to authenticate the user and identify the database.

Prerequisites

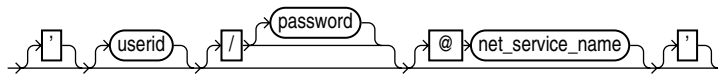
You must have SYSDBA privileges to connect to a target or auxiliary database. Do not connect to the recovery catalog database as user SYS.

Usage Notes

It is not recommended that you connect to the recovery catalog database as user SYS.

Syntax

connectStringSpec::=



Semantics

Syntax Element	Description
/	If you do not specify a user ID or password when connecting to the target database, then a forward slash establishes a connection as user SYS by using operating system authentication (see Example 3–12). Note: The forward slash depends on platform-specific environment variables.
<i>userid</i>	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 3–11). The characters will not be echoed to the terminal. You must have SYSDBA authority when connecting to the target or auxiliary database, but must <i>not</i> connect as SYS to the recovery catalog database. Note: The connect string must not contain any white space, but it can contain punctuation characters such as a forward slash (/) and an at sign (@).
<i>/password</i>	Establishes a connection for the specified user by using a password. If the target database is not open, then a password file must exist.
<i>@net_service_name</i>	Establishes a connection to the database through an optional Oracle Net net service name (see Example 3–10).

Examples

Example 3–10 Connecting to a Target Database Without a Recovery Catalog

This example connects to the target database by using a password and the Oracle Net service name `prod1` in the default NOCATALOG mode:

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

Example 3–11 Connecting to a Target Database and Entering the Password Interactively

This example connects to the target database as user SYS but without specifying a password at the command line. Note that you are prompted for a 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:
```

Example 3–12 Connecting with Operating System Authentication

This example starts RMAN and then connects to the target database prod by using operating system authentication. The example also connects to the recovery catalog database catdb by using a net service name.

```
% rman
```

```
RMAN> CONNECT TARGET /
```

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

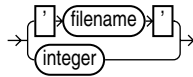
datafileSpec

Purpose

Use the *datafileSpec* subclause to specify a datafile by filename or absolute file number.

Syntax

***datafileSpec*::=**



Semantics

Syntax Element	Description
<i>'filename'</i>	<p>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 3-14).</p> <p>Double and single quotes are both legal (although only single quotes are shown in the diagram).</p> <p>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</p>
<i>integer</i>	<p>Specifies the datafile by using its absolute file number (see Example 3-13). Obtain the file number from the V\$DATAFILE, V\$DATAFILE_COPY, or V\$DATAFILE_HEADER views or REPORT SCHEMA command output.</p>

Examples

Example 3-13 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 3-14 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';

```

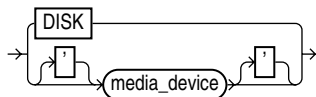
deviceSpecifier

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 3–16 on page 3-15).
<i>media_device</i>	Specifies a sequential I/O device or access method for storage (see Example 3–15 on page 3-15). The <i>media_device</i> 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 <i>sbt</i> or <i>sbt_tape</i> (which are synonymous values). Note: RMAN stores the value <i>sbt</i> in the recovery catalog as <i>sbt_tape</i> for backward compatibility.

Examples

Example 3–15 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 3–16 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 would create an image copy of datafile `/disk1/dbs/users.dbf` as `/disk1/newdbs/users.dbf`:

```
BACKUP AS COPY TABLESPACE users
  DB_FILE_NAME_CONVERT = ('dbs', 'newdbs');
```

When multiple possible matches exist for a given filename being converted, RMAN uses first match in the list of patterns to generate the new filename. The following command would have 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 TABLESPACE users
  DB_FILE_NAME_CONVERT = ('dbs', 'newdbs', '/disk1', '/newdisk');
```

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
```

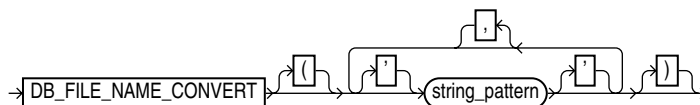
An OMF filename in ASM storage might be of the following form:

```
+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
<i>string_pattern</i>	<p>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:</p> <pre>DB_FILE_NAME_CONVERT = ('str1','str2','str3', 'str4' ...)</pre> <p>In this example, <i>str1</i> is a pattern matching the original filename, <i>str2</i> is the pattern replacing <i>str1</i> in the generated filename, <i>str3</i> is a pattern matching the original filename, and <i>str4</i> is the pattern replacing <i>str3</i> in the generated filename.</p>

Examples

Example 3–17 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 3–18 Using DB_FILE_NAME_CONVERT with Multiple Conversion Pairs

This example creates image copies of the same datafiles described in [Example 3–17](#). 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.

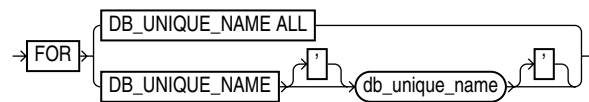
Usage Notes

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 <code>SET DBID</code> command).
FOR DB_UNIQUE_NAME <i>db_unique_name</i>	Specifies the primary or standby database in the recovery catalog with the specified <i>db_unique_name</i> .

Examples

Example 3-19 Listing Expired Backups Associated with a Standby Database

This example connects to the 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/password@catdb;
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 logs for use in RMAN operations.

Usage Notes

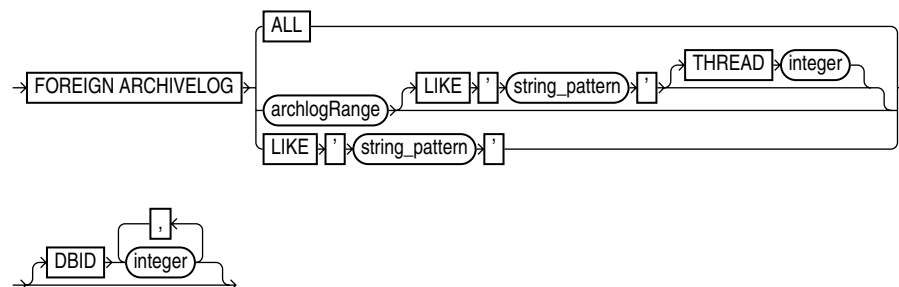
Foreign archived redo logs are received by a logical standby database for a LogMiner session. Unlike normal archived logs, foreign archived logs have a different DBID. For this reason, they cannot be backed up or restored on a logical standby database.

A logical standby database creates foreign archived logs in the flash recovery area if the following conditions are met:

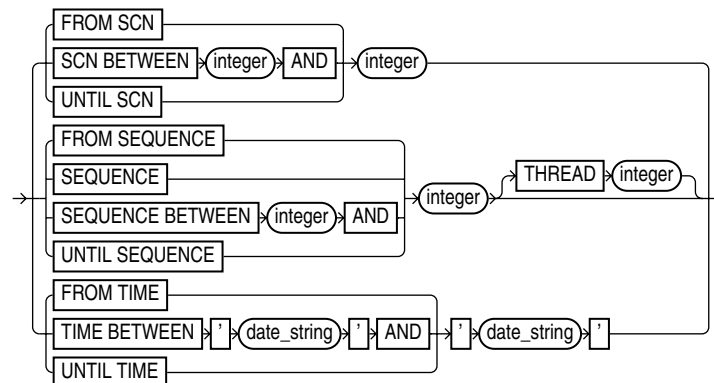
- A flash recovery area is configured on the logical standby database.
- One of the LOG_ARCHIVE_DEST_1 initialization parameters is set to 'LOCATION=USE_DB_RECOVERY_FILE_DEST' with a proper VALID_FOR setting to receive foreign archived logs.
- The COMPATIBLE initialization parameter is set to a value greater than or equal to 11.0.0.

Syntax

foreignlogRecordSpecifier::=



archlogRange::=



Semantics

The semantics are exactly the same as for [archiveLogRecordSpecifier](#) on page 3-6 except that the logs are foreign archived redo logs.

Examples

Example 3–20 Crosschecking Foreign Archived Redo Logs

This example crosschecks all foreign archived redo logs:

```
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 flash recovery area if it is enabled, or in a platform-specific directory (for example, `*/dbs` on UNIX) if a flash 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 legal 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 legal 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.

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 3-23 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*::=**

→ `'` `format_string` `'` →

Semantics

formatSpec

The following table lists RMAN substitution variables that are legal in format strings.

Syntax Element	Description
%a	Specifies the activation ID of the database.
%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 3–22 on page 3-24).
%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 3–22 on page 3-24).
%F	Combines the DBID, day, month, year, and sequence into a unique and repeatable generated name. This variable translates into <i>c-<i>IIIIIIIIII</i>-<i>YYYYMMDD</i>-<i>QQ</i></i> , where: <ul style="list-style-type: none"> ■ <i>IIIIIIIIII</i> stands for the DBID. The DBID is printed in decimal so that it can be easily associated with the target database. ■ <i>YYYYMMDD</i> is a time stamp in the Gregorian calendar of the day the backup is generated ■ <i>QQ</i> 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 legal when backing up datafiles as image copies.
%n	Specifies the name of the database, padded on the right with <i>x</i> characters to a total length of eight characters. For example, if <i>prod1</i> is the database name, then the padded name is <i>prod1xxx</i> .
%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 <i>PROXY</i> , then the %p variable must be included in the <i>FORMAT</i> 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, <i>CREATE CONTROLFILE</i> 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: <i>YYYYMMDD</i> .
%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	<p>Specifies a system-generated unique filename (default).</p> <p>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:</p> <pre>data-D-%d_id-%I_TS-%N_FNO-%f_%u</pre> <p>For an image copy of an archived redo log, %U means the following:</p> <pre>arch-D_%d-id-%I_S-%e_T-%h_A-%a_%u</pre> <p>For an image copy of a control file, %U means the following:</p> <pre>cf-D_%d-id-%I_%u</pre>
%Y	Specifies the year in this format: YYYY.
%%	Specifies the percent (%) character. For example, %%Y translates to the string %Y.

Example

Example 3–21 Specifying an ASM Disk Group

This example copies the database to ASM disk group DISK1:

```
BACKUP AS COPY DATABASE FORMAT '+DATAFILE';
```

Example 3–22 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';
```

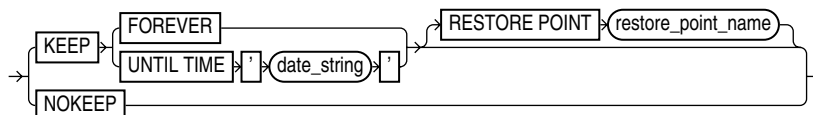
keepOption

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	<p>Specifies the backup as an archival backup, which is a self-contained backup that is exempt from the configured retention policy.</p> <p>An archival backup is self-contained because it 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 logs needed to make the database backup consistent (see Example 2-26 on page 2-49).</p> <p>RMAN does not consider backup pieces with the <code>KEEP</code> 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.</p> <p>Note: You cannot use <code>KEEP</code> to override the retention policy for files stored in the flash recovery area. If you specify <code>KEEP</code> when backing up to the recovery area, then RMAN issues an error.</p> <p>When <code>KEEP</code> is specified, RMAN creates multiple backup sets. RMAN backs up datafiles, archived redo logs, the control file, and the server parameter file with the options specified in the first <i>backupOperand</i>. RMAN uses the <code>FORMAT</code>, <code>POOL</code>, and <code>TAG</code> parameters for all the backups. For this reason, the <code>FORMAT</code> string must allow for the creation of multiple backup pieces. Specifying <code>%U</code> is the easiest way to meet this requirement.</p> <p>Note: A recovery catalog is only required for <code>KEEP FOREVER</code>. No other <code>KEEP</code> options require a catalog.</p>
FOREVER	<p>Specifies that the backup or copy never becomes obsolete (see Example 2-27 on page 2-49). A recovery catalog is required when <code>FOREVER</code> is specified because the backup records eventually age out of the control file.</p>

Syntax Element	Description
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 2-49). 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 3-23 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 logs 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 3-24 Removing the KEEP Attributes for a Backup

This example backs up all archived redo logs. 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
```


dbObject

Syntax Element	Description
DATABASE	Specifies backup sets or image copies of all files in the current database.
DATAFILE <i>datafileSpec</i>	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: <i>datafileSpec</i> on page 3-14
TABLESPACE <i>tablespace_name</i>	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 3–25 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 3–26 Crosschecking Archived Redo Logs

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 3–27 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_2z15k15h_.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_2z15k15h_.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_2z15k15h_.bkp
```

```
RECID=191 STAMP=616690994
```

```
Deleted 1 EXPIRED objects
```

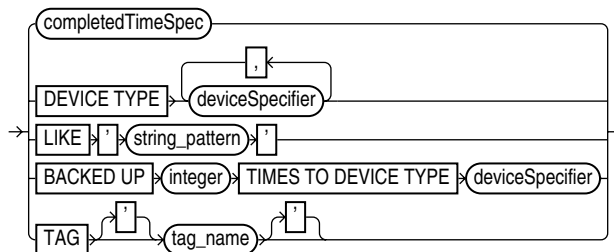
maintQualifier

Purpose

Use the *maintQualifier* subclause to specify database files and archived redo logs.

Syntax

***maintQualifier*::=**



(*completedTimeSpec*::= on page 3-10, *deviceSpecifier*::= on page 3-15)

Semantics

maintQualifier

Syntax Element	Description
<i>completedTimeSpec</i>	Specifies a range of time for completion of the backup or copy. See Also: <i>completedTimeSpec</i> on page 3-10
DEVICE TYPE <i>deviceSpecifier</i>	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: <i>deviceSpecifier</i> on page 3-15
LIKE ' <i>string_pattern</i> '	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 <i>integer</i> TIMES TO DEVICE TYPE <i>deviceSpecifier</i>	Restricts the command to archived logs that have been successfully backed up <i>integer</i> or more times to the specified media. This option applies only to archived redo logs. 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 TO BACKED UP is overridden. Use FORCE with DELETE ARCHIVELOG to override this configuration as well as any mismatches between media and repository.

Syntax Element	Description
TAG <i>tag_name</i>	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 3–28 Listing Backups in a Specific Location

The following command lists all image copies located in /disk1:

```

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 3–29 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

```

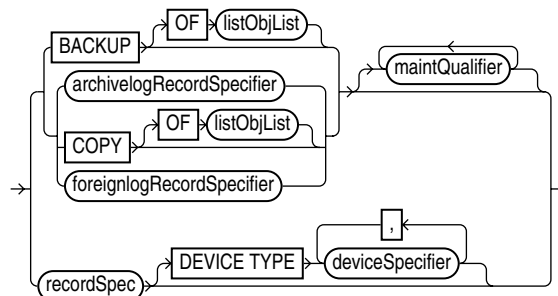
maintSpec

Purpose

Use the *maintSpec* subclause to specify the backup files operated on by the [CHANGE](#), [CROSSCHECK](#), and [DELETE](#) commands.

Syntax

***maintSpec*::=**



([listObjList](#)::= on page 3-28, [maintQualifier](#)::= on page 3-31, [archivelogRecordSpecifier](#)::= on page 3-6, [recordSpec](#)::= on page 3-36, [deviceSpecifier](#)::= on page 3-15)

Semantics

maintSpec

Syntax Element	Description
BACKUP	Processes backup sets and proxy copies. If you do not use the OF clause with CHANGE BACKUP, then RMAN changes all backup sets and proxy copies 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 3-28
archivelogRecordSpecifier	Processes the specified archived redo logs. 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 3-6

Syntax Element	Description
COPY	Processes datafile copies, control file copies, and archived redo logs. 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 3-28
foreignlogRecordSpecifier	Processes the specified foreign archived redo logs. See Also: foreignlogRecordSpecifier on page 3-6
maintQualifier	Restricts the command based on the specified options. See Also: maintQualifier on page 3-31
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 2-160 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 3-36
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 3-15

Examples

Example 3-30 Crosschecking Backups

The following command crosschecks backups of archived redo logs:

```
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

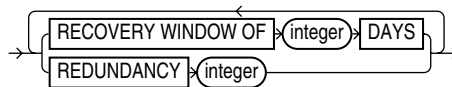
Use the *obsOperandList* subclause used to specify which criteria are used to mark backups as obsolete.

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
<code>RECOVERY WINDOW OF <i>integer</i> DAYS</code>	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 <i>integer</i> days. See Also: CONFIGURE on page 2-64 for an explanation of the recovery window
<code>REDUNDANCY <i>integer</i></code>	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 <i>integer</i> more recent backup sets or image copies of this file; a datafile backup set is obsolete if there are at least <i>integer</i> more recent backup sets or image copies of each datafile contained in the backup set. For example, <code>REDUNDANCY 2</code> 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 3–31 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

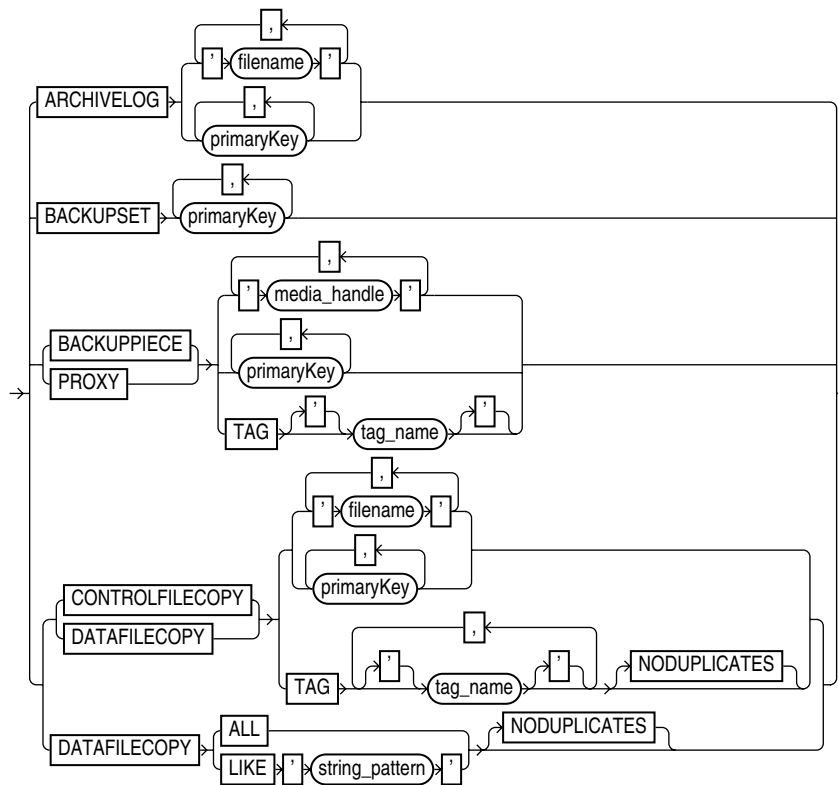
Purpose

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.
DATAFILECOPY	Specifies a datafile copy by primary key, filename pattern ('filename'), or TAG tag_name. If you crosscheck a datafile copy, then you must specify a filename rather than a primary key.
DATAFILECOPY	Specifies a datafile copy by ALL, LIKE, or string_pattern. If you crosscheck a datafile copy, then you must specify a string pattern rather than a primary key.
	NODUPLICATES

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 3-32 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 3-33 Deleting Datafile Copies

This example deletes the specified datafile copy:

```

RMAN> DELETE NOPROMPT DATAFILECOPY '/disk1/oradata/prod/users01.dbf';

```

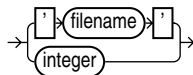
tempfileSpec

Purpose

Use the *tempfileSpec* subclause to specify a tempfile by name or absolute file number.

Syntax

***tempfileSpec* ::=**



Semantics

Syntax Element	Description
<i>'filename'</i>	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 legal (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.
<i>integer</i>	Specifies the datafile by absolute file number. Obtain the file number from the V\$TEMPFILE view or REPORT SCHEMA output.

Examples

Example 3–34 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.

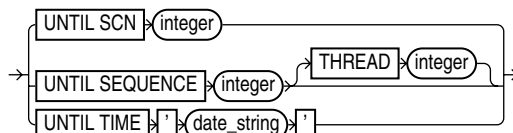
untilClause

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 <i>integer</i>	<p>Specifies an SCN as an upper, noninclusive limit.</p> <p>RMAN selects only files that can be used to restore or recover up to but not including the specified SCN (see Example 3-35 on page 3-40). For example, <code>RESTORE DATABASE UNTIL SCN 1000</code> chooses only backups that could be used to recover to SCN 1000.</p>
UNTIL SEQUENCE <i>integer</i>	<p>Specifies a redo log sequence number and thread as an upper, noninclusive limit.</p> <p>RMAN selects only files that can be used to restore or recover up to but not including the specified sequence number. For example, <code>REPORT OBSOLETE UNTIL SEQUENCE 8000</code> reports only backups that could be used to recover through log sequence 7999.</p>
THREAD <i>integer</i>	<p>Specifies the number of the redo thread.</p>
UNTIL TIME ' <i>date_string</i> '	<p>Specifies a time as an upper, noninclusive limit (see Example 3-36 on page 3-40).</p> <p>RMAN selects only files that can be used to restore and recover up to but not including the specified time. For example, <code>LIST BACKUP UNTIL TIME 'SYSDATE-7'</code> lists all backups that could be used to recover to a point one week ago.</p> <p>When specifying dates in RMAN commands, the date string must be either:</p> <ul style="list-style-type: none"> ■ A literal string whose format matches the <code>NLS_DATE_FORMAT</code> setting. ■ A SQL expression of type <code>DATE</code>, for example, <code>'SYSDATE-10'</code> or <code>"TO_DATE('01/30/2007', 'MM/DD/YYYY')"</code>. Note that the second example includes its own date format mask and so is independent of the current <code>NLS_DATE_FORMAT</code> setting. <p>Following are examples of typical date settings in RMAN commands:</p> <pre> BACKUP ARCHIVELOG FROM TIME 'SYSDATE-31' UNTIL TIME 'SYSDATE-14'; RESTORE DATABASE UNTIL TIME "TO_DATE('09/20/06', 'MM/DD/YY')"; </pre>

Examples

Example 3–35 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 3–36 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';
```

Recovery Catalog Views

This chapter contains descriptions of recovery catalog views. You can only access these views if you have created a recovery catalog. For a summary of the recovery catalog views, refer to "[Summary of RMAN Recovery Catalog Views](#)" on page 4-1.

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 4–1 *Recovery Catalog Views*

Recovery Catalog View	Corresponding V\$ View	Catalog View Describes ...
RC_ARCHIVED_LOG	V\$ARCHIVED_LOG	Archived and unarchived redo logs
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_CONTROLFILE	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
RC_BACKUP_DATAFILE_DETAILS	V\$BACKUP_DATAFILE_DETAILS	Details about datafile backups for Enterprise Manager

Table 4–1 (Cont.) Recovery Catalog Views

Recovery Catalog View	Corresponding V\$ View	Catalog View Describes ...
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 logs 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\$CONTROLFILE_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 logs for Enterprise Manager
RC_PROXY_ARCHIVELOG_SUMMARY	V\$PROXY_ARCHIVELOG_SUMMARY	Summary of information about proxy archived redo logs for Enterprise Manager
RC_PROXY_CONTROLFILE	V\$PROXY_CONTROLFILE	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
RC_PROXY_DATAFILE	V\$PROXY_DATAFILE	Datafile backups that were taken using the proxy copy functionality

Table 4–1 (Cont.) Recovery Catalog Views

Recovery Catalog View	Corresponding V\$ View	Catalog View Describes ...
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\$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 logs. 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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a 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 form a 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 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	Datatype	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 flash recovery area: YES or NO.
COMPRESSED	VARCHAR2 (3)	Internal only
CREATOR	VARCHAR2 (7)	Creator of the archived redo log: <ul style="list-style-type: none"> ■ ARCH - Archiver process ■ FGPD - Foreground process ■ RMAN - Recovery Manager ■ SRMN - RMAN at standby ■ LGWR - Logwriter 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	Datatype	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 form a 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	Datatype	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 form a 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 one 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 logs 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 nM, nG, nT, nP.
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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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 form a 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	Datatype	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	Datatype	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 form a 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	Datatype	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 form a 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	Datatype	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 form a 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 flash 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	Datatype	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 form a 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. It corresponds to the V\$BACKUP_CORRUPTION view in the control file. Note that corruptions are not tolerated in control file and archived redo log backups.

Column	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 form a 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 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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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 form a 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	Datatype	Description
BLOCKS	NUMBER	The number of data blocks written to the backup. This value is often less than <code>DATAFILE_BLOCKS</code> 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 <code>DATAFILE_BLOCKS</code> .
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 <code>INCREMENTAL_LEVEL</code> 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, <code>BS_LEVEL</code> is 1 and <code>INCREMENTAL_LEVEL</code> 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 <code>DATAFILE_BLOCKS</code> . Otherwise, the value of this column will be the same as <code>DATAFILE_BLOCKS</code> . Even when change tracking data is used, the value of this column may be larger than <code>BLOCKS</code> , 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	Datatype	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 form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a 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	Datatype	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, 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	Datatype	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 form a 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 logs). 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 DATAFILE.

Column	Datatype	Description
DF_CHECKPOINT_CHANGE#	NUMBER	Checkpoint change number of the datafile described by this row. This column populated only if FILE_TYPE is DATAFILE.
DF_CKP_MOD_TIME	DATE	Checkpoint time of the datafile described by this row. Valid only if FILE_TYPE is DATAFILE.
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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a 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 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 form a 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 flash recovery area: YES or NO.

Column	Datatype	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	Datatype	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 form a 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 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 form a 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	Datatype	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 flash 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 logs in backup sets. It corresponds to the V\$BACKUP_REDOLOG view.

You cannot back up online logs directly: you must first archive them to disk and then back them up. An archived log backup set contains one or more archived logs.

Column	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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 form a 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	Datatype	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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a 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 connected to the recovery catalog, then this value appears in the KEY column of the output. Use this column to form a 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	Datatype	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	Datatype	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 form a 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 form a joint 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	Datatype	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	Datatype	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 form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a 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 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 form a 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	Datatype	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 form a 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 form a 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	Datatype	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 form a 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 4-62 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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 flash recovery area: YES or NO.
RSR_KEY	NUMBER	Unique key for the row in RC_RMAN_STATUS that created this backup piece.

Column	Datatype	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 <code>SITE_KEY</code> value. You can use <code>SITE_KEY</code> in a join with the <code>RC_SITE</code> view to obtain the <code>DB_UNIQUE_NAME</code> of the database.

RC_COPY_CORRUPTION

This view lists corrupt block ranges in datafile copies. It corresponds to the V\$COPY_CORRUPTION view.

Column	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 connected to the recovery catalog, then this value appears in the KEY column of the output. Use this column to form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the current incarnation. Use this column to form a join with RC_DATABASE_INCARNATION.
DBID	NUMBER	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.

Column	Datatype	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the current incarnation. Use this column to form a join with RC_DATABASE_INCARNATION.
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: <ul style="list-style-type: none"> ▪ 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 is for logically corrupt blocks. CORRUPTION_CHANGE# will have a nonzero value.

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	Datatype	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	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 11g 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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 IMMEDIATE (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	Datatype	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 flash 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 11g 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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	Datatype	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 logs that have been performed. This view is primarily intended to be used internally by Enterprise Manager.

Column	Datatype	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 form a 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 logs.

This view is primarily intended to be used internally by Enterprise Manager.

Column	Datatype	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 form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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	Datatype	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	Datatype	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 form a 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	Datatype	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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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 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	Datatype	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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	Description
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	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 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	Datatype	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 <i>hh:mm:ss</i> .

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	Datatype	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 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	Datatype	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	Datatype	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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database corresponding to this configuration. Use this column to form a 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 11g 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the 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: <ul style="list-style-type: none"> ▪ 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. <ul style="list-style-type: none"> ▪ 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.
STATUS	VARCHAR2 (33)	The status of the operation described by this row. Possible values are: COMPLETED, COMPLETED WITH WARNINGS, COMPLETED WITH ERRORS, and FAILED.
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.

Column	Datatype	Description
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	Datatype	Description
<code>SITE_KEY</code>	NUMBER	The unique key of this database. You can join the <code>RC_SITE.SITE_KEY</code> column with the <code>RC_SITE</code> column of other views to determine which <code>DB_UNIQUE_NAME</code> is associated with a backup.
<code>DB_KEY</code>	NUMBER	The primary key for this database in the recovery catalog. Use this column to form a join with almost any other catalog view.
<code>DATABASE_ROLE</code>	VARCHAR2 (7)	The role of the database in the Data Guard environment.
<code>CF_CREATE_TIME</code>	DATE	The creation date of the control file.
<code>DB_UNIQUE_NAME</code>	VARCHAR2 (30)	The <code>DB_UNIQUE_NAME</code> of the database. All databases in a Data Guard environment share the same <code>DBID</code> but different <code>DB_UNIQUE_NAME</code> 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the database that owns this stored script. Use this column to form a 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the database that owns this stored script. Use this column to form a 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 <code>SCRIPT_NAME</code> and <code>LINE</code> .
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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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: <ul style="list-style-type: none"> ■ 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	Datatype	Description
DB_KEY	NUMBER	The primary key for the target database. Use this column to form a join with almost any other catalog view.
DBINC_KEY	NUMBER	The primary key for the incarnation of the target database. Use this column to form a join with RC_DATABASE_INCARNATION.
DB_NAME	VARCHAR2 (8)	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	Datatype	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 11g 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	Datatype	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 form a 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.

Deprecated RMAN Commands

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
11.1.0	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
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

Table A-1 (Cont.) Deprecated RMAN Syntax

Deprecated in Release	Deprecated Syntax	Preferred Current Syntax
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 3-39)	UNTIL SEQUENCE (see untilClause on page 3-39)
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)

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](#)

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/rman@catdb
```

2. Query the `rcvcr` catalog table. For example, run this query:

```
SQL> SELECT * FROM rcvcr;
```

```

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 `rcvcr` 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. Oracle9i RMAN clients cannot connect to a virtual private catalog. This version restriction does not affect RMAN client connections to an Oracle Database 11g 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 legal combinations, listed in [Table B-2](#)). The RMAN client cannot be of a greater version than the target or auxiliary database.
- While backing up an Oracle Database 10g or later database with the Oracle9i 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.

[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

Table B–2 (Cont.) RMAN Compatibility Table

Target/Auxiliary Database	RMAN client	Recovery Catalog Database	Recovery Catalog Schema
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

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 Oracle9i RMAN client to back up an Oracle Release 10g database, you will not have access to features like the flash 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
- 11.1.0

You want to record RMAN repository data about these databases in a single recovery catalog database. According to [Table B–2](#) on page B-2, you can use a single 11.1.0 recovery catalog database with a 11.1.0 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 11.1.0 database.

Symbols

- ? symbol
 - in RMAN quoted strings, 1-3
- @ command, 2-2
 - in RMAN quoted strings, 1-3
- @@ command, 2-4

A

- ADVISE FAILURE command, 2-6
- ALLOCATE CHANNEL command, 2-10, 3-2
 - and shared server, 2-13
 - FOR MAINTENANCE option, 2-13
- ALTER DATABASE command, 2-17
- archivelogRecordSpecifier clause, 3-6, 3-20
- autobackups
 - control file, 2-76

B

- BACKUP command, 2-19
- backup sets
 - and unused block compression, 2-41
 - binary compression, 2-20, 2-42
 - unused block compression, 2-41
- backups
 - binary compression, 2-20, 2-42
 - datafiles, 2-41
 - unused block compression, 2-41
- binary compression of backup sets, 2-20, 2-42

C

- CATALOG command, 2-51
- CHANGE command, 2-56
- channels
 - allocating to shared server sessions, 2-13
- command line
 - arguments for RMAN, 2-232
- commands, Recovery Manager
 - @, 2-2
 - @@, 2-4
 - ADVISE FAILURE, 2-6
 - ALLOCATE CHANNEL, 2-10, 3-2
 - ALLOCATE CHANNEL FOR MAINTENANCE, 2-13

- ALTER DATABASE, 2-17
- archivelogRecordSpecifier clause, 3-6, 3-20
- BACKUP, 2-19
- CATALOG, 2-51
- CHANGE, 2-56
- completedTimeSpec clause, 3-10
- CONFIGURE, 2-64
- CONNECT, 2-83
- connectStringSpec clause, 3-12
- CONVERT, 2-86
- CREATE CATALOG, 2-98
- CREATE SCRIPT, 2-101
- CROSSCHECK, 2-104
- DELETE, 2-108
- DELETE SCRIPT, 2-114
- deprecated, A-1
- DROP CATALOG, 2-116
- DROP DATABASE, 2-118
- DUPLICATE, 2-119
- EXECUTE SCRIPT, 2-136
- EXIT, 2-139
- fileNameConversionSpec clause, 3-16
- FLASHBACK, 2-140
- GRANT, 2-146
- HOST, 2-149, 2-151
- LIST, 2-154
- listObjList clause, 3-28
- PRINT SCRIPT, 2-175
- QUIT, 2-177
- recordSpec, 3-36
- RECOVER, 2-178
- REGISTER, 2-191
- RELEASE CHANNEL, 2-193
- REPAIR FAILURE, 2-195
- REPLACE SCRIPT, 2-199
- REPORT, 2-202
- RESET DATABASE, 2-209
- RESTORE, 2-211
- RESYNC CATALOG, 2-226
- REVOKE, 2-230
- RMAN, 2-232
- RUN, 2-238
- SEND, 2-241
- SET, 2-243
- SHOW, 2-252
- SHUTDOWN, 2-255

SPOOL, 2-257
SQL, 2-258
STARTUP, 2-260
summary, 1-6, 4-1
SWITCH, 2-262
TRANSPORT TABLESPACE, 2-266
UNREGISTER DATABASE, 2-271
untilClause, 3-39
UPGRADE CATALOG, 2-274
VALIDATE, 2-276
comments in RMAN commands, 1-3
compatibility
 recovery catalog, B-1
 Recovery Manager, B-1
completedTimeSpec clause, 3-10
CONFIGURE command, 2-64
CONNECT command, 2-83
connectStringSpec clause, 3-12
control files
 automatic backups, 2-76
CONVERT command, 2-86
corrupt datafile blocks
 maximum acceptable number, 2-247
corruption detection
 using SET MAXCORRUPT command, 2-247
CREATE CATALOG command, 2-98
CREATE SCRIPT command, 2-101
CROSSCHECK command, 2-104

D

datafiles
 backing up, 2-41
 unused block compression, 2-41
dates
 specifying in RMAN commands, 3-39
DELETE command, 2-108
DELETE SCRIPT command, 2-114
deprecated commands
 Recovery Manager, A-1
DROP CATALOG command, 2-116
DROP DATABASE command, 2-118
DUPLICATE command, 2-119

E

environment variables
 in RMAN strings, 1-3
EXECUTE SCRIPT command, 2-136
EXIT command, 2-139

F

fileNameConversionSpec clause, 3-16
FLASHBACK command, 2-140

G

GRANT command, 2-146

H

HOST command, 2-149, 2-151

K

keywords
 in syntax diagrams, 1-2

L

LIST command, 2-154
listObjList clause, 3-28

M

MAXCORRUPT parameter
 SET command, 2-247
MAXSETSIZE parameter
 BACKUP command, 2-34, 2-281

P

parameters
 in syntax diagrams, 1-2
PRINT SCRIPT command, 2-175

Q

QUIT command, 2-177
quoted strings
 environment variables, 1-3

R

RC_ARCHIVED_LOG view, 4-4
RC_BACKUP_ARCHIVELOG_DETAILS view, 4-6
RC_BACKUP_ARCHIVELOG_SUMMARY
 view, 4-7
RC_BACKUP_CONTROLFILE view, 4-8
RC_BACKUP_CONTROLFILE_DETAILS view, 4-10
RC_BACKUP_CONTROLFILE_SUMMARY
 view, 4-11
RC_BACKUP_COPY_DETAILS view, 4-12
RC_BACKUP_COPY_SUMMARY view, 4-13
RC_BACKUP_CORRUPTION view, 4-14
RC_BACKUP_DATAFILE view, 4-15
RC_BACKUP_DATAFILE_DETAILS view, 4-17
RC_BACKUP_DATAFILE_SUMMARY view, 4-18
RC_BACKUP_PIECE view, 4-22
RC_BACKUP_REDOLOG view, 4-26
RC_BACKUP_SET view, 4-28
RC_BACKUP_SPFILE view, 4-33
RC_CHECKPOINT view, 4-36
RC_CONTROLFILE_COPY view, 4-37
RC_COPY_CORRUPTION view, 4-39
RC_DATABASE recovery catalog view, 4-40
RC_DATABASE view, 2-249
RC_DATABASE_BLOCK_CORRUPTION
 view, 4-41
RC_DATABASE_INCARNATION view, 2-249, 4-42

- RC_DATAFILE view, 4-43
- RC_DATAFILE_COPY view, 4-45
- RC_LOG_HISTORY view, 4-47
- RC_OFFLINE_RANGE view, 4-48
- RC_PROXY_CONTROLFILE view, 4-53
- RC_PROXY_DATAFILE view, 4-57
- RC_REDO_LOG view, 4-59, 4-70
- RC_REDO_THREAD view, 4-60
- RC_RESTORE_POINT view, 4-61
- RC_RESYNC view, 4-62
- RC_RMAN_CONFIGURATION view, 4-68
- RC_SITE view, 4-72
- RC_STORED_SCRIPT view, 4-73
- RC_STORED_SCRIPT_LINE view, 4-74
- RC_TABLESPACE view, 4-75, 4-76
- recordSpec clause, 3-36
- RECOVER command, 2-178
- recovery
 - database
 - in NOARCHIVELOG mode, 2-189
- recovery catalog
 - views, 4-1
- Recovery Manager
 - backups
 - control file autobackups, 2-76
 - commands
 - @@, 2-4
 - ALLOCATE CHANNEL, 2-10, 3-2
 - ALLOCATE CHANNEL FOR MAINTENANCE, 2-13
 - ALTER DATABASE, 2-17
 - archiveLogRecordSpecifier clause, 3-6, 3-20
 - BACKUP, 2-19
 - CATALOG, 2-51
 - CHANGE, 2-56
 - completedTimeSpec, 3-10
 - CONFIGURE, 2-64
 - CONNECT, 2-83
 - connectStringSpec, 3-12
 - CREATE CATALOG, 2-98
 - CREATE SCRIPT, 2-101
 - CROSSCHECK, 2-104
 - DELETE, 2-108
 - DELETE SCRIPT, 2-114
 - DROP CATALOG, 2-116
 - DROP DATABASE, 2-118
 - DUPLICATE, 2-119
 - EXECUTE SCRIPT, 2-136
 - EXIT, 2-139
 - fileNameConversionSpec, 3-16
 - FLASHBACK, 2-140
 - GRANT, 2-146
 - HOST, 2-149, 2-151
 - LIST, 2-154
 - listObjList, 3-28
 - PRINT SCRIPT, 2-175
 - QUIT, 2-177
 - recordSpec, 3-36
 - RECOVER, 2-178
 - REGISTER, 2-191

- RELEASE CHANNEL, 2-193
- REPLACE SCRIPT, 2-199
- REPORT, 2-202
- RESET DATABASE, 2-209
- RESTORE, 2-211
- RESYNC CATALOG, 2-226
- REVOKE, 2-230
- RUN, 2-238
- SEND, 2-241
- SET, 2-243
- SHOW, 2-252
- SHUTDOWN, 2-255
- SPOOL, 2-257
- SQL, 2-258
- STARTUP, 2-260
- SWITCH, 2-262
- TRANSPORT TABLESPACE, 2-266
- UNREGISTER DATABASE, 2-271
 - untilClause, 3-39
- UPGRADE CATALOG, 2-274
 - VALIDATE, 2-276
- compatibility, B-1
- dates in commands, 3-39
 - syntax conventions, 1-2
- REGISTER command, 2-191
- RELEASE CHANNEL command (RMAN), 2-193
- REPAIR FAILURE command, 2-195
- REPLACE SCRIPT command, 2-199
- REPORT command, 2-202
- RESET DATABASE command, 2-209
- RESTORE command, 2-211
- RESYNC CATALOG command, 2-226
- REVOKE command, 2-230
- RMAN command, 2-232
- RUN command, 2-238

S

- scenarios, Recovery Manager
 - recovering pre-resetlogs backup, 2-189
- SECTION SIZE parameter
 - BACKUP command, 2-35
- SEND command, 2-241
- SET command, 2-243
- shared server
 - allocating channels, 2-13
- SHOW command, 2-252
- SHUTDOWN command, 2-255
- SPOOL command, 2-257
- SQL command, 2-258
- STARTUP command, 2-260
- Structured Query Language (SQL)
 - syntax, 1-1
- SWITCH command, 2-262
- syntax conventions, Recovery Manager, 1-2
- syntax diagrams
 - keywords, 1-2
 - parameters, 1-2

T

TRANSPORT TABLESPACE command, 2-266

U

UNREGISTER DATABASE command, 2-271

untilClause, 3-39

unused block compression, 2-41

UPGRADE CATALOG command, 2-274

V

V\$ARCHIVED_LOG view, 2-168, 3-7, 3-8

V\$BACKUP_DEVICE view, 2-11, 2-14

V\$BACKUP_PIECE view, 2-161, 2-163, 2-165

V\$BACKUP_SET view, 2-161, 2-164, 2-165, 2-166

V\$DATABASE view, 2-249

V\$DATABASE_BLOCK_CORRUPTION
view, 2-184, 2-185, 2-189, 2-216, 2-280

V\$DATAFILE view, 3-14, 3-38

V\$DATAFILE_COPY view, 2-167, 2-168, 3-14

V\$DATAFILE_HEADER view, 3-14

V\$PROXY_DATAFILE view, 2-163

V\$SESSION view, 2-249

VALIDATE command, 2-276

views, recovery catalog, 4-1