

Oracle® Database

Recovery Manager Reference

10g Release 2 (10.2)

B14194-01

June 2005

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

Oracle Database Recovery Manager Reference, 10g Release 2 (10.2)

B14194-01

Copyright © 1996, 2005, Oracle. All rights reserved.

Primary Author: Antonio Romero

Contributing Author: Lance Ashdown

Contributors: Tammy Bednar, Anand Beldalker, Timothy Chien, Raymond Guzman, Alex Hwang, Ashok Joshi, J. William Lee, Valarie Moore, Muthu Olagappan, Samitha Samaranayake, Francisco Sanchez, 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 Corporation, 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 Retek 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

Send Us Your Comments	vii
Preface	ix
Audience	ix
Organization	ix
Related Documentation	x
Conventions	x
Documentation Accessibility	xii
 1 About RMAN Commands	
Conventions Used in this Reference	1-1
RMAN Text Conventions	1-1
RMAN Command Format, Syntax Diagrams and Notation	1-1
Format of RMAN Commands	1-1
Syntax Diagrams	1-2
Keywords	1-2
Placeholders	1-2
RMAN Reserved Words	1-3
RMAN Command Entries	1-5
RMAN Code Examples	1-5
 2 RMAN Commands	
Summary of RMAN Commands	2-1
@	2-5
@@	2-6
ALLOCATE CHANNEL	2-7
ALLOCATE CHANNEL FOR MAINTENANCE	2-10
allocOperandList	2-13
ALTER DATABASE	2-16
archiveLogRecordSpecifier	2-18
BACKUP	2-22
BLOCKRECOVER	2-48
CATALOG	2-51
CHANGE	2-54
cmdLine	2-57

completedTimeSpec.....	2-61
CONFIGURE.....	2-63
CONNECT	2-75
connectStringSpec	2-77
CONVERT	2-79
CREATE CATALOG.....	2-89
CREATE SCRIPT	2-91
CROSSCHECK	2-93
datafileSpec	2-95
DELETE	2-96
DELETE SCRIPT	2-100
deviceSpecifier.....	2-101
DROP CATALOG.....	2-102
DROP DATABASE.....	2-103
DUPLICATE	2-104
EXECUTE SCRIPT.....	2-112
EXIT	2-113
fileNameConversionSpec	2-114
FLASHBACK.....	2-116
formatSpec	2-121
HOST	2-124
keepOption	2-125
LIST	2-127
listObjList	2-143
maintQualifier	2-145
maintSpec.....	2-147
obsOperandList	2-149
PRINT SCRIPT	2-150
QUIT	2-151
recordSpec.....	2-152
RECOVER.....	2-154
REGISTER	2-163
RELEASE CHANNEL.....	2-165
releaseForMaint	2-166
REPLACE SCRIPT.....	2-167
REPORT	2-169
RESET DATABASE.....	2-175
RESTORE.....	2-178
RESYNC	2-192
RUN.....	2-194
SEND	2-197
SET	2-198
SHOW.....	2-206
SHUTDOWN	2-209
SPOOL	2-211
SQL.....	2-212
STARTUP	2-214

SWITCH	2-216
tempfileSpec	2-219
TRANSPORT TABLESPACE	2-221
UNREGISTER DATABASE	2-225
untilClause	2-227
UPGRADE CATALOG	2-229
VALIDATE	2-230

3 Recovery Catalog Views

Summary of RMAN Recovery Catalog Views	3-1
RC_ARCHIVED_LOG	3-4
RC_BACKUP_ARCHIVELOG_DETAILS	3-6
RC_BACKUP_ARCHIVELOG_SUMMARY	3-7
RC_BACKUP_CONTROLFILE	3-8
RC_BACKUP_CONTROLFILE_DETAILS	3-10
RC_BACKUP_CONTROLFILE_SUMMARY	3-11
RC_BACKUP_COPY_DETAILS	3-12
RC_BACKUP_COPY_SUMMARY	3-13
RC_BACKUP_CORRUPTION	3-14
RC_BACKUP_DATAFILE	3-15
RC_BACKUP_DATAFILE_DETAILS	3-17
RC_BACKUP_DATAFILE_SUMMARY	3-18
RC_BACKUP_FILES	3-19
RC_BACKUP_PIECE	3-22
RC_BACKUP_PIECE_DETAILS	3-23
RC_BACKUP_REDOLOG	3-25
RC_BACKUP_SET	3-27
RC_BACKUP_SET_DETAILS	3-29
RC_BACKUP_SET_SUMMARY	3-31
RC_BACKUP_SPFILE	3-32
RC_BACKUP_SPFILE_DETAILS	3-33
RC_BACKUP_SPFILE_SUMMARY	3-34
RC_CHECKPOINT	3-35
RC_CONTROLFILE_COPY	3-36
RC_COPY_CORRUPTION	3-37
RC_DATABASE	3-38
RC_DATABASE_BLOCK_CORRUPTION	3-39
RC_DATABASE_INCARNATION	3-40
RC_DATAFILE	3-41
RC_DATAFILE_COPY	3-42
RC_LOG_HISTORY	3-44
RC_OFFLINE_RANGE	3-45
RC_RMAN_OUTPUT	3-46
RC_PROXY_ARCHIVEDLOG	3-47
RC_PROXY_ARCHIVELOG_DETAILS	3-48
RC_PROXY_ARCHIVELOG_SUMMARY	3-49
RC_PROXY_CONTROLFILE	3-50

RC_PROXY_COPY_DETAILS	3-52
RC_PROXY_COPY_SUMMARY	3-53
RC_PROXY_DATAFILE	3-54
RC_REDO_LOG	3-56
RC_REDO_THREAD	3-57
RC_RESYNC	3-58
RC_RMAN_BACKUP_JOB_DETAILS	3-59
RC_RMAN_BACKUP_SUBJOB_DETAILS	3-61
RC_RMAN_BACKUP_TYPE	3-62
RC_RMAN_CONFIGURATION	3-63
RC_RMAN_STATUS	3-64
RC_STORED_SCRIPT	3-66
RC_STORED_SCRIPT_LINE.....	3-67
RC_TABLESPACE	3-68
RC_TEMPFILE	3-69
RC_UNUSABLE_BACKUPFILE_DETAILS.....	3-70

A Deprecated RMAN Commands

B RMAN Compatibility

About RMAN Compatibility	B-1
Determination of 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

Send Us Your Comments

Oracle Database Recovery Manager Reference, 10g Release 2 (10.2) B14194-01

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

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

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

- Electronic mail: infodev_us@oracle.com
- FAX: 650-506-7227 Attn: Server Technologies Documentation Manager
- Postal service:

Oracle Corporation
Server Technologies Documentation
500 Oracle Parkway, Mailstop 4op11
Redwood Shores, CA 94065
USA

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

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

Preface

This preface contains these topics:

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

Audience

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

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

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

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

Organization

This document contains:

Chapter 1, "About RMAN Commands"

This chapter describes the basic conventions of RMAN syntax.

Chapter 2, "RMAN Commands"

This chapter displays the RMAN syntax diagrams, describes the elements of the syntax, and provides examples.

Chapter 3, "Recovery Catalog Views"

This chapter describes the recovery catalog views.

Appendix A, "Deprecated RMAN Commands"

This appendix describes RMAN syntax that is deprecated (that is, no longer supported) but still functional.

Appendix B, "RMAN Compatibility"

This appendix shows the compatible combinations of the RMAN client, target database, recovery catalog database, and recovery catalog schema.

Related Documentation

For more information, see these Oracle resources:

- *Oracle Database Backup and Recovery Basics*
- *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.

Printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

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

<http://www.oracle.com/technology/membership/>

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

<http://www.oracle.com/technology/documentation/>

Conventions

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

- [Conventions in Text](#)
- [Conventions in Code Examples](#)

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an index-organized table .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle Database Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> reside on the same disk.

Convention	Meaning	Example
UPPERCASE monospace (fixed-width) font	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a NUMBER column. You can back up the database by using the BACKUP command. Query the TABLE_NAME column in the USER_TABLES data dictionary view. Use the DBMS_STATS.GENERATE_STATS procedure.
lowercase monospace (fixed-width) font	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter sqlplus to open SQL*Plus. The password is specified in the orapwd file. Back up the datafiles and control files in the /disk1/oracle/dbs directory. The department_id, department_name, and location_id columns are in the hr.departments table. Set the QUERY_REWRITE_ENABLED initialization parameter to true. Connect as oe user. The JRepUtil class implements these methods.
lowercase italic monospace (fixed-width) font	Lowercase italic monospace font represents placeholders or variables.	You can specify the <i>parallel_clause</i> . Run <i>Uold_release</i> .SQL where <i>old_release</i> refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[]	Brackets enclose one or more optional items. Do not enter the brackets.	DECIMAL (<i>digits</i> [, <i>precision</i>])
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	{ENABLE DISABLE}
	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	{ENABLE DISABLE} [COMPRESS NOCOMPRESS]
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> That we have omitted parts of the code that are not directly related to the example That you can repeat a portion of the code 	CREATE TABLE ... AS <i>subquery</i> ; SELECT <i>col1</i> , <i>col2</i> , ... , <i>coln</i> FROM employees;

Convention	Meaning	Example
.	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	<pre>SQL> SELECT NAME FROM V\$DATAFILE; NAME ----- /fsl/dbs/tbs_01.dbf /fsl/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.</pre>
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	<pre>acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/system_password DB_NAME = database_name</pre>
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

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.

About RMAN Commands

This chapter describes the basic elements of RMAN syntax. It includes the following sections:

- [Conventions Used in this Reference](#)
- [RMAN Command Entries](#)

Conventions Used in this Reference

This section explains the conventions used in this chapter including:

- [RMAN Text Conventions](#)
- [RMAN Command Format, Syntax Diagrams and Notation](#)
- [RMAN Code Examples](#)

RMAN Text 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.

RMAN Command Format, Syntax Diagrams and Notation

This section describes the conventions for RMAN command syntax.

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. A command can span multiple lines.

A comment can be inserted by using a `#` character at any point in a line. After the `#` character, the remainder of the line is ignored. For example:

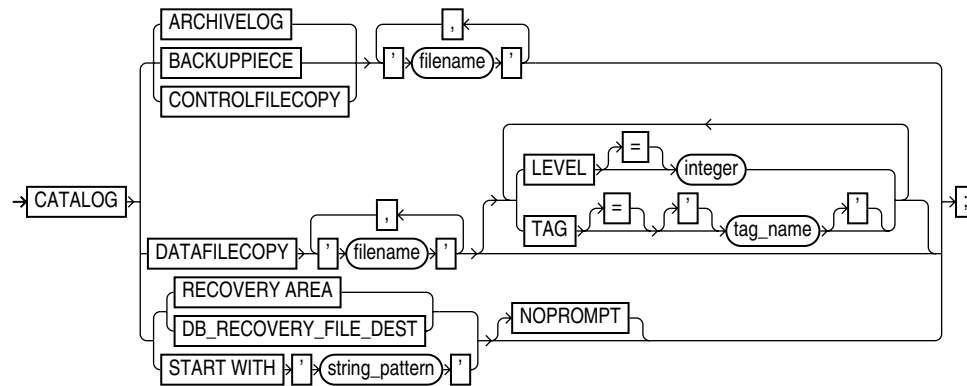
```
# run this command once each day
BACKUP INCREMENTAL LEVEL 1
    FOR RECOVER OF COPY          # using incrementally updated backups
```

```
WITH TAG "DAILY_BACKUP" # daily backup routine
DATABASE;
```

Syntax Diagrams

This reference uses syntax diagrams to show Recovery Manager commands. These syntax diagrams use lines and arrows to show syntactic structure, as shown in the following example for the RMAN CATALOG command.

lcatalog::=



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](#)
- [Placeholders](#)

Keywords 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 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.

Some placeholder values are 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 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.

The only system-independent, legal environment variables in RMAN quoted strings are ? for the Oracle home and @ 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 <i>'filename', 'tablespace_name', 'channel_name', 'channel_parms'</i>	A string of characters contained in either single or double quotes. A quoted string may contain white space, punctuation, and RMAN and SQL keywords.	" ? /dbs/cf.f " 'dev1'
Nonquoted strings such as <i>channel_id, tag_name, date_string</i>	A sequence of characters containing no white space and no punctuation characters and starting with an alphabetic character.	ch1
<i>integer</i>	Any sequence of only number characters.	67843

RMAN Reserved Words

The RMAN language contains a number of reserved words, which are or have been in the past used in RMAN commands. The following table lists all of the current reserved words.

Reserved Word	Reserved Word	Reserved Word	Reserved Word	Reserved Word	Reserved Word
ABORT	CONSISTENT	FORMAT	MAXPIECESIZE	PLSQL	SETLIMIT
AFFINITY	CONTROLFILE	FROM	MAXSEQ	PLUS	SETSIZE
AFTER	CONTROLFILECOPY	FULL	MAXSETSIZE	POLICY	SHOW
ALL	COPIES	G	MAXSIZE	POOL	SHUTDOWN
ALLOCATE	COPY	GET	MISC	PRINT	SINCE
ALTER	CORRUPTION	GROUP	MOUNT	PROXY	SIZE
AND	CREATE	HIGH	MSGLOG	PUT	SKIP
APPEND	CROSSCHECK	HOST	MSGNO	QUIT	SLAXDEBUG
ARCHIVELOG	CUMULATIVE	ID	NAME	RATE	SNAPSHOT
AT	CURRENT	IDENTIFIER	NEED	RCVCAT	SPFILE
ATALL	DATABASE	IMMEDIATE	NEW	RCVMAN	SPOOL
AUTOBACKUP	DATAFILE	INACCESSIBLE	NEW-LINE	READONLY	SQL

Reserved Word	Reserved Word	Reserved Word	Reserved Word	Reserved Word	Reserved Word
AUTOLOCATE	DATAFILECOPY	INCARNATION	NEWNAME	READRATE	STANDBY
AUXILIARY	DAYS	INCLUDE	NOCATALOG	RECOVER	STARTUP
AUXNAME	DBA	INCREMENTAL	NOCFAU	RECOVERABLE	STEP
AVAILABLE	DBID	INPUT	NOCHECKSUM	RECOVERY	SUMMARY
BACKED	DEBUG	IO	NOEXCLUDE	REDUNDANCY	SWITCH
BACKUP	DEFAULT	JOB	NOFILENAMECHECK	REGISTER	TABLESPACE
BACKUPPIECE	DEFINE	K	NOFILEUPDATE	RELEASE	TAG
BACKUPSET	DELETE	KBYTES	NOKEEP	RELOAD	TARGET
BEFORE	DESTINATION	KEEP	NOLOGS	REMOVE	TEST
BETWEEN	DEVICE	LEVEL	NOMOUNT	RENORMALIZE	THREAD
BLOCK	DISK	LIBNAME	NONE	REPLACE	TIME
BLOCKRECOVER	DISKRATIO	LIBPARM	NOPROMPT	REPLICATE	TIMEOUT
BLOCKS	DISPLAY	LIBRARY	NOREDO	REPORT	TIMES
BY	DORECOVER	LIBTEXT	NORMAL	RESET	TO
CANCEL	DROP	LIKE	NOT	RESETLOGS	TRACE
CATALOG	DUMP	LIMIT	NULL	RESTART	TRANSACTIONAL
CHANGE	DUPLEX	LIST	OBSOLETE	RESTORE	TXT
CHANNEL	DUPLICATE	LOG	OF	RESYNC	TYPE
CHARSET	ECHO	LOGFILE	OFF	RETENTION	UNAVAILABLE
CHECK	EXCLUDE	LOGICAL	OFFLINE	REUSE	UNCATALOG
CLEAR	EXECUTE	LOGS	ON	RPC	UNLIMITED
CLONE	EXIT	LOGSCN	ONLY	RPCTEST	UNRECOVERABLE
CLONENAME	EXPIRED	LOGSEQ	OPEN	RUN	UNTIL
CLONE_CF	FILE	LOW	OPTIMIZATION	SAVE	UNUSED
CMDFILE	FILES	M	ORPHAN	SCHEMA	UP
COMMAND	FILESPERSET	MAINTENANCE	PACKAGES	SCN	UPGRADE
COMPATIBLE	FINAL	MASK	PARALLELISM		VALIDATE
COMPLETED	FOR	MAXCORRUPT	PARMS	SEND	VERBOSE
CONFIGURE	FORCE	MAXDAYS	PFILE	SEQUENCE	WINDOW
CONNECT	FOREVER	MAXOPENFILES	PIPE	SET	

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), surround it with quotes. Otherwise, RMAN cannot parse your command correctly and generates an error. These are examples of correct and incorrect commands that use RMAN reserved words with quotes:

```
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, you should avoid using reserved words in ways that conflict with their primary meaning in the RMAN command language.

RMAN Command Entries

The description of each command or subclause contains the following sections:

Table 1–1 Sections of a Command Description

Section	Content
Syntax	Shows the keywords and parameters that make up the statement. Note: Not all keywords and parameters are valid in all circumstances. Be sure to refer to the "Keywords and Parameters" section of each statement to learn about any restrictions on the syntax.
Purpose	Describes the basic uses of the statement.
Restrictions and Usage Notes	Lists requirements, restrictions, and guidelines for proper use of the command.
Keywords and Parameters	Describes the purpose of each keyword and parameter. Restrictions and usage notes can also appear in this section.
Examples	Shows how to use various clauses and options of the statement.

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

RMAN Code Examples

The command entries in this reference include many examples of RMAN commands, which illustrate how to use the different elements of each RMAN command for common tasks. These examples are generally set off from the text and appear in a monospace font, as in the following example:

```
BACKUP DATABASE;
```


RMAN Commands

This chapter describes, in alphabetical order, Recovery Manager commands and subclauses. For a summary of the RMAN commands and command-line options, refer to ["Summary of RMAN Commands"](#) on page 2-1.

Summary of RMAN Commands

[Table 2-1](#) 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.

For command line options for the RMAN client, refer to ["cmdLine"](#) on page 2-57.

Table 2-1 Recovery Manager Commands

Command	Purpose
"@" on page 2-5	Run a command file.
"@@" on page 2-6	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.
"ALLOCATE CHANNEL" on page 2-7	Establish a channel, which is a connection between RMAN and a database instance.
"ALLOCATE CHANNEL FOR MAINTENANCE" on page 2-10	Allocate a channel in preparation for issuing maintenance commands such as DELETE .
"allocOperandList" on page 2-13	A subclause that specifies channel control options such as PARMS and FORMAT.
"ALTER DATABASE" on page 2-16	Mount or open a database.
"archivelogRecordSpecifier" on page 2-18	Specify a range of archived redo logs files.
"BACKUP" on page 2-22	Back up database files, copies of database files, archived logs, or backup sets.
"BLOCKRECOVER" on page 2-48	Recover an individual data block or set of data blocks within one or more datafiles.
"CATALOG" on page 2-51	Add information about file copies and user-managed backups to the repository.
"CHANGE" on page 2-54	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.
"completedTimeSpec" on page 2-61	Specify a time range during which the backup or copy completed.

Table 2–1 (Cont.) Recovery Manager Commands

Command	Purpose
"CONFIGURE" on page 2-63	Configure persistent RMAN settings. These settings apply to all RMAN sessions until explicitly changed or disabled.
"CONNECT" on page 2-75	Establish a connection between RMAN and a target, auxiliary, or recovery catalog database.
"connectStringSpec" on page 2-77	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.
"CONVERT" on page 2-79	Converts datafile formats for transporting tablespaces and databases across platforms.
"CREATE CATALOG" on page 2-89	Create the schema for the recovery catalog.
"CREATE SCRIPT" on page 2-91	Create a stored script and store it in the recovery catalog.
"CROSSCHECK" on page 2-93	Determine whether files managed by RMAN, such as archived logs, datafile copies, and backup pieces, still exist on disk or tape.
"datafileSpec" on page 2-95	Specify a datafile by filename or absolute file number.
"DELETE" on page 2-96	Delete backups and copies, remove references to them from the recovery catalog, and update their control file records to status <code>DELETED</code> .
"DELETE SCRIPT" on page 2-100	Delete a stored script from the recovery catalog.
"deviceSpecifier" on page 2-101	Specify the type of storage device for a backup or copy.
"DROP CATALOG" on page 2-102	Remove the schema from the recovery catalog.
"DROP DATABASE" on page 2-103	Deletes the target database from disk and unregisters it.
"DUPLICATE" on page 2-104	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 2-112	Run an RMAN stored script.
"EXIT" on page 2-113	Quit the RMAN executable.
"fileNameConversionSpec" on page 2-114	Specify patterns to transform source to target filenames during <code>BACKUP AS COPY</code> , <code>CONVERT</code> and <code>DUPLICATE</code> .
"FLASHBACK" on page 2-116	Returns the database to its state at a previous time or SCN.
"formatSpec" on page 2-121	Specify a filename format for a backup or copy.
"HOST" on page 2-124	Invoke an operating system command-line subshell from within RMAN or run a specific operating system command.
"keepOption" on page 2-125	Specify that a backup or copy should or should not be exempt from the current retention policy.
"LIST" on page 2-127	Produce a detailed listing of backup sets or copies.
"listObjList" on page 2-143	A subclause used to specify which items will be displayed by the <code>LIST</code> command.
"maintQualifier" on page 2-145	A subclause used to specify additional options for maintenance commands such as <code>DELETE</code> and <code>CHANGE</code> .
"maintSpec" on page 2-147	A subclause used to specify the files operated on by maintenance commands such as <code>CHANGE</code> , <code>CROSSCHECK</code> , and <code>DELETE</code> .
"obsOperandList" on page 2-149	A subclause used to determine which backups and copies are obsolete.

Table 2–1 (Cont.) Recovery Manager Commands

Command	Purpose
"PRINT SCRIPT" on page 2-150	Display a stored script.
"QUIT" on page 2-151	Exit the RMAN executable.
"recordSpec" on page 2-152	A subclause used to specify which objects the maintenance commands should operate on.
"RECOVER" on page 2-154	Apply redo logs and incremental backups to datafiles restored from backup or datafile copies, in order to update them to a specified time.
"REGISTER" on page 2-163	Register the target database in the recovery catalog.
"RELEASE CHANNEL" on page 2-165	Release a channel that was allocated with an ALLOCATE CHANNEL command.
"releaseForMaint" on page 2-166	Release a channel allocated with an ALLOCATE CHANNEL FOR MAINTENANCE command.
"REPLACE SCRIPT" on page 2-167	Replace an existing script stored in the recovery catalog. If the script does not exist, then REPLACE SCRIPT creates it.
"REPORT" on page 2-169	Perform detailed analyses of the content of the recovery catalog.
"RESET DATABASE" on page 2-175	Inform RMAN that the SQL statement ALTER DATABASE OPEN RESETLOGS has been executed and that a new incarnation of the target database has been created, or reset the target database to a prior incarnation.
"RESTORE" on page 2-178	Restore files from backup sets or from disk copies to the default or a new location.
"RESYNC" on page 2-192	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.
"RUN" on page 2-194	Execute a sequence of one or more RMAN commands, which are one or more statements executed within the braces of RUN.
"SEND" on page 2-197	Send a vendor-specific quoted string to one or more specific channels.
"SET" on page 2-198	Sets the value of various attributes that affect RMAN behavior for the duration of a RUN block or a session.
"SHOW" on page 2-206	Displays the current CONFIGURE settings.
"SHUTDOWN" on page 2-209	Shut down the target database. This command is equivalent to the SQL*Plus SHUTDOWN command.
"SPOOL" on page 2-211	Write RMAN output to a log file.
"SQL" on page 2-212	Execute a SQL statement from within Recovery Manager.
"STARTUP" on page 2-214	Start up the target database. This command is equivalent to the SQL*Plus STARTUP command.
"SWITCH" on page 2-216	Specify that a datafile copy is now the current datafile , that is, the datafile pointed to by the control file. This command is equivalent to the SQL statement ALTER DATABASE RENAME FILE as it applies to datafiles.
"tempfileSpec" on page 2-219	Specifies a tempfile by path or by file number.
"TRANSPORT TABLESPACE" on page 2-221	Creates transportable tablespace sets from backup for one or more tablespaces.
"UNREGISTER DATABASE" on page 2-225	Unregisters a database from the recovery catalog.

Table 2–1 (Cont.) Recovery Manager Commands

Command	Purpose
untilClause on page 2-227	A subclause specifying 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.
"UPGRADE CATALOG" on page 2-229	Upgrade the recovery catalog schema from an older version to the version required by the RMAN executable.
"VALIDATE" on page 2-230	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.

@

Syntax

at::=

→ @ filename →

Purpose

To execute a series of RMAN commands stored in an operating system file with the specified full path name, for example, `@/oracle/dbs/cmd/cmd1.rman`. If you do not specify the full path name, then the current working directory is assumed, for example, `@cmd1.rman`. Do not use quotes around the string or leave whitespace between the @ and filename. RMAN processes the specified file as if its contents had appeared in place of the @ command.

Note: The file must contain complete RMAN commands; partial commands generate syntax errors.

Restrictions and Usage Notes

None.

Example

Running a Command File from the Command Line: Example This example creates a command file and then runs it from the operating system command line:

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

Running a Command File Within RMAN: Example This example runs a command file from the RMAN prompt and from within a RUN command:

```
@backup_db.rman
RUN { @backup_db.rman }
```

@@

Syntax

atat::=

> @@ > filename :

Purpose

To execute a series of RMAN commands stored in an operating system file with the specified filename, for example, `@@cmd2.rman`. 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. For example, assume that you invoke RMAN as follows:

```
% rman @$ORACLE_HOME/rdbms/admin/dba/scripts/cmd1.rman
```

Assume that the command `@@cmd2.rman` appears inside the `cmd1.rman` script. In this case, the `@@` command directs RMAN to look for the file `cmd2.rman` in the directory `$ORACLE_HOME/rdbms/admin/dba/scripts/`. Note that the file must contain complete RMAN commands.

Restrictions and Usage Notes

None.

Example

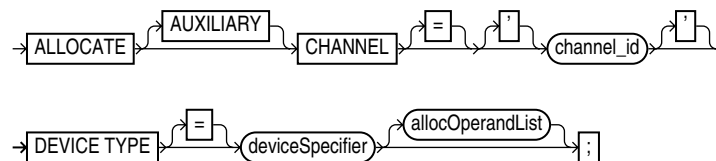
Calling a Command File Within Another Command File: Example Assume that you create command files called `backup_logs.rman` and `backup_db.rman` as in the following example. Then, you execute `bkup_db.rman` from the command line, which specifies that RMAN should look for the `bkup_logs.rman` script in the Oracle home directory:

```
echo "BACKUP ARCHIVELOG ALL;" > $ORACLE_HOME/bkup_logs.rman
echo "BACKUP DATABASE;" > $ORACLE_HOME/bkup_db.rman
echo "@@bkup_logs.rman" >> $ORACLE_HOME/bkup_db.rman
rman TARGET / @$ORACLE_HOME/bkup_db.rman
```

ALLOCATE CHANNEL

Syntax

allocate::=



Purpose

To manually allocate a **channel**, which is a connection between RMAN and a database instance. Each connection initiates an database server session on the target or auxiliary instance: this server session performs the work of backing up, restoring, or recovering RMAN backups.

Manually allocated channels (allocated by using `ALLOCATE`) should be distinguished from automatically allocated channels (specified by using `CONFIGURE`). Manually allocated channels apply only to the `RUN` job in which you issue the command. Automatic channels apply to any RMAN job in which you do *not* manually allocate channels. You can always override automatic channel configurations by manually allocating channels within a `RUN` command.

Each channel operates on one backup set or image copy at a time. RMAN automatically releases the channel at the end of the job.

You can control the degree of parallelism within a job by allocating the desired number of channels. Allocating multiple channels simultaneously allows a single job to read or write multiple backup sets or disk copies in parallel. If you establish multiple connections, then each connection operates on a separate backup set or disk copy.

Whether `ALLOCATE CHANNEL` causes operating system resources to be allocated 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.

Note: When you specify `DEVICE TYPE DISK`, no operating system resources are allocated other than for the creation of the server session.

Restrictions and Usage Notes

- You can only use `ALLOCATE CHANNEL` within a `RUN` block.
- The target instance must be started.
- You cannot make a connection to a shared server session.
- You must either allocate a channel manually or configure a channel for automatic allocation before executing a `BACKUP`, `DUPLICATE`, `CREATE CATALOG`, `RESTORE`, `RECOVER`, or `VALIDATE` command.
- You cannot use `BACKUP DEVICE TYPE` or `RESTORE DEVICE TYPE` to use automatic channels after specifying manual channels with `ALLOCATE CHANNEL`.

- You must use a recovery catalog when backing up a standby database.
- You cannot prefix `ORA_` to a channel name. RMAN reserves channel names beginning with the `ORA_` prefix for its own use.

Keywords and Parameters

Syntax Element	Description
AUXILIARY	Specifies a connection between RMAN and an auxiliary database instance. An auxiliary instance is used when executing the <code>DUPLICATE</code> or <code>TRANSPORT TABLESPACE</code> command, and when performing <code>TSPITR</code> with <code>RECOVER TABLESPACE</code> . An auxiliary database can reside on the same host as its parent or on a different host. When specifying this option, the auxiliary database must be mounted but not open. See Also: " DUPLICATE " on page 2-104 to learn how to duplicate a database, and " CONNECT " on page 2-75 to learn how to connect to a duplicate database
CHANNEL ' <i>channel_id</i> '	Specifies a connection between RMAN and the target database instance. Each connection initiates a server session on the database instance: this server session performs the work of backing up, restoring, and recovering backups and copies. Specify a channel id, which is the case-sensitive name of the channel, after the <code>CHANNEL</code> keyword. The database uses the <i>channel_id</i> to report I/O errors.
DEVICE TYPE = <i>deviceSpecifier</i>	Specifies the type of storage device. See Also: " deviceSpecifier " on page 2-101 Note: If you do not specify the <code>DEVICE TYPE</code> parameter, then you must specify the <code>NAME</code> parameter to identify a particular sequential I/O device. Query the <code>V\$BACKUP_DEVICE</code> view for information about available device types and names.
allocOperandList	Specifies control options for the allocated channel. See Also: " allocOperandList " on page 2-13

Examples

Allocating a Single Channel for a Backup: Example This command allocates a tape channel for a whole database and archived redo log backup:

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

Spreading a Backup Across Multiple Disks: Example 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 filenames are on different disks:

```
RUN
{
  ALLOCATE CHANNEL disk1 DEVICE TYPE DISK FORMAT '/disk1/backups/%U';
  ALLOCATE CHANNEL disk2 DEVICE TYPE DISK FORMAT '/disk2/backups/%U';
  BACKUP DATABASE PLUS ARCHIVELOG; # AS COPY is default when backing up to disk
}
```

Creating Multiple Copies of a Backup: Example When creating multiple copies of a backup, you can specify the [SET BACKUP COPIES](#) command. The following example

generates a single backup of the database to disk, and then creates two identical backups of datafile 1 to two different file systems:

```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE DISK MAXPIECESIZE 5M;
  BACKUP DATABASE PLUS ARCHIVELOG; # AS COPY is the default, so RMAN creates image copies
  SET BACKUP COPIES = 2;
  BACKUP DATAFILE 1 FORMAT '/disk1/backups/%U', '/disk2/backups/%U';
}
```

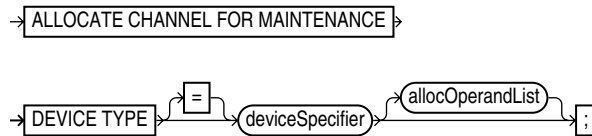
Allocating an Auxiliary Channel for Database Duplication: Example When creating a duplicate database, allocate a channel by using the AUXILIARY option:

```
RUN
{
  ALLOCATE AUXILIARY CHANNEL c1 DEVICE TYPE sbt;
  ALLOCATE AUXILIARY CHANNEL c2 DEVICE TYPE sbt;
  DUPLICATE TARGET DATABASE TO ndbnewh
    LOGFILE
      '?/oradata/aux1/redo01.log' SIZE 200K,
      '?/oradata/aux1/redo02.log' SIZE 200K
      '?/oradata/aux1/redo03.log' SIZE 200K
  SKIP READONLY
  NOFILENAMECHECK;
}
```

ALLOCATE CHANNEL FOR MAINTENANCE

Syntax

allocateForMaint::=



Purpose

To manually allocate a channel in preparation for issuing a [CHANGE](#), [DELETE](#), or [CROSSCHECK](#) command.

Note: If you [CONFIGURE](#) at least one channel for each device type you use in your configuration, then you do not need to use `ALLOCATE CHANNEL FOR MAINTENANCE` at all.

Oracle recommends that you use configured channels instead of maintenance channels. Configured channels can be used for all RMAN I/O to the specified device, not just the maintenance tasks supported by maintenance channels, and they persist across RMAN sessions.

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt. This command cannot be used within a RUN block.
- The target instance must be started.
- Do not specify a channel ID.
- You cannot allocate a maintenance channel to a shared session.
- You cannot prefix `ORA_` to a channel name. RMAN reserves channel names beginning with the `ORA_` prefix for its own use.
- Manually allocated channels (maintenance or normal channels) and automatic channels based on configured settings are never mixed. To perform maintenance on both disk and SBT simultaneously using manually allocated maintenance channels, you must allocate both SBT and DISK channels explicitly.
- If you use `ALLOCATE CHANNEL FOR MAINTENANCE`, then RMAN uses the following convention for channel naming: `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
```
- You can allocate multiple maintenance channels for a single job, but you should only use this feature in these situations:
 - To allow crosschecking or deletion of all backup pieces or proxy copies, both on disk and tape, with a single command

- To make crosschecking and deleting work correctly in an Oracle Real Application Clusters configuration in which each backup piece or proxy copy exists only on one node

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

Keywords and Parameters

Syntax Element	Description
DEVICE TYPE = <i>deviceSpecifier</i>	Specifies the type of storage device. See Also: " deviceSpecifier " on page 2-101 Note: If you do not specify the DEVICE TYPE parameter, then you must specify the NAME parameter to identify a particular sequential I/O device. Query the V\$BACKUP_DEVICE view for information about available device types and names.
<i>allocOperandList</i>	Specifies control options for the allocated channel. See Also: " allocOperandList " on page 2-13

Examples

Deleting a Backup Set: Example This example deletes backup sets from tape created more than a week ago:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
DELETE NOPROMPT BACKUP OF DATABASE COMPLETED BEFORE 'SYSDATE-7';
```

Crosschecking Archived Logs: Example This example crosschecks all archived logs on disk and tape. For disk, the preconfigured disk channel is used; for tape, an SBT channel is allocated manually. If the logs are not found, then RMAN marks them as EXPIRED in the repository:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK ARCHIVELOG ALL;
```

Crosschecking on Multiple Nodes of an Oracle Real Application Clusters

Configuration: Example In this example, you perform a crosscheck of backups on two nodes of an Oracle Real Application Clusters configuration, where each node has access to a subset of backups. It is assumed here 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 CONNECT 'SYS/change_on_install@inst1';
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK CONNECT 'SYS/change_on_install@inst2';
CROSSCHECK BACKUP;
```

Note: Oracle recommends that all nodes in a Real Application Clusters configuration have the same access to all backups on all storage devices.

Deleting on Disk and sbt Channels with One Command: Example In this example, you delete a backup from both disk and tape:

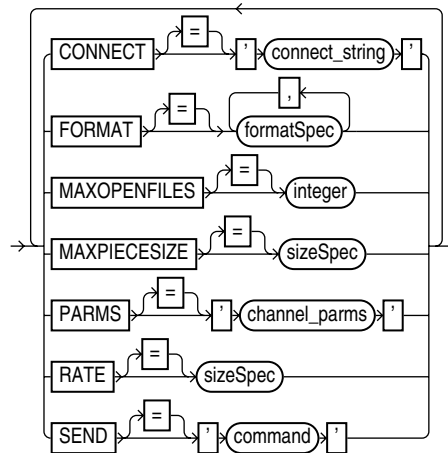
```
# back up datafile to disk and tape
BACKUP DEVICE TYPE DISK DATAFILE 1 TAG "weekly_bkup";
BACKUP DEVICE TYPE sbt DATAFILE 1 TAG "weekly_bkup";
```

```
# manually allocate gsbt channel and disk channel
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE DISK;
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
DELETE BACKUPSET TAG "weekly_bkup";
```


allocOperandList

Syntax

allocOperandList::=



Purpose

A subclause specifying control options on a **channel**, which is a connection between RMAN and a database instance. Specify this clause on the following commands:

- [ALLOCATE CHANNEL](#)
- [ALLOCATE CHANNEL FOR MAINTENANCE](#)
- [CONFIGURE](#)

Keywords and Parameters

Syntax Element	Description
CONNECT = connectStringSpec	<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 of backup or restore operations across different instances in an Oracle Real Application Clusters configuration.</p> <p>If you do not specify this parameter, and if you did not specify the <code>AUXILIARY</code> option, then RMAN conducts all operations on the target database instance specified by the command-line <code>CONNECT</code> parameter or the instance connected to when you issued the <code>CONNECT</code> command. Typically, you should not use the <code>CONNECT</code> parameter in conjunction with the <code>AUXILIARY</code> option.</p> <p>See Also: "connectStringSpec" on page 2-77 and "cmdLine" on page 2-57</p>

Syntax Element	Description
FORMAT = <i>formatSpec</i>	<p>Specifies the format to use for the names of backup pieces that are created on this channel.</p> <p>If you do not specify FORMAT, RMAN uses %U by default, which guarantees a unique identifier. If the flash recovery area is configured, then the files are created in the default disk location. Otherwise, the default disk location is operating system specific (for example, ?/dbs on Solaris).</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 machine and the target database runs on a UNIX machine, then the FORMAT string must adhere to the naming conventions of a UNIX file system or raw device.</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>This parameter is useful if you allocate multiple disk channels and want each channel to write to a different directory. 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>See Also: "formatSpec" on page 2-121 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>integer</i>	<p>Specifies the maximum size of each backup piece created on this channel. 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>
PARMS = ' <i>channel_parms</i> '	<p>Specifies parameters for the device to allocate. Do not use this port-specific string if you have specified DEVICE TYPE DISK.</p> <p>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> to learn how to integrate media management libraries</p>
'ENV=(<i>var1=val1</i> , <i>var2=val2</i> ,...)'	<p>Specifies one or more environment variables required by the media management vendor in the server session corresponding to this RMAN client. Because RMAN is a client program, the ENV parameter can be used to set server session specific variables that perform backup and restore operations on behalf of the RMAN client. For example:</p> <pre>PARMS="ENV=(TAPE_SERVER=srv1)"</pre>
'SBT_LIBRARY = <i>lib_name</i> '	<p>Specifies which media library should be used on this sbt channel. The default library is operating system specific (for example, libobk.so in the Solaris Operating Environment and ORASBT.DLL on Windows NT). For example:</p> <pre>PARMS="SBT_LIBRARY=/oracle/lib/mmvs.so"</pre>
RATE = <i>integer</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.</p> <p>See Also: Your media manager documentation to see whether this feature is supported and when it should be used.</p>

Examples

Configuring an Automatic Channel: Example This example configures a persistent disk channel:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT = '?/oradata/bkup_%U';
```

Configuring a Channel for a Backup: Example This example manually allocates an sbt channel and then runs a whole database backup:

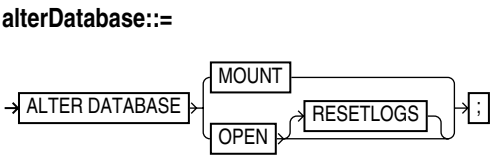
```
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt MAXPIECESIZE 800M;
  BACKUP DATABASE;
}
```

Allocating a Channel for a Backup: Example This example configures a default media management library, then makes a database backup by using this library. Then, the example backs up the database again using a different library, then finally makes a third backup using the default library:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS="SBT_LIBRARY=/mediavendor/lib/mm_lib1.so";
BACKUP DATABASE;
RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt PARMS="SBT_LIBRARY=/mediavendor/lib/mm_lib2.so";
  BACKUP DATABASE;
}
BACKUP ARCHIVELOG ALL;
```

ALTER DATABASE

Syntax



Purpose

To mount or open a database.

See Also: *Oracle Database SQL Reference* for ALTER DATABASE syntax

Restrictions and Usage Notes

- Execute this command either within the braces of a [RUN](#) command or at the RMAN prompt.
- The target instance must be started.

Keywords and Parameters

Syntax Element	Description
MOUNT	Mounts the database without opening it. This option is equivalent to the SQL statement ALTER DATABASE MOUNT.
OPEN	Opens the database.
RESETLOGS	<p>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 RESETLOGS option is equivalent to the SQL statement ALTER DATABASE OPEN RESETLOGS.</p> <p>If you use a recovery catalog, then RMAN performs an implicit RESET DATABASE after the database is opened to make this new incarnation the current one in the catalog. If you execute the SQL statement ALTER DATABASE OPEN RESETLOGS (not the RMAN command of the same name), then you must manually run the RESET DATABASE command.</p>

Examples

Opening the Database After a Backup: Example This example mounts the database, takes a whole database backup, then opens the database. At the RMAN prompt enter:

```
STARTUP MOUNT;
BACKUP DATABASE;
# now that the backup is complete, open the database.
ALTER DATABASE OPEN;
```

Mounting the Database After Restoring the Control File: Example To restore the control file to its default location when connected to a recovery catalog, enter the following:

```
STARTUP NOMOUNT;
```

```
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
# you must run the RECOVER command after restoring a control file even if no datafiles
# require recovery
RECOVER DATABASE;
ALTER DATABASE OPEN RESETLOGS;
```

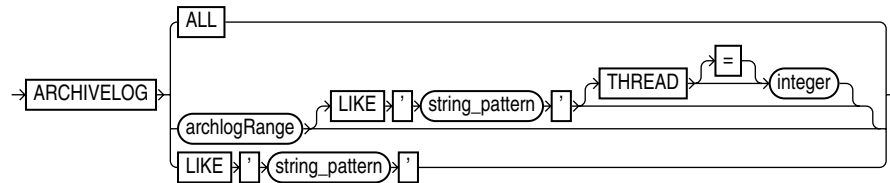
Performing RESETLOGS After Incomplete Recovery: Example This example uses a manually allocated channel to perform incomplete recovery and then resets the online redo logs:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt;
  SET UNTIL SCN 1024;
  RESTORE DATABASE;
  RECOVER DATABASE;
  ALTER DATABASE OPEN RESETLOGS;
}
```

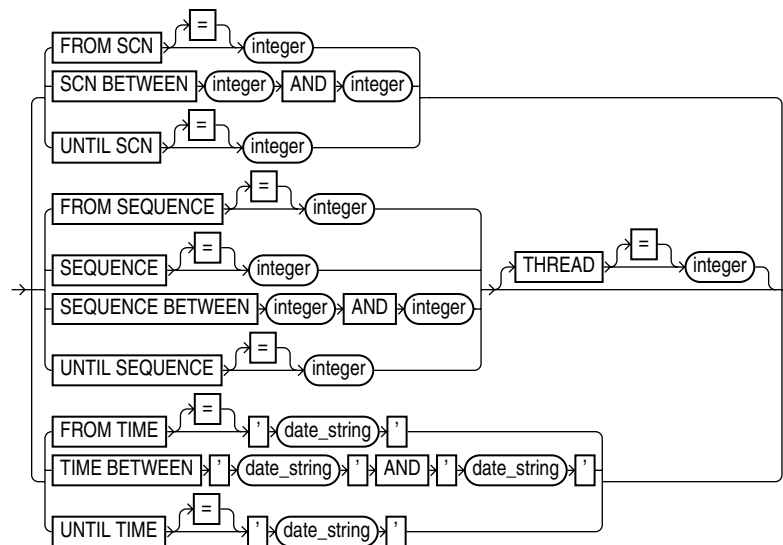
archivelogRecordSpecifier

Syntax

archivelogRecordSpecifier::=



archlogRange::=



Purpose

A subclause used to specify an archived log or range of archived redo logs files for use in backup, restore, recovery, and maintenance operations.

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.

Specifying a range of archived redo logs does not guarantee that RMAN includes all redo data in the range: for example, the last available archived log file may end before the end of the range, or an archived log file in the range may be missing from all archiving destinations. RMAN includes the archived redo logs it finds and does not issue a warning for missing files.

Note: Query the V\$ARCHIVED_LOG view or RC_ARCHIVED_LOG recovery catalog view to determine the time stamps, SCNs, and log sequence numbers for an archived log. For information on how to use the NLS_LANG and NLS_DATE_FORMAT environment variables to specify the format for the time, see *Oracle Database Reference*.

Keywords and Parameters

archivelogRecordSpecifier

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> . The same pattern matching characters that are valid in the LIKE operator in the SQL language can be used to match multiple files. See Also: <i>Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide</i> to learn about using RMAN in a RAC configuration

archlogRange

Syntax Element	Description
FROM SCN = <i>integer</i>	Specifies the beginning SCN for a sequence of archived redo log files. If you do not specify the UNTIL SCN parameter, RMAN will include all available log files beginning with SCN specified in the from SCN parameter.
SCN BETWEEN <i>integer</i> AND <i>integer</i>	Specifies a range of SCNs. SCN BETWEEN <i>integer1</i> AND <i>integer2</i> is exactly equivalent to FROM SCN <i>integer1</i> UNTIL SCN <i>integer2</i> .
UNTIL SCN = <i>integer</i>	Specifies the ending SCN for a sequence of archived redo log files. If you do not specify the FROM SCN parameter, then RMAN will start with the earliest available archived log. If the database is open when you run BACKUP ARCHIVELOG, and if the UNTIL clause is specified, then RMAN does not run ALTER SYSTEM ARCHIVE LOG CURRENT.
FROM SEQUENCE <i>integer</i>	Specifies the beginning log sequence number for a sequence of archived redo log files. If you do not specify the UNTIL SEQUENCE parameter, RMAN will include all available log files beginning with log sequence number specified in the FROM SEQUENCE parameter. Note: You can specify all log sequence numbers in a thread by using the following syntax, where <i>thread_number</i> is an integer referring to the thread: ... ARCHIVELOG FROM SEQUENCE 0 THREAD <i>thread_number</i>
SEQUENCE	Specifies either a single log sequence number or a range of sequence numbers. If the database is open when you run BACKUP ARCHIVELOG, and if the SEQUENCE keyword is specified, then RMAN does not run ALTER SYSTEM ARCHIVE LOG CURRENT.
<i>integer</i>	Specifies a single log sequence number.
BETWEEN <i>integer</i> AND <i>integer</i>	Specifies a range of log sequence numbers. SEQUENCE BETWEEN <i>integer1</i> AND <i>integer2</i> is exactly 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. If you do not specify the FROM SEQUENCE parameter, RMAN uses the lowest available log sequence number to begin the sequence. If the database is open when you run BACKUP ARCHIVELOG, and if the UNTIL clause is specified, then RMAN does not run ALTER SYSTEM ARCHIVE LOG CURRENT.

Syntax Element	Description
FROM TIME = 'date_string'	<p>Specifies the beginning date for a sequence of archived redo log files. The clause specifies those logs that could be used in a recovery starting at the indicated time.</p> <p>The time specified in the string must be formatted according to the Globalization Technology date format specification currently in effect, but can also be any SQL expression with the DATE datatype, such as SYSDATE. The TO_DATE function can be used to specify hard-coded dates that work regardless of the current Globalization Technology settings.</p> <p>If you do not specify the UNTIL TIME parameter, RMAN will include all available log files beginning with the date specified in the FROM TIME parameter. Use the V\$ARCHIVED_LOG data dictionary view to determine the time stamps for the first and last entries in a log file.</p> <p>Note: The FROM TIME clause is not the same as the COMPLETED AFTER clause. FROM TIME specifies logs that could be used to <i>recover</i> starting at the indicated time, whereas COMPLETED AFTER specifies logs that were <i>created</i> after the indicated time (refer to "completedTimeSpec" on page 2-61).</p> <p>See Also: <i>Oracle Database Reference</i> for information on how to use the NLS_LANG and NLS_DATE_FORMAT environment variables to specify the format for the time</p>
TIME BETWEEN 'date_string' AND 'date_string'	<p>Specifies a range of times. Note that TIME BETWEEN 'date_string' AND 'date_string' is exactly equivalent to FROM TIME 'date_string' UNTIL TIME 'date_string'.</p>
UNTIL TIME = 'date_string'	<p>Specifies the end date for a sequence of archived redo log files. The clause specifies those logs that could be used to recover to the indicated time.</p> <p>The time specified in the string must be formatted according to the Globalization Technology date format specification currently in effect, but can also be any SQL expression with the DATE datatype, such as SYSDATE. The TO_DATE function can be used to specify hard-coded dates that work regardless of the current Globalization Technology settings.</p> <p>If you do not specify the FROM TIME parameter, then the beginning time for the sequence will be the earliest available archived redo log.</p> <p>If the database is open when you run BACKUP ARCHIVELOG, and if the UNTIL clause is specified, then RMAN does not run ALTER SYSTEM ARCHIVE LOG CURRENT.</p> <p>Note: The UNTIL TIME clause is not the same as the COMPLETED BEFORE clause. UNTIL TIME specifies logs that could be used to <i>recover</i> to the indicated time, whereas COMPLETED BEFORE specifies logs that were <i>created</i> before the indicated time (refer to "completedTimeSpec" on page 2-61).</p> <p>See Also: <i>Oracle Database Reference</i> for information on how to use the NLS_LANG and NLS_DATE_FORMAT environment variables to specify the format for the time</p>
THREAD = integer	<p>Specifies the thread containing the archived redo log files you wish to include. You only need to specify this parameter when running the database in an Oracle Real Application Clusters (RAC) configuration.</p> <p>THREAD is only valid when SEQUENCE is also specified. 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 check the thread number for a log.</p>

Examples

Specifying Records by Time: Example This example backs up all logs that could be used to recover to a point one week ago, and then deletes all archived redo logs that were created more than two weeks ago:


```
BACKUP ARCHIVELOG UNTIL TIME 'SYSDATE-7';
DELETE ARCHIVELOG ALL COMPLETED BEFORE 'SYSDATE-14';
```

Specifying Records by SCN: Example This example restores backup archived redo log files from tape that fall within a range of SCNs:

```
RESTORE ARCHIVELOG SCN BETWEEN 94097 AND 106245;
```

Specifying a Single Log Sequence Number: Example This example backs up only archived log 30 of thread 1 and then deletes it.

```
BACKUP ARCHIVELOG SEQUENCE 30 DELETE INPUT;
```

Specifying a Range of Records by Log Sequence Number: Example This example backs up all archived logs from sequence 431 to sequence 440 on thread 1 and deletes the archived logs after the backup is complete. If the backup fails, the logs are not deleted.

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE sbt;
  BACKUP ARCHIVELOG
    SEQUENCE BETWEEN 31 AND 40 THREAD 1
    # delete original archived redo logs after backup completes
    DELETE INPUT;
}
```

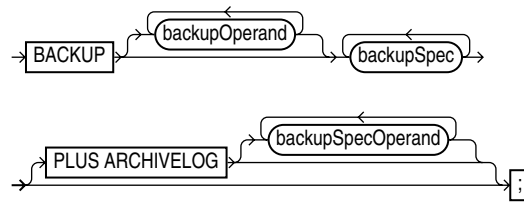
Specifying All Log Sequence Numbers in a Thread This example crosschecks all archived redo logs in thread 1:

```
CROSSCHECK ARCHIVELOG FROM SEQUENCE 0 THREAD 1;
```

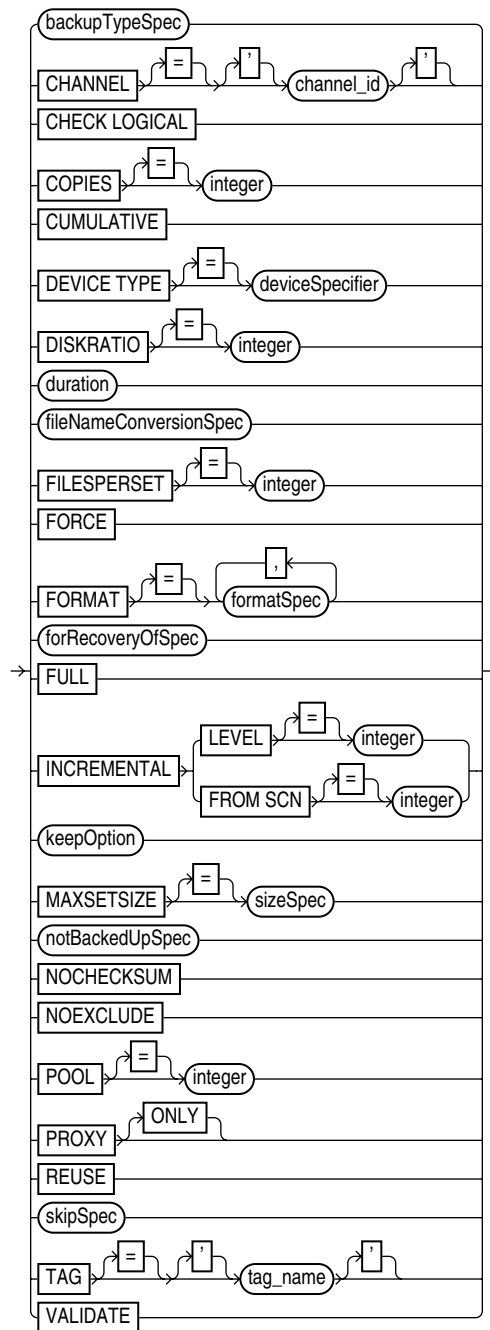
BACKUP

Syntax

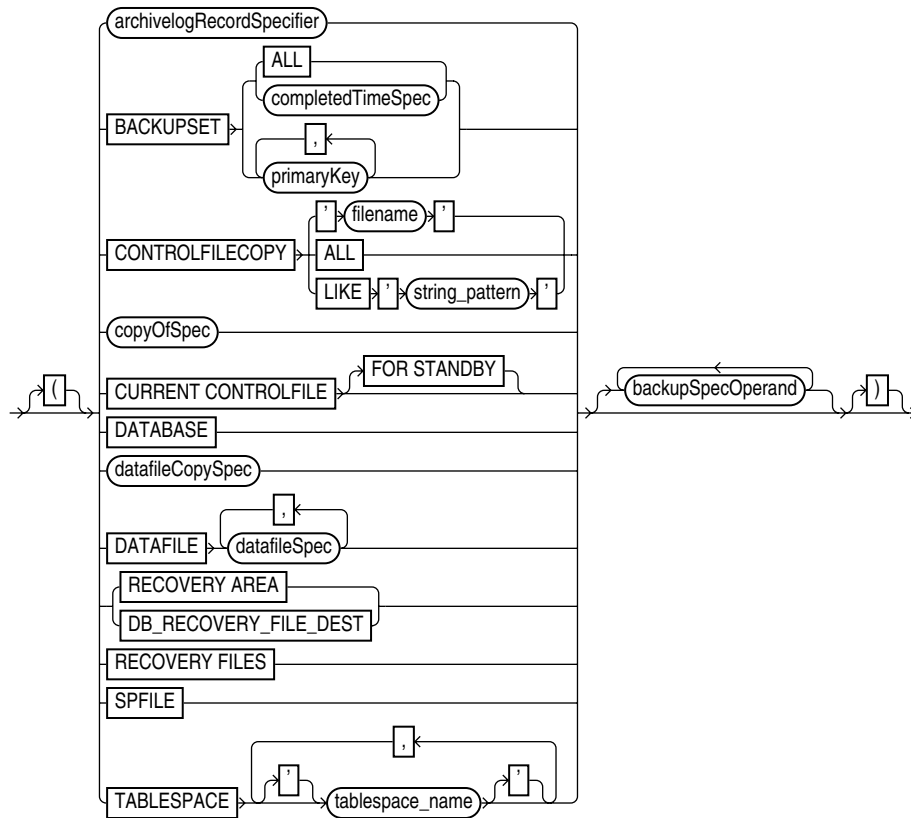
backup::=



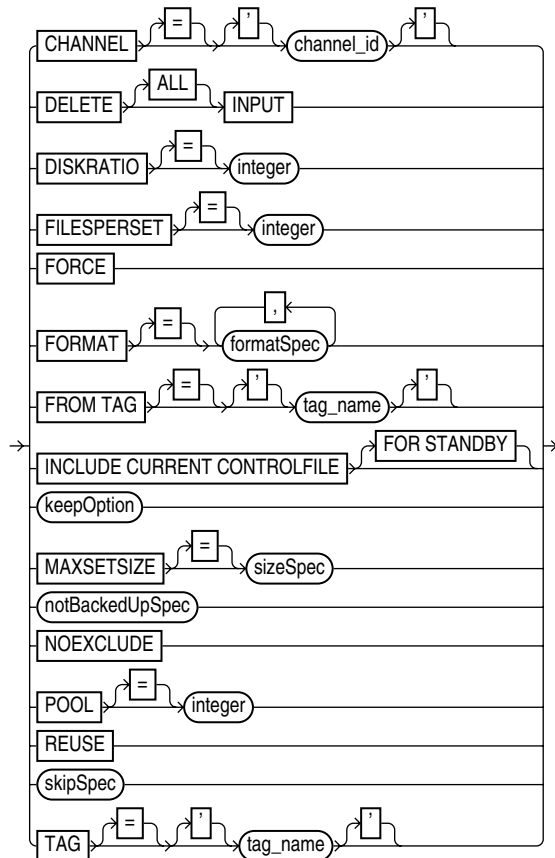
backupOperand::=



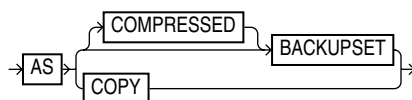
backupSpec::=



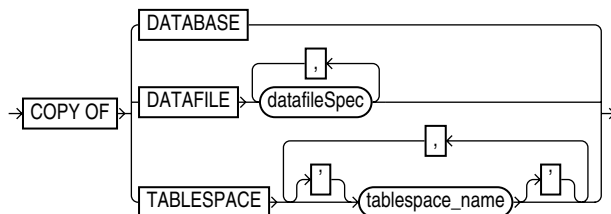
backupSpecOperand::=



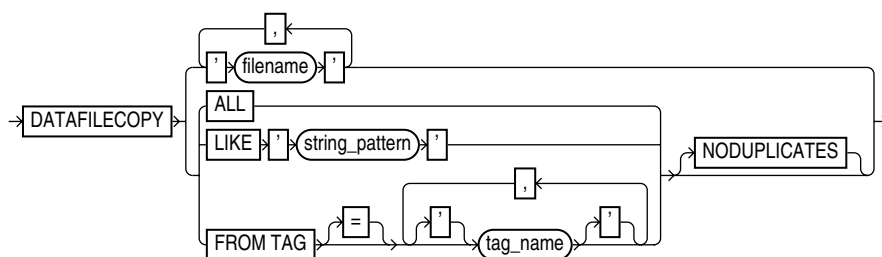
backupTypeSpec::=



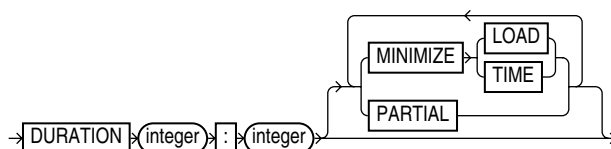
copyOfSpec::=



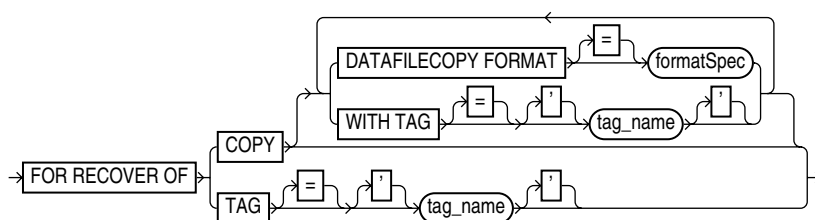
datafileCopySpec::=



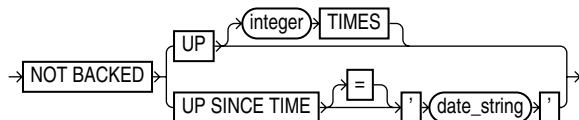
duration::=



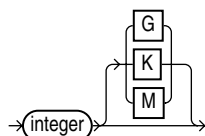
forRecoveryOfSpec::=



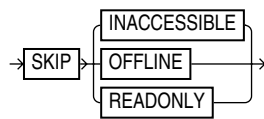
notBackedUpSpec::=



sizeSpec::=



skipSpec::=



Purpose

To back up a database, tablespace, datafile (current or copy), control file (current or copy), SPFILE, archived log, or backup set.

Note: RMAN can only back up datafiles, controlfiles, SPFILES, archived redo log files, as well as backups of these files created by RMAN.

Although the database depends on other types of files for operation, such as network configuration files, password files, and the contents of the Oracle home, these files cannot be backed up with RMAN. Likewise, some features of Oracle, such as external tables or the BFILE datatype, store data in files other than those listed above. RMAN cannot back up those files. You must use some non-RMAN backup solution for any files not in the preceding list.

RMAN can back up a target or standby database. The term **backup** refers to the files created by an RMAN BACKUP command.

Backup Sets, Backup Pieces, Image Copies, and Proxy Copies

When the AS BACKUPSET option is used, RMAN generates one or more **backup sets**, which are RMAN-specific logical structures. The backup set is the smallest unit of a backup. By default, each backup set contains 4 or fewer datafiles or 16 or fewer archived logs.

Note: RMAN only records complete backup sets in the RMAN repository. There are no partial backup sets. When a BACKUP command creates backup pieces but does not produce a complete backup set for any reason, the backup pieces are discarded.

Each backup set contains at least one **backup piece**, which is an RMAN-specific physical file containing the backed up data. You can also use the BACKUP command to generate a **proxy copy**, which is a backup to a third-party medium in which the entire data transfer is conducted by a media manager.

Backup sets can be created on disk, or on media manager devices such as tape drives.

When the AS COPY option is used, RMAN generates **image copies** of the input files. 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.

You can create and restore image copy backups with RMAN, or with a native operating system command for copying files. However, when you use RMAN to create image copies, they are recorded in the RMAN repository, and are more easily available for use in restore and recovery operations. Otherwise, you must use the

CATALOG command to add the image copies to the RMAN repository before RMAN can use them.

By default, RMAN creates all backups as backup sets, on tape or on disk. You can change the default backup type for disk backups to be image copies using the **CONFIGURE** command. Backups to tape can only be backup sets.

Incremental Backups

Incremental backups copy only those blocks that have changed since a previous backup. A level 0 incremental backup captures all data blocks in a datafile. Level 1 incremental backups capture changes since a previous backup. Level 1 backups can be cumulative, in which case they capture changes since the last level 0 incremental backup, or differential, in which case they capture changes since the last level 0 or level 1 incremental backup.

Block Change Tracking for Incremental Backups

You can improve incremental backup performance by enabling **block change tracking**, in which case RMAN keeps a record of which blocks have changed, and uses this record whenever possible to avoid scanning entire datafiles.

For details on block change tracking, including how to enable and disable it, see *Oracle Database Backup and Recovery Basics*.

Incrementally Updated Backups: Rolling Forward Datafile Image Copies

By using the `BACKUP INCREMENTAL... FOR RECOVER OF COPY WITH TAG...` syntax you can create level 1 incremental backups suitable for rolling forward a level 0 incremental image copy backup of your database. `RECOVER COPY OF... WITH TAG...` is used to perform incremental update of a backup.

Note: This technique is used in Enterprise Manager's Oracle-suggested Strategy for backups to disk.

For more details on incrementally updated backups, see *Oracle Database Backup and Recovery Basics*.

Incremental Roll Forward of Database Copy: Updating Standby Databases

RMAN's Incremental Roll Forward of Database Copy feature enables you to synchronize a standby database with a source database by creating an incremental backup at the source database containing all changed blocks since the standby was created or last synchronized, and then applying that incremental backup at the standby.

The incremental backup is created at the source database using the `BACKUP INCREMENTALFROM SCN =n` form of the `BACKUP` command. All blocks changed at SCNs greater than or equal to the specified SCN are included in the incremental created by this `BACKUP` command.

Note: The resulting backup is not considered to be part of any backup strategy at the source database. It is not suitable for use in a normal `RECOVER DATABASE` operation at the source database.

At the standby database, the `RECOVER` command is used with the `NOREDO` option to apply the incremental backup to the standby. All changed blocks captured in the

incremental backup are updated at the standby, bringing it up to date with the original.

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, based upon several different encryption algorithms (listed in `V$RMAN_ENCRYPTION_ALGORITHMS`). RMAN offers three modes of encryption:

- **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.

For an overview of the encrypted backups mechanism, a guide to its use and information on choosing among the different modes of encryption, see *Oracle Database Backup and Recovery Advanced User's Guide*. The RMAN [CONFIGURE](#), [SET](#) and [SHOW](#) commands are used to manage the encryption settings for your database backups. See the reference entries for those commands for more details.

Binary Compression of Backup Sets

RMAN can apply a binary compression algorithm as it writes data to backup sets. This compression is similar to the compression provided by many tape vendors when backing up data to tape. (The two types of compression should not be used together; see the discussion of `BACKUP AS BACKUPSET` below for details on choosing between using RMAN compression and the tape compression for backup sets.)

Unused Block Compression Of Datafile Backups to Backup Sets

When backing up datafiles into backup sets, RMAN does not back up the contents of data blocks that have never been allocated. (In previous releases, this behavior was referred to as NULL compression.)

RMAN also skips other datafile blocks that do not currently contain data, if all of the following conditions apply:

- The `COMPATIBLE` initialization parameter is set to 10.2
- 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 being created on disk.

Skipping unused data blocks where possible enables RMAN to back up datafiles using less space, and can make I/O more efficient.

Backup Optimization

The `BACKUP` command optimizes backups, that is, does not back up files that are identical to files that are already backed up, when the following conditions are met:

- The `CONFIGURE BACKUP OPTIMIZATION ON` command has been run.
- You run `BACKUP DATABASE`, `BACKUP ARCHIVELOG` with `ALL` or `LIKE` options, or `BACKUP BACKUPSET ALL`.
- You specify a channel of only one device type, that is, you do not mix channels that use different device types.

Channel Failover

A `BACKUP` command is decomposed into multiple independent backup steps by RMAN. Each independent step can be executed on any channel allocated for a specific device. If you have multiple channels allocated, and 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.

Control File Autobackups

If `CONFIGURE CONTROLFILE AUTOBACKUP` is set to `ON`, then RMAN automatically backs up the control file after `BACKUP` commands. "CONFIGURE" on page 2-63 describes the complete set of circumstances in which autobackups occur.

Restrictions and Usage Notes

- Although the database depends on other types of files for operation, such as network configuration files, password files, and the contents of the Oracle home, these files cannot be backed up with RMAN. Likewise, some features of Oracle, such as external tables or the `BFILE` datatype, store data in files other than those listed above. RMAN cannot back up those files. You must use some non-RMAN backup solution for any files not in the preceding list.
- The target database must be mounted or open when using the `RMAN BACKUP` command.
- RMAN can make an **inconsistent backup** when the database is in `ARCHIVELOG` mode, but you must apply redo logs after restoring such a backup, to make the database consistent.
- You must use a current control file when using the `BACKUP` command.
- If no automatic channel is configured for the specified device type, then you must manually allocate a channel for each execution of the `BACKUP` command. If no manual channel is allocated, then RMAN uses the default channels (as set using the `CONFIGURE` command). RMAN comes with a `DISK` channel preconfigured but no preconfigured channels for `sbt` devices.

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.

- Backup pieces must have unique names.
- When backing up Oracle files to `DISK`, the logical block size of the Oracle file to be backed up must be an even multiple of the physical block size of the destination device. For example, a `DISK` device with a block size of 2K can only be used as a destination for backups of Oracle files with logical block sizes of 2K, 4K, 6K 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 writable CD or DVD, or some other device which has a larger physical block size.

- You must back up files onto valid media. If you specify `DEVICE TYPE DISK`, then RMAN will back up to random access disks. You can make a backup on any device that can store a datafile: in other words, if the statement `CREATE TABLESPACE tablespace_name DATAFILE 'filename'` works, then '*filename*' is a valid backup path name. If you specify `DEVICE TYPE sbt`, then you can back up to any media supported by the media manager.
- You can only backup a database running in `NOARCHIVELOG` mode after a consistent shutdown. You cannot make a backup (either normal or incremental) in `NOARCHIVELOG` mode when the database is open or is closed after an instance failure or `SHUTDOWN ABORT`.
- You cannot stripe a single backup set across multiple channels.
- You cannot stripe a single input file across multiple backup sets.
- Archived redo log files and datafiles are never combined into a single backup set.
- When using encrypted backups, datafiles from tablespaces with different encryption settings are never written into the same backup set.
- There is no persistent configuration that controls whether archivelog backups as backupsets are encrypted. 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 archive log backup is being created.
 - Encryption is configured for backups of the whole database or at least one tablespace.
- You cannot specify multiple, identical `FORMAT` strings within a single `backupSpec` (for example, `BACKUP DATAFILE 3 TO '/tmp/foo', DATAFILE 4 TO '/tmp/foo'`). However, RMAN permits a single `FORMAT` string to exist in multiple `backupSpec` clauses.
- You 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.
- You cannot back up locally-managed temporary tablespaces (although you can back up dictionary-managed tablespaces)
- You cannot back up transportable tablespaces that were not made read-write after being transported.
- You cannot use the `DELETE INPUT` option when backing up objects other than datafile copies, archived redo logs, or backup sets.
- You cannot specify the number of backup pieces that should go in a backup set.
- You cannot create image copies (that is, use `BACKUP AS COPY`) on a nondisk media device.
- You cannot back up a backup set from tape to disk, or from one tape to another tape.
- You cannot make an image copy of a backup set, although you can make an image copy of an image copy. To backup a backupset, use `BACKUP BACKUPSET`.
- You cannot specify the `PLUS ARCHIVELOG` clause on the `BACKUP ARCHIVELOG` command.

- Do not open a NOARCHIVELOG mode database while it is being backed up. If you do, and some data blocks in the files being backed up are modified before being read by the backup session, then the backup is not usable after being restored because it requires recovery.
- The maximum length of a backup piece filename is platform-specific. For SBT backups using 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 SBT1.1 vendors may support longer filenames. Vendors supporting SBT 2.0 must support filenames up to 512 characters. Some SBT2.0 vendors may support longer filenames.
- You cannot use the `DEVICE TYPE` option for a device other than `DISK` if you have not already run `CONFIGURE DEVICE TYPE` for this device.
- You cannot allocate channels and run `BACKUP` with the `DEVICE TYPE` option.
- You cannot validate the backup of backup sets.
- You cannot specify `DEVICE TYPE DISK` when running the `BACKUP RECOVERY AREA` command.
- You cannot back up the change tracking file with `RMAN`.

Keywords and Parameters

backup

Syntax Element	Description
<i>backupOperand</i>	Specifies various options for the <code>BACKUP</code> command.
<i>backupSpec</i>	<p>A <code>BACKUP</code> specification list contains a list of one or more <i>backupSpec</i> clauses. A <i>backupSpec</i> clause minimally contains a list of one or more objects to be backed up.</p> <p>Each <i>backupSpec</i> clause generates one or more backup sets (<code>AS BACKUPSET</code>) or image copies (<code>AS COPY</code>). For <code>AS BACKUPSET</code>, 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 default limit of 4 datafiles or 16 archived logs in each backup set.</p>
<code>PLUS ARCHIVELOG</code>	<p>When you specify <code>PLUS ARCHIVELOG</code>, <code>RMAN</code> performs these steps:</p> <ol style="list-style-type: none"> 1. Runs an <code>ALTER SYSTEM ARCHIVE LOG CURRENT</code> statement. 2. Runs the <code>BACKUP ARCHIVELOG ALL</code> command. Note that if backup optimization is enabled, then <code>RMAN</code> only backs up logs that have not yet been backed up. 3. Backs up the files specified in the <code>BACKUP</code> command. 4. Runs an <code>ALTER SYSTEM ARCHIVE LOG CURRENT</code> statement. 5. Backs up any remaining archived redo logs.
<i>backupSpecOperand</i>	Specifies a variety of options and parameters that affect the <i>backupSpec</i> clause.

backupOperand

Syntax Element	Description
<i>backupOperand</i>	Specifies various options for the <code>BACKUP</code> command.

Syntax Element	Description
<i>backupTypeSpec</i>	Specifies the type of backup being created, either backup sets or image copies. See " backupTypeSpec " on page 2-41 for details.
CHANNEL <i>channel_id</i>	<p>Specifies the case-sensitive name of a channel to use when creating backups. Use any name that is meaningful, for example <i>ch1</i> or <i>dev1</i>. 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.</p> <p>Note: You can also specify this parameter in the backupSpec clause.</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 <code>alert.log</code> and server session trace file.</p> <p>If the sum of physical and logical corruptions detected for a file is no more than its <code>MAXCORRUPT</code> setting, then the backup command completes, and <code>V\$DATABASE_BLOCK_CORRUPTION</code> is populated with any corrupt block ranges. If there are more than <code>MAXCORRUPT</code> corrupt blocks, then the command terminates without updating the view and no output file is created for the backup.</p> <p>If the initialization parameter <code>DB_BLOCK_CHECKSUM=TRUE</code>, and if <code>MAXCORRUPT</code> and <code>NOCHECKSUM</code> are 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 specifies the total number of physical and logical corruptions permitted on a file.</p>
COPIES = <i>integer</i>	<p>Sets the number of identical backups (1 - 4) that RMAN should create. The default value is 1. 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. SET <code>BACKUP COPIES</code> 3. CONFIGURE . . . <code>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 backup at level 0 or lower, where <i>n</i> is 1. For example, in a cumulative level 1 backup RMAN backs up all blocks used since the most recent level 0 backup.</p> <p>Note: This option does not apply with <code>AS COPY</code> and results in an error message.</p>
DEVICE TYPE deviceSpecifier	<p>Allocates automatic channels for the specified device type only. This option is valid only if you have configured channels and have not manually allocated channels. For example, if you configure disk and tape channels, then configure <code>sbt</code> as the default device type, this command allocates disk channels only:</p> <pre>BACKUP DEVICE TYPE DISK DATABASE;</pre> <p>See Also: "deviceSpecifier" on page 2-101</p>

Syntax Element	Description
<code>DISKRATIO [=] integer</code>	<p>Directs RMAN to populate each backup set with datafiles from at least integer disks. This parameter is only enabled when you are backing up datafiles or control files, and when the operating system can give RMAN disk contention and node affinity information. To manually disable this feature, set <code>DISKRATIO = 0</code>.</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 = 5</code> to direct RMAN to include datafiles from at least 5 disks in each backup set.</p> <p>If you set <code>FILESERSET</code> but not <code>DISKRATIO</code>, then <code>DISKRATIO</code> defaults to the same value as <code>FILESERSET</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>
<code>duration</code>	<p>Specifies options related to the maximum time for a backup command to run.</p> <p>See Also: "duration" on page 2-43</p>
<code>filenameConversionSpec</code>	<p>This option is valid only when BACKUP is creating image copies. Files being copied are renamed according to the specified patterns. If a file being backed up has a name that does not match any of the specified rename patterns, then RMAN uses <code>FORMAT</code> to name the output image copies. If no <code>FORMAT</code> was specified, then RMAN uses the default format <code>%U</code>.</p> <p>See "fileNameConversionSpec" on page 2-114 for details about file renaming patterns.</p>
<code>FILESERSET [=] integer</code>	<p>When used with commands that create backupsets, specifies the maximum number of files to include in each backupset created.</p> <p>By default, RMAN divides files among backupsets 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 it is the number of files placed in each backupset. Otherwise, 64 files will be placed in each backupset.</p>
<code>FORCE</code>	<p>Causes RMAN to ignore backup optimization. In other words, even if <code>CONFIGURE BACKUP OPTIMIZATION</code> is set to ON, RMAN backs up all specified files.</p> <p>Note: You can also specify this option in the <i>backupSpecOperand</i> clause.</p>
<code>FORMAT = formatSpec</code>	<p>Specifies a pattern to use in creating a filename for the backup pieces or image copies created by this command. For <code>AS COPY</code>, if one or more of the directories mentioned in the specified format do not exist, then RMAN signals an error.</p> <p>See Also: "formatSpec" on page 2-121 for legal substitution variables</p>
<code>forRecoveryOfSpec</code>	<p>Identifies the backup being created as an incremental backup to be used in rolling forward an image copy. See "forRecoveryOfSpec" on page 2-43 for details.</p>

Syntax Element	Description
FULL	<p>Creates a backup of all blocks of datafiles included in the backup. FULL is the opposite of INCREMENTAL.</p> <p>RMAN makes full backups by default if neither FULL nor INCREMENTAL 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 (though a full image copy backup can be incrementally updated by applying incremental backups with the RECOVER command).</p> <p>Note:Backup Unused Space Compression, described in "Unused Block Compression Of Datafile Backups to Backup Sets" on page 2-28, causes some datafile blocks to be skipped during full backups of datafiles as backup sets.</p>
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. An incremental backup at level 0 backs up all data blocks in datafiles being backed up. An incremental backup at level 1 backs up only changed blocks. An incremental backup can be either differential or cumulative. In a cumulative level 1 incremental backup, RMAN backs up all blocks changed since the most recent level 0 backup. In a differential level 1 incremental backup, RMAN backs up blocks updated since the last level 0 or level 1 incremental backup.</p> <p>Incremental backups at level 0 can be either backup sets or image copies. Incremental backups at level 1 can only be backup sets.</p> <p>A level 0 backup must exist as the base backup for an incremental strategy. An incremental backup at level 0 is identical in content to a full backup, but unlike a full backup the level 0 backup is considered a part of the incremental strategy. If no level 0 backup exists when you run a level 1 backup, then RMAN makes a level 0 backup automatically.</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 RECOVER command. Among the checks performed are:</p> <ul style="list-style-type: none"> ■ A level 0 backup must exist for each datafile in the BACKUP command. These backups must not have status UNAVAILABLE. If no level 0 backup exists, then RMAN makes one. ■ Sufficient incremental backups taken since the level 0 must exist and be available such that the incremental backup to be created is usable. <p>If you specify INCREMENTAL, then in the <i>backupSpec</i> clause you must set one of the following parameters: DATAFILE, DATAFILECOPY, TABLESPACE, or DATABASE. RMAN does not support incremental backups of control files, archived redo logs, or backup sets.</p> <p>Note: You cannot make inconsistent incremental backups when the database is in NOARCHIVELOG mode. Hence, you cannot generate incremental backups when a NOARCHIVELOG database is open and in use.</p> <p>Note: When creating an incremental backup, RMAN considers backups from parent incarnations as valid. For example, assume you make a level 0 backup and then OPEN RESETLOGS. If you make a level 1 incremental backup, then RMAN backs up all blocks changed since the pre-RESETLOGS level 0 backup.</p> <p>See Also: "CHANGE" on page 2-54</p>

Syntax Element	Description
INCREMENTAL FROM SCN= <i>integer</i>	<p>Creates an incremental backup of all specified datafiles which includes all datafile blocks changed at SCNs greater than or equal to the specified SCN.</p> <p>The expected use of this feature is in refreshing a standby database with changes from a primary database. See <i>Oracle Database Backup and Recovery Advanced User's Guide</i> for details.</p> <p>Note:</p> <ul style="list-style-type: none"> ■ The default location of the incremental is the dbs subdirectory under the Oracle home. ■ The incremental backup created by <code>BACKUP INCREMENTAL FROM SCN</code> is not part of any incremental strategy at the primary database. It can only be used to refresh a standby with the changes at the primary. In the RMAN repository, this backup is not marked with an incremental level when it is recorded in the RMAN repository. The algorithms that control which backups RMAN selects for use in a restore and recovery do not select incremental backups created using <code>FROM SCN</code>.
keepOption	<p>Overrides any configured retention policy for this backup so that the backup is not considered obsolete. You can use CHANGE to alter the keep status. Note that you must be connected to a recovery catalog when you specify <code>KEEP FOREVER</code>.</p> <p>See Also: "keepOption" on page 2-125</p>
MAXSETSIZE = <i>integer [K M G]</i>	<p>Specifies a maximum size for a backup set. RMAN limits all backup sets to this size. Use <code>MAXSETSIZE</code> to configure backup sets so that each fits on one tape rather than spanning multiple tapes. Otherwise, if one tape of a multivolume backup set fails, then you lose the data on all the tapes rather than just one.</p> <p>Specify size in bytes (default), kilobytes (K), megabytes (M), or gigabytes (G). Thus, to limit a backup set to 3 MB, specify <code>MAXSETSIZE = 3M</code>. The default size is in bytes, rounded down from kilobytes. For example, <code>MAXSETSIZE = 3000</code> 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 4 for datafiles and 16 for archived logs. When you specify <code>MAXSETSIZE</code>, RMAN attempts to limit the size in bytes of the backup sets according to the <code>MAXSETSIZE</code> 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 <code>MAXSETSIZE</code>s.</p> <p>Note: This option cannot be used with <code>BACKUP AS COPY</code> and results in an error message. If you run <code>BACKUP AS COPY</code> on a channel that has <code>MAXSETSIZE</code> set, then <code>MAXSETSIZE</code> is silently ignored.</p>
NOCHECKSUM	<p>Suppresses block checksums. A checksum is a number that is computed from the contents of a data block. If the <code>DB_BLOCK_CHECKSUM</code> initialization parameter is <code>true</code>, then the database computes a checksum for each block during normal operations and stores it in the block before writing the block 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>By default, the database also computes a checksum for each block and stores it in the backup. The checksum is verified when restoring from the backup and written to the datafile when restored.</p> <p>If the database is already maintaining block checksums, then this flag has no effect. The checksum is always verified and stored in the backup in this case.</p> <p>If you wish to prevent the use of block checksums in your backup, use the <code>NOCHECKSUM</code> option.</p> <p>See Also: <i>Oracle Database Reference</i> for more information about the <code>DB_BLOCK_CHECKSUM</code> initialization parameter</p>

Syntax Element	Description
notBackedUpSpec	Limits the set of files to be backed up according to whether a specified number of backups are already present (and not obsolete), or whether the files have been backed up since a specified date. See " notBackedUpSpec " on page 2-43 for details.
NOEXCLUDE	When specified on 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.
POOL = <i>integer</i>	Specifies the media pool in which the backup should be stored. Consult your media management documentation to see whether the POOL option is supported. Note: This option does not apply with AS COPY and results in an error message.
PROXY	Backs up the specified files by means of the proxy copy functionality, which gives the media management software control over the data transfer between storage devices and the datafiles on disk. The media manager—not RMAN—decides how and when to move data. When you run BACKUP with the PROXY option, RMAN performs these steps: <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, 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>If you do not want RMAN to try a conventional copy when a proxy copy fails, use the ONLY option.</p> <p>Note: If you specify PROXY, then the %p variable must be included in the FORMAT string either explicitly or implicitly within %U.</p> <p>Note: This option cannot be used with BACKUP AS COPY and results in an error message.</p>
ONLY	Causes the database to issue an error message when it cannot proxy copy rather than creating conventional backup sets.
REUSE	Enables RMAN to overwrite an already existing backup or copy with the same filename as the file that BACKUP is currently creating.
skipSpec	Excludes datafiles or archived redo logs from the backup if they are inaccessible, offline or read-only. See " skipSpec " on page 2-44 for details.

Syntax Element	Description
<code>TAG tag_name</code>	<p>Creates a user-specified tag name for a backup. The tag is not case sensitive.</p> <p>If you do not specify a tag name, then by default RMAN creates a tag for backups (except for control file autobackups) in the format <code>TAGYYYYMMDDTHHMMSS</code>, where <code>YYYY</code> is the year, <code>MM</code> is the month, <code>DD</code> is the day, <code>HH</code> is the hour (in 24-hour format), <code>MM</code> is the minutes, and <code>SS</code> is the seconds. For example, a backup of datafile 1 can receive the tag <code>TAG20020208T133437</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>A tag applies to 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 piece in the backup set 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> <p>Typically, a tag is a meaningful name such as <code>MONDAY_EVENING_BACKUP</code> or <code>WEEKLY_FULL_BACKUP</code>. Tags are reusable, so that backup set 100 can have the tag <code>MONDAY_EVENING_BACKUP</code> one week while backup set 105 has the same tag the next week.</p> <p>Tags must be 30 characters or less. The characters used in a tag must be limited to the characters that are legal in filenames on the target filesystem. For example, ASM does not support the use of the <code>-</code> character in the filenames it uses internally, so a tag including a <code>-</code> (for example, <code>weekly-incremental</code>) is not a legal tag name if you are storing backups in ASM disk groups.</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.
<code>VALIDATE</code>	<p>Causes RMAN to scan the specified files and verify their contents, testing whether this file can be backed up. RMAN creates no output files. Use this command periodically to check for physical and logical errors in database files.</p> <p>Note: You cannot validate backups of backup sets.</p>

backupSpec

Syntax Element	Description
<i>backupSpec</i>	<p>A <code>BACKUP</code> specification list contains a list of one or more <i>backupSpec</i> clauses. A <i>backupSpec</i> clause contains, at a minimum, a list of one or more objects to be backed up.</p> <p>Each <i>backupSpec</i> clause generates one or more backup sets (<code>AS BACKUPSET</code>) or image copies (<code>AS COPY</code>). For <code>AS BACKUPSET</code>, 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 default limit of 4 datafiles or 16 archived logs in each backup set.</p>

Syntax Element	Description
<i>archivelogRecordSpecifier</i>	<p>Specifies a range of archived redo logs to be backed up. 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 BACKUP ARCHIVELOG ALL DELETE INPUT command.</p> <p>If you specify BACKUP ARCHIVELOG ALL, then RMAN backs up exactly one copy of each distinct log sequence number. For example, if you archive to multiple destinations, RMAN backs up <i>one</i> copy of each log sequence number—not each copy of each log sequence number. For other commands, such as DELETE, ALL does refer to every log, even duplicate log sequences.</p> <p>If the database is open when you run BACKUP ARCHIVELOG, and if the UNTIL clause or SEQUENCE parameter is not specified, then RMAN runs ALTER SYSTEM ARCHIVE LOG CURRENT.</p> <p>Note: If you run BACKUP ARCHIVELOG ALL, or if the specified log range includes logs from prior incarnations, then RMAN backs up logs from prior incarnations to ensure availability of all logs that may be required for recovery through an OPEN RESETLOGS.</p> <p>See Also: "archivelogRecordSpecifier" on page 2-18 for syntax, and <i>Oracle Database Backup and Recovery Advanced User's Guide</i> explanations of backup failover for logs and automatic log switching</p>
<pre> BACKUPSET { ALL completedTimeSpec primaryKey [, primaryKey] } </pre>	<p>Backs up either ALL backup sets or backup sets specified by <i>primaryKey</i> or completion time. Use this parameter in conjunction with the DEVICE TYPE sbt clause to back up all backups on disk to tape. You cannot back up from tape to tape or from tape to disk: only from disk to disk or disk to tape.</p> <p>Note if you specify the DELETE INPUT option, then RMAN deletes all copies of the backup set that exist on disk. For example, if you duplexed a backup to 4 locations, then RMAN deletes all 4 backup sets. The ALL option is redundant, that is, it 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>Note: You can duplex backups of backup sets by using BACKUP COPIES and SET BACKUP COPIES.</p> <p>See Also: "completedTimeSpec" on page 2-61</p>
<pre> CONTROLFILECOPY { 'filename' ALL LIKE 'string_pattern' } </pre>	<p>Specifies a control file copy in one of the following ways:</p> <ul style="list-style-type: none"> ▪ 'filename' specifies a control file copy by filename ▪ ALL specifies that all control file copies should be backed up ▪ LIKE 'pattern' specifies a filename pattern. The percent sign (%) as a wildcard meaning 0 or more characters; an underscore (_) is a wildcard meaning 1 character. <p>The control file copy can be:</p> <ul style="list-style-type: none"> ▪ A copy of a normal control file (that is, not a standby control file) created with the BACKUP AS COPY CURRENT CONTROLFILE command or the SQL statement ALTER DATABASE BACKUP CONTROLFILE TO '...' ▪ A standby control file copy created with the BACKUP AS COPY STANDBY CONTROLFILE command or the SQL statement ALTER DATABASE CREATE STANDBY CONTROLFILE <p>RMAN inspects the header of the control file copy to determine whether it is a standby or nonstandby control file.</p>
<i>copyOfSpec</i>	<p>Makes a backup of previous image copies of datafiles and possibly control files. See "copyOfSpec" on page 2-42 for details.</p>

Syntax Element	Description
CURRENT CONTROLFILE [FOR STANDBY]	<p>Specifies the current control file.</p> <p>If you specify FOR STANDBY, then RMAN generates a backup of the control file that is usable during creation of a standby database. The backup contains only the standby control file.</p> <p>Note: You cannot assign a tag to a backup of the current control file.</p>
DATABASE	<p>Creates a backup set (AS BACKUPSET) or group of image copies (AS COPY) for 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 CONFIGURE CONTROLFILE AUTOBACKUP is OFF, then RMAN automatically includes the control file in the backup. If the instance is started with a server parameter file, then RMAN also includes this parameter file in the backup.</p> <p>If the <i>backupSpec</i> includes datafile 1, and if CONTROLFILE AUTOBACKUP 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>Note: To force RMAN to include the current control file in the backup when CONTROLFILE AUTOBACKUP is ON, specify the INCLUDE CURRENT CONTROLFILE clause.</p>
<i>datafileCopySpec</i>	<p>Specifies the filenames of one or more datafile image copies. See "<i>datafileCopySpec</i>" on page 2-42 for details.</p>
DATAFILE <i>datafileSpec</i>	<p>Specifies a list of one or more datafiles. Refer to description of BACKUP DATABASE for RMAN behavior when datafile 1 is backed up.</p> <p>See Also: "<i>datafileSpec</i>" on page 2-95</p>
RECOVERY AREA DB_RECOVERY_FILE_DEST	<p>Backs up recovery files created in the current and all previous flash recovery area destinations. Recovery files are full and incremental backup sets, control file autobackups, archived logs, and datafile copies. Flashback logs, the current control file, and online redo logs are <i>not</i> backed up.</p> <p>RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms.</p> <p>Backup optimization is always ON for this option. The backup must go to sbt, so RMAN issues an error if no sbt channels are allocated or configured.</p> <p>Note: If the flash recovery area is not enabled but has been enabled in the past, then files that were created in the previous flash recovery area location are backed up.</p>
RECOVERY FILES	<p>Backs up <i>all</i> recovery files on disk, whether they are stored in the flash recovery area or another locations on disk. Recovery files include full and incremental backup sets, control file autobackups, archived logs, and datafile copies. Backup optimization is always ON for this option. The backup must go to sbt, so RMAN issues an error if no sbt channels are allocated or configured.</p>
SPFILE	<p>Backs up the server parameter file currently used by the database. RMAN cannot back up other copies of the server parameter file, and cannot back up the server parameter file when the instance was started with an initialization parameter file. RMAN cannot make incremental backups of the SPFILE.</p>
TABLESPACE <i>tablespace_name</i> [, <i>tablespace_name</i>]	<p>Specifies the names of one or more tablespaces. RMAN backs up all datafiles that are currently part of the tablespaces. If the SYSTEM tablespace (and thus datafile 1) is included in the backup, and if CONTROLFILE AUTOBACKUP is not configured, then RMAN creates a copy of the control file.</p> <p>The TABLESPACE keyword is merely a convenience; tablespace names are translated internally into a list of datafiles. If you rename a tablespace (for example, from users to customers), then RMAN detects that the tablespace has changed its name and updates the recovery catalog on the next resynchronization.</p>

Syntax Element	Description
<i>backupSpecOperand</i>	The <i>backupSpecOperand</i> that follows a <i>backupSpec</i> specifies options that apply to the backupSpec.

backupSpecOperand

Syntax Element	Description
<i>backupSpecOperand</i>	Specifies a variety of options and parameters that affect the <i>backupSpec</i> clause. Many subclauses of <i>backupSpecOperand</i> are also used with <i>backupOperand</i> . For those, see the description of " <i>backupOperand</i> " on page 2-31. Those which are not shared in common with <i>backupOperand</i> are listed here.
DELETE [ALL] INPUT	<p>Deletes the input files upon successful creation of the backup. Specify this option only when backing up archived logs, datafile copies (COPY OF or DATAFILECOPY), or backup sets. It is equivalent to issuing DELETE for the input files.</p> <p>The ALL option applies only to archived 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. For example, if you specify the SEQUENCE <i>n</i> clause, then RMAN deletes all archive logs with same sequence number <i>n</i>, including duplicate archived logs (that is, logs with same log sequence number and thread).</p> <p>Note: 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>See Also: "<i>CONNECT</i>" on page 2-75 for information on the effect of recovery catalog compatibility on this command</p>
FROM TAG = 'tag_name'	Allows specifying a backup by tag. Defined in context with several other commands.
INCLUDE CURRENT CONTROLFILE [FOR STANDBY]	<p>Creates a snapshot of the current control file and places it into each backup set produced by this clause.</p> <p>If you specify FOR STANDBY, then RMAN creates a backup of the control file usable for creation of a standby database. The backup set includes the standby control file and the object backed up.</p> <p>Note: This option does not apply with AS COPY and results in an error message.</p>

backupTypeSpec

Syntax Element	Description
AS [COMPRESSED] BACKUPSET	<p>Creates backup sets (rather than image copies) on the specified device type.</p> <p>AS BACKUPSET is the only possibility when backing up to tape, and for creating level 1 incremental backups to any destination.</p> <p>When backing up datafiles into backup sets, RMAN only backs up blocks that are currently in use.</p> <p>With the COMPRESSED option, binary compression is used. The data written into the backup set is compressed 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>Note:</p> <ul style="list-style-type: none"> There is some CPU overhead associated with compressing backup sets. If the database being backed up is running at or near its maximum load, you may find the overhead from using AS COMPRESSED BACKUPSET unacceptable. In most other circumstances, compressing backupsets saves enough disk space to be worth the CPU overhead. 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 BACKUP AS COMPRESSED BACKUPSET. Therefore, use uncompressed backup sets and turn on the compression provided by the media management vendor when backing up to locally attached tape devices. In any case, you should not use both forms of compression together. When using incrementally updated backups, the level 0 incremental must be an image copy backup. Full database backups should usually be either image copies or compressed backupsets. Image copies are more flexible than backup sets for some purposes (such as use in an incrementally updated backups strategy), and compressed backupsets make more efficient use of storage space, if the CPU overhead involved in creating them is tolerable.
AS COPY	<p>Creates image copies (rather than backup sets). Can only be used with backups created on disk.</p> <p>Note:</p> <p>When using BACKUP AS COPY, if you do not specify the destination for the image copies, RMAN will choose a location according to the following rules:</p> <ul style="list-style-type: none"> If the channel used for output has a default CONFIGURE... FORMAT setting, that setting is used to generate output file names If a flash recovery area is configured, the backups are placed in the flash recovery area Otherwise a platform-specific default FORMAT (which includes a %U for generating a unique filename) is used.

copyOfSpec

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 BACKUP DATABASE are expected to have copies: if not, then RMAN signals an error. It is not necessary for all copies to have been produced by a single BACKUP command. If multiple copies exist of datafile, then RMAN backs up the latest. Optionally, specify the copies by tag name (for example, FULL_COLD_COPY).</p> <p>Note: The output of this command can be image copies or backup sets.</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 BACKUP TABLESPACE should have copies: if not, then RMAN signals an error. It is not necessary for all copies to have been produced by a single BACKUP command. If multiple copies exist of datafile, then RMAN backs up the latest.</p> <p>Specify the tablespaces in the list by tablespace name (for example, users) or specify a particular copy by tag name (for example, 0403_CPY_OF_USERS). If you do not specify TAG, then RMAN backs up the most recent datafile copy for each datafile in the tablespace.</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 (DATAFILE 3) or filename (DATAFILE ' ?/oradata/trgt/users01.dbf '). 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 BACKUP command.</p> <p>Note: The output of this command can be image copies or backup sets.</p> <p>See Also: "datafileSpec" on page 2-95</p>

datafileCopySpec

Syntax Element	Description
<pre>DATAFILECOPY { filename [,filename...] { ALL LIKE 'string_pattern' FROM TAG 'tag_name' } }</pre>	<p>Specifies the filenames of one or more datafile image copies. You can use the following parameters:</p> <ul style="list-style-type: none"> ▪ 'filename' specifies an image copy by filename ▪ ALL specifies that all datafile image copies should be backed up ▪ LIKE 'pattern' specifies a filename pattern. The percent sign (%) is a wildcard that means zero or more characters; an underscore (_) is a wildcard that means one character. ▪ FROM TAG 'tag_name' specifies a list of one or more datafile copies, identified by the tag <i>tag_name</i>. 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	<p>Prevents the inclusion of identical datafile copies in a backup operation. For each set of duplicate datafile copies, the file with the most recent timestamp will be selected.</p>

duration

Syntax Element	Description
DURATION <i>hh:mm</i> [PARTIAL] [MINIMIZE (TIME LOAD)]	<p>Specifies a maximum time for a backup command to run. If a backup command does not complete in the specified duration, the backup being taken stops.</p> <p>With the PARTIAL option, the command is considered to have completed successfully and no error is reported by RMAN even if the whole backup is not completed in the specified duration.</p> <p>Without the PARTIAL option, the backup command is considered to have failed if it does not complete in the specified duration, and RMAN reports an error. If the backup command is part of a RUN block, subsequent commands in the RUN block do not execute.</p> <p>Whether PARTIAL is used or not, all backupsets completed before the backup is interrupted are retained and can be used in restore and recover operations.</p> <p>With disk backups, you can use MINIMIZE TIME run the backup at maximum speed (default) , or MINIMIZE LOAD to slow the rate of backup to lessen the load on the system. With MINIMIZE LOAD the backup will take the full specified duration.</p> <p>When DURATION is used with MINIMIZE TIME, the 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>

forRecoveryOfSpec

Syntax Element	Description
FOR RECOVER OF TAG [=] ' <i>tag_name</i> '	<p>Lets you identify any tagged level 0 incremental to serve as the basis for this level 1 incremental. Useful in strategies other than the incrementally updated backups strategy, which uses the FOR RECOVER OF COPY clause.</p>
FOR RECOVER OF COPY	<p>Lets you specify that this incremental should contain all changes since the SCN of a specified datafile copy (level 0 incremental backup) of your database. The datafile copies should be identified with either a DATAFILE COPY or WITH TAG clause, to keep the incremental backup strategy for which this backup will be used separate from the rest of your backup strategies. Otherwise, the most recent copy of each datafile will be used as the basis for the incremental.</p>
WITH TAG [=] ' <i>tag_name</i> '	<p>Used with FOR RECOVER OF COPY, specifies a tag to identify the level 0 incremental backup to serve as the basis of the incremental. If no level 0 with the tag specified in WITH TAG option is found, then FOR RECOVER OF COPY option will create a level 0 datafile copy tagged with the value specified in the WITH TAG option. BACKUP . . . FOR RECOVER OF COPY WITH TAG... is key to backup strategies based on incrementally updated backups, as described in <i>Oracle Database Backup and Recovery Basics</i> and used in some Enterprise Manager backup strategies.</p>
DATAFILE COPY [=] <i>formatSpec</i>	<p>Used with FOR RECOVER OF COPY, identifies the datafile copies to use as the basis for this incremental.</p>

notBackedUpSpec

Syntax Element	Description
NOT BACKED UP	<p>Backs up only those files (of the files specified on the command) that RMAN has never backed up. This option is a convenient way to back up new files after adding them to the database.</p>

Syntax Element	Description
<code>SINCE TIME = 'date_string'</code>	<p>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 NLS_DATE_FORMAT, or a SQL date expression such as 'SYSDATE-1'. When calculating the number of backups for a file, RMAN only considers backups created on the same device type as the current backup.</p> <p>This option is a convenient way to back up files that were not backed up during a previous failed backup. For example, you back up the database, but the instance fails halfway through. You can restart the backup with the NOT BACKED UP SINCE TIME clause and avoid backing up those files that you already backed up. If AS BACKUPSET is set, then this feature is only useful if RMAN generates multiple backup sets during the backup.</p> <p>When determining whether a file has been backed up, the SINCE date is compared with the completion time of the most recent backup. For BACKUP AS BACKUPSET, the completion time for a file in a backup set is the completion time of the entire backup set. In other words, all files in the same backup set have the same completion time.</p>
<code>integer TIMES</code>	<p>Backs up only those archived logs that have not been backed up at least <i>integer</i> times. To determine the number of backups for a file, RMAN only considers backups created on the same device type as the current backup.</p> <p>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).</p>

sizeSpec

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

skipSpec

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

Examples

Backing Up a Database: Example This example assumes that CONFIGURE CONTROLFILE AUTOBACKUP is OFF. The command backs up all datafiles to tape, as well as the current control file, the server parameter file, and archived logs:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```


Scripting Incremental Backups: Example This example shows a series of scripts that you can run to make regular incremental backups of the database:

```
# Run once a week to back up database to disk as level 0:
BACKUP INCREMENTAL LEVEL 0 DATABASE;

# Run every day to back up blocks that have changed since most recent level 0 or 1:
BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
BACKUP INCREMENTAL LEVEL 1 DIFFERENTIAL TABLESPACE users;
```

Performing a Cumulative Incremental Backup: Example This example backs up all blocks changed in the database since the most recent level 0 backup:

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

Backing Up Tablespaces and Datafiles to Disk as Backup Sets: Example This command uses two *backupSpec* clauses to back up tablespaces and datafiles and lets RMAN perform automatic parallelization of the backup:

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK FORMAT '/disk1/%U';
  ALLOCATE CHANNEL dev2 DEVICE TYPE DISK FORMAT '/disk2/%U';
  BACKUP AS BACKUPSET
    (TABLESPACE SYSTEM, tools, users, undotbs MAXSETSIZE 5M)
    (DATAFILE 2,4,5);
}
```

Backing Up Tablespaces and Datafiles to Disk as Image Copies: Example This command uses two *backupSpec* clauses to back up tablespaces and datafiles and lets RMAN perform automatic parallelization of the backup:

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK FORMAT '/disk1/%U';
  ALLOCATE CHANNEL dev2 DEVICE TYPE DISK FORMAT '/disk2/%U';
  BACKUP AS COPY
    (TABLESPACE SYSTEM, tools, users, undotbs)
    (DATAFILE 2,4,5);
}
```

Backing Up Datafile Copies: Example This example finds three datafile copies with the tag *LATESTCOPY*, copies them to directory */copies* and names the new copies by means of substitution variables:

```
BACKUP AS COPY
  FROM TAG 'LATESTCOPY'
  COPY OF DATAFILE 4, 6, 14
  FORMAT '/copies/Datafile%f_Database%d';
```

Backing Up Archived Logs and Deleting the Input: Example This example assumes that you have two archive destinations set: */arch1* and */arch2*. The command backs up one log for each unique sequence number and then deletes all logs from both archiving directories.

```
BACKUP ARCHIVELOG LIKE '/arch%' DELETE ALL INPUT;
```

Backing Up Backup Sets to Tape: Example In this example, the goal is to keep recent backup sets on disk and older backup sets on tape, and to avoid keeping copies of the same backup set on disk and tape simultaneously. This command backs up backup sets created more than two weeks ago to tape and then deletes the backed-up backup pieces from disk:

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

```
DELETE INPUT;
```

Specifying DEVICE TYPE on the BACKUP Command: Example This example configures DISK as the default device type, then backs up the server parameter file and all archived logs to tape:

```
# when disk is the default device and you do not specify a FORMAT parameter, then the
# default backup location is the flash recovery area (if configured) or
# a platform-specific default location (if not configured)
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
BACKUP DEVICE TYPE sbt SPFILE ARCHIVELOG ALL;
```

Duplexing a Backup Set: Example This example duplexes a backup of datafile 1 (which includes the current control file and server parameter file) to separate disks:

```
BACKUP DEVICE TYPE DISK
  COPIES 2 DATAFILE 1
  FORMAT '/disk1/df1_%U', '/disk2/df1_%U';
```

Specifying How Channels Divide Workload: Example This example explicitly parallelizes a backup operation by specifying which channels should back up which files and to which locations:

```
RUN
{
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt PARMS="ENV=(NSR_SERVER=tape_server_1)";
  ALLOCATE CHANNEL ch2 DEVICE TYPE DISK;
  ALLOCATE CHANNEL ch3 DEVICE TYPE sbt PARMS="ENV=(NSR_SERVER=tape_server_2)";
  BACKUP
    (DATAFILE 1,2,3,4          # channel ch1 backs up datafiles to tape drive #1
     CHANNEL ch1)
    (CONTROLFILECOPY '/oracle/copy/cf.f'
     CHANNEL ch2)             # channel ch2 backs up control file copy to disk
    (ARCHIVELOG FROM TIME 'SYSDATE-14'
     CHANNEL ch3);            # channel ch3 backs up archived redo logs to tape drive #2
}
```

Creating a Control File for a Standby Database: Example This example creates a backup of the current control file that can be used to create a standby database:

```
BACKUP CURRENT CONTROLFILE FOR STANDBY;
```

Creating an Incremental Backup for Refresh of a Standby Database: Example 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, as described in *Oracle Database Backup and Recovery Advanced User's Guide*.

```
BACKUP DEVICE TYPE DISK INCREMENTAL FROM SCN 750983 DATABASE
  FORMAT '/tmp/incr_standby_%U';
```

Backing Up Datafiles, Tolerating Corrupt Blocks: Example This example backs up datafile 3 and specifies that no more than two blocks with corruption should be tolerated:

```
RUN
{
  SET MAXCORRUPT FOR DATAFILE 3 TO 2;
  BACKUP CHECK LOGICAL
    DATAFILE 3;
}
```

Making an Image Copy of a Database Copy: Example This example makes an image copy of the database copy with tag TEST to the default destination, gives the output copy the tag DUPTEST, and performs logical checking:

```
BACKUP AS COPY COPY OF DATABASE FROM TAG 'TEST' CHECK LOGICAL TAG 'DUPTEST';
```

Creating a Long-Term Database Backup: Example This example creates a consistent backup of the database and server parameter file that is exempt from the retention policy. The command instructs RMAN to keep the backup for the next year, but not to keep the archived logs necessary to recover it:

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
BACKUP DATABASE
    KEEP UNTIL TIME 'SYSDATE+365'
    NOLOGS;
ALTER DATABASE OPEN;
```

Exempting Copies from the Retention Policy: Example The following example copies the control file and two datafiles and exempts them from the retention policy forever. (Note that KEEP FOREVER requires a recovery catalog.)

```
rman TARGET / CATALOG rman/rman@rcat
RMAN> SHUTDOWN IMMEDIATE;
RMAN> STARTUP MOUNT;
RMAN> BACKUP AS COPY
    KEEP FOREVER NOLOGS
    CURRENT CONTROLFILE FORMAT '?/oradata/cf_longterm.cpy',
    DATAFILE 1 FORMAT '?/oradata/df1_longterm.cpy',
    DATAFILE 2 FORMAT '?/oradata/df2_longterm.cpy';
ALTER DATABASE OPEN;
```

Backing Up Files That Need Backups: Example This example backs up all datafiles that have not been backed up to tape in the last month, and then backs up all archived logs that do not have at least two backups on tape:

```
BACKUP DEVICE TYPE sbt DATABASE NOT BACKED UP SINCE TIME 'SYSDATE-31';
BACKUP DEVICE TYPE sbt ARCHIVELOG ALL NOT BACKED UP 2 TIMES;
```

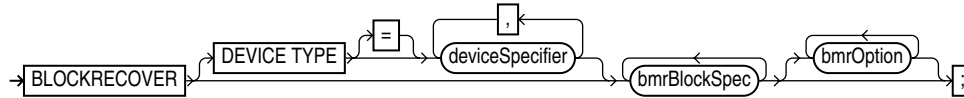
Using NODUPLICATES To Back Up Datafile Copies: Example This example creates several duplicate datafiles, and then backs up only the most recent of the duplicates:

```
BACKUP AS COPY DATAFILE 10 FORMAT 'foo' TAG my_tag;
BACKUP AS COPY DATAFILECOPY 'foo' FORMAT 'bar';
BACKUP AS COPY DATAFILECOPY 'bar' FORMAT 'foobar';
BACKUP AS BACKUPSET DATAFILECOPY ALL NODUPLICATES; # backs up only 'foobar'
BACKUP AS BACKUPSET DATAFILECOPY ALL; # backs up all files
```

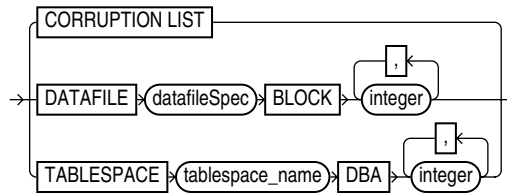
BLOCKRECOVER

Syntax

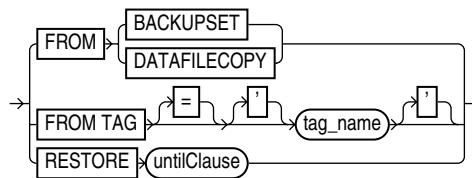
blockrecover::=



bmrBlockSpec::=



bmrOption::=



Purpose

Block media recovery recovers an individual data block or set of data blocks within a datafile. This type of recovery is useful if 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. Block-level data loss usually results from:

- I/O errors causing minor data loss
- Memory corruptions that get flushed to disk

You can either use `BLOCKRECOVER 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) when executing the `BLOCKRECOVER` command.

Restrictions and Usage Notes

- The target database must be mounted or open. You do *not* have to take a datafile offline if you are performing block media recovery on it.
- You can only perform complete media recovery of individual blocks. Point-in-time recovery of individual data blocks is not supported.
- You can only perform block media recovery on corrupt blocks.
- Blocks marked media corrupt are not accessible until recovery completes.
- You cannot perform block media recovery when using a backup control file.

- You cannot use proxy backups to perform block media recovery. If the only backups that you have are proxy backups, then you can restore them to a nondefault location on disk, which causes RMAN to view the restored files as datafile copies. You can then use the datafile copies for block media recovery.
- You must have a full backup of the file containing the corrupt blocks: block media recovery cannot use incremental backups.
- If RMAN fails to access a specific archived redo log file needed for block media recovery, it performs restore failover, trying all other backups listed in the RMAN repository that are suitable for use in this operation, and only fails if no suitable backup is available. See *Oracle Database Backup and Recovery Advanced User's Guide* for details on restore failover.
- Block media recovery cannot survive a missing or inaccessible archived log, although it can sometimes survive missing or inaccessible records (*Oracle Database Backup and Recovery Advanced User's Guide*).
- The datafile header block (block 1) cannot be recovered.
- You cannot perform block media recovery in NOARCHIVELOG mode.

Keywords and Parameters

blockrecover

Syntax Element	Description
DEVICE TYPE <i>deviceSpecifier</i>	Specifies the device type for the backup used in the block recovery. See Also: " <i>deviceSpecifier</i> " on page 2-101

bmrBlockSpec

Syntax Element	Description
<i>bmrBlockSpec</i>	Specifies the data blocks that require recovery.
CORRUPTION LIST	Recovers all blocks listed in the V\$DATABASE_BLOCK_CORRUPTION view. This view displays blocks marked corrupt by the most recent BACKUP (with or without the VALIDATE option), VALIDATE , or CREATE CATALOG command. The following types of corruption result in rows added to this view: <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. Physical corruption checking is on by default, and can be turned off with the NOCHECKSUM option. ■ Logical corruption. The block has a valid checksum, the header and footer match, and so forth, but the contents are logically inconsistent. Logical corruption checking is off by default, and can be turned on with the CHECK LOGICAL option. <p>Note: Logical corruption cannot be repaired using block media recovery. To repair logical corruption, restore the datafile from backup and perform media recovery.</p>
DATAFILE <i>datafileSpec</i>	Specifies a list of one or more datafiles that contain blocks requiring recovery. See Also: " <i>datafileSpec</i> " on page 2-95
BLOCK <i>integer</i>	Specifies the block number of the block requiring media recovery. Typically, the block number is obtained from error message output.

Syntax Element	Description
TABLESPACE <i>tablespace_name</i>	Specifies the tablespace name or number containing the corrupt blocks.
DBA <i>integer</i>	Specifies the data block address (DBA) of the corrupt block.

bmrOption

Syntax Element	Description
<i>bmrOption</i>	Specifies various restore options relating to the block recovery.
FROM BACKUPSET	Indicates that only backup sets should be restored.
FROM DATAFILECOPY	Indicates that only datafile image copies should be restored.
FROM TAG= ' <i>tag_name</i> '	Indicates that only the copy of the backup with the specified tag should be restored. Tag names are not case sensitive. See Also: " BACKUP " on page 2-22 to learn how a tag is applied to a copy of a backup.
RESTORE <i>untilClause</i>	Specifies that only backups and copies created before the specified time, SCN, or log sequence number should be restored. See Also: " untilClause " on page 2-227

Examples

Recovering a Group of Corrupt Blocks: Example This example recovers corrupt blocks in three datafiles:

```
BLOCKRECOVER DATAFILE 2 BLOCK 12, 13 DATAFILE 3 BLOCK 5, 98, 99 DATAFILE 4 BLOCK 19;
```

Limiting Block Media Recovery by Type of Restore: Example The following example recovers a series of blocks and restores only from datafile copies:

```
RUN
{
  BLOCKRECOVER DATAFILE 3 BLOCK 2,3,4,5 TABLESPACE sales DBA 4194405, 4194409, 4194412
  FROM DATAFILECOPY;
}
```

Limiting Block Media Recovery by Backup Tag: Example This example recovers blocks and restores only from the backup with the tag `weekly_backup`:

```
BLOCKRECOVER TABLESPACE SYSTEM DBA 4194404, 4194405 FROM TAG "weekly_backup";
```

Limiting Block Media Recovery by Time: Example The following example recovers two blocks in the `SYSTEM` tablespace. It restores only from backups that could be used to recover the database to a point two days ago:

```
BLOCKRECOVER TABLESPACE SYSTEM DBA 4194404, 4194405 RESTORE UNTIL TIME 'SYSDATE-2';
```

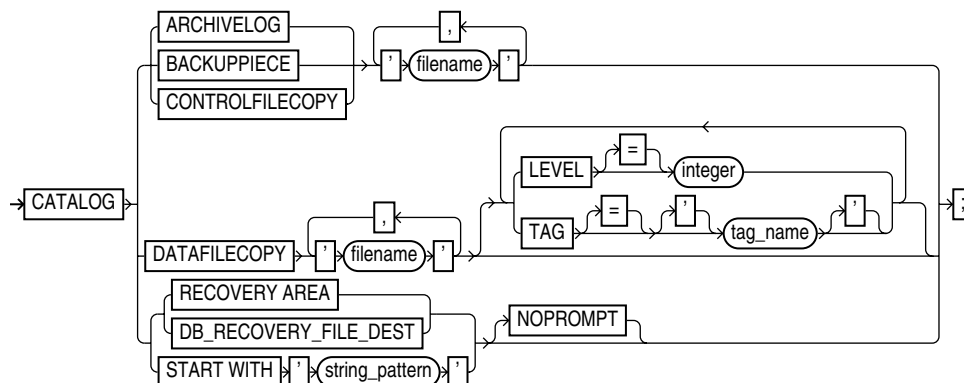
Repairing All Block Corruption in the Database: Example The following example runs a backup validation to populate `V$DATABASE_BLOCK_CORRUPTION`, then repairs any corrupt blocks recorded in the view:

```
BACKUP VALIDATE DATABASE;
BLOCKRECOVER CORRUPTION LIST;
```

CATALOG

Syntax

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.
- Record the existence of the last user-managed datafile copies made after the final shutdown in Oracle7, before migrating the database to Oracle8.

See Also: *Oracle Database Backup and Recovery Basics* to learn how to manage target database records stored in the catalog

Restrictions and Usage Notes

- 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.
- For a user-managed copy to be cataloged, it must be:
 - A datafile copy, control file copy, archived log, or backup piece.
 - Accessible on disk.

RMAN treats 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.

- You cannot catalog any datafile copies that were created in Oracle7 unless they were made after the final consistent shutdown in Oracle7 and before running the migration utility, or were made of a tablespace that was offline normal or read-only at the time of the migration. In other words, it must be possible to use the Oracle7 datafile copies without applying any archived logs.
- You cannot use CATALOG to catalog a file that belongs to a different database.
- You cannot use CATALOG to catalog a backup piece that exists on an `sbt` device.

Keywords and Parameters

Syntax Element	Description
ARCHIVELOG ' <i>filename</i> '	Specifies the filename of an archived log to be added to or updated in the RMAN repository.
BACKUPPIECE	<p>Specifies the name of a backup piece to be added to the RMAN repository. The backup piece must be on disk. RMAN verifies the backup piece header before cataloging it. RMAN can catalog a backup piece from a prior incarnation.</p> <p>Typically, you would catalog a backup piece in the following situations:</p> <ul style="list-style-type: none"> ■ You have copied a backup piece with an operating system utility. In this case, the original backup piece is cataloged but the piece copy is not. ■ You want to move a backup piece from one disk to another under a different absolute filename. ■ You run in NOCATALOG mode and must re-create the control file, thereby losing all RMAN repository data. Cataloging your backups makes them available again. <p>If you specify a list of backup pieces, RMAN attempts to catalog all pieces in the given list even if some of them fail. Cataloging a backup piece creates a new row in V\$BACKUP_PIECE. A backup set is only usable when <i>all</i> backup pieces are cataloged. Otherwise it would be in partially available state.</p> <p>Note: When cataloging backup pieces from releases prior to Oracle9i, performance improves when you catalog higher copy numbers first. For example, if you need to catalog copies 1, 2, and 3 of a backup piece, then specify copy 3 as the first item in the CATALOG list.</p>
CONTROLFILECOPY ' <i>filename</i> '	<p>Specifies the filename of a control file copy to be added to or updated in the RMAN repository. The control file copy can be:</p> <ul style="list-style-type: none"> ■ A copy of a normal control file (that is, not a standby control file) created with the RMAN command BACKUP AS COPY CURRENT CONTROLFILE command or the SQL statement ALTER DATABASE BACKUP CONTROLFILE ■ A standby control file copy created with the RMAN command BACKUP AS COPY STANDBY CONTROLFILE or the SQL statement ALTER DATABASE CREATE STANDBY CONTROLFILE
DATAFILECOPY ' <i>filename</i> '	Specifies the filename of a datafile copy to be added to or updated in the RMAN repository.
LEVEL = 0	For datafile copies only, indicates that the copy should be recorded as a level 0 incremental backup. You can perform incremental backups by using this datafile copy as the base level 0 backup.
(RECOVERY AREA DB_RECOVERY_FILE_DEST) [NOPROMPT]	Catalogs all valid backup sets, datafile copies, and archived redo logs in the flash recovery area. RMAN must be connected to the target database and the target database must be mounted or open. Specify NOPROMPT if you do not want to be prompted after every match. The keywords RECOVERY AREA and DB_RECOVERY_FILE_DEST are exact synonyms.
START WITH ' <i>string_pattern</i> ' [NOPROMPT]	<p>Catalogs all valid backups in locations that match the string pattern, which can be an Automatic Storage Management disk group, Oracle Managed Files directory, or part of a filename. RMAN will report any files in the disk location that cannot be cataloged. RMAN must be connected to the target database and the target database must be mounted. Specify NOPROMPT if you do not want to be prompted after every match.</p> <p>If the string pattern specifies a filename, then it matches the left part of the filename pattern. For example, /tmp/arc matches everything in directory /tmp/arc_dest and /tmp/archive/january as well as a file /tmp/arc.cpy.</p> <p>Note: You cannot use wildcard characters in the string pattern, only a strict prefix.</p>

Examples

Cataloging an Archived Redo Log: Example This example assumes that you made operating system copies of archived logs or transferred them from another location, and then added them to the RMAN repository:

```
CATALOG ARCHIVELOG '?/oradata/archive1_30.dbf', '?/oradata/archive1_31.dbf',  
                  '?/oradata/archive1_32.dbf';
```

Cataloging a File Copy as an Incremental Backup: Example The following example catalogs datafile copy users01.bak as an incremental level 0 backup:

```
CATALOG DATAFILECOPY '?/oradata/users01.bak' LEVEL 0;
```

Note that you can create datafile copies either using the RMAN `BACKUP AS COPY` command, or by using operating system utilities in conjunction with `ALTER TABLESPACE BEGIN/END BACKUP`.

Cataloging Multiple Copies in a Directory: Example The following example catalogs a directory full of archived logs that were copied into the `/tmp/arch_logs` directory with an operating system utility:

```
CATALOG START WITH '/tmp/arch_logs';
```

Cataloging Files in the Flash Recovery Area: Example The following example catalogs all files in the currently enabled flash recovery area without prompting the user for each one:

```
CATALOG RECOVERY AREA NOPROMPT;
```

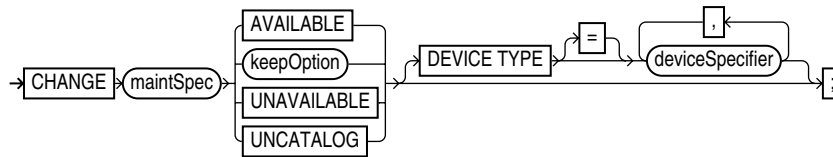
Cataloging Backup Pieces: Example The following example catalogs a backup piece that was copied with an operating system utility:

```
CATALOG BACKUPPIECE '?/oradata/01dmsbj4_1_1.bcp';
```

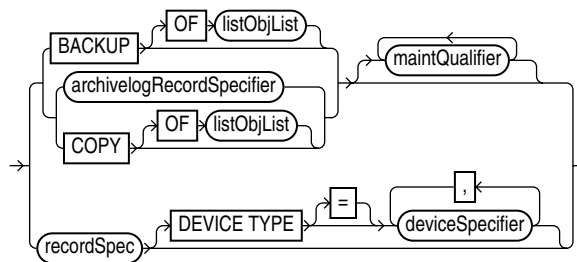
CHANGE

Syntax

change::=



maintSpec::=



Purpose

To make the following changes:

- To change the status of backups, copies, and archived logs in the repository to `AVAILABLE` or `UNAVAILABLE`. This feature is useful when a previously unavailable file is made available again, or you do not want a specific backup or copy to be eligible to be restored but also do not want to delete it.
- To alter the repository status of usable backups and copies from prior incarnations.
- To remove catalog records for backups and copies, and update the corresponding records in the target control file to status `DELETED`. This feature is useful when you remove a file by using an operating system command rather than the RMAN `CHANGE` command, and want to remove its repository record as well.
- To specify that a backup or copy should either abide by the currently configured retention policy or be exempt from it.

See Also: *Oracle Database Backup and Recovery Basics* to change the availability status of a backup or copy

Restrictions and Usage Notes

- The target instance must be started.
- The `KEEP FOREVER` clause requires use of a recovery catalog.
- You cannot use `CHANGE . . . UNAVAILABLE` or `KEEP` attributes for files stored in the flash recovery area.
- The only `CHANGE` command that requires either a manual or automatic maintenance channel is the `CHANGE . . . AVAILABLE` command. However, a maintenance channel is not required when `CHANGE . . . AVAILABLE` is used with a file that is disk only (that is, an `ARCHIVELOG`, `DATAFILECOPY`, or `CONTROLFILECOPY`).

If you use `CHANGE . . . AVAILABLE` on files that are not disk-only, and have objects created on device types that are not configured for automatic channels, then issue manual maintenance commands on these channels. For example, if you created a backup on an `sbt` channel, but have only a `DISK` channel automatically configured, then you must manually allocate an `sbt` channel before `CHANGE . . . AVAILABLE` can operate on the backup.

Keywords and Parameters

To obtain the primary keys of the records whose status you want to change, run a [LIST](#) command or query the recovery catalog views.

Syntax Element	Description
maintSpec	Specifies which files you want to <code>CHANGE</code> . Refer to " maintSpec " on page 2-147 for descriptions of the options in this clause.
<code>AVAILABLE</code>	Changes the status of a backup or copy to <code>AVAILABLE</code> in the repository. View the status in the <code>LIST</code> output or recovery catalog views.
keepOption	<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 make a backup that is currently exempt from the retention policy eligible for <code>OBSOLETE</code> status.</p> <p>You can also specify <code>KEEP</code> in the <i>backupSpec</i> clause.</p> <p>Note: You cannot use <code>keepOption</code> with flash recovery area files.</p> <p>See Also: "keepOption" on page 2-125</p>
<code>UNAVAILABLE</code>	Changes the status of a backup or copy to <code>UNAVAILABLE</code> in the repository. View the status in the <code>LIST</code> output or recovery catalog views. This option is provided for cases when the file cannot be found or has migrated offsite. RMAN does not use a file that is marked <code>UNAVAILABLE</code> in a RESTORE or RECOVER command. If the file is later found or returns to the main site, then use the <code>AVAILABLE</code> option to update its status.
<code>UNCATALOG</code>	<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 <code>DELETED</code>. The <code>CHANGE . . . UNCATALOG</code> 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 <code>DELETE</code> command.</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 <code>CHANGE... UNCATALOG</code> may reappear in the recovery catalog.</p>
<code>DEVICE TYPE</code> deviceSpecifier	Executes the <code>CHANGE</code> for the specified device type only (see " deviceSpecifier " on page 2-101). This option is valid only if you have configured automatic channels and have not manually allocated channels. For example, if you run <code>CHANGE UNCATALOG . . . DEVICE TYPE DISK</code> , then RMAN only uncatalogs files on disk.

Examples

Updating Backups to Status UNAVAILABLE: Example This example changes the status of backup set 100 as well as all backups of server parameter files created more than a day ago to `UNAVAILABLE`:

```
CHANGE BACKUPSET 100 UNAVAILABLE;
CHANGE BACKUP OF SPFILE COMPLETED BEFORE 'SYSDATE-1' UNAVAILABLE;
```

You do not need to allocate a maintenance channel.

Uncataloging and Cataloging Archived Logs: Example In this example, you move all archived logs to a new directory, uncatalog them, and then recatalog them in the new location:

```
HOST 'mv $ORACLE_HOME/oradata/trgt/arch/* /fs2/arch';  
CHANGE ARCHIVELOG ALL UNCATALOG;  
CATALOG START WITH '/fs2/arch';
```

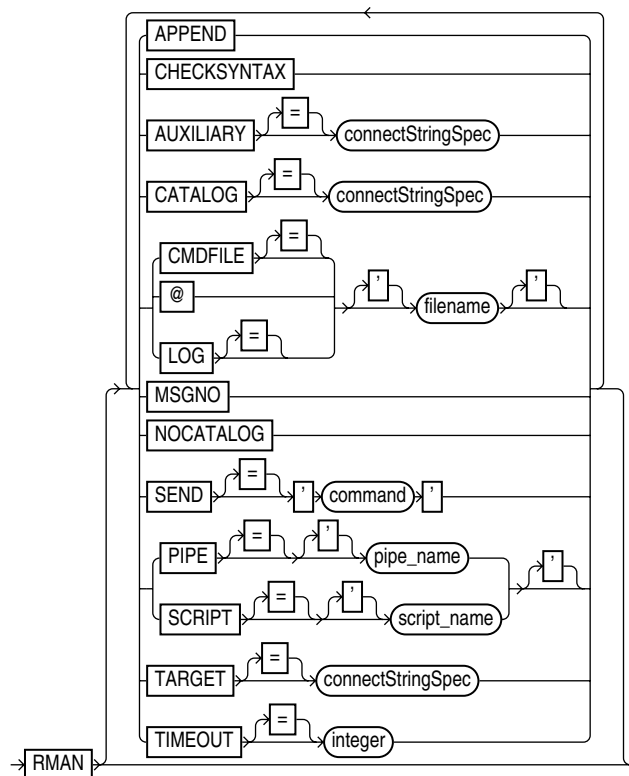
Changing the Retention Status of a Backupset: Example This example, which requires a recovery catalog, changes an ordinary backup into a long-term backup:

```
CHANGE BACKUP TAG 'consistent_db_bkup'  
KEEP FOREVER NOLOGS;
```

cmdLine

Syntax

cmdLine::=



Purpose

To start RMAN from the operating system command line. Use these arguments to:

- Connect to the target, recovery catalog, or auxiliary database.

Note: On some platforms, you may not want to connect at the operating system command line because the credentials can be 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 is an alternative method that avoids this problem.

- Specify whether you are using RMAN without a recovery catalog.
- Run a command file (text file containing commands) or stored script (from the recovery catalog) on startup, and exit on completion, instead of starting an interactive session.
- Start an interactive session which only checks commands entered for correct syntax, or perform a syntax check on the contents of a command file.
- Specify the file in which RMAN records the results of processed commands.

- Append output to the existing RMAN log file.
- Send a command to the media manager.
- Cause RMAN to print message numbers in the RMAN output.

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

See Also: *Oracle Database Backup and Recovery Basics* to learn how to connect RMAN to database instances

Restrictions and Usage Notes

Use these arguments at the operating system command line rather than at the RMAN prompt.

Keywords and Parameters

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.
AUXILIARY = <i>connectStringSpec</i>	Specifies a connect string to an auxiliary database, for example, AUXILIARY SYS/change_on_install@dupdb. See Also: " <i>connectStringSpec</i> " on page 2-77
CATALOG = <i>connectStringSpec</i>	Specifies a connect string to the database containing the recovery catalog, for example, CATALOG rman/rman@inst2. See Also: " <i>connectStringSpec</i> " on page 2-77
CATALOG = <i>connectStringSpec</i>	Specifies a connect string to the database containing the recovery catalog, for example, CATALOG rman/rman@inst2. See Also: " <i>connectStringSpec</i> " on page 2-77
CHECKSYNTAX	Causes RMAN to start in a mode where commands entered are checked for syntax errors, but no other processing is performed. 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.
CMDFILE = ' <i>filename</i> '	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 runtime error during the execution phase. If no errors are found, then RMAN exits after the job completes. If the first character of the filename is alphabetic, then you can omit the quotes around the filename. The contents of the command file should be identical to commands entered at the RMAN prompt. Note: If you run a command file at the RMAN prompt rather than as an option on the operating system command line, then RMAN does <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.
@ <i>filename</i>	Equivalent to CMDFILE.

Syntax Element	Description
LOG = 'filename'	<p>Specifies the file where RMAN records its output, that is, the commands that were processed and their results. If you do not specify this argument, then RMAN writes its message log file to standard output. The RMAN output is also stored in the V\$RMAN_OUTPUT view (a memory-only view for jobs in progress) and in V\$RMAN_STATUS (a control file view for completed jobs and jobs in progress).</p> <p>The LOG parameter does not cause RMAN to terminate if the specified file cannot be opened. Instead, RMAN writes to standard output.</p>
MSGNO	Causes RMAN to print message numbers, that is, RMAN-xxxx, for the output of all commands. By default, RMAN does not print the RMAN-xxxx prefix.
NOCATALOG	<p>Indicates that you are using RMAN without a recovery catalog.</p> <p>Note: If you do not specify either CATALOG or NOCATALOG on the command line, then RMAN defaults to NOCATALOG mode when it requires a repository connection (assuming that you have not issued CONNECT CATALOG).</p>
SEND = 'command'	<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 2-197</p>
PIPE = 'pipe_name'	<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 PIPE parameter. For example, you can invoke the RMAN pipe interface with the following options: PIPE rpi TARGET SYS/pwd@tdb.</p> <p>RMAN opens the following pipes in the target database:</p> <ul style="list-style-type: none"> ORA\$RMAN_RPI_IN, which RMAN uses to receive user commands ORA\$RMAN_RPI_OUT, which RMAN uses to send all output <p>All messages on both the input and output pipes are of type VARCHAR2.</p> <p>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> to learn how to pass commands to RMAN through a pipe</p>
SCRIPT = 'script_name'	<p>Once connected to the target database and recovery catalog (which must be specified using the TARGET and CATALOG options), RMAN will run the named stored script from the recovery catalog against the target database. If there are both a global script and a local stored script on the target database with the name <i>script_name</i>, RMAN will run 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. You should avoid creating script names that begin with a number or that match RMAN reserved words.</p> <p>See "CREATE SCRIPT" on page 2-91 for more details about stored scripts.</p>
TARGET = <i>connectStringSpec</i>	<p>Specifies a connect string to the target database, for example, TARGET SYS/mypassword@inst1.</p> <p>See Also: "connectStringSpec" on page 2-77</p>
TIMEOUT = <i>integer</i>	<p>Causes RMAN to exit automatically if it does not receive input from an input pipe within <i>integer</i> seconds. The PIPE parameter must be specified when using TIMEOUT.</p> <p>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> to learn how to pass commands to RMAN through a pipe</p>

Examples

Connecting Without a Recovery Catalog: Example This example connects to the target database prod1 without a recovery catalog:

```
% rman TARGET SYS/oracle@inst1 NOCATALOG
```

Connecting in Default NOCATALOG Mode: Example This example connects to the target database `prod1` 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;
```

Connecting to an Auxiliary Instance: Example This example connects to target database `prod1`, recovery catalog database `rcat`, and auxiliary instance `aux1`:

```
% rman TARGET SYS/sys_pwd@prod1 CATALOG rman/rman@rcat AUXILIARY sys/aux_pwd@aux1
```

Specifying a Command File: Example This example connects to the target database `prod1` and the recovery catalog database `rcat`, and then runs the command file `b_whole_10.rcv`:

```
% rman TARGET SYS/sys_pwd@prod1 CATALOG rman/rman@rcat @'/oracle/dbs/b_whole_10.rcv'
```

Syntax Check of a Command File: Example This example checks syntax of the contents of a command file `b_whole_10.rcv`:

```
% rman CHECKSYNTAX @'b_whole_10.rcv'
```

Syntax Check in an Interactive Session: Example This example starts an interactive session to perform syntax checking:

```
% rman CHECKSYNTAX
```

Specifying a Stored Script: Example This example connects to the target database `prod1` and the recovery catalog database `rcat`, and then runs the stored script `full_backup`:

```
% rman TARGET SYS/sys_pwd@prod1 CATALOG rman/rman@rcat SCRIPT full_backup
```

Specifying a Message Log in Append Mode: Example This example connects to the target database `prod1` without a recovery catalog and then specifies that RMAN should append messages to the message log:

```
% rman TARGET / NOCATALOG LOG = $ORACLE_HOME/dbs/log/msglog.f APPEND
```

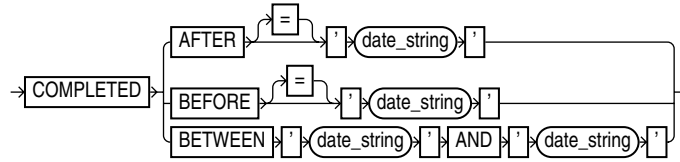
Invoking the RMAN Pipe Interface: Example This example invokes the RMAN pipe `newpipe` with a 90 second timeout option:

```
% rman PIPE newpipe TARGET SYS/oracle@inst1 TIMEOUT = 90
```


completedTimeSpec

Syntax

completedTimeSpec::=



Purpose

A subclause that specifies when a backup or copy completed.

Restrictions and 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 technique specifies dates independently of the current Global Technology environment variable settings.

Note: In Oracle8i, the FROM/UNTIL . . . TIME syntax in the LIST, CROSSCHECK, and DELETE commands was replaced with *completedTimeSpec*. If you are adapting an RMAN script from before Oracle8i for use in the current release, then you must update these commands for the script to work correctly.

Keywords and Parameters

Syntax Element	Description
AFTER 'date_string'	Specifies the time after which the backup was completed.
BEFORE 'date_string'	Specifies the time before which the backup was completed.
BETWEEN 'date_string' AND 'date_string'	Specifies a time range during which the backup was completed. Note that BETWEEN 'date1' AND 'date2' is exactly equivalent to AFTER 'date1' BEFORE 'date2'.

Examples

Crosschecking Backups Within a Time Range: Example This example crosschecks the backup sets of the database made last month:

```
CROSSCHECK BACKUP OF DATABASE COMPLETED BETWEEN 'SYSDATE-62' AND 'SYSDATE-31';
```

Deleting Expired Backups: Example This example deletes expired backup sets of datafile 1 made in the last two weeks:

```
DELETE EXPIRED BACKUP OF DATAFILE 1 COMPLETED AFTER 'SYSDATE-14';
```

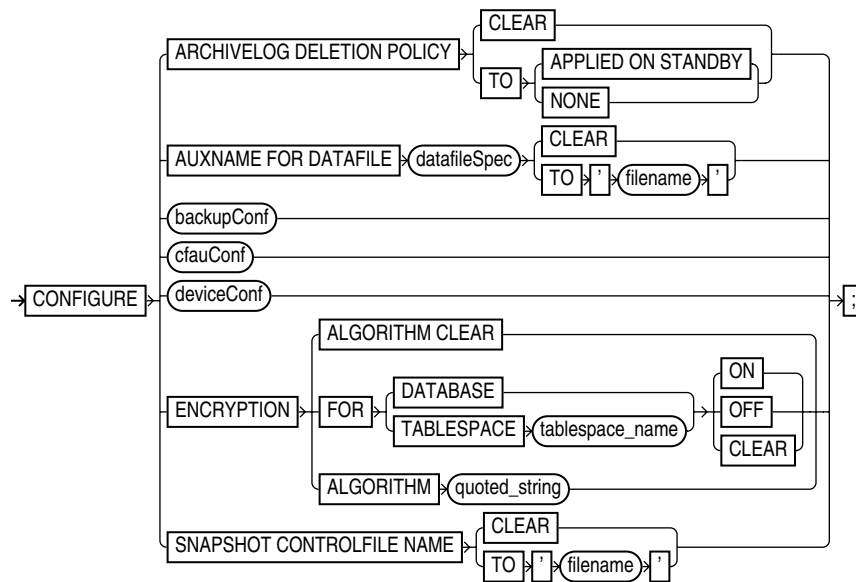
Listing Copies: Example This example lists image copies of datafile `*/oradata/trgt/users01.dbf` made before September 27, 2001:

```
LIST COPY OF DATAFILE '*/oradata/trgt/users01.dbf'  
COMPLETED BEFORE '27-SEP-01';
```

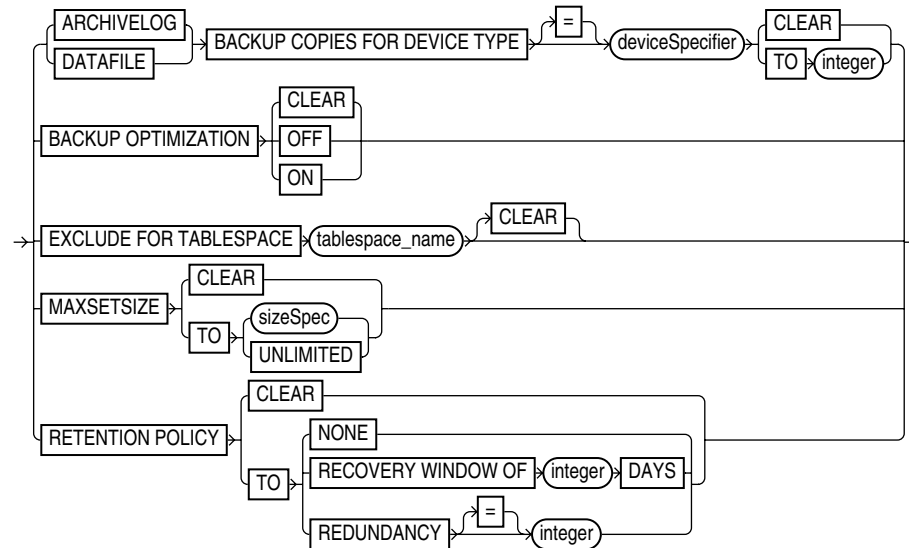
CONFIGURE

Syntax

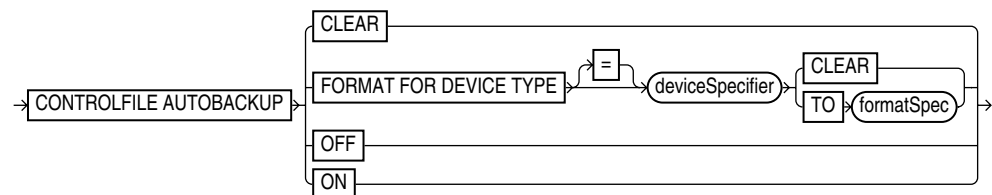
configure::=



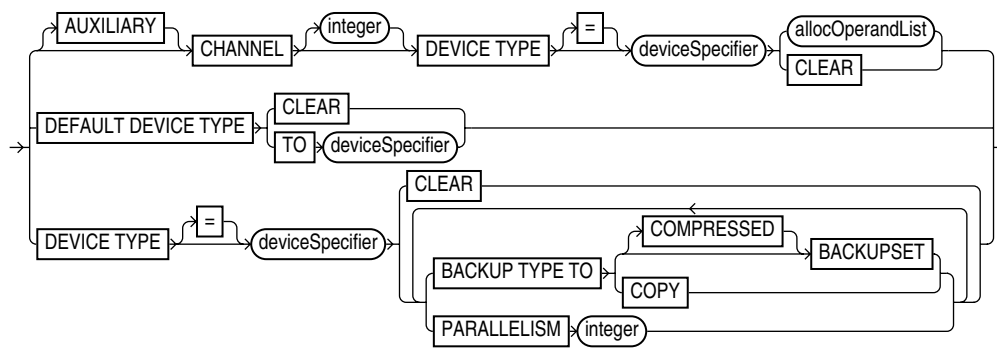
backupConf::=



cfauConf::=



deviceConf::=



Purpose

To configure persistent settings affecting RMAN backup, restore, duplication, and maintenance jobs. These configurations are in effect for any RMAN session until the configuration is cleared or changed.

Use `CONFIGURE` to set the following:

- An ongoing retention policy that automatically determines which backups and copies are eligible for deletion because they are no longer needed
- The device type (for example, `DISK` or `sbt`) for RMAN jobs
- The default number of channels of each device type that RMAN should allocate for automated backup and restore jobs
- The settings for automatic channels for a specified device type
- The maximum size of backup pieces and sets created on automatic channels
- Backup optimization either `ON` or `OFF`
- The exclusion policy for tablespaces in whole database backups
- The filename of the snapshot control file
- Filenames for files in an auxiliary database
- The control file autobackup feature to `ON` or `OFF`
- The default format for the control file autobackup output files

RMAN uses default settings for `CONFIGURE` options. You can return to the default value for any `CONFIGURE` command by running the same command with the `CLEAR` option.

See Also: *Oracle Database Backup and Recovery Basics* to learn how to configure the RMAN environment

Restrictions and Usage Notes

- Execute this command at the RMAN prompt. `CONFIGURE` cannot be used within a `RUN` block.
- The target database must be mounted or open, because configuration settings are stored in the control file.
- Channels allocated with `ALLOCATE CHANNEL` override any configured automatic channels.

- RMAN does not simultaneously allocate automatic channels for multiple device types in BACKUP command.
- To direct backups or restores to specific channels, use the RMAN-generated channel names. If you specify channel numbers in the CONFIGURE CHANNEL command, then RMAN uses the same numbers in the system-generated channel names.
- If you configure channels by using the nondefault CONNECT or PARMS options to create backups or copies, then you must either use the same configured channels or manually allocate channels with the same options to restore or crosscheck these backups.
- You cannot exclude the SYSTEM tablespace from whole database backups.
- The REDUNDANCY and RECOVERY WINDOW options are mutually exclusive. Only one type of retention policy can be in effect at any time.
- You cannot clear individual parameters when running CONFIGURE . . . CLEAR. For example, you can run CONFIGURE CHANNEL DEVICE TYPE sbt CLEAR but not CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 5M CLEAR.
- The channel number in a manually numbered channel must be less than 255.
- You must specify at least one channel option when running CONFIGURE CHANNEL. In other words, you cannot issue a command such as CONFIGURE CHANNEL 2 DEVICE TYPE DISK, but you can issue a command such as CONFIGURE CHANNEL 2 DEVICE TYPE DISK MAXPIECESIZE 2500K.
- The CONFIGURE CONTROLFILE AUTOBACKUP FORMAT format string must include the %F substitution variable. It cannot contain any other substitution variable.
- With Oracle Database Release 10g in a Data Guard environment, configurations can be set for standby databases as well as primary databases. All configurations except for retention policy, tablespace exclude and auxiliary names can be set to node-specific values. This means that the primary and standby databases can have different channel configurations, autobackup locations, and so on.

Keywords and Parameters

configure

Syntax Element	Description
<pre> ARCHIVELOG DELETIONPOLICYTO (APPLIED ON STANDBY NONE CLEAR) </pre>	<p>Governs archived redo log deletion policy for the flash recovery area. Possible settings are:</p> <ul style="list-style-type: none"> ■ APPLIED ON STANDBY - enables flash recovery area to delete archivelogs that are applied on mandatory standby. See <i>Oracle Data Guard Concepts and Administration</i> for details. ■ NONE - enables flash recovery area to delete archivelogs that are backed up to tertiary device and that are obsolete based on the configured backup retention policy. This is the default configuration. ■ CLEAR - clears the deletion policy and returns the specified configuration to default value. The default value is NONE.

Syntax Element	Description
<pre>AUXNAME FOR DATAFILE datafileSpec TO 'filename'</pre>	<p>Configures the auxiliary filename for the specified target datafile to 'filename'. For example, you can set the auxiliary name for datafile 2 to /df2.f, and then unspecify this auxiliary name by running <code>CONFIGURE AUXNAME FOR DATAFILE 2 CLEAR</code>.</p> <p>If you are performing TSPITR or running the <code>DUPLICATE</code> command, then by setting <code>AUXNAME</code> you can 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 <code>AUXNAME</code> parameter in TSPITR for the datafiles of the <code>SYSTEM</code> tablespace and the tablespaces containing rollback segments. Do not overlay files which are in use by the production database and can be discarded after TSPITR completes. In essence, the <code>AUXNAME</code> of a datafile is the location where TSPITR can create a temporary copy of it.</p> <p>When renaming files with the <code>DUPLICATE</code> command, <code>CONFIGURE AUXNAME</code> is an alternative to <code>SET NEWNAME</code>. The difference is that after you set the <code>AUXNAME</code> the first time, you do not need to reset the filename when you issue another <code>DUPLICATE</code> command: the <code>AUXNAME</code> setting remains in effect until you issue <code>CONFIGURE AUXNAME . . . CLEAR</code>. In contrast, you must reissue the <code>SET NEWNAME</code> command every time you rename files.</p> <p>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> to learn how to perform RMAN TSPITR, and <i>Oracle Database Backup and Recovery Advanced User's Guide</i> to learn how to duplicate a database with RMAN</p>
<code>backupConf</code>	Configures default backup options such as duplexing, optimization, excluding tablespaces, backup set sizes, and retention policies.
<code>cfauConf</code>	Configures control file autobackup settings
<code>deviceConf</code>	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	Used to specify encryption settings for the database or tablespaces within the database, which apply unless overridden using the <code>SET</code> command. Options specified for an individual tablespace take precedence over options specified for the whole database.
<pre>ALGORITHM { algorithm_name CLEAR }</pre>	<p>Specifies the default algorithm to use for encryption, when writing encrypted backup sets. Possible values are listed in <code>V\$RMAN_ENCRYPTION_ALGORITHMS</code>. With <code>CLEAR</code>, resets the database to the default algorithm, which is AES128.</p>

Syntax Element	Description
<pre>FOR { DATABASE TABLESPACE <i>tablespace_name</i> } { ON OFF CLEAR }</pre>	<p>Specifies whether to use encryption for the database or specified tablespaces. With FOR DATABASE, the effect is as follows:</p> <p>ON enables encryption for all database files, OFF turns off encryption for all database files CLEAR restores the default setting of OFF.</p> <p>Configured settings for a tablespace always override configuration set at the database level. With FOR TABLESPACE <i>tablespace_name</i>, the effect is as follows:</p> <p>ON means encryption is on for that tablespace, unless SET ENCRYPTION OFF FOR ALL TABLESPACES is used.</p> <p>OFF means encryption is off for that tablespace, unless SET ENCRYPTION ON FOR ALL TABLESPACES is used.</p> <p>CLEAR means that encryption for that tablespace is determined by the current default for the whole database.</p> <p>With ON, turns on encryption for the specified tablespaces or the whole database. With OFF turns off encryption for the specified tablespaces or the whole database. With CLEAR, returns the database or tablespaces to the default setting of OFF.</p>
<pre>SNAPSHOT CONTROLFILE NAME [TO '<i>filename</i>' CLEAR]</pre>	<p>Configures the snapshot control file filename to '<i>filename</i>'. If you run CONFIGURE SNAPSHOT CONTROLFILE NAME CLEAR, 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 you change the Oracle home, then the default location of the snapshot control file changes as well.</p> <p>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> for more information about snapshot control files</p>

backupConf

Syntax Element	Description
<pre>{ARCHIVELOG DATAFILE} BACKUP COPIES FOR DEVICE TYPE [=] <i>deviceSpecifier</i> [CLEAR TO <i>integer</i>]</pre>	<p>Specifies the number of copies of each backup set for DATAFILE (both datafiles and control files) or ARCHIVELOG files on the specified device type, from 1 (default) to 4. If duplexing is specified in the BACKUP command or in a SET BACKUP COPIES command, then the CONFIGURE setting is overridden.</p> <p>Note: Control file autobackups on disk are a special case and are never duplexed. RMAN always writes one and only copy.</p> <p>Note: RMAN raises an error if you try to duplex backups to the flash recovery area. You cannot duplex backups to the flash recovery area.</p>

Syntax Element	Description
<pre> BACKUP OPTIMIZATION [CLEAR OFF ON] </pre>	<p>Toggles backup optimization ON or OFF (default). Specify CLEAR to return optimization to its default value of OFF.</p> <p>Optimization does not back up a file to a device type if the identical file is already backed up on the device type. For two files to be identical, their content must be exactly the same. You can override backup optimization by using the FORCE option of the BACKUP command.</p> <p>RMAN does not signal an error if optimization causes all files to be skipped during a backup. Note that BACKUP . . . DELETE INPUT deletes all specified files whether or not optimization would skip these files during a backup.</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 ALL or LIKE options, or BACKUP BACKUPSET ALL. ■ The RMAN job uses a channel of only one device type. <p>The retention policy has an effect on which files backup optimization skips.</p> <p>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> for a description of how RMAN determines that it can skip the backup of a file</p>
<pre> EXCLUDE FOR TABLESPACE tablespace_name [CLEAR] </pre>	<p>Excludes the specified tablespace from BACKUP DATABASE commands. Note that you cannot exclude the SYSTEM tablespace. 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 "not excluded."</p> <p>You can still back up the configured tablespace by explicitly specifying it in a BACKUP command or by specifying the NOEXCLUDE option on a BACKUP DATABASE command.</p>
<pre> MAXSETSIZE [CLEAR TO [sizeSpec UNLIMITED] </pre>	<p>Specifies the maximum size of each backup set created on a channel. By default MAXSETSIZE is set to UNLIMITED, meaning that it is disabled.</p> <p>Note: This option is ignored by BACKUP AS COPY.</p>
RETENTION POLICY	<p>Specifies a persistent, ongoing policy for datafile and control file backups and copies that RMAN marks as obsolete, that is, not needed and eligible for deletion. As time passes, RMAN marks backups and copies as obsolete according to the criteria you specify in the retention policy. RMAN does not automatically delete any backups or copies: manually run the DELETE OBSOLETE command to remove obsolete files. By default, RETENTION POLICY is configured to REDUNDANCY 1.</p> <p>For backups, the basic unit of the retention policy is a backup set (not a backup piece) or image copy. For example, BACKUP AS BACKUPSET COPIES 4 TABLESPACE users generates 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>
CLEAR	Resets the retention policy to its default (REDUNDANCY = 1).
<pre> TO RECOVERY WINDOW OF integer DAYS </pre>	<p>Specifies a time window in which RMAN should be able to recover the database. The window stretches from the current time (SYSDATE) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is SYSDATE - integer days in the past.</p>

Syntax Element	Description
TO REDUNDANCY <i>integer</i>	Specifies that RMAN should retain <i>integer</i> backups or copies of each datafile and control file. If more than <i>integer</i> backups or copies exist, RMAN marks these extra files as obsolete. Then, RMAN determines the oldest of the retained backups and copies, and marks all archived logs and log backups older than this backup or copy as obsolete. The DELETE OBSOLETE command removes obsolete backups and copies as well as archived log backups and copies.
TO NONE	Disables the retention policy feature. RMAN does not consider any backups or copies as obsolete.

cfauConf

Syntax Element	Description
CONTROLFILE AUTOBACKUP	Controls the control file autobackup feature. By default, this feature is not enabled.
CLEAR	Returns the feature to its default setting of OFF.
FORMAT FOR DEVICE TYPE <i>deviceSpecifier</i> [CLEAR TO <i>formatSpec</i>]	<p>Configures the default filename format for the control file autobackup on the specified device type. If a flash recovery area is enabled, then RMAN creates the disk autobackup in the flash recovery area. Otherwise, RMAN creates it in an operating system specific location (? /dbs on Unix and Windows).</p> <p>By default, the initial format is %F for all devices. Any default format string specified with CONFIGURE must include the %F substitution variable.</p> <p>%F is the only legal substitution variable for use in a control file autobackup format. Use of any other substitution variable is an error.</p> <p>Specify CLEAR to return the format to the default %F.</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 2-121, for the semantics of the %F substitution variable.</p>
OFF	Disables the autobackup feature. (OFF is the default value.) When this command is OFF, any BACKUP command that includes datafile 1 (including BACKUP DATABASE) automatically includes the current control file and server parameter file in the backup set. Otherwise, RMAN does not include these files.

Syntax Element	Description
ON	<p>If <code>CONFIGURE CONTROLFILE AUTOBACKUP</code> is ON (by default it is OFF), then RMAN performs 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 database structural changes such as adding a new tablespace, altering the state of a tablespace or datafile (for example, bringing it online), adding a new online redo log, renaming a file, adding a new redo thread, and so forth. This type of autobackup, unlike autobackups that occur in the preceding circumstances, goes only to disk. You can run <code>CONFIGURE CONTROLFILE AUTOBACKUP FOR DEVICE TYPE DISK</code> to set a nondefault disk location. <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 the 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> (refer to entry for <code>CONFIGURE CONTROLFILE AUTOBACKUP FORMAT</code> for an explanation of this substitution variable). 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>

deviceConf

Syntax Element	Description
[AUXILIARY] CHANNEL [<i>integer</i>] DEVICE TYPE <i>deviceSpecifier</i>	<p>Specifies the standard or AUXILIARY channel that you are configuring or clearing, as well as the device type (DISK or sbt) of the channel. Either configure a generic channel or specify a channel number, where <i>integer</i> is less than 255.</p> <p>If you configure a generic channel (that is, if you do not specify a channel number), then RMAN uses the generic settings for every parallelized channel <i>except</i> any channel number that you have explicitly configured. A generic channel setting specifies options for all channels not configured explicitly.</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 MAXPIECESIZE setting of the first command.</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>See Also: <i>Oracle Database Backup and Recovery Advanced User's Guide</i> to learn how configure automatic channels specified by channel number</p>

Syntax Element	Description
<code>allocOperandList</code>	<p>Specifies control options for the allocated channel. 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: "<code>allocOperandList</code>" on page 2-13</p>
<code>CLEAR</code>	<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>
<code>DEFAULT DEVICE TYPE [TO deviceSpecifier CLEAR]</code>	<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>DISK</code>, then <code>RMAN</code> only allocates disk 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.</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>
<code>DEVICE TYPE [=] deviceSpecifier</code>	<p>Specifies the device type (disk or <code>sbt</code>) to which to apply the settings specified in this <code>CONFIGURE</code> command.</p>
<code>CLEAR</code>	<p>Resets backup type and parallelism settings for this device to their defaults..</p>
<code>BACKUP TYPE TO [COPY [COMPRESSED] BACKUPSET]</code>	<p>Configures the default backup type for disk or tape backups to either <code>BACKUPSET</code>, <code>COMPRESSED BACKUPSET</code> or <code>COPY</code>.</p> <p>For <code>sbt</code> devices the <code>COPY</code> option is not supported.</p> <p>The default for <code>DISK</code> is <code>BACKUPSET</code>.</p> <p>If the backup type is set to <code>BACKUPSET</code>, the <code>BACKUP</code> command always produces backup sets regardless of which media the backup is produced on. With the <code>COMPRESSED</code> option, the backupsets produced will use binary compression.</p> <p>The default location for disk backups is the flash recovery area, if one is configured; otherwise, backups are stored in a platform-specific location (for Unix and Windows, this is <code>\$ORACLE_HOME/dbs</code>). The default format for backup filenames is <code>%U</code>.</p>

Syntax Element	Description
<code>PARALLELISM integer</code>	<p>Configures the device types that are eligible for use in jobs that use automatic channels and sets the degree of channel parallelism (DISK is the default).</p> <p>The PARALLELISM parameter sets the number of automatic channels of the specified device type allocated for RMAN jobs. RMAN always allocates the number of channels set by PARALLELISM, although it may use only a subset of these channels.</p> <p>By default, PARALLELISM = 1. Specifying CLEAR for a device type resets its settings to the default. For example, you can set PARALLELISM for disk backups to 3. If you configure automatic channels of type disk and tape, and set the default device type as disk, then RMAN allocates three disk channels when you run BACKUP DATABASE at the RMAN prompt.</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> <p>Note: If you configure <i>n</i> manually numbered channels, 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>

Examples

Configuring Backup Optimization: Example This example configures RMAN so that the BACKUP command does not back up files to a device type if the identical file has already been backed up to the device type:

```
CONFIGURE BACKUP OPTIMIZATION ON;
```

Configuring a Retention Policy: Example This example configures a retention policy with a recovery window of 2 weeks, and then resets the retention policy to its default value of REDUNDANCY = 1:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 14 DAYS;
CONFIGURE RETENTION POLICY CLEAR;
```

Configuring Automatic Disk and Tape Channels: Example This example configures generic DISK and sbt channels, sets the default device type to sbt, and sets PARALLELISM to 3:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/?/%U';
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(NSR_SERVER=bksrv1)';
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
```

Overriding the Default Device Type: Example This example configures the default device type to sbt, backs up the archived logs on the default sbt channel, and then backs up the database to disk on the default disk channel:

```
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS 'ENV=(NSR_SERVER=bksrv1)';
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
BACKUP ARCHIVELOG ALL;
BACKUP DEVICE TYPE DISK DATABASE;
```

Configuring Automatic Channels Across File Systems: Example This example configures automatic disk channels across three file systems:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT '/disk1/backup/%U';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT '/disk2/backup/%U';
```

```
CONFIGURE CHANNEL 3 DEVICE TYPE DISK FORMAT '/disk3/backup/%U';
BACKUP DEVICE TYPE DISK DATABASE PLUS ARCHIVELOG;
```

Configuring Automatic Channels in an Oracle Real Application Clusters

Configuration: Example This example allocates automatic sbt channels for two nodes of an Oracle Real Application Clusters database:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'SYS/change_on_install@node1'
  PARS 'ENV=(NSR_SERVER=bkserv1)';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'SYS/change_on_install@node2'
  PARS ENV=(NSR_SERVER=bkserv2)';
```

Clearing Automatic Channels: Example This example clears manually numbered DISK channels 2 and 3 and the generic sbt channel:

```
CONFIGURE CHANNEL 2 DEVICE TYPE DISK CLEAR;
CONFIGURE CHANNEL 3 DEVICE TYPE DISK CLEAR;
CONFIGURE CHANNEL DEVICE TYPE sbt CLEAR;
```

Configuring and Clearing Parallelism: Example This example sets DISK parallelism to 2, then changes it to 3, then returns it to the default parallelism of 1:

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

Configuring Backup Copies: Example This example configures duplexing to 3 for DISK backups of datafiles and control files (control file autobackups on disk are a special case and are never duplexed) and then runs a database backup, specifying three different file systems for the copies:

```
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 3;
BACKUP DEVICE TYPE DISK DATABASE
  FORMAT '/disk1/backup/%U', '/disk2/backup/%U', '/disk3/backup/%U';
```

Configuring the Snapshot Control File Location: Example This example configures a new location for the snapshot control file and then resynchronizes the recovery catalog.

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '?/oradata/snap.cf';
```

Excluding a Tablespace from a Whole Database Backup: Example This example excludes the example tablespace from whole database backups, then returns the tablespace to its default value of "not excluded":

```
CONFIGURE EXCLUDE FOR TABLESPACE example;
CONFIGURE EXCLUDE CLEAR;
```

Specifying Auxiliary Filenames: Example This example duplicates a database to a remote host with a different directory structure, by using CONFIGURE AUXNAME to specify new filenames for the datafiles:

```
# 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 200K,
GROUP 2 ('?/dbs/dupdb_log_2_1.f',
        '?/dbs/dupdb_log_2_2.f') SIZE 200K 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;
```

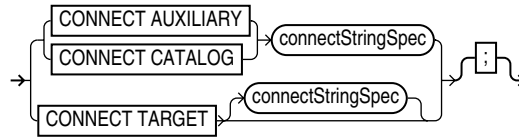
Specifying the Default Format for the Control File Autobackup: Example This example turns on the autobackup feature, then changes the default format for the DISK and sbt devices, then clears the autobackup setting:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '?/oradata/%F';
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'cf_auto_%F';
CONFIGURE CONTROLFILE AUTOBACKUP CLEAR; # returns to default setting of OFF
```

CONNECT

Syntax

connect::=



Purpose

To establish a connection between RMAN and a target, auxiliary, or recovery catalog database.

Note: When connecting from the command line, the password may be visible to other users on the system. The [CONNECT](#) command used from within RMAN avoids this problem.

See Also: ["cmdLine"](#) on page 2-57 for command line connection options

Restrictions and Usage Notes

- You can only run the `CONNECT TARGET`, `CONNECT CATALOG`, and `CONNECT AUXILIARY` commands if you are at the RMAN prompt and if you are not already connected to the specified databases.
- If you need to connect to a different target, catalog, or auxiliary database, then you must start a new RMAN session.
- You cannot use the `CONNECT CATALOG` command when RMAN is in the default `NOCATALOG` mode, that is, when these conditions are met:
 - You started RMAN without specifying either `CATALOG` or `NOCATALOG`.
 - You have already run a command such as `BACKUP` that requires a repository connection.

Keywords and Parameters

Syntax Element	Description
<code>CONNECT AUXILIARY</code> connectStringSpec	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 <code>TSPITR</code> .
<code>CONNECT CATALOG</code> connectStringSpec	Establishes a connection between RMAN and the recovery catalog database. You must run this command <i>before</i> running any command that requires a repository. Otherwise, RMAN defaults to <code>NOCATALOG</code> mode and invalidates the use of <code>CONNECT CATALOG</code> in the session.
<code>CONNECT TARGET</code> connectStringSpec	Establishes a connection between RMAN and the target database.

Examples

Connecting Without a Recovery Catalog: Example This example starts RMAN and then connects to the target database with an Oracle Net service name `prod1`:

```
% rman NOCATALOG
RMAN> CONNECT TARGET sys/change_on_install@prod1;
```

Connecting in the Default NOCATALOG Mode: Example This example starts RMAN and then connects to the target through Oracle Net. Because `BACKUP` is run and no `CONNECT CATALOG` has been run, RMAN defaults to NOCATALOG mode:

```
% rman
RMAN> CONNECT TARGET sys/change_on_install@prod1;
RMAN> BACKUP DATAFILE 7;
# You cannot run CONNECT CATALOG after this point because RMAN has defaulted to NOCATALOG
```

Connecting with a Recovery Catalog: Example This example starts RMAN and then connects to the target database `prod1` by using operating system authentication and the recovery catalog database `rcat` by using a password file:

```
% rman
RMAN> CONNECT TARGET / ;
RMAN> CONNECT CATALOG rman/rman@rcat;
```

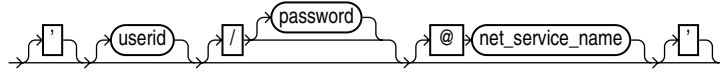
Connecting to Target, Recovery Catalog, and Auxiliary Databases: Example This example connects to three databases specifying a username and password for each:

```
% rman
RMAN> CONNECT TARGET SYS/sysdba@prod1;
RMAN> CONNECT CATALOG rman/rman@rcat;
RMAN> CONNECT AUXILIARY SYS/sysdba@dupdb;
```


connectStringSpec

Syntax

connectStringSpec::=



Purpose

A subclause specifying the username, password, and net service name for connecting to a target, recovery catalog, or auxiliary database. The connection is necessary to authenticate the user and identify the database.

Restrictions and Usage Notes

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

Keywords and Parameters

Syntax Element	Description
<code>/</code>	<p>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 <code>SYS</code> by using operating system authentication. For example, enter the following to connect to the target database:</p> <pre>% rman TARGET /</pre> <p>Note: The forward slash depends on platform-specific environment variables.</p>
<code>userid</code>	<p>Establishes a connection to the database for the specified user. If you do not specify a password, RMAN obtains the password interactively by displaying a prompt. The characters will not be echoed to the terminal.</p> <p>You must have <code>SYSDBA</code> authority when connecting to the target or auxiliary database, but must <i>not</i> connect as <code>SYS</code> to the recovery catalog database.</p> <p>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 (@).</p>
<code>/password</code>	<p>Establishes a connection for the specified user by using a password. If the target database is not open, then a password file must exist.</p>
<code>@net_service_name</code>	<p>Establishes a connection to the database through an optional Oracle Net service name.</p>

Examples

Connecting to a Target Database Without a Recovery Catalog: Example This example connects to the target database by using a password and the Oracle Net service name `prod1` in the default `NOCATALOG` mode:

```
% rman TARGET SYS/change_on_install@prod1
```

Connecting to a Target Database and Entering the Password Interactively: Example 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 10.1.0.2.0 - Production
```

```
Copyright (c) 1995, 2003, Oracle. All rights reserved.
```

```
target database Password:
```

Connecting with Operating System Authentication: Example This example starts RMAN and then connects to the target database `prod1` by using operating system authentication and the recovery catalog database `rcat` using a net service name:

```
% rman
```

```
RMAN> CONNECT TARGET /
```

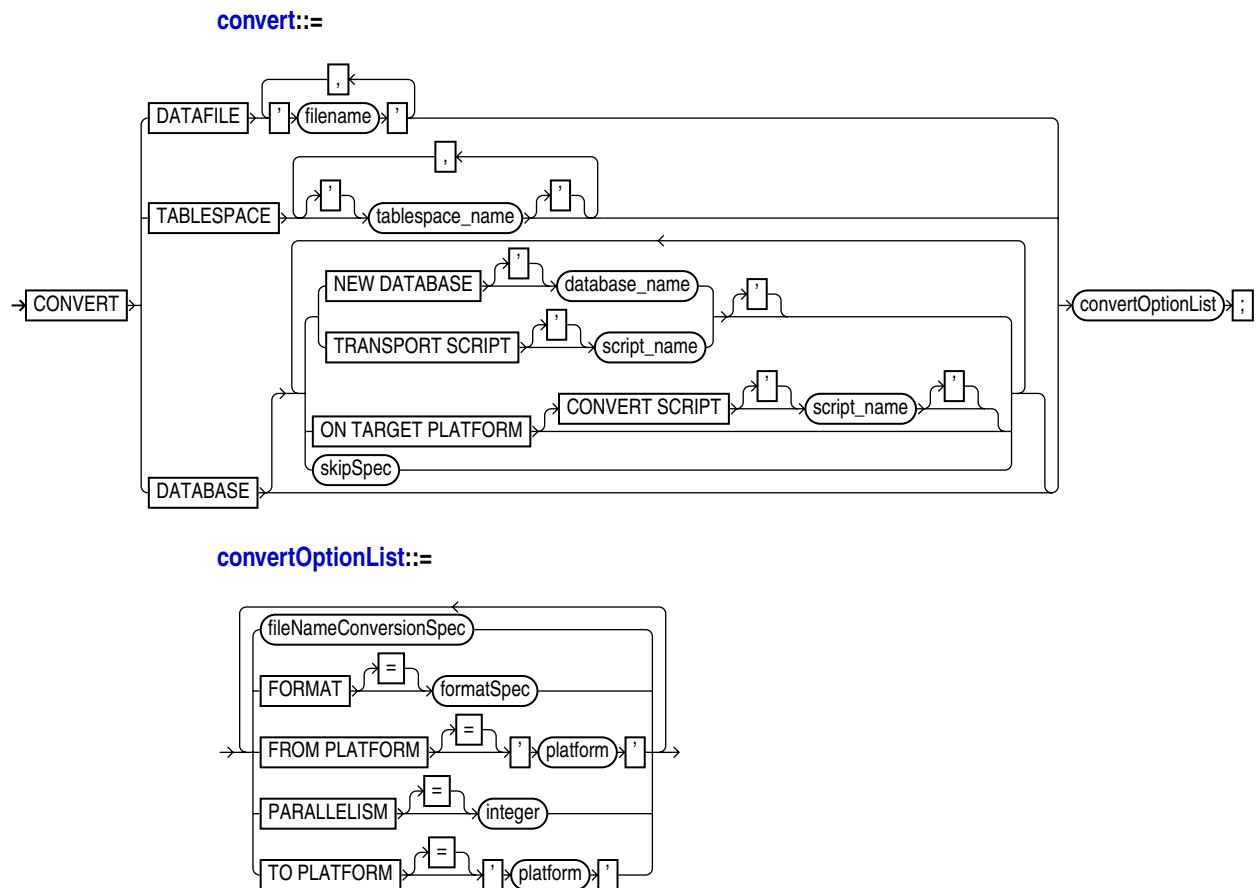
```
RMAN> CONNECT CATALOG rman/rman@inst1
```

Connecting to a Target Database, Recovery Catalog, and Auxiliary Instance: Example This example connects to three different databases from the command line, specifying a username, password, and net service name for each:

```
% rman TARGET SYS/pwd1@prod1 CATALOG rman/rman@rcat AUXILIARY SYS/pwd2@dupdb
```

CONVERT

Syntax



Purpose

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

- **CONVERT TABLESPACE** is used at the source database to produce datafiles for the specified tablespaces in the format of a different destination platform. The converted files can then be transported to the destination platform.
- **CONVERT DATAFILE** is used on the destination database to convert datafiles that are in the format of a different source platform. Once all of the datafiles required for a tablespace have been converted, the datafiles can be transported into the destination database.
- **CONVERT DATABASE** is used to transport an entire database from a source platform to a destination platform, converting the datafiles to the format of the destination platform and ensuring the creation of other required database files. Depending upon the requirements of your situation, **CONVERT DATABASE** on either the source or destination platform.

The following list describes some situations in which **CONVERT TABLESPACE** and **CONVERT DATAFILE** can be useful:

- Content providers can publish structured data as transportable tablespaces and distribute it to customers who can easily and efficiently integrate this data into their Oracle databases, regardless of their chosen platform.
- Data from a large data warehouse server can be distributed to data marts on smaller computers such as Windows-based workstations or servers.
- Read-only tablespaces can be shared across a heterogeneous cluster.

Note: The `CONVERT TABLESPACE` and `CONVERT DATAFILE` commands are only one part of a multiple-step process for transporting datafiles and tablespaces across platforms. You can transport datafiles and tablespaces using your live datafiles with the process described in *Oracle Database Administrator's Guide* or from backups using the process described in *Oracle Database Backup and Recovery Advanced User's Guide*. You should refer to that document before attempting to transport a tablespace across platforms.

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

`CONVERT DATAFILE` and `CONVERT TABLESPACE` can also be used to move files into and out of Automated Storage Management (ASM) disk groups. This capability is needed because native operating system file manipulation commands like Unix `cp` and Windows `COPY` cannot read from or write to ASM disk groups.

Restrictions and Usage Notes

This discussion includes the following sections:

- [Restrictions and Usage Notes on All Forms of RMAN CONVERT](#)
- [Restrictions and Usage Notes on CONVERT DATAFILE and CONVERT TABLESPACE](#)
- [Restrictions on CONVERT DATABASE](#)

Restrictions and Usage Notes on All Forms of RMAN CONVERT

The following restrictions apply to `CONVERT DATAFILE`, `CONVERT TABLESPACE` and `CONVERT DATABASE`:

- The input files are not altered by the `CONVERT` process. The conversion is not performed in place. Instead, converted files are written to a specified output destination.
- Both source and destination databases must be running with initialization parameter `COMPATIBLE` set to 10.0 or higher.
- Prior to Release 10g, CLOBs were created in a variable width character set and stored in an endian-dependent 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 will 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 Release 10g are stored in character set `AL16UTF16`, which is platform independent.

- CONVERT does not process user datatypes that require endian conversions. If you need to transport objects between databases that are built on underlying types that store data in a platform-specific format, then use the Data Pump Import and Export utilities.
- A tablespace must have been made read-write at least once in Release 10g before it can be transported to another platform. Hence, any read-only tablespaces (or currently existing transported tablespaces) that exist from an earlier release must be first made read-write at least once before they can be transported to a different platform. (It is sufficient to open the tablespace read-write and then immediately make it read-only again.)

Restrictions and Usage Notes on CONVERT DATAFILE and CONVERT TABLESPACE

The following usage notes apply to CONVERT DATAFILE and CONVERT TABLESPACE:

- 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 cannot convert individual datafiles on the source platform using CONVERT DATAFILE.

When converting on the destination host, use CONVERT DATAFILE . . . FROM to identify the source platform (and the destination platform is, implicitly, the platform of the destination host). You must use CONVERT DATAFILE, rather than CONVERT TABLESPACE, on the destination, because the destination database does not have any information to associate the datafiles being converted with tablespaces during the conversion process.

- 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, 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](#)" reference entry on page 2-41.
- Not all platforms support the use of CONVERT. Query V\$TRANSPORTABLE_PLATFORM to determine the platforms supported by the CONVERT command. Cross-platform tablespace transport is only supported when both the source and destination platforms are contained in this view.

The database has a list of its own internal names for each platform it runs on. You may need the exact name of the source or target platform as a parameter to the CONVERT command. Query V\$TRANSPORTABLE_PLATFORM to get the platform name from SQL*Plus as follows:

```
SQL> SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
       FROM V$TRANSPORTABLE_PLATFORM
       WHERE UPPER(PLATFORM_NAME) LIKE 'LINUX%';
```

Note: In Release 10g, the `CONVERT DATAFILE` or `CONVERT TABLESPACE` command is required when transporting datafiles between platforms for which the value in `V$TRANSPORTABLE_PLATFORM.ENDIAN_FORMAT` is different.

When transporting between platforms for which the `ENDIAN_FORMAT` column is the same, you do not need to use either `CONVERT DATAFILE` or `CONVERT TABLESPACE` command to convert the files. You can move or copy the file from the source to the destination with operating system utilities.

- Because you cannot use operating system utilities to move files into ASM, you may want to use `CONVERT TABLESPACE` or `CONVERT DATAFILE` to move files into ASM even if no change in endian format is required. Using `CONVERT` in this manner provides the equivalent of an operating system-level file copy command for copying files into ASM. `BACKUP AS COPY` provides similar functionality but catalogs the file copies created in the RMAN repository, which is only desirable if the file copies created in ASM are intended for use as backups at the target database.

Restrictions on CONVERT DATABASE

Because `CONVERT DATABASE` uses the same mechanism as `CONVERT TABLESPACE` and `CONVERT DATAFILE` to convert the datafiles of the database, the usage notes and restrictions in ["Restrictions and Usage Notes on CONVERT DATAFILE and CONVERT TABLESPACE"](#) on page 2-81 also apply to the conversion of databases.

The primary additional restriction on transporting entire databases is that the source and target platform must share the same endian format. For example, while you can transport a database from Microsoft Windows to Linux for x86 (both little-endian), or from HP-UX to AIX (both big-endian), you cannot transport a whole database from Solaris to Linux to x86 using this feature. You can, however, create a new database on a target platform manually, and transport individual tablespaces from the source database using the RMAN `CONVERT TABLESPACE` or `CONVERT DATAFILE` commands.

Note: In spite of the fact that the endian formats for the source and destination platform are the same, the datafiles for a transportable database must undergo a conversion process, 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.

Also note that some parts of the database are not transported directly:

- Redo log files and control files from the source database are not transported. New control files and redo log files are created for the target database during the transport process, and an `OPEN RESETLOGS` is performed once the new database is created.

Note: The control file for the converted database does not contain a copy of the RMAN repository information from the source database. Backups from the source database cannot be used with the converted database.

- BFILEs are not transported. RMAN provides a list of objects using the BFILE datatype in the output for the `CONVERT DATABASE` command, but users must copy the BFILEs themselves 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. RMAN provides a list of affected objects as part of the output of the `CONVERT DATABASE` command, but users must redefine these on the target platform. See *Oracle Database Administrator's Guide* for more information on managing external tables and directories.
- Password files are not transported. If a password file was used with the source database, the output of `CONVERT DATABASE` includes a list of all usernames and their associated privileges. Create a new password file on the target database using this information. See *Oracle Database Security Guide* for more information on managing password files.

Keywords and Parameters

convert

Syntax Element	Description
DATABASE	Used to transport entire databases across platforms (where the source and destination platforms have the same endian format).
NEW DATABASE <i>'database_name'</i>	Specifies the DB_NAME for the new database produced by the <code>CONVERT DATABASE</code> command.
ON TARGET PLATFORM	Specifies that any <code>CONVERT</code> commands required for datafiles should be performed on the destination platform rather than the source database. Useful if you do not want the overhead of the conversion on the source platform, or if you do not know the destination platform (for example, if you are publishing a transportable tablespace to be used by recipients with many different target platforms).
CONVERT SCRIPT <i>'script_name'</i>	Specifies the location of the file to contain the convert script generated by <code>CONVERT DATABASE... ON TARGET PLATFORM</code> . If not specified, the convert script is not generated.
skipSpec	Specifies that <code>CONVERT DATABASE</code> should skip inaccessible, offline or read-only files during the conversion process.
TRANSPORT SCRIPT <i>'script_name'</i>	Specifies the location of the file to contain the transport script generated by <code>CONVERT DATABASE</code> . If omitted, the transport script is not generated.
DATAFILE <i>datafile_name</i>	Specifies the name of a datafile that you want to transport into the destination database. <code>CONVERT DATAFILE</code> can only be used at the destination database.

Syntax Element	Description
<p>TABLESPACE</p> <p><i>tablespace_name</i></p>	<p>Specifies the name of a tablespace in the source database that you want to transport into the destination database on a different platform. CONVERT TABLESPACE can only be used when connected to the source database and converting on the source platform.</p>

convertOptionList

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.
FORMAT <i>formatSpec</i>	Specifies the name template for the output file(s). See the BACKUP AS COPY command for the format values that are valid here.
FROM PLATFORM = <i>platform_name</i>	Specifies the name of the source platform. This must be one of the platforms listed in the PLATFORM_NAME column of the V\$TRANSPORTABLE_PLATFORM view.
PARALLELISM [=] <i>integer</i>	Specifies the number of channels to be used to performing the operation. If not used, then channels allocated or configured for disk determine the number of channels.
TO PLATFORM = <i>platform_name</i>	Specifies the name of the destination platform as displayed in the V\$TRANSPORTABLE_PLATFORM.PLATFORM_NAME output. If used on the destination platform, then this must be the value for the destination platform in V\$TRANSPORTABLE_PLATFORM.PLATFORM_NAME.

Examples

Examples for Creating Transportable Tablespace Sets

The procedure for creating and using a transportable tablespace set is documented at length in *Oracle Database Administrator's Guide*. RMAN's CONVERT command is only required in cases where you are moving transportable tablespaces between platforms with different byte ordering. If your platforms have the same byte ordering, then you can either use CONVERT or copy the files directly.

The basic outline of the process is as follows:

1. Identify the tablespaces that will be transported. Depending on relations between objects in the tablespaces you want to transport and objects in other tablespaces, this may require careful planning.
2. Make the tablespaces to be moved read-only.
3. Use the Original Export utility to generate a file containing structural information from the data dictionary for the tablespaces to be transported. This file will be used when plugging the tablespaces into the destination database.
4. If you need to convert your datafiles for transport and you wish to use the source system's resources for the conversion, then use the RMAN CONVERT TABLESPACE command at this point on the source platform to convert the tablespaces for the target platform. See the first example following this outline. (If you prefer to use the destination system's resources for the conversion, do nothing in this step.)

5. Copy the datafiles (converted, if necessary) and the export dump file to the target database. You may move these by any means that is convenient: an operating system copy, ftp, or even distribution on removable media like CDs or tapes.
6. If you are transporting across platforms where endian conversion is required, and you did not perform the conversion on the source platform, then perform the conversion on the destination platform. See the second example following this outline.
7. Invoke the Original Import utility to plug the set of tablespaces into the target database.

For more details on this process, see *Oracle Database Administrator's Guide*. Read that discussion *in its entirety* before attempting any part of the tablespace transport process. The discussion in this document will focus on the specifics of using the CONVERT TABLESPACE and CONVERT DATAFILE commands.

Converting Tablespaces on the Source Platform: Example Suppose you need to convert tablespaces `finance` (datafiles `'/orahome/fin/fin01.dbf'` and `'/orahome/fin/fin02.dbf'`) and `hr` (datafiles `'/orahome/fin/hr01.dbf'` and `'/orahome/fin/hr02.dbf'`) from a source database running on a Sun Solaris host to a destination database running on an Linux PC. You plan to store the converted datafiles in the temporary directory `/tmp/transport_linux/` on the source host.

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

- You have set the tablespaces to be transported to be read-only.
- You have looked up the name for the destination platform in `V$TRANSPORTABLE_PLATFORM`.

The `PLATFORM_NAME` for Linux on a PC is `'Linux IA (32-bit)'`.

Now use RMAN to convert the datafiles to be transported to the destination host's format on the source host. The `FORMAT` argument controls the name and location of the converted datafiles.

```
% rman TARGET /
RMAN> CONVERT TABLESPACE finance,hr
      TO PLATFORM 'Linux IA (32-bit)'
      FORMAT='/tmp/transport_linux/%U';
```

The result is a set of converted datafiles in the `/tmp/transport_linux/` directory, with data in the right endian-order for the Linux IA (32-bit) platform.

From this point, you follow the rest of the general outline for tablespace transport. Use the 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_linux/` to the desired directories on the destination host, and plug the tablespace into the new database with the Import utility.

Converting Datafiles on the Target Platform: Example Suppose you need to transport tablespaces `finance` (datafiles `'/orahome/fin/fin01.dbf'` and `'/orahome/fin/fin02.dbf'`) and `hr` (datafiles `'/orahome/hr/hr01.dbf'` and `'/orahome/hr/hr02.dbf'`) from a source database running on a Sun Solaris host to a destination database running on an Linux PC host. You plan to perform conversion on the target host. You will temporarily store the unconverted datafiles in the directory `/tmp/transport_solaris/` on the target host. When the datafiles are plugged into the destination database, they will be stored in `/orahome/dbs`.

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

- You have set the source tablespaces to be transported to be read-only, used the Original Export utility to create the structural information file (named, in our example, `expdat.dmp`), gathered `expdat.dmp` and the unconverted tablespace datafiles to be transported, and copied these files to the destination host, in the `/tmp/transport_solaris/` directory. You have preserved the subdirectory structure from the files' original location, that is, the datafiles are stored as:
 - `/tmp/transport_solaris/fin/fin01.dbf`
 - `/tmp/transport_solaris/fin/fin02.dbf`
 - `/tmp/transport_solaris/hr/hr01.dbf`
 - `/tmp/transport_solaris/hr/hr02.dbf`

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

Note the following:

- You have to identify the datafiles by filename, not by tablespace name. Until the datafiles are plugged in, the local instance has no way of knowing the desired 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 using the `FROM` argument. Otherwise, RMAN will assume that the source platform is the same as the platform of the host performing the conversion.

```
% rman TARGET /
RMAN> CONVERT DATAFILE
      '/tmp/transport_solaris/fin/fin01.dbf',
      '/tmp/transport_solaris/fin/fin02.dbf',
      '/tmp/transport_solaris/hr/hr01.dbf',
      '/tmp/transport_solaris/hr/hr02.dbf'
DB_FILE_NAME_CONVERT
      '/tmp/transport_solaris/fin','/orahome/dbs/fin',
      '/tmp/transport_solaris/hr','/orahome/dbs/hr'
;
```

The result is a set of converted datafiles in the `/orahome/dbs/` directory, named thus:

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

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

Copying Datafiles To and From ASM Using `CONVERT DATAFILE`: Examples

The following example illustrates copying datafiles into ASM from normal storage. Use `CONVERT DATAFILE` without specifying a source or destination platform, and specifying 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 backup 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 backup 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 backup 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 backup at 03-JUN-05

```

Transporting Databases: Examples

Transporting databases is documented in *Oracle Database Backup and Recovery Advanced User's Guide*. Read that discussion *in its entirety* before attempting any part of the process of transporting a database. The discussion in this document will focus on the specifics of using the CONVERT DATABASE command.

The arguments to CONVERT DATABASE vary depending upon whether the conversion of the datafiles is performed 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 Advanced User's Guide*.

The following example illustrates using the CONVERT DATABASE command and converting the datafiles to the destination platform format on the source host:

```
RMAN> CONVERT DATABASE NEW DATABASE 'newdb'
      transport script '/tmp/convertedb/transportscript'
      to platform 'Microsoft Windows IA (32-bit)'
      db_file_name_convert '/disk1/oracle/dbs' '/tmp/convertedb'
      ;
```

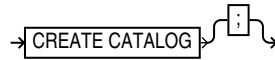
The following example illustrates using the CONVERT DATABASE command and converting the datafiles to the destination platform format on the destination host:

```
CONVERT DATABASE ON TARGET PLATFORM
  CONVERT SCRIPT '/tmp/convertedb/convertscript.rman'
  TRANSPORT SCRIPT '/tmp/convertedb/transportscript.sql'
  new database 'newdb'
  FORMAT '/tmp/convertedb/%U'
```

CREATE CATALOG

Syntax

createCatalog::=



Purpose

To create a schema for the recovery catalog. Typically, you create this schema in a separate recovery catalog database. The catalog is created in the default tablespace of the recovery catalog owner.

Note: In releases prior to 8.1.5, you created the recovery catalog schema by connecting to the recovery catalog database and executing the `catrman.sql` script.

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

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt.
- RMAN must be connected to the recovery catalog 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 must be granted the `RECOVERY_CATALOG_OWNER` role, and also be granted space privileges in the tablespace where the recovery catalog tables will reside.
- If you specify the tablespace name in the `CREATE CATALOG` command, *and* if the tablespace name is an RMAN reserved word (as listed in "[RMAN Reserved Words](#)" on page 1-3), then it *must* be uppercase and enclosed in quotes.
- Do not create the recovery catalog in the `SYS` schema.

See Also:

- *Oracle Database Administrator's Guide* for more information about the `RECOVERY_CATALOG_OWNER` role
- "[cmdLine](#)" on page 2-57 for information about RMAN command-line options

Keywords and Parameters

None

Example

Creating a Catalog Schema: Example

The following example creates a user `rman`, grants `rman` the `RECOVERY_CATALOG_OWNER` role, then creates the recovery catalog in the schema `rman.cattbs` of the database `rcat`:

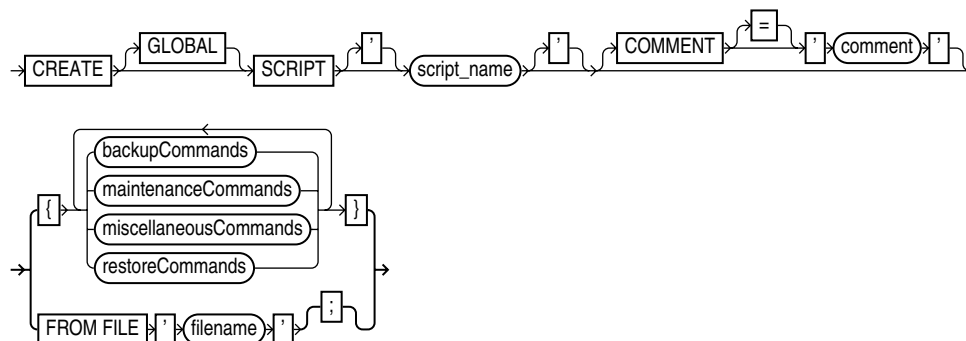
```
#!/usr/bin/tcsh
# create user rman in recovery catalog database as catalog owner
% sqlplus 'SYS/change_on_install@rcat AS SYSDBA'
SQL> CREATE USER rman IDENTIFIED BY rman
      DEFAULT TABLESPACE cattbs
      QUOTA UNLIMITED ON cattbs;
SQL> GRANT recovery_catalog_owner TO rman;
SQL> EXIT

# connect to database as catalog owner and create catalog
% rman CATALOG rman/rman@rcat
RMAN> CREATE CATALOG;
```

CREATE SCRIPT

Syntax

createScript::=



Purpose

To create a stored script in the recovery catalog.

A stored script is a sequence of RMAN commands, given a name and stored in the recovery catalog for later execution. A stored script may be local (that is, associated with one target database) or global (available for use with any database registered in the recovery catalog).

Any command that is legal within a [RUN](#) command is permitted in the stored script.

Several other commands are used with stored scripts:

- The [PRINT SCRIPT](#) command is used to view the contents of a stored script.
- The [REPLACE SCRIPT](#) command is used to update the contents of a stored script.
- The [EXECUTE SCRIPT](#) command is used to execute the commands in the stored script.
- The `SCRIPT` command line arguments for RMAN (described in "[cmdLine](#)" on page 2-57) runs a stored script automatically when starting RMAN.
- The [LIST SCRIPT NAMES](#) command is used to find out what stored scripts are defined for the current target database and recovery catalog.
- The [DELETE SCRIPT](#) command is used to delete a stored script from the recovery catalog.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide* to learn how to use stored scripts

Restrictions and Usage Notes

Note the following restrictions:

- Execute `CREATE SCRIPT` only at the RMAN prompt, not within a `RUN` block.
- When creating a script, RMAN must be connected to a target database and a recovery catalog, and the catalog database must be open.
- You cannot run `CREATE SCRIPT` once to create a local script, and then use this same script on multiple target databases. If you want to create a global script, you

must connect to some target database (as well as a recovery catalog) and then use `CREATE GLOBAL SCRIPT` instead of `CREATE SCRIPT`.

- You cannot execute a [RUN](#) command within a stored script.
- The `@` and `@@` commands do not work within `CREATE SCRIPT`.
- Quotes must be used around the script name when the name contains either spaces or reserved words.
- RMAN returns an error `RMAN-20401: script already exists` if you try to create a local script when another local script already exists for the same target database with the same name. The same error is returned if you try to create a global script and a global script already exists with the same name in the recovery catalog. If the script already exists, you must use `REPLACE SCRIPT` to update its contents.

Keywords and Parameters

Syntax Element	Description
<code>GLOBAL</code>	Identifies the script being created as global. If omitted, RMAN creates a local stored script <i>script_name</i> defined on the current target database. If no such script is defined on the target database, RMAN creates for a global stored script <i>script_name</i> .
<code>'script_name'</code>	The name of the script to create.
<code>COMMENT [=] 'comment'</code>	Associates an explanatory comment with the stored script in the catalog.
<code>FROM FILE 'filename'</code>	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 '{' and the last line must contain a '}'. The RMAN commands in the file must be valid in a stored script.
<i>backupCommands</i>	Commands valid in a stored script. The statements allowable within the brackets of the <code>CREATE SCRIPT 'script_name' {...}</code> command are the same commands supported within a RUN block. See "RUN" on page 2-194 for more details.
<i>maintenanceCommands</i>	
<i>miscellaneousCommands</i>	
<i>restoreCommands</i>	

Example

Creating a Local Stored Script: Example This example creates a stored script called `backup_whole` that backs up the database and archived redo logs:

```
# creates recovery catalog script to back up database and archived logs
CREATE SCRIPT backup_whole
COMMENT "backup whole database and logs"
{
    BACKUP INCREMENTAL LEVEL 0 TAG b_whole_l0
    DATABASE PLUS ARCHIVELOG;
}
```

Creating a Global Stored Script: Example This example creates a stored script called `backup_whole` that backs up the database and archived redo logs:

```
# creates recovery catalog script to back up database and archived logs
CREATE GLOBAL SCRIPT global_backup_db
COMMENT "backup any database from the recovery catalog, with logs"
{
    BACKUP DATABASE PLUS ARCHIVELOG;
}
```

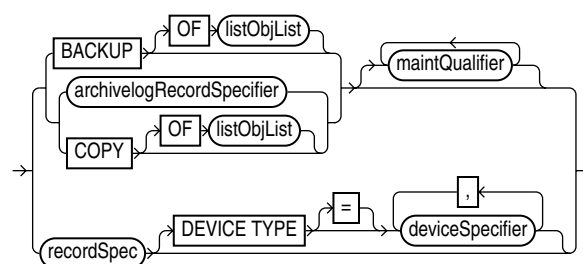

CROSSCHECK

Syntax

crosscheck::=

→ CROSSCHECK maintSpec ;

maintSpec::=



Purpose

To verify the status of backups and copies recorded in the RMAN repository against media such as disk or tape. The **CROSSCHECK** command only processes files created on the same device type as the channels running the crosscheck.

Status of RMAN Backups

The **CROSSCHECK** command checks only objects marked **AVAILABLE** or **EXPIRED** by examining the files on disk for **DISK** channels or by querying the media manager for **sbt** channels. [Table 2–2](#) describes the meaning of each status.

Table 2–2 *Meaning of Crosscheck Status*

Status	Description
EXPIRED	Object is not found either in file system (for DISK) or in the media manager (for sbt). Note that for a backup set to be EXPIRED , all backup pieces in the set must be EXPIRED . Note: EXPIRED does not mean the same as OBSOLETE .
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 .
UNAVAILABLE	Object is not available for use by RMAN. For a backup set to be UNAVAILABLE , all backup pieces in the set must have the status UNAVAILABLE .

The **CROSSCHECK** command does not delete any files that it is unable to find, but updates their repository records to **EXPIRED**. Then, you can run **DELETE EXPIRED** to remove the repository records for all expired files as well as any existing physical files whose records show the status **EXPIRED**.

If some backup pieces or copies were erroneously marked as **EXPIRED**, for example, because the media manager was misconfigured, then after ensuring that the files really do exist in the media manager, run the **CROSSCHECK BACKUP** command again to restore those files to **AVAILABLE** status.

See Also: *Oracle Database Backup and Recovery Basics* to learn how to manage target database records in the catalog

Restrictions and Usage Notes

- The target instance must be started.
- A maintenance channel is not required when CROSSCHECK is used with a file that is on disk. However, if you run CROSSCHECK on files stored on a media manager, and you have not configured automatic channels for the media manager, then you must manually allocate maintenance channels for these objects. For example, if you created a backup on an sbt channel, but have not configured automatic channels for your sbt device, then you must manually allocate an sbt channel before the CROSSCHECK command can check the backup.
- Crosscheck validates all specified backups and copies, even if they were created in prior incarnations (that is, before the most recent OPEN RESETLOGS).

Keywords and Parameters

Syntax Element	Description
maintSpec	Crosschecks files output by the BACKUP command. For <i>maintSpec</i> options, refer to the parameter descriptions in " maintSpec " on page 2-147.

Examples

Crosschecking All Backups and Copies: Example The following example, which assumes that the default configured channel is DEVICE TYPE sbt, queries the status of all backups and copies on tape and disk. Because RMAN preconfigures a disk channel, you do not need to manually allocate a disk channel:

```
CROSSCHECK BACKUP; # crosschecks backup sets and image copies
```

Crosschecking Within a Range of Dates: Example The following example queries the media manager for the status of the backup sets in a given six month range. Note that RMAN uses the date format specified in the NLS_DATE_FORMAT parameter, which is 'DD-MON-YY' in this example:

```
# if you manually allocate an sbt channel, then RMAN does not crosscheck disk
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;
CROSSCHECK BACKUP
  COMPLETED BETWEEN '01-JAN-01' AND '01-SEP-01';
RELEASE CHANNEL;

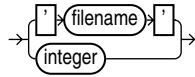
# to crosscheck only disk, specify CROSSCHECK DEVICE TYPE DISK
CROSSCHECK BACKUP DEVICE TYPE DISK
  COMPLETED BETWEEN '01-JAN-01' AND '01-SEP-01';

# assuming that the default channel is sbt, you can crosscheck on both disk
# and sbt backups by simply running CROSSCHECK with the default channels
CROSSCHECK BACKUP COMPLETED BETWEEN '01-JAN-01' AND '01-SEP-01';
```

datafileSpec

Syntax

datafileSpec::=



Purpose

A subclause that specifies a datafile by filename or absolute file number.

Restrictions and Usage Notes

- You can specify the relative or absolute path name.
- Double and single quotes are both legal (although only single quotes are shown in the diagram). Double quotes are recommended in the SQL command.
- Use ? to represent the Oracle home and @ for the Oracle SID.

See Also: ["Placeholders"](#) on page 1-2 to learn about the difference between single and double quotes, as well as the behavior of environment variables in RMAN quoted strings

Keywords and Parameters

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 port-specific manner by the target database.
<i>integer</i>	Specifies the datafile by using its absolute file number. Obtain the file number from the V\$DATAFILE, V\$DATAFILE_COPY, or V\$DATAFILE_HEADER views or REPORT SCHEMA command output.

Examples

Specifying a Datafile by Filename: Example This example copies datafile ?/oradata/trgt/users01.dbf to disk, specifying it by filename:

```

BACKUP AS COPY
  DATAFILE '?/oradata/trgt/users01.dbf'
  FORMAT '?/oradata/users01.cpy';

```

Specifying a Datafile by Absolute File Number: Example This example copies datafiles 3 and 4 to disk, specifying them by file number:

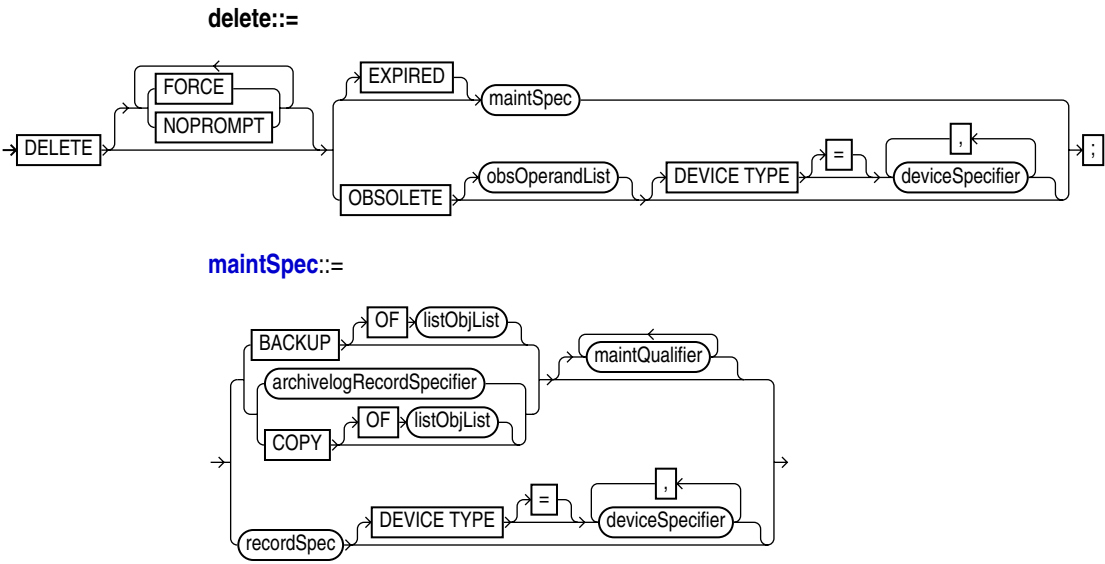
```

BACKUP AS COPY
  DATAFILE 3 FORMAT '?/oradata/df3.cpy',
  DATAFILE 4 FORMAT '?/oradata/df4.cpy';

```

DELETE

Syntax



Purpose

To delete physical backups and copies as well as do the following:

- Update their repository records in the target control file to status DELETED
- Remove their repository records from the recovery catalog (if you use a catalog)

When running RMAN interactively, DELETE displays a list of the files and prompts you for confirmation before deleting any file in the list. When reading commands from a command file, RMAN will not prompt for confirmation.

Relationship Between Repository and Media

The repository record for a backup can sometimes fail to reflect the physical status of the backup. For example, files backed up to disk can be deleted using an operating system utility. The record in the RMAN repository is not updated by the operating system utility, so the RMAN repository still shows the object as available even though it has been deleted. If you do not run the CROSSCHECK command to update the repository, and if you then run DELETE against the backup, then the repository shows that the object is AVAILABLE while the object is in fact missing. The following table indicates the behavior of DELETE in such situations.

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

Repository Status	Physical Status	Behavior of DELETE Command
FORCE	Any	Removes repository record and deletes object if it exists. All I/O errors are ignored. RMAN displays the number of objects deleted at the end of the job.

See Also: ["BACKUP"](#) on page 2-22 to learn about the BACKUP . . .
DELETE INPUT command

Restrictions and Usage Notes

- The target instance must be started.
- The DELETE command can delete usable backups and copies from prior incarnations.
- A maintenance channel is not required when DELETE is used with a file that is disk-only (that is, an ARCHIVELOG, DATAFILECOPY, CONTROLFILECOPY). Otherwise, you must use a manual or automatic maintenance channel.

If you use DELETE on files that are not disk-only, and if you have objects created on device types that are not configured for automatic channels, then run manual maintenance commands on these channels. For example, if you created a backup using an sbt channel, but have only a DISK channel automatically configured, you must manually allocate an sbt channel for DELETE.

Keywords and Parameters

Syntax Element	Description
FORCE	Deletes specified files (whether or not they exist on the media) and removes repository records. RMAN ignores any I/O errors for the deleted objects. RMAN displays the number of deleted objects at the end of the job.
NOPROMPT	Deletes specified files without first listing the files or prompting for confirmation. The DELETE NOPROMPT command still displays each item as it is deleted. By default, DELETE displays a list of files to be deleted and prompts for confirmation. If the user confirms, then RMAN shows each item as it is deleted. If you are running commands from a command file, then NOPROMPT is the default.
EXPIRED	Removes only files whose status in the repository is EXPIRED. RMAN marks backups and copies as expired when you run a CROSSCHECK command and the files are absent or inaccessible. To determine which files are expired, run a LIST EXPIRED command. If for some reason a backup or copy marked EXPIRED exists when you run the DELETE EXPIRED command, then RMAN deletes the physical files.
<i>maintSpec</i>	Deletes files output by the BACKUP command. For maintSpec options, refer to the parameter descriptions in "maintSpec" on page 2-147.

Syntax Element	Description
OBSOLETE	<p>Deletes backups and datafile copies recorded in the RMAN repository that are obsolete, that is, no longer needed. In addition to obsolete datafile backups, RMAN deletes obsolete archived logs and archived log backups. RMAN determines which backups and copies of datafiles are no longer needed, which in turn determines when logs (and backups of logs) are no longer needed. RMAN considers the creation of a datafile as a backup when deciding which logs to keep.</p> <p>RMAN first uses the options that you specify with <i>obsOperandList</i> to determine what is 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: Even if you use a <code>KEEP UNTIL</code> clause to specify a time after which a backup is to be considered obsolete, RMAN does not consider that backup obsolete if it is needed to satisfy the specified retention policy. A <code>KEEP UNTIL</code> clause can cause a backup to be kept longer than the retention policy requires, but never a shorter time.</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 2-149</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 2-101</p>

Examples

Deleting Expired Backups: Example The following 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 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';
```

Deleting Obsolete Backups: Example The following example deletes backups and copies that are not needed to recover the database to a random point within the last week. RMAN also deletes archived redo logs that are no longer needed:

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

Deleting Files That Have Already Been Backed Up: Example The following example deletes backups and copies (including archived redo logs) that have already been backed up at least twice to tape:

```
DELETE NOPROMPT BACKUP BACKED UP 2 TIMES TO DEVICE TYPE sbt;
DELETE NOPROMPT COPY BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

Forcing the Deletion of a Backup Set: Example The following example attempts to delete the backup set copy with tag `weekly_bkup`:

```
DELETE NOPROMPT BACKUPSET TAG weekly_bkup;
```

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

```
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
List of Mismatched objects
=====
Object Type      Filename/Handle
```

```
-----  
Backup Piece      01d270ud_1_1
```

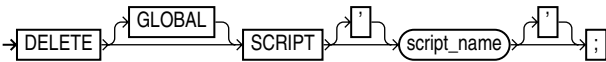
The following command forces RMAN to delete the backup set:

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

DELETE SCRIPT

Syntax

deleteScript::=



Purpose

To delete a local or global stored script from the recovery catalog.

Restrictions and Usage Notes

- Execute `DELETE SCRIPT` only at the RMAN prompt.
- RMAN must be connected to a recovery catalog and target database, and the catalog database must be open.
- To delete a local script, you must be connected to the target database for which the local script is defined.
- Quotes must be used around the script name when the name contains either spaces or reserved words.

Keywords and Parameters

Syntax Element	Description
GLOBAL	Specifies that the script to delete is a global stored script. Otherwise, RMAN will look for a local stored script called <i>script_name</i> defined on the current target database. (If no such script is defined on the target database, RMAN will check for a global stored script named <i>script_name</i> and delete that script if it exists.)
'script_name'	Specifies the name of the script to delete. See Also: "CREATE SCRIPT" on page 2-91, "EXECUTE SCRIPT" on page 2-112, "REPLACE SCRIPT" on page 2-167, and "LIST" on page 2-91 for LIST SCRIPT NAMES.

Example

Deleting a Script: Example The following example deletes a stored script `b_whole_10` from the recovery catalog:

```
rman TARGET / CATALOG rman/cat@catdb
RMAN> DELETE SCRIPT b_whole_10;
```

If a local stored script `b_whole_10` is defined, it is deleted. If no local stored script `b_whole_10` is defined but a global stored script `b_whole_10` is defined, the global script is deleted.

deviceSpecifier

Syntax

deviceSpecifier::=



Purpose

A subclause specifying the type of storage for a backup or copy.

Keywords and Parameters

Syntax Element	Description
DISK	Specifies disk storage device.
'media_device'	<p>Specifies a sequential I/O device or access method for storage. The syntax and semantics of sequential I/O device types are platform-specific. Example values are <code>sbt</code> and <code>sbt_tape</code> (with or without quotes). These values are synonymous.</p> <p>The <code>media_device</code> variable specifies a media manager. Media device names are case insensitive. The <code>sbt</code> variable is legal as input, but RMAN output always displays its synonym <code>sbt_tape</code>. It is stored in the catalog as <code>sbt_tape</code> for backward compatibility.</p>

Examples

Allocating a Tape Channel: Example This example allocates a maintenance channel for a media management device:

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

Backing Up to Disk: Example This example backs up the database to disk:

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

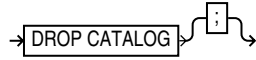
Restoring from Tape: Example This example restores archived logs from tape:

```
RESTORE DEVICE TYPE sbt ARCHIVELOG ALL;
```

DROP CATALOG

Syntax

dropCatalog::=



Purpose

To remove the schema from the recovery catalog.

Caution: This command deletes all RMAN repository data from the recovery catalog. If you have no backups of the catalog, then all backups of all databases managed by this recovery catalog become unusable.

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

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt.
- You must be connected to the recovery catalog database through the `CATALOG` command-line option or the [CONNECT CATALOG](#) command. The catalog database must be open. You do not have to be connected to the target database.
- Enter the command twice to confirm that you want to drop the schema.

Example

Deleting the Catalog: Example This example drops the schema from the recovery catalog (you must enter the command twice to confirm):

```
RMAN> DROP CATALOG
```

```
recovery catalog owner is RMAN
```

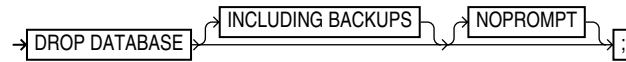
```
enter DROP CATALOG command again to confirm catalog removal
```

```
RMAN> DROP CATALOG
```

DROP DATABASE

Syntax

dropDatabase::=



Purpose

Deletes the target database and, if RMAN is connected to a recovery catalog, unregisters it. RMAN removes all datafiles, online logs, and control files belonging to the target database.

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt.
- You must be connected to the target database, which must be mounted exclusive and not open, and started in RESTRICT mode.
- If you want RMAN to delete archived logs, copies, and backups belonging to the database, then you must use the `DROP DATABASE INCLUDING BACKUPS` form of the command.

Keywords and Parameters

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

Example

Deleting a Database: Example In this example, you want to delete a test database called `test1` that is registered in the recovery catalog. You connect to `test1` which is started in RESTRICT mode, and delete the database files, as well as all backups, copies, and archived logs associated with the database:

```

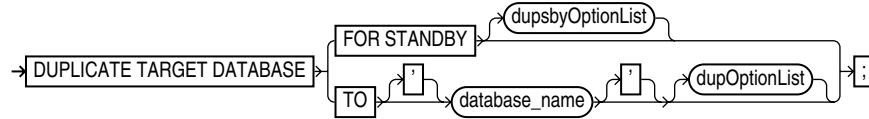
% sqlplus SYS/oracle@test1 AS SYSDBA;
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
SQL> QUIT;
% rman TARGET SYS/oracle@test1 CATALOG test1/test1@catdb
RMAN> DROP DATABASE INCLUDING BACKUPS NOPROMPT;

```

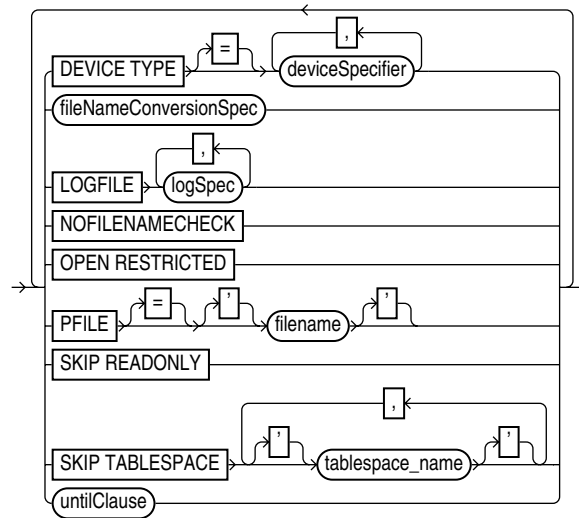
DUPLICATE

Syntax

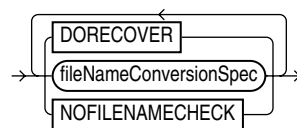
duplicate::=



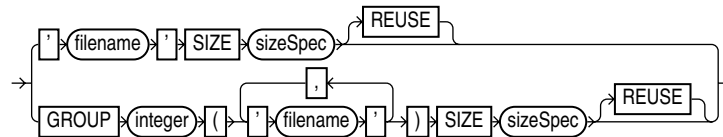
dupOptionList::=



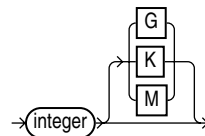
dupsbyOptionList::=



logSpec::=



sizeSpec::=



Purpose

To use backups (backup sets or image copies) of the target database to create either of the following:

- A **duplicate database**, which is a copy of the target database (or a subset of the target database) with a unique DBID. Because a duplicate database has a unique DBID, it is entirely independent of the primary database and can be registered in the same recovery catalog as the primary database. Typically, duplicate databases are used for testing.
- A **standby database**, which is a special copy of the primary database that is updated by applying archived redo logs from the primary database. A standby database does *not* get a new DBID.

To create a standby database with the `DUPLICATE` command you must specify the `FOR STANDBY` option. The `DUPLICATE . . . FOR STANDBY` command creates the standby database by restoring a standby control file, mounting the standby control file, and then restoring and recovering backups of the target datafiles. The standby database is left mounted after duplication is complete. Note that backups of the standby database are interchangeable with backups of the primary database.

When duplicating a database that is currently in `NOARCHIVELOG` mode, recovery occurs with the `NOREDO` option. Hence, if incremental backups exist, RMAN applies only these backups to the restored files during recovery. For databases in `ARCHIVELOG` mode, `DUPLICATE` recovers by default up to the last archived redo log generated at the time the command was executed, or until a time specified with a `SET UNTIL` clause.

See Also:

Oracle Database Backup and Recovery Advanced 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

Restrictions and Usage Notes

These restrictions apply to all uses of the `DUPLICATE` command (both for creation of a standby database and creation of a nonstandby duplicate database):

- The target SCN for a `DUPLICATE` command cannot be before the most recent `OPENRESETLOGS`. `DUPLICATE` to previous incarnations is not supported.
- Issue one or more `ALLOCATE AUXILIARY CHANNEL` commands before executing the `DUPLICATE` command, or [CONFIGURE](#) automatic auxiliary channels. RMAN uses the automatic target channel configuration for auxiliary channels in the following circumstances:
 - You have not manually allocated auxiliary channels.
 - You have not configured automatic auxiliary channels.
 - The automatic target channels do not have `CONNECT` strings.

The `DUPLICATE` command does not require non-`AUXILIARY` channels (that is, normal target database channels).

- You must be connected to both the target database and auxiliary instance. The auxiliary instance must be started with the `NOMOUNT` option, and the target database must be mounted or open. The target database cannot be a standby database.
- If you need to duplicate a database when some backups of the target database do not exist then you must specify `SKIP TABLESPACE`. If you do not specify `SKIP TABLESPACE`, then RMAN attempts to duplicate the following:

- All datafiles in online tablespaces, whether or not the datafiles are online.
- All tablespaces taken offline with an option *other than* `NORMAL`. For example, RMAN attempts to duplicate tablespaces taken offline with the `IMMEDIATE` option. You cannot duplicate `OFFLINE NORMAL` tablespaces, although you can add these tablespaces manually after duplication.

If no valid backups exist of any tablespace or datafile, then the `DUPLICATE` command fails.

- You can skip all tablespaces in the target database except the `SYSTEM` tablespace, undo tablespaces, and tablespaces containing rollback segments. 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.
- If the target and duplicate databases reside on the same host, set the `CONTROL_FILES` parameter appropriately so that the `DUPLICATE` command does not generate an error because the target control file is in use.
- If the target and duplicate databases share the same host, set all `*_PATH` and `*_DEST` initialization parameters appropriately so that the target database files are not overwritten by the duplicate database files.
- You cannot set the `DB_NAME` parameter in the duplicate parameter file to a value different from the database name specified in the `DUPLICATE` command.
- You cannot use the same database name for the target and duplicate databases when the duplicate database resides in the same Oracle home as the target. Note that if the duplicate database resides in a different Oracle home from the target, then its database name just has to differ from other database names in that same Oracle home.
- If the target and duplicate databases reside on different hosts, then you must do one of the following tasks for duplication to be successful:
 - Move backups and disk copies from the target host to the duplicate host to the same location as the target host so that the path names are identical
 - Move backups and disk copies from the target host to the duplicate host to a new location (so that the path names are different), and then [CATALOG](#) them.
 - Make sure that all backups and copies (disk or `sbt`) on the target host are remotely accessible from the duplicate host. Make sure that the archived redo logs are available in the expected location in the new host.
- Duplication must be done to the same platform as the source database.
- You cannot recover the duplicate database to the current point in time, that is, the most recent SCN. RMAN recovers the duplicate database up to or before the most recent available archived log; it cannot recover into the online redo logs.
- Specify new filenames or convert target filenames for the datafiles and online redo logs when the duplicate filenames must be different from the target filenames (as when duplicating to the same host as the primary). If you do not specify filenames for duplicate online redo logs and datafiles, then RMAN reuses the target datafile names.
- If you want the duplicate filenames to be the same as the target filenames, and if the databases are in different hosts, then you must specify `NOFILENAMECHECK`.
- If duplicating a database on the same host as the target database, do not specify the `NOFILENAMECHECK` option. Otherwise, RMAN may signal this error:

```

RMAN-10035: exception raised in RPC: ORA-19504: failed to create file
           "/oracle/dbs/tbs_01.f"
ORA-27086: skgfglk: unable to lock file - already in use
SVR4 Error: 11: Resource temporarily unavailable
Additional information: 8
RMAN-10031: ORA-19624 occurred during call to
DBMS_BACKUP_RESTORE.RESTOREBACKUPPIECE

```

The following restrictions and notes apply when you use the `DUPLICATE` command with the `FOR STANDBY` option:

- All backups and copies located on disk must be available at the standby host with the same path names as in the target host.
- Backups on tape must be accessible from the standby host.
- If archived logs have not been backed up, then archived logs must be available at the standby host with the same path names as in the target host.
- If RMAN recovers the standby database, then the checkpoint SCN of the control file must be included in an archived redo log that is either available at the standby site or included in an RMAN backup. For example, assume that you create the standby control file and then immediately afterward archive the current log, which has a sequence of 100. In this case, you must recover the standby database up to at least log sequence 100, or the database signals an ORA-1152 error message because the standby control file backup or copy was taken after the point in time.
- You cannot use `SET NEWNAME` or `CONFIGURE AUXNAME` to transform the filenames for the online redo logs on the standby database.
- You cannot use the `DUPLICATE` command to activate a standby database.
- You cannot connect to the standby database and then `DUPLICATE . . . FOR STANDBY` to create an additional standby database. To create additional standby databases, connect to the original *primary* database and run `DUPLICATE . . . FOR STANDBY`.
- Do not attempt to register the standby database in the primary database repository.
- When creating a standby or duplicate database and using Oracle Managed Files, tempfiles are re-created 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, `DB_FILE_NAME_CONVERT` is used 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 2-216.

Keywords and Parameters

duplicate

Syntax Element	Description
FOR STANDBY	Specifies that database being duplicated is to be used as a standby database. RMAN restores the most recent files, unless SET UNTIL is specified. If DORECOVER is specified, then RMAN also recovers database. RMAN always leaves standby database in mounted state after executing DUPLICATE command.
<i>dupsbyOptionList</i>	Specifies options that only apply when creating a standby database.
DORECOVER	Specifies that RMAN should recover the database after creating it. If you specify an <i>untilClause</i> , then RMAN recovers to the specified point and leaves the database mounted.
NOFILENAMECHECK	Prevents RMAN from checking whether target datafiles sharing the same names as the duplicated files are in use. Note that the NOFILENAMECHECK option is required when the standby and primary datafiles and logs have identical filenames. See Also: The description in <i>dupOptionList</i>
TO 'database_name'	Specifies the name of the duplicate database. The name should match the name in the initialization parameter file of the duplicate database or the database signals an error when creating the control file.

dupOptionList

Syntax Element	Description
<i>dupOptionList</i>	Specifies options that apply when creating a duplicate database not intended for use as a standby database.
DEVICE TYPE <i>deviceSpecifier</i>	Allocates automatic channels for the specific <i>deviceSpecifier</i> only (for example, DISK or sbt). 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 DUPLICATE . . . DEVICE TYPE DISK, then RMAN allocates only disk channels. See Also: "deviceSpecifier" on page 2-101
<i>fileNameConversionSpec</i>	Specifies one or more patterns to map original to duplicate filenames. Note that this parameter overrides the initialization parameter DB_FILE_NAME_CONVERT (if it is set). If a file in the specification list is not affected by the conversion parameter in DUPLICATE, then you must rename it by other means, such as SET NEWNAME. See Also: "fileNameConversionSpec" on page 2-114
LOGFILE <i>logSpec</i>	Specifies the online redo logs when creating a nonstandby duplicate database. The syntax is the same used in the LOGFILE option of the CREATE DATABASE statement. Refer to the description of <i>logSpec</i> for the legal options.

Syntax Element	Description
NOFILENAMECHECK	<p>Prevents RMAN from checking whether target datafiles sharing the same names as the duplicated files are in use. The user is responsible for determining that the duplicate operation will not overwrite useful data.</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 target database. For example, assume that you have a small database located in the /dbs directory of host1:</p> <pre>/oracle/dbs/system_prod1.dbf /oracle/dbs/users_prod1.dbf /oracle/dbs/tools_prod1.dbf /oracle/dbs/rbs_prod1.dbf</pre> <p>Assume that you want to duplicate the database in machine host2, which has the same file system /oracle/dbs/*, and you want to use the same filenames in the duplicate database as in the primary. In this case, specify the NOFILENAMECHECK option to avoid an error message. Because RMAN is not aware of the different hosts, RMAN cannot determine automatically that it should not check the filenames.</p>
OPEN RESTRICTED	Enables a restricted session in the duplicate database by issuing the following SQL statement: ALTER SYSTEM ENABLE RESTRICTED SESSION. RMAN issues this statement immediately before the duplicate database is opened.
PFILE = 'filename'	Specifies a client-side initialization parameter used by the auxiliary instance. RMAN automatically shuts down and restarts the auxiliary instance during duplication. If the auxiliary does not use a server-side parameter file in the default location, you must specify the client-side parameter file that RMAN should use when starting the auxiliary instance. Otherwise, you do not need to specify PFILE.
SKIP READONLY	Excludes datafiles in read-only tablespaces from the duplicate database. 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 CD-ROM, then mount the CD-ROM later and view the data.
SKIP TABLESPACE 'tablespace_name'	Excludes the specified tablespace from the duplicate database. Note that you cannot exclude the SYSTEM tablespace, undo tablespaces, and tablespaces with rollback segments.
<i>untilClause</i>	Sets the end point for incomplete recovery of the duplicate database. You can achieve the same result by running SET UNTIL before the DUPLICATE command. See Also: " untilClause " on page 2-227

logSpec

Syntax Element	Description
<i>logSpec</i>	<p>Specifies the online redo logs when creating a nonstandby duplicate database. 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 target log filenames for the duplicate files. You must specify the NOFILENAMECHECK option in this case.</p> <p>See Also: <i>Oracle Database SQL Reference</i> for CREATE DATABASE syntax</p>
'filename' SIZE integer	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.

Syntax Element	Description
<i>GROUP integer</i>	Specifies the group containing the online redo log members.

dupsbbyOptionList

Syntax Element	Description
<i>dupsbbyOptionList</i>	Specifies options that only apply when creating a standby database.
DORECOVER	Specifies that RMAN should recover the database after creating it. If you specify an <i>untilClause</i> , then RMAN recovers to the specified point and leaves the database mounted.
<i>fileNameConversionSpec</i>	Specifies how to convert original datafile names to new datafile names in the standby database. See Also: "fileNameConversionSpec" on page 2-114
NOFILENAMECHECK	Prevents RMAN from checking whether target datafiles sharing the same names as the duplicated files are in use. Note that the NOFILENAMECHECK option is required when the standby and primary datafiles and logs have identical filenames. See Also: The description in <i>dupOptionList</i> on page 2-108

Examples

Setting New Filenames Manually: Example This example assumes that the target datafiles are on host1 in directory /h1/oracle/dbs/trgt. You wish to duplicate the database to newdb on host2 in the directory /h2/oracle/oradata/newdb. The DUPLICATE command uses backup sets stored on tape to duplicate the target database to database newdb, and recovers it to a point 24 hours ago:

```

RUN
{
  ALLOCATE AUXILIARY CHANNEL newdb1 DEVICE TYPE sbt;
  DUPLICATE TARGET DATABASE TO newdb
    DB_FILE_NAME_CONVERT=('/h1/oracle/dbs/trgt/', '/h2/oracle/oradata/newdb/')
    UNTIL TIME 'SYSDATE-1' # specifies incomplete recovery
    SKIP TABLESPACE cmwlite, drsys, example # skip desired tablespaces
    PFILE = ?/dbs/initNEWDB.ora
    LOGFILE
      GROUP 1 ('?/oradata/newdb/redo01_1.f',
              '?/oradata/newdb/redo01_2.f') SIZE 200K,
      GROUP 2 ('?/oradata/newdb/redo02_1.f',
              '?/oradata/newdb/redo02_2.f') SIZE 200K
      GROUP 3 ('?/oradata/newdb/redo03_1.f',
              '?/oradata/newdb/redo03_2.f') SIZE 200K REUSE;
}

```

Reusing the Target Filenames: Example This example assumes the following:

- You are restoring to a new host without a catalog.
- You have configured automatic channels.
- The target host and duplicate host have the same file structure.
- You wish to name the duplicate files exactly like the target database files.
- You do not want to duplicate read-only tablespaces.
- You want to prevent RMAN from checking whether files on the target database that have the same names as the duplicated files are in use.

```
CONNECT TARGET
CONNECT AUXILIARY SYS/aux_pwd@newdb
DUPLICATE TARGET DATABASE TO ndbnewh
LOGFILE
    '?/dbs/log_1.f' SIZE 200K,
    '?/dbs/log_2.f' SIZE 200K
SKIP READONLY
NOFILENAMECHECK;
```

Creating a Standby Database: Example This example creates a standby database on a remote host with the same directory structure as the primary host. In this example, the `NOFILENAMECHECK` option is specified because the standby and primary datafiles and logs have the same names. Note that an automatic auxiliary channel is already configured, so you do not need to manually allocate a channel:

```
DUPLICATE TARGET DATABASE FOR STANDBY
NOFILENAMECHECK;
```

EXECUTE SCRIPT

Syntax

`executeScript::=`



Purpose

To run a local or global RMAN script stored in the recovery catalog.

See Also: ["CREATE SCRIPT"](#) on page 2-91 and *Oracle Database Backup and Recovery Advanced User's Guide* for more details about stored scripts

Restrictions and Usage Notes

- Use EXECUTE SCRIPT only within the braces of a [RUN](#) command.
- RMAN must be connected to the catalog with the CATALOG command-line option or the [CONNECT CATALOG](#) command, and the catalog must be open.
- For a local script, RMAN must be connected to the target database for which the local script is created.
- 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.

Keywords and Parameters

Syntax Element	Description
GLOBAL	Specifies the execution of a global stored script instead of a local one.
'script_name'	<p>The name of the stored script to execute.</p> <p>If no local stored script defined for the current target database is found with the name specified, RMAN searches for a global script by the same name and executes it if one is found.</p> <p>See Also: "LIST" on page 2-127 for more information about listing the scripts stored in the recovery catalog, and "CREATE SCRIPT" on page 2-91 for information about creating scripts</p>

Example

Executing a Script: Example This example runs a stored script called backup_whole_10:

```
RUN { EXECUTE script backup_whole_10; }
```

EXIT

Syntax

exit::=

→ EXIT →

Purpose

To shut down the Recovery Manager utility.

Restrictions and Usage Notes

Execute only at the RMAN prompt.

Example

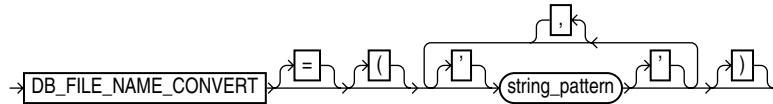
Exiting RMAN: Example This example starts RMAN and then shuts it down:

```
% rman
RMAN> EXIT
```

fileNameConversionSpec

Syntax

fileNameConversionSpec::=



Purpose

A subclause that specifies one or more patterns to be used in generating new database file names based on old ones. Used with **BACKUP**, **CONVERT** and **DUPLICATE** as one way of generating output file names.

Restrictions and Usage Notes

The rules for these 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 a new filename is generated based on an old one, the original filename is compared to the first member of each pair of string patterns. The first time a pattern is found which is a substring of the original filename, the new filename is generated by substituting the second member of the pair for the substring that matched.

Set the `string_pattern` to a value such as:

```
DB_FILE_NAME_CONVERT = ('string1' , 'string2' , 'string3' , 'string4' ...)
```

Where:

- *string1* is a pattern matching the original filename
- *string2* is the pattern replacing *string1* in the generated filename
- *string3* is a pattern matching the original filename
- *string4* is the pattern replacing *string3* in the generated filename

You can use as many pairs of primary and standby replacement strings as required.

For example, when making image copy backups of tablespaces `users` (with datafiles in directory `/disk1/dbs/users`) and `tools` (with datafiles in `/disk1/dbs/tools/`), to direct the converted datafiles to `/newdisk/users` and `/newdisk/tools` respectively, use the `DB_FILE_NAME_CONVERT` pattern shown here:

```
BACKUP AS COPY TABLESPACE users, tools
  DB_FILE_NAME_CONVERT = ('disk1/dbs','newdisk');
```

For each datafile to be converted where `'disk1/dbs'` is a substring of the filename, the new filename is created by replacing `'disk1/dbs'` with `'newdisk'`. For example, the converted datafile corresponding to `/disk1/dbs/users/users01.dbf` is stored in `/newdisk/users/users01.dbf`, the converted datafile corresponding to `/disk1/dbs/tools/tools01.dbf` is stored in `/newdisk/tools/tools.dbf`, and so on.

Be aware of the following details:

- The pattern does not have to match at the beginning of the filename. In the previous example, the match of the pattern to the original filename began at the second character. The command

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

would direct the image copies to /disk1/newdbs/users and /disk1/newdbs/tools.

- When there are multiple possible matches for a given filename being converted, the first match in the list of patterns is used to generate the new filename. The command

```
BACKUP AS COPY TABLESPACE users
    DB_FILE_NAME_CONVERT = ('dbs', 'newdbs', '/disk1', '/newdisk');
```

would have the same effect as the previous example, because the pattern 'dbs' matches the filename and there is no reason to compare it to the second pattern '/disk1'.

Keywords and Parameters

Syntax Element	Description
<i>'string_pattern'</i>	Specifies the pattern, consisting of the pairs of strings used to convert the filenames..

Examples

Using DB_FILE_NAME_CONVERT with Multiple String Patterns: Example This example shows the use of DB_FILE_NAME_CONVERT with BACKUP AS COPY to create image copies of the users and tools tablespaces from the previous discussion, directing users to /newdisk1 and tools to /newdisk2:

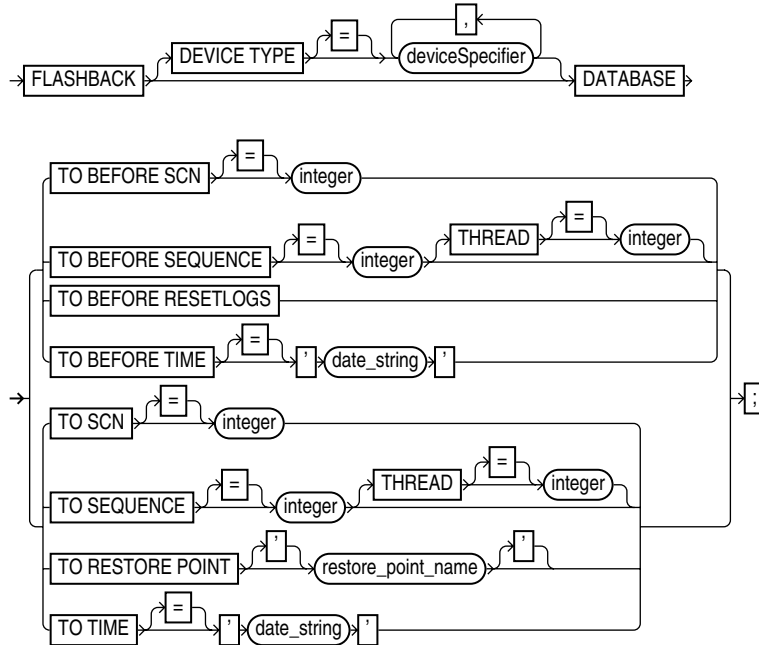
```
BACKUP AS COPY DEVICE TYPE DISK
    DB_FILE_NAME_CONVERT=('/disk1/dbs/users', '/newdisk1',
                          '/disk1/dbs/tools', '/newdisk2')
    TABLESPACE tools, users;
```

See Also: ["BACKUP"](#) on page 2-22, ["CONVERT"](#) on page 2-79, and ["DUPLICATE"](#) on page 2-104 for commands that use fileNameConversionSpec

FLASHBACK

Syntax

flashback::=



Purpose

Performs a Flashback Database operation, returning the database to (or to just before) target time, as specified by time, SCN or log sequence number.

The result of using flashback database is generally similar to a database point-in-time recovery performed with [RECOVER](#), except for the following principal differences:

- You do not need to restore a backup.
- RMAN uses **flashback logs** to undo changes to a point before the target time or SCN, and then uses archived redo logs to recover the database forward to make it consistent. RMAN automatically restores from backup any archived logs that are needed. (Flashback logs are stored as Oracle-managed files in the flash recovery area, and cannot be created if no flash recovery area is configured.)
- Some **NOLOGGING** changes may be reflected in the flashback which would not be reflected in the results of a point-in-time recovery, because flashback database uses backed-up block images as the basis for undoing changes to your current datafiles instead of relying on full and incremental backups and redo logs.

Because **FLASHBACK DATABASE** does not require you to restore a backup, it is usually much faster than incomplete recovery.

Flashback Database also has a number of applications in the context of standby databases, including providing functionality comparable to storage snapshots for reporting and testing databases, more flexibility in testing disaster recovery, and recovery from unwanted changes at a standby such as application of corrupted or erroneous updates at the standby database. These uses are discussed in more detail in *Oracle Data Guard Concepts and Administration*.

Restrictions and Usage Notes

- You can run this command from the RMAN prompt or from within a RUN command.
- The target database must be in ARCHIVELOG mode.
- The target database must be a Release 10g database.
- You must be connected to the target database.
- You must have enabled the flashback logging functionality before the target time for flashback, using the SQL statement `ALTER DATABASE . . . FLASHBACK ON`. You can check `V$DATABASE.FLASHBACK_ON` to determine whether flashback logging has been enabled.
- The target database must be mounted with a current control file, that is, the control file cannot be a backup or have been re-created.
- The flash recovery area must be enabled to enable logging for Flashback Database (that is, `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` must be set). Flashback logs can only be stored in the flash recovery area.
- The `FLASHBACK` 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.
- RMAN performs restore failover when unable to restore an archived redo log file for use in a Flashback Database operation. See *Oracle Database Backup and Recovery Advanced User's Guide* for details on restore failover.
- 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 upon the actual retention of flashback logs permitted by available disk space
 If the `FLASHBACK` command does not have enough flashback data to return the database to the requested SCN or time, then RMAN issues an error and does not modify the database.
- Flashback logs may be deleted if disk space in the flash recovery area is low. If there is not enough space in the flash recovery area for the retention of other files, then the database may delete an archived redo log file that is eligible for deletion because it is backed up to another location, or because it is obsolete. In such a case, flashback logs that cannot be used without that redo log file are also deleted. To ensure that this does not occur, make sure the flash recovery area is large enough to accomodate your flashback logs and other backups.
- `FLASHBACK DATABASE` can be used to return your database to a point prior to the most recent `OPEN RESETLOGS` operation, if your database has been upgraded to 10g Release 2.
- If the database has not retained enough flashback logs to allow a Flashback Database operation on all of your datafiles, then you can identify the datafiles for which there are not enough flashback logs, and take these datafiles offline. You can then run `FLASHBACK DATABASE` on the database and only the datafiles that are online will be affected. Then, you can perform point-in-time recovery on those files where there was not sufficient flashback log data.
- RMAN issues an error if you attempt to perform Flashback Database on online tablespaces on which flashback was disabled using the SQL statement `ALTER TABLESPACE . . . FLASHBACK OFF`.

- RMAN never flashes back data for temporary tablespaces.
- If a datafile has changed status between the current SCN and the SCN to which you are flashing back, then the `FLASHBACK` command behaves differently depending on the nature of the status change. Refer to [Table 2–3](#) for details.
- If the `FLASHBACK DATABASE` command fails or is interrupted, then the database is left mounted. At that point you can issue another `FLASHBACK DATABASE` command or use the `RECOVER` to perform complete recovery, returning the database to its original state.
- When performing a `FLASHBACK DATABASE` operation, your database may not be left at the SCN most immediately before the target time you specify. There are events other than transactions which cause the SCN for your database to be updated. If you use the `FLASHBACK DATABASE TO` form of the command and there is a transaction associated with your specified SCN, the database after the flashback operation will include all changes up to and including that transaction. Otherwise, all changes up to but not including that transaction will be included in your datafiles, whether you use the `FLASHBACK DATABASE TO` or `FLASHBACK DATABASE TO BEFORE` form of the command. Changes after the specified target time or SCN are never applied as a result of a `FLASHBACK DATABASE` operation.
- After the `FLASHBACK DATABASE` operation is complete, you may wish to open the database read-only and run some queries to see if you have achieved the desired result. If you are not satisfied with your flashback operation, you can perform `RECOVER DATABASE` to re-apply all changes and bring the database back to its state when you started the flashback operation. You can then attempt flashback again. If you are satisfied, you can either perform an `OPEN RESETLOGS` to abandon all changes after the target time for the flashback, or you can export lost data, use `RECOVER DATABASE` to return your database to its state before the flashback database operation, and then re-import the lost data.
- If you perform a `FLASHBACK DATABASE` operation, and some datafiles are not flashed back because they are offline, then you may encounter an error when you attempt to open the database with an `OPEN RESETLOGS`. In such a case you must do one of the following:
 - Use `FLASHBACK DATABASE` to bring the affected datafiles to the same time or SCN as the other datafiles.
 - Run [RESTORE](#) and then [RECOVER](#) to bring the affected datafiles to the same time or SCN as the rest of the database.
 - Take the affected datafiles offline and then drop them.

Table 2–3 How FLASHBACK Responds to Datafile Operations

If this datafile operation occurred during the flashback window ...	Then the FLASHBACK 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.

Table 2–3 (Cont.) How FLASHBACK Responds to Datafile Operations

If this datafile operation occurred during the flashback window ...	Then the FLASHBACK command ...
Resized	May fail. You can take the datafile offline and then rerun the FLASHBACK 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.

Keywords and Parameters

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, the FLASHBACK operation will fail. See Also: "deviceSpecifier" on page 2-101
DATABASE	Returns the database to the specified point. Query <code>OLDEST_FLASHBACK_SCN</code> and <code>OLDEST_FLASHBACK_TIME</code> in <code>V\$FLASHBACK_DATABASE_LOG</code> to display the approximate lowest SCN and time to which you can flash back. View the current database SCN in <code>V\$DATABASE.CURRENT_SCN</code> .
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 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. The DBA can override the default by using the <code>RMAN RESET DATABASE INCARNATION</code> command to set the recovery target incarnation.
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. The DBA can override the default by using the <code>RMAN RESET DATABASE INCARNATION</code> command to set the recovery target incarnation.
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 BEFORE 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 (but not including) the last change in the log with the specified sequence and thread number.
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 UNTIL TIME 'SYSDATE-7'</code> .
TO BEFORE TIME = ' <i>date_string</i> '	Similar to the TO TIME clause, but returns the database to its state including all changes up to but not including changes at the specified time.

Syntax Element	Description
TO BEFORE RESETLOGS	Returns the database to its state including all changes up to the SCN of the lastOPEN RESETLOGS.

Examples

FLASHBACK DATABASE to a Specific SCN: Example The following command uses Flashback Database to return the database to the specified SCN:

```
RMAN> FLASHBACK DATABASE TO SCN 46963;
```

FLASHBACK DATABASE to One Hour Ago: Example The following command uses Flashback Database to return the database to 1/24 of a day (one hour) in the past:

```
RMAN> FLASHBACK DATABASE TO TIMESTAMP (SYSDATE-1/24);
```

FLASHBACK DATABASE to a Specific Time: Example The following command uses SQL date conversion functions to specify the target time:

```
RMAN> FLASHBACK DATABASE TO TIMESTAMP  
TO_TIMESTAMP('2002-03-11 16:00:00', 'YYYY-MM-DD HH24:MI:SS');
```

FLASHBACK DATABASE to a Restore Point: Example The following command uses a restore point to identify the target time for a Flashback Database operation:

```
RMAN> FLASHBACK DATABASE TO RESTORE POINT "before_batch";
```

FLASHBACK DATABASE to Before the Last Resetlogs: Example The following command returns the database to immediately before the most recent OPEN RESETLOGS:

```
RMAN> FLASHBACK DATABASE TO BEFORE RESETLOGS;
```

formatSpec

Syntax

formatSpec::=

→ **'** *format_string* **'**

Purpose

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 port-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.

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 "Keywords and Parameters" are available in `FORMAT` strings to aid in generating unique filenames. The formatting of this information varies by platform.

Order of Precedence for Multiple Format Strings

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 these 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.

Restrictions and 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.

Keywords and Parameters

Syntax Element	Description
<code>%a</code>	Specifies the activation ID of the database.
<code>%c</code>	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 <code>%c</code> is 256.

Syntax Element	Description
%d	Specifies the name of the database.
%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.
%F	<p>Combines the DBID, day, month, year, and sequence into a unique and repeatable generated name. This variable translates into <i>c-FFFFFFFF-YYYYMMDD-QQ</i>, where:</p> <ul style="list-style-type: none"> ■ <i>FFFFFFFF</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.
%n	Specifies the name of the database, padded on the right with x characters to a total length of eight characters. For example, if the <code>prod1</code> is the database name, then the padded name is <code>prod1xxx</code> .
%p	<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.</p> <p>Note: If you specify <code>PROXY</code>, then the %p variable must be included in the <code>FORMAT</code> string either explicitly or implicitly within %U.</p>
%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, <code>CREATE CONTROLFILE</code> 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). The meaning of %U is different for image copies and backup pieces.</p> <p>For a backup piece, %U specifies a convenient shorthand for %u_%p_%c that guarantees uniqueness in generated backup filenames. If you do not specify a format when making a backup, then RMAN uses %U by default.</p> <p>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 literal '%' character. For example, %%Y translates to the string %Y.

Example

Specifying an ASM Disk Group: Example This example copies the database to ASM disk group disk1:

```
BACKUP AS COPY DATABASE FORMAT '+disk1';
```

Specifying a Format for Datafile Copies: Example This example creates copies of three datafiles with tag 'LATESTCOPY' to directory /copies:

```
# Create copies of 3 datafiles with tag 'LATESTCOPY' to directory /copies
BACKUP AS COPY
  FROM TAG 'LATESTCOPY'
  COPY OF DATAFILE 4, 6, 14
  FORMAT '/copies/Datafile%f_Database%d';
```

Creating a Database Copy for Use as a Standby Database: Example This example creates an image copy of the database to instantiate a physical standby in /stby:

```
# Create an image copy of the database to instantiate physical standby in /stby
BACKUP AS COPY
  DATABASE
  FORMAT '/stby/standby_file_%f_of_db_%I';
```

HOST

Syntax

host::=



Purpose

To invoke an operating system command-line sub-shell from within RMAN.

Restrictions and Usage Notes

Execute this command at the RMAN prompt or within the braces of a [RUN](#) command.

Keywords and Parameters

Syntax Element	Description
HOST	Enables you to execute an operating system command. Use this parameter: <ul style="list-style-type: none">With a '<i>command</i>', in which case RMAN runs the command in the specified string and then continues.Without a '<i>command</i>', in which case RMAN displays a command prompt and resumes after you exit the subshell.

Examples

Executing an Operating System Copy Within RMAN: Example This example shuts down the database, makes a backup of datafile `system01.dbf`, then executes the UNIX `ls` command to display all backed up datafiles:

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
BACKUP DATAFILE '?/oradata/trgt/system01.dbf'
  FORMAT '/tmp/system01.dbf';
HOST 'ls -l /tmp/*dbf';
ALTER DATABASE OPEN;
```

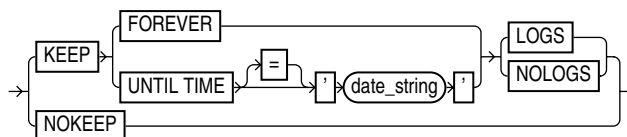
Hosting to the Operating System Within a Backup: Example This example makes an image copy of datafile 3, hosts out to the UNIX prompt to check that the copy is in the directory (the UNIX session output is indented and displayed in bold), then resumes the RMAN session:

```
RMAN> BACKUP DATAFILE 3 FORMAT '?/oradata/df3.cpy';
RMAN> HOST;
  % ls $ORACLE_HOME/oradata/df3.cpy
  /net/oracle/oradata/df3.cpy
  % exit
RMAN> LIST COPY;
```


keepOption

Syntax

keepOption::=



Purpose

A subclause specifying the status of a backup or copy in relation to a retention policy. The **KEEP** option marks the backup or copy as exempt from the retention policy (that is, not obsolete), and the **NOKEEP** option undoes any existing exemptions.

Limitations and Restrictions

- This option cannot be used to override the retention policy for files stored in the flash recovery area.
- The **KEEP UNTIL** clause never causes RMAN to consider a backup obsolete, if it is still required to satisfy the retention policy. **KEEP UNTIL** can cause backups to be kept longer than the retention policy would otherwise require, but never causes a backup to become obsolete sooner than the retention policy requires.

Keywords and Parameters

Syntax Element	Description
KEEP	Overrides any configured retention policy for this backup or copy so that the backup is not obsolete. The BACKUP . . . KEEP command specifies a new retention time for this backup. Use this option to create a long-term backup , that is, a backup that you want to archive.
FOREVER	Specifies that the backup or copy never expires. You must use a recovery catalog when FOREVER is specified, because the backup records eventually age out of the control file.
UNTIL TIME = 'date_string'	Specifies the date until which the backup or copy must be kept. You can either specify a specific time by using the current <code>NLS_DATE_FORMAT</code> , or a SQL date expression, such as <code>'SYSDATE+365'</code> .
LOGS	Specifies that all of the archived logs required to recover this backup or copy must remain available as long as this backup or copy is available.
NOLOGS	Specifies that this backup or copy cannot be recovered because the archived logs needed to recover this backup will not be kept. The only use for this backup or copy is to restore the database to the point in time that the backup or copy was taken. This is the only valid recoverability option when the database operates in NOARCHIVELOG mode. This option is not valid if the backup or copy is inconsistent.
NOKEEP	Specifies that the backup or copy expires according to the user's retention policy. This is the default behavior if no KEEP option is specified.

Examples

Making a Long-Term Backup: Example This example makes a long-term backup of the database and specifies that it should never become obsolete and that the logs required to recover it should not be retained:

```
SHUTDOWN IMMEDIATE;  
STARTUP MOUNT;  
BACKUP DATABASE  
    KEEP FOREVER NOLOGS;  
ALTER DATABASE OPEN;
```

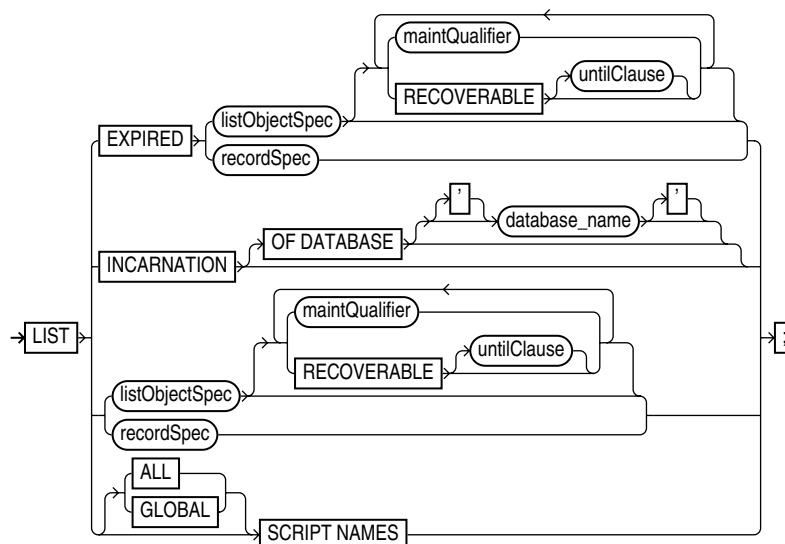
Changing the Status of a Copy: Example This example specifies that any long-term image copies of datafiles and control files should lose their exempt status and so become eligible to be obsolete according to the existing retention policy:

```
CHANGE COPY OF DATABASE CONTROLFILE NOKEEP;
```

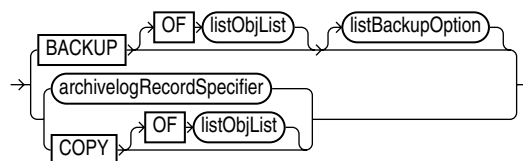
LIST

Syntax

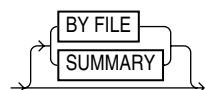
list::=



listObjectSpec::=



listBackupOption::=



Purpose

To display information about backup sets, proxy copies, and image copies recorded in the repository. The **LIST** command displays the files against which you can run **CROSSCHECK** and **DELETE** commands. Use this command to list:

- Backups and copies that do not have the status **AVAILABLE** in the RMAN repository
- Backups and copies of datafiles that are available and can possibly be used in a restore operation
- Specified archived logs, backup sets, backup pieces, control file copies, datafile copies, and proxy copies
- Backups and copies restricted by tag, completion time, recoverability, or device
- Incarnations of a specified database or of all databases known to the repository
- Stored scripts in the recovery catalog

RMAN records the output to either standard output or the message log, but not to both at the same time. You can control how the output is organized (BY BACKUP or BY FILE) as well as the level of detail in the output (VERBOSE or SUMMARY).

See Also: *Oracle Database Backup and Recovery Basics* to learn how to make lists and reports, and ["cmdLine"](#) on page 2-57

Restrictions and Usage Notes

- Execute LIST only at the RMAN prompt.
- RMAN must be connected to the target database. If RMAN is connected in NOCATALOG mode, then the database must be mounted. If RMAN is connected to a recovery catalog, then the target instance must be started but the target database does not need to be mounted.

Keywords and Parameters

list

Syntax Element	Description
EXPIRED	Displays backup sets, proxy copies, and image copies marked in the repository as EXPIRED, that is, "not found." To ensure that LIST EXPIRED shows up-to-date output, issue a CROSSCHECK command periodically. When you issue a CROSSCHECK command, RMAN searches on disk and tape for the backups and copies recorded in the repository. If it does not find them, then it updates their repository records to status EXPIRED.
RECOVERABLE	Specifies datafile backups or copies whose status in the repository is AVAILABLE and which can be used for restore and recovery in the target database's current incarnation. This list includes all backups and copies except the incremental backups that have no valid parent to which the incremental can be applied.
untilClause	Specifies an end time, SCN, or log sequence number. See "untilClause" on page 2-227.
recordSpec	Specifies the object or objects that you are listing. Refer to "recordSpec" on page 2-152.
INCARNATION	Displays information about the incarnations of a database. Whenever you open a database with the RESETLOGS option, then you create a new incarnation of the database. So, if LIST INCARNATION displays <i>n</i> incarnations of a database, then you have reset the online logs for this database <i>n</i> -1 times. The LIST output includes the primary keys of all database incarnation records for the specified database name (in the column "Inc Key," short for "incarnation key"). Use the key in a RESET DATABASE command to change the incarnation that RMAN considers to be current to a previous incarnation. See Also: Table 2–19 for an explanation of the column headings of the LIST INCARNATION output table
OF DATABASE ' <i>database_name</i> '	Specifies the name of the database. If you do not specify the OF DATABASE option, then the command lists all databases registered in the recovery catalog.
maintQualifier	Restricts the range of the listing. Refer to "maintQualifier" on page 2-145.

Syntax Element	Description
SCRIPT NAMES	<p>Lists names of RMAN stored scripts in the currently connected recovery catalog, along with any descriptive comments.</p> <p>Without ALL or GLOBAL options, 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.</p> <p>You must connect to a target database and a recovery catalog to list names for scripts defined for that target database.</p>
ALL	<p>RMAN lists all global and local scripts defined for all databases in the connected recovery catalog.</p> <p>You must be connected to a recovery catalog to use LIST ALL SCRIPT NAMES, but you do not need to be connected to a target database.</p>
GLOBAL	<p>RMAN lists only global scripts defined in the connected recovery catalog.</p> <p>You must be connected to a recovery catalog to use LIST GLOBAL SCRIPT NAMES, but you do not need to be connected to a target database.</p>

listObjectSpec

Syntax Element	Description
<i>listObjectSpec</i>	<p>Specifies the type of object or objects that you are listing.</p> <p>See Also: "recordSpec" on page 2-152</p>
BACKUP	Displays information about BACKUP output: backup sets (including detail on backup pieces), proxy copies, and image copies.
BACKUPSET	<p>Displays only information about backup sets, backup pieces, and proxy copies. The output displays a unique key for each. The LIST BACKUPSET command defaults to BY BACKUP.</p> <p>By default, RMAN lists both usable and unusable backups, even those that cannot be restored, are expired or unavailable, or are incrementals that cannot be restored because their parent full backup or copy no longer exists. To see only backups that can be used for recovery, use the RECOVERABLE option.</p> <p>See Also: "LIST Output" on page 2-130 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.</p>
COPY	<p>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.</p> <p>See Also: Table 2-16 and Table 2-18 for an explanation of the column headings of the LIST COPY output tables</p>
OF listObjList	<p>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.</p> <p>Note: The LIST BACKUP . . . LIKE command is not valid. The only valid exception is LIST BACKUP OF ARCHIVELOG LIKE.</p> <p>See Also: "listObjList" on page 2-143</p>
archiveLogRecordSpecifier	Displays information about a range of archived redo logs.

listBackupOption

Syntax Element	Description
<i>listBackupOption</i>	Specifies how you want the list backup sets. See Also: "recordSpec" on page 2-152
BY BACKUP	Lists backup sets, then the contents of each backup set (pieces and files), and then proxy copies. This is the default option for LIST BACKUP. If you specify the SUMMARY option, then this command is equivalent to LIST BACKUP SUMMARY.
VERBOSE	Gives detailed description of contents of each backup set (default).
SUMMARY	Gives a one-line summary for each datafile (when using BY FILE) or backup (when using BY BACKUP).
BY FILE	Lists a datafile, then its backup sets, and then proxy copies.

LIST Output

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

- [Table 2–4, "List of Backup Sets \(for datafile backup sets\)"](#)
- [Table 2–5, "List of Backup Pieces \(for sets with only one piece\)"](#)
- [Table 2–6, "List of Datafiles in backup set ..."](#)
- [Table 2–7, "List of Archived Logs in backup set ..."](#)
- [Table 2–8, "Backup Set Copy ... of backup set ... \(only if multiple pieces\)"](#)
- [Table 2–9, "List of Backup Pieces for backup set ... Copy ... \(if multiple pieces\)"](#)
- [Table 2–10, "List of Proxy Copies"](#)
- [Table 2–11, "List of Backup Sets \(LIST BACKUP ... SUMMARY\)"](#)
- [Table 2–12, "List of Backup Pieces \(LIST BACKUPPIECE ...\)"](#)
- [Table 2–13, "List of Datafile Backups \(LIST BACKUP ... BY FILE\)"](#)
- [Table 2–14, "List of Archived Log Backups \(LIST BACKUP ... BY FILE\)"](#)
- [Table 2–15, "List of Controlfile Backups \(LIST BACKUP ... BY FILE\)"](#)
- [Table 2–16, "List of Datafile Copies"](#)
- [Table 2–17, "List of Controlfile Copies"](#)
- [Table 2–18, "List of Archived Log Copies"](#)
- [Table 2–19, "List of Database Incarnations"](#)
- [Table 2–20, "List Script Names Output"](#)

Table 2–4 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.

Table 2–5 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).
Tag	The tag applied to the backup set; NULL if none. Note that tag names are not case sensitive and display in all uppercase.
Piece Name	The filename or handle of the backup piece. If the backup piece is on sbt, the Media ID is displayed with the name.
Controlfile Included	A control file is included in the backup. Note: This row appears only if the current control file is included in the backup.

Table 2–5 (Cont.) List of Backup Pieces (for sets with only one piece)

Column	Indicates
SPFILE Included	A server parameter 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 2–6 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 2-198

Table 2–7 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 2–8 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.

Table 2–8 (Cont.) Backup Set Copy ... of backup set ... (only if multiple pieces)

Column	Indicates
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; NULL if none. Note that tag names are not case sensitive and display in all uppercase.

Table 2–9 List of Backup Pieces for backup set ... Copy ... (if multiple pieces)

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 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.</p> <p>Note: The values for KEY in the recovery catalog and the control file are different.</p>
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, the media ID is also displayed.

Table 2–10 List of Proxy Copies

Column	Indicates
PC Key	<p>A unique key identifying this proxy copy.</p> <p>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.</p>
File	The absolute datafile number of the file that was copied.
Status	The proxy copy status: AVAILABLE, UNAVAILABLE, or EXPIRED (see 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 2–10 (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" on page 2-198
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; NULL if none. Note that tag names are not case sensitive and display in all uppercase.

Table 2–11 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.
Tag	The tag applied to the backup set; NULL if none. An asterisk (*) indicates multiple copies with different tags. Note that tag names are not case sensitive and display in all uppercase.

Table 2–12 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 (see 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 2–13 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 you are 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 you are connected 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.

Table 2–13 (Cont.) List of Datafile Backups (LIST BACKUP ... BY FILE)

Column	Indicates
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.
#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.
Tag	The tag applied to the backup set; NULL if none. Note that tag names are not case sensitive and display in all uppercase.

Table 2–14 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.
Tag	The tag applied to the backup set; NULL if none. Note that tag names are not case sensitive and display in all uppercase.

Table 2–15 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.

Table 2–15 (Cont.) List of Controlfile Backups (LIST BACKUP ... BY FILE)

Column	Indicates
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.
Tag	The tag applied to the backup set; NULL if none. Note that tag names are not case sensitive and display in all uppercase.

Table 2–16 List of Datafile Copies

Column	Indicates
Key	<p>The unique identifier for the datafile copy. Use this value in a CHANGE command to alter the status of the datafile copy.</p> <p>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.</p> <p>Note: The values for KEY in the recovery catalog and the control file are different.</p>
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.

Table 2–17 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	<p>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.</p>
Completion Time	<p>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.</p>
Ckp SCN	<p>The checkpoint of this control file when it was copied.</p>
Ckp TIME	<p>The checkpoint of this control file when it was copied.</p>
Name	<p>The filename of the control file copy.</p>

Table 2–18 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	<p>The redo log thread number.</p>
Seq	<p>The log sequence number.</p>
S	<p>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.</p>
Low Time	<p>The time when the database switched into the redo log having this sequence number.</p>
Name	<p>The filename of the archived redo log copy.</p>

Table 2–19 List of Database Incarnations

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

Table 2–20 List Script Names Output

Column	Indicates
Script Name	The name of the stored script.
Description	The comment provided when the script was created.

Examples

Listing Backups: Example This example lists all backups in default verbose mode:

```
LIST BACKUP;
```

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

```
BS Key Device Type Elapse Time Completion Time
-----
236    DISK          00:00:08    21-SEP-00
      BP Key: 237    Status: AVAILABLE    Tag: TAG20011121T053733
      Piece Name: /oracle/oradata/09c5unih_1_1
```

```
List of Archived Logs in backup set 236
```

```
Thrd Seq      Low SCN      Low Time    Next SCN    Next Time
-----
1     141        49173       21-SEP-01  49784       21-SEP-01
1     142        49784       21-SEP-01  50331       21-SEP-01
```

```
BS Key Type LV Size      Device Type Elapse Time Completion Time
-----
244    Full    61M          DISK          00:00:18    21-SEP-01
      BP Key: 245    Status: AVAILABLE    Tag: TAG20011121T053816
      Piece Name: /oracle/oradata/0ac5unj5_1_1
      Controlfile Included: Ckp SCN: 51554      Ckp time: 21-SEP-01
      SPFILE Included: Modification time: 21-SEP-01
      List of Datafiles in backup set 244
      File LV Type Ckp SCN    Ckp Time    Name
```

```

-----
1      Full 51555      21-SEP-01 /oracle/oradata/trgt/system01.dbf
2      Full 51555      21-SEP-01 /oracle/oradata/trgt/undotbs_01.dbf

```

List of Proxy Copies

```
=====
```

PC Key	File	Status	Completion time	Ckp SCN	Ckp time
552	1	AVAILABLE	10/07/2001 03:05:21	78022	10/07/2001 03:05:10
		Datafile name: /oracle/oradata/trgt/system01.dbf			
		Handle: 0jb81876_1_0			
561	1	AVAILABLE	10/07/2001 03:38:22	78025	10/07/2001 03:38:09
		Datafile name: /oracle/oradata/trgt/system01.dbf			
		Handle: 0lb81a51_1_0			
		Tag: WKLYBKUP			

Listing a Summary of Backups: Example The following example lists a summarized version of all RMAN backups:

```
LIST BACKUP SUMMARY;
```

List of Backups

```
=====
```

Key	TY	LV	S	Device	Type	Completion Time	#Pieces	#Copies	Tag
35	B	A	A	SBT_TAPE		FEB 08 2002 05:37:37	1	1	TAG20020208T053733
42	B	F	A	SBT_TAPE		FEB 08 2002 05:38:21	1	1	TAG20020208T053744

Listing Backups by File: Example This example groups all backups by file:

```
LIST BACKUP BY FILE;
```

List of Datafile Backups

```
=====
```

File Key	TY	LV	S	Ckp SCN	Ckp Time	#Pieces	#Copies	Tag
1	502	B	0	A 37973	09/28/01 19:28:36	1	3	*
	552	P	F	X 78022	10/07/01 03:05:10	1	1	DF_1
	561	P	0	U 78025	10/07/01 03:38:09	2	1	DF_1
2	502	B	0	A 37973	09/28/01 19:28:36	1	2	*
	562	P	0	U 78027	10/07/01 03:38:22	1	1	DF_2

List of Archived Log Backups

```
=====
```

Thrd	Seq	Low SCN	Low Time	BS Key	S	#Pieces	#Copies	Tag
1	141	49463	14-SEP-01	213	A	1	1	TAG20011114T125431

List of Controlfile Backups

```
=====
```

CF Ckp SCN	Ckp Time	BS Key	S	#Pieces	#Copies	Tag
51593	14-SEP-01	222	A	1	1	

List of SPFILE Backups

```
=====
```

Modification Time	BS Key	S	#Pieces	#Copies	Tag
OCT 08 2001 05:38:55	251	A	1	1	

Listing Archived Redo Logs: Example The following example lists archived logs and copies of logs:

```
LIST COPY OF DATABASE ARCHIVELOG ALL;
```

List of Archived Log Copies

Key	Thrd	Seq	S	Low Time	Name
153	1	30	A	14-SEP-01	/oracle/oradata/trgt/arch/archive1_30.dbf
154	1	31	A	14-SEP-01	/oracle/oradata/trgt/arch/archive1_31.dbf

Listing Backups of Specific Datafiles: Example The following example lists backups of datafile 3 in summary mode:

```
LIST BACKUP OF DATAFILE 3 SUMMARY;
```

List of Backups

=====

Key	TY	LV	S	Device	Type	Completion Time	#Pieces	#Copies	Tag
180	B	0	A	DISK		14-SEP-01	1	2	TAG20011114T125431

Listing Database Incarnations: Example This example lists all database incarnations recorded in the recovery catalog:

```
LIST INCARNATION;
```

List of Database Incarnations

DB Key	Inc Key	DB Name	DB ID	STATUS	Reset SCN	Reset Time
1	1	RDBMS	774627068	PARENT	1	21-OCT-03
2	2	RDBMS	774627068	CURRENT	173832	21-OCT-03

Listing Stored Scripts: Example This example shows the output of running LIST ALL SCRIPT NAMES:

```
RMAN> LIST ALL SCRIPT NAMES;
```

List of Stored Scripts in Recovery Catalog

Scripts of Target Database TEST

Script Name

Description

configure_Rman

Script to configure retention policy and device parameters.

backup_schema_dfs

Backups only tablespaces that are being currently used.

Scripts of Target Database PROD

Script Name

Description

nightly_backup

Script used to backup PROD database every weekday.

Global Scripts

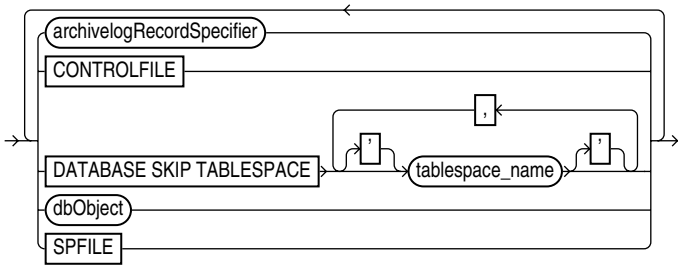
Script Name	Description

<code>purge_backups</code>	General script to enforce retention policy
<code>validate_archived_logs</code>	General script to synchronize archived logs on disk with RMAN.

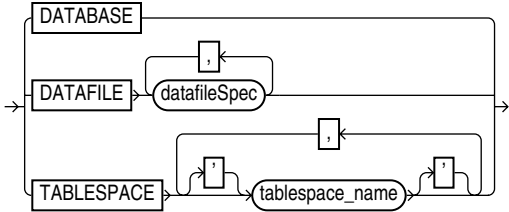
listObjList

Syntax

listObjList::=



dbObject::=



Purpose

A subclause used to specify database files and archived redo logs.

Restrictions and Usage Notes

Use this clause in the following commands:

- [CHANGE](#)
- [CROSSCHECK](#)
- [DELETE](#)
- [LIST](#)

Keywords and Parameters

listObjList

Syntax Element	Description
archivelogRecordSpecifier	Specifies a range of archived redo logs. See Also: " archivelogRecordSpecifier " on page 2-18
CONTROLFILE	Specifies the current control file.
DATABASE SKIP TABLESPACE 'tablespace_name' [, 'tablespace_name']	Omits the specified tablespaces from the DATABASE specification.
SPFILE	Specifies the current server parameter file.

dbObject

Syntax Element	Description
DATABASE	Specifies backup sets or image copies of all files in the current database.
DATAFILE datafileSpec	Specifies datafiles by filename or file number. The clause specifies datafile image copies or backup sets that contain at least one of the datafiles. See Also: " datafileSpec " on page 2-95
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

Listing Datafile Copies: Example 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;
```

Crosschecking Archived Redo Logs: Example The following example queries the media manager for the status of server parameter file and archived redo log backups (either backup sets or image copies) created in the last three months:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;  
CROSSCHECK BACKUP OF SPFILE ARCHIVELOG FROM TIME 'SYSDATE-90';
```

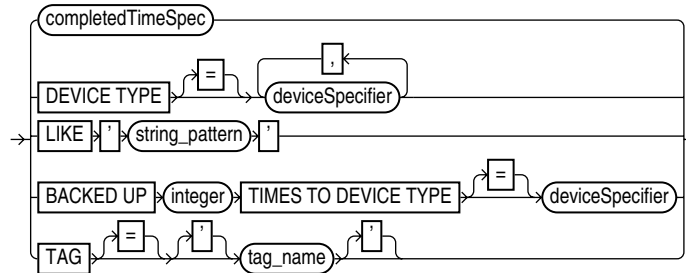
Deleting Expired Control File Backup Sets: Example The following command deletes expired backups (either backup sets or image copies) of the control file:

```
DELETE EXPIRED BACKUP OF CONTROLFILE;
```

maintQualifier

Syntax

maintQualifier::=



Purpose

A subclause used to specify database files and archived redo logs.

Restrictions and Usage Notes

- Use this clause in the following commands:
 - [LIST](#)
 - [CROSSCHECK](#)
 - [DELETE](#)
 - [SWITCH](#)
- The BACKED UP *integer* TIMES clause applies only to archived redo logs.
- You cannot use LIKE with backup pieces.

Keywords and Parameters

Syntax Element	Description
completedTimeSpec	Specifies a range of time for completion of the backup or copy. See Also: " completedTimeSpec " on page 2-61
DEVICE TYPE deviceSpecifier	Allocates automatic channels for the specified device type only. This option is valid only if you have configured automatic channels and have not manually allocated channels. For example, if you configure automatic disk and tape channels, and issue CHANGE . . . DEVICE TYPE DISK, then RMAN allocates only disk channels. See Also: " deviceSpecifier " on page 2-101
LIKE ' <i>string_pattern</i> '	Restricts datafile copies by specifying a filename pattern. The pattern can contain Oracle pattern matching characters % and _. RMAN only operates on those files whose name matches the pattern. Note: You cannot use the LIKE option with the LIST . . . ARCHIVELOG command.
BACKED UP <i>integer</i> TIMES TO DEVICE TYPE deviceSpecifier	Restricts the command to archived logs that have been successfully backed up <i>integer</i> or more times to the specified media.

Syntax Element	Description
<code>TAG = 'tag_name'</code>	<p>Specifies the datafile copies and backup sets by tag. Tag names are not case sensitive and display in all uppercase.</p> <p>See Also: "BACKUP" on page 2-22 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</p>

Example

Listing Backups on a Specific Device: Example The following command lists all backups located on tape and copies located in `/tmp`:

```
LIST BACKUP DEVICE TYPE sbt;  
LIST COPY LIKE '/tmp';
```

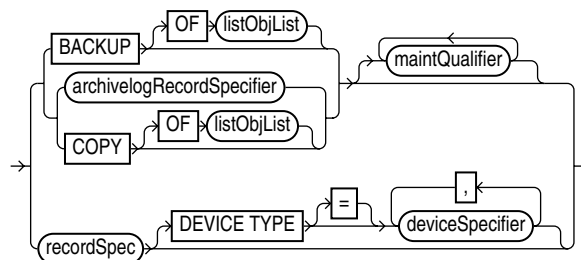
Deleting Archived Logs That Are Already Backed Up: Example The following command deletes only those archived logs that have been successfully backed up three or more times to tape:

```
DELETE ARCHIVELOG ALL BACKED UP 3 TIMES TO DEVICE TYPE sbt;
```

maintSpec

Syntax

maintSpec::=



Purpose

To specify the backup files operated on by the CHANGE, CROSSCHECK, and DELETE commands.

Keywords and Parameters

Syntax Element	Description
BACKUP	<p>Processes files output by the BACKUP command.</p> <p>With CHANGE BACKUP:</p> <p>Specify one or more objects using the "CHANGE BACKUP OF <i>listObjList</i>" form of the command. Otherwise, CHANGE BACKUP operates on all backup sets, image copies, and proxy copies recorded in the repository.</p> <p>With CROSSCHECK BACKUP:</p> <p>If you are running CROSSCHECK BACKUP, then the command crosschecks BACKUP output that has status AVAILABLE or EXPIRED: backup sets (including backup pieces), proxy copies, and image copies. RMAN crosschecks all backups not marked UNAVAILABLE, even if they cannot be restored or if they are incremental backups whose parent full backup or copy no longer exists.</p> <p>Specify one or more objects using the "CROSSCHECK BACKUP OF <i>listObjList</i>" form of the command. Otherwise, RMAN crosschecks backups of the whole database.</p> <p>With DELETE BACKUP:</p> <p>Deletes backup sets and proxy copies.</p> <p>Specify one or more objects using the "DELETE BACKUP OF <i>listObjList</i>" form of the command. Otherwise, RMAN deletes backup sets of the whole database.</p>
BACKUPSET	<p>When followed by a list of primary keys: this is actually a case of the <i>recordSpec</i> syntax element in the syntax diagram for <i>maintSpec</i>. In such a case, the keys identify backupsets for use with the CHANGE, CROSSCHECK and DELETE commands. For more details, see "recordSpec" on page 2-152 and "LIST Output" on page 2-130 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.</p> <p>When not followed by a list of primary keys: A synonym for BACKUP.</p>

Syntax Element	Description
COPY	<p>Processes datafile copies, control file copies, archived redo logs, and image copies of archived redo logs.</p> <p>If you do not specify an option for CHANGE COPY, then the command operates on all image copies recorded in the repository.</p> <p>If you are running CROSSCHECK COPY, then by default the command checks all image copies of all files in the database with status <code>AVAILABLE</code> or <code>EXPIRED</code>.</p> <p>By default, <code>COPY</code> removes copies of all files in the database. Specify the <code>EXPIRED</code> option to remove only copies that are marked <code>EXPIRED</code> in the repository.</p>
OF listObjList	<p>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 the command defaults to all copies. Note that <code>CHANGE COPY OF DATABASE</code> includes datafiles but not control files.</p> <p>See Also: "listObjList" on page 2-143</p>
archivelogRecordSpecifier	<p>Processes the specified archived redo logs.</p> <p>See Also: "archivelogRecordSpecifier" on page 2-18</p>
maintQualifier	<p>Restricts the command based on the specified options.</p> <p>See Also: "maintQualifier" on page 2-145</p>
recordSpec	<p>Specifies the file that you are performing maintenance on. See "recordSpec" on page 2-152.</p>
DEVICE TYPE deviceSpecifier	<p>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>CROSSCHECK . . . DEVICE TYPE DISK</code>, then RMAN allocates only disk channels.</p> <p>See Also: "deviceSpecifier" on page 2-101</p>

Examples

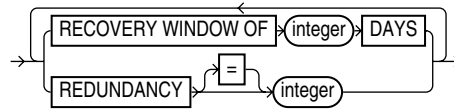
To see the *maintSpec* clause in use, refer to the commands where it is used:

- "[CHANGE](#)" on page 2-54
- "[CROSSCHECK](#)" on page 2-93
- "[DELETE](#)" on page 2-96

obsOperandList

Syntax

obsOperandList::=



Purpose

A subclause used to specify which criteria are used to mark backups and copies as obsolete.

Restrictions and Usage Notes

- Use this clause in the following commands:
 - [DELETE](#)
 - [REPORT](#)
- Using both `RECOVERY WINDOW` and `REDUNDANCY` clause in a single `REPORT OBSOLETE` or `DELETE OBSOLETE` command is not supported.

Keywords and Parameters

Syntax Element	Description
<code>RECOVERY WINDOW OF <i>integer</i> DAYS</code>	Specifies that RMAN should report as obsolete those backups and copies that are not needed to recover the database to any point within the last <i>integer</i> days. See Also: "CONFIGURE" on page 2-63 for an explanation of the recovery window
<code>REDUNDANCY = <i>integer</i></code>	Specifies the minimum level of redundancy considered necessary for a backup or copy to be obsolete. A datafile copy is obsolete if there are at least <i>integer</i> more recent backups or image copies of this file; a datafile backup set is obsolete if there are at least <i>integer</i> more recent backups or image copies of each file contained in the backup set. For example, <code>REDUNDANCY 2</code> means that there must be at least two more recent backups or copies of a datafile for any other backup or copy to be obsolete.

Example

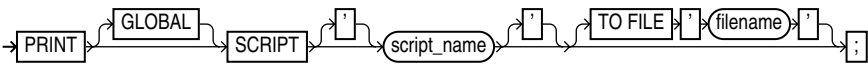
Deleting Obsolete Backups: Example The following command deletes all backups and copies not needed to recover the database to a random point within the last 30 days:

```
DELETE OBSOLETE RECOVERY WINDOW OF 30 DAYS;
```

PRINT SCRIPT

Syntax

printScript::=



Purpose

To print a local or global stored script to standard output or to a file.
For more information about stored scripts, see ["CREATE SCRIPT"](#) on page 2-91.

Restrictions and Usage Notes

- Use this command only at the RMAN prompt.
- You must be connected to the target database that you connected to when you created or replaced the script.
- You must be connected to the recovery catalog, and the recovery catalog database must be open.

Keywords and Parameters

Syntax Element	Description
GLOBAL	With the GLOBAL option, PRINT SCRIPT prints the global script named 'script_name' instead of the local script. If omitted, RMAN looks for a local or global script script_name to print. If a local script is found, it is printed. If no local script is found, but a global script script_name is found, the global script is printed. See "CREATE SCRIPT" on page 2-91 for more details about local and global stored scripts.
'script_name'	The name of the script to print.
TO FILE ' filename '	With TO FILE 'filename' option, PRINT SCRIPT sends its output to the specified file instead of standard output.

Examples

Printing a Script to a File: Example The following example prints a script to the file '/tmp/backup_db.rman':

```
PRINT SCRIPT backup_db TO FILE '/tmp/backup_db.rman';
```

Printing a Script to the Screen: Example This example prints a stored script to standard output (includes sample output):

```
RMAN> PRINT SCRIPT 'backup_db';

printing stored script: backup_db
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt
  BACKUP DATABASE;
}
```

QUIT

Syntax

quit::=

→ QUIT →

Purpose

To shut down the Recovery Manager utility.

Restrictions and Usage Notes

Execute only at the RMAN prompt, not in a RUN block.

Example

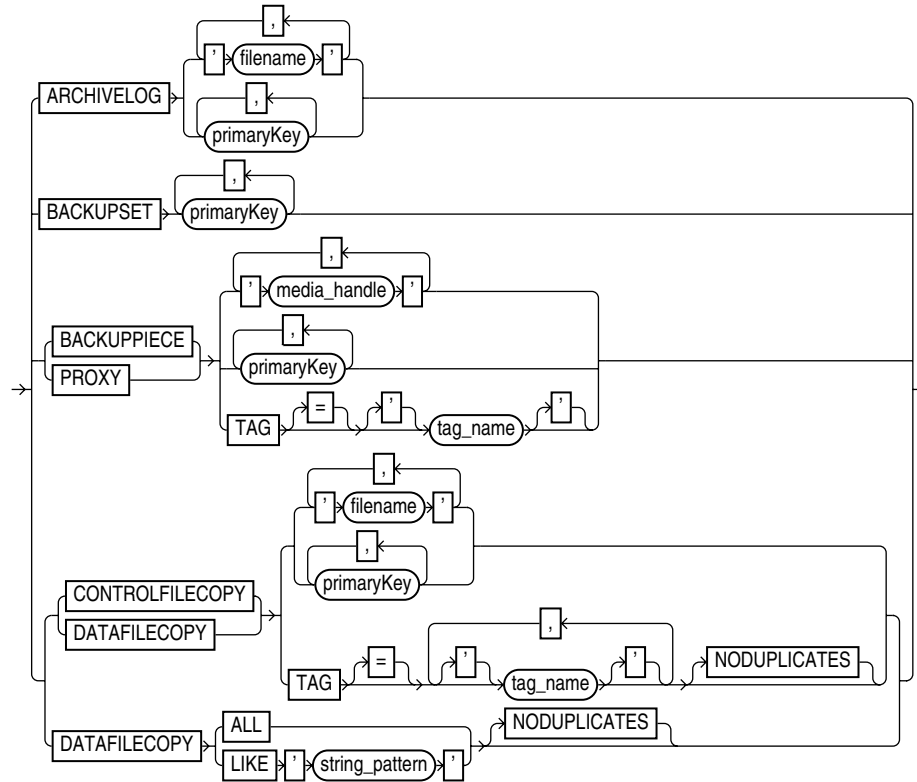
Quitting RMAN: Example This example starts RMAN and then shuts it down:

```
% rman
RMAN> QUIT
```

recordSpec

Syntax

recordSpec::=



Purpose

A subclause that specifies which objects the [CHANGE](#), [CROSSCHECK](#), [DELETE](#), and [LIST](#) commands should operate on.

Most *recordSpec* options allow you to specify a primary key. Use the output of the [LIST](#) command to obtain keys.

Keywords and Parameters

Syntax Element	Description
ARCHIVELOG	Specifies an archived redo log by either <i>primary_key</i> or ' <i>filename</i> '.
BACKUPSET <i>primary_key</i>	Specifies a backup set by <i>primary_key</i> .
BACKUPPIECE	Specifies a backup piece by ' <i>media_handle</i> ', <i>primary_key</i> , or <i>tag_name</i> .
PROXY	Specifies a proxy copy by ' <i>media_handle</i> ', <i>primary_key</i> , or <i>tag_name</i> .
CONTROLFILECOPY	Specifies a control file copy by <i>primary_key</i> , filename pattern (' <i>filename</i> '), or TAG = <i>tag_name</i> . If you crosscheck a control file copy, you must specify a filename rather than a primary key.
DATAFILECOPY	Specifies a datafile copy by either <i>primary_key</i> , filename pattern (' <i>filename</i> '), or TAG = <i>tag_name</i> .

Syntax Element	Description
NODUPLICATES	With CONTROLFILECOPY or DATAFILECOPY, 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

Crosschecking Backups: Example This example crosschecks backup sets specified by primary key:

```
CROSSCHECK BACKUPSET 507, 509;
```

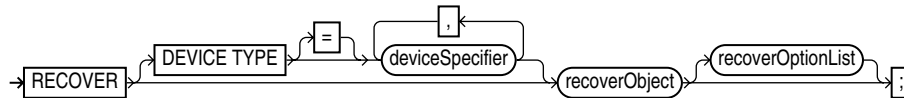
Deleting Datafile Copies: Example This example deletes a specified datafile copy:

```
DELETE NOPROMPT DATAFILECOPY '?/oradata/users01.cpy';
```

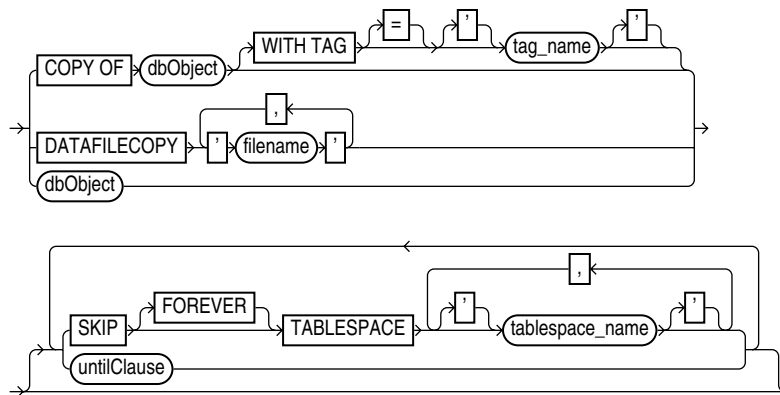
RECOVER

Syntax

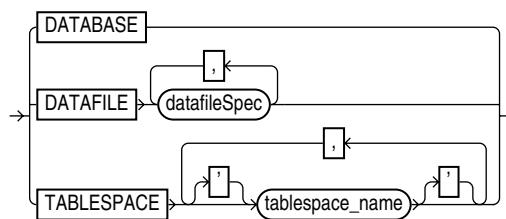
recover::=



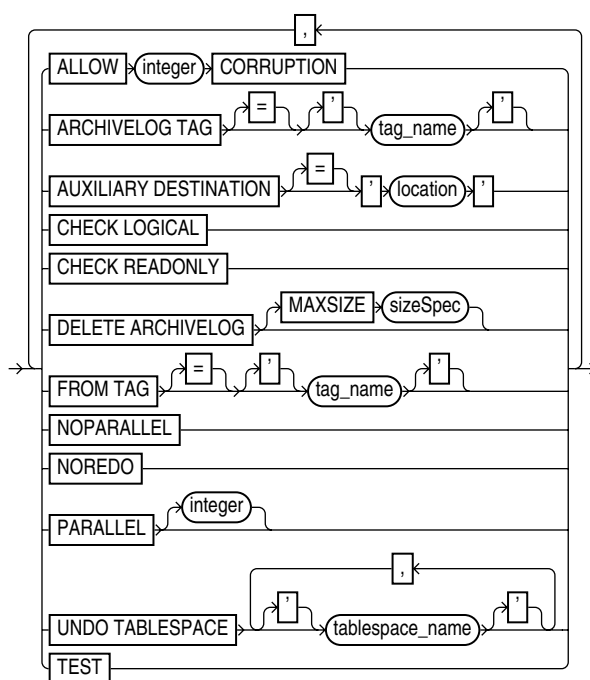
recoverObject::=



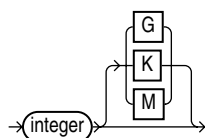
dbObject::=



recoverOptionList::=



sizeSpec::=



Purpose

The RECOVER command has three distinct uses:

- Performing complete recovery of one or more restored datafiles, or the entire database.
- Performing point-in-time recovery of a database (DBPITR) or tablespace (TSPITR).
- Applying incremental backups to a datafile image copy (*not* a restored datafile) to roll it forward in time.

When performing media recovery, RMAN first looks for archived logs on disk, and if none are available, then it restores logs from backups to the LOG_ARCHIVE_DEST_1 destination (or the SET ARCHIVELOG DESTINATION) as needed for the recovery.

Complete and point-in-time recovery of a database can use both incremental backups and archived logs. If RMAN has a choice between applying an incremental backup or applying redo, then it always chooses the incremental backup. Applying incremental backups is faster and captures NOLOGGING changes. If overlapping levels of incremental backup are available, then RMAN automatically chooses the one covering the longest period of time.

RMAN also applies incremental backups to restored files that were not restored from an incremental backup.

Note: When RMAN applies incremental backups, it also recovers changes to objects created with the `NOLOGGING` option. Applying archived redo logs to datafiles does not recover these changes.

See Also:

- *Oracle Database Backup and Recovery Basics* to learn how to recover datafiles.
- ["RESTORE"](#) on page 2-178 command for explanation of the default location for restoring archived logs. Note that RMAN automatically specifies the `MAXSIZE` option when staging logs in the flash recovery area.

Restrictions and Usage Notes

- You cannot arbitrarily recover individual datafiles to different points in time. You can recover the whole database to a single point in time (DBPITR, in which case you should use `SET UNTIL`, followed by `RESTORE DATABASE` and `RECOVER DATABASE`) or recover wholly contained tablespaces to a point in time different from the rest of the database (TSPITR, in which case you must use `RECOVER TABLESPACE . . . UNTIL . . .`). For more information on DBPITR, see *Oracle Database Backup and Recovery Advanced User's Guide*. For more information on TSPITR, see the procedure described in *Oracle Database Backup and Recovery Advanced User's Guide*.
- For whole database recovery, the database must be mounted but not open. For recovery of datafiles or tablespaces, the target database must be mounted or open, but if the database is open, then the datafiles or tablespaces to be recovered must be offline.
- The `RECOVER DATABASE` command does not recover any files that are offline normal or read-only at the point in time to which the files are being recovered. RMAN omits offline normal files with no further checking. If `CHECK READONLY` is specified, then RMAN checks each read-only file on disk to ensure that it is already current at the desired point in time. If `CHECK READONLY` is not specified, then RMAN skips read-only files completely during the `RECOVER` command.
- You must open the database with the `RESETLOGS` option after incomplete recovery or recovery with a backup control file.
- If RMAN is unable to restore files needed for the recovery operation from the first backup it tries, it uses restore failover to try to restore the file from other suitable backups. See *Oracle Database Backup and Recovery Advanced User's Guide* for details on restore failover.
- If the recovery operation requires that RMAN restore archived log redo or incremental backups from backup, then you must either configure channels for the devices from which files are to be restored (disk or SBT) or use [ALLOCATE CHANNEL](#) commands in a `RUN` block with your `RECOVER` command.
- 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.
- You cannot manually allocate channels and then run `RECOVER` with the `DEVICE TYPE` option.

- If incremental backups or archived logs need to be restored during recovery, then you must either use configured channels or manually allocate channels of the same type that created these backups.
- 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 prior incarnations.
- If, during recovery of a tablespace or database, the database encounters redo for adding a datafile, 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.
- When opening a database after recovery, any locally managed tempfiles recorded in the RMAN repository are re-created if necessary. If you are recovering using a backup control file and no recovery catalog, then tempfiles created after the control file backup are not recorded in the RMAN repository and are not created automatically.
- If you want to perform DBPITR, the best practice is to enter a SET UNTIL command before both the RESTORE and RECOVER commands. If you run SET UNTIL after the RESTORE operation, then you may not be able to perform media recovery on the database to the target time, because the restored files may have time stamps later than the target time.

Keywords and Parameters

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 issue RECOVERDEVICE TYPE DISK . . . , then RMAN allocates only disk channels. See Also: " deviceSpecifier " on page 2-101

recoverObject

Syntax Element	Description
COPY OF	<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. RECOVER COPY is a method for updating a 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 BACKUP . . . FOR RECOVER OF COPY 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 'tag_name'	Specifies a tag name to identify the image copy to be rolled forward.
DATAFILECOPY	Applies incremental backups to the specified datafile image copy. Refer to description of RECOVER COPY OF..
SKIP [FOREVER] TABLESPACE	<p>Specifies tablespaces that should not be recovered, which is useful for avoiding recovery of tablespaces containing only temporary data or for postponing recovery of some tablespaces. The SKIP clause takes the datafiles in the specified tablespaces offline before starting media recovery. These files are left offline after the media recovery is complete.</p> <p>If you perform incomplete recovery, then SKIP is not allowed. Instead, use SKIP FOREVER, with the intention of dropping the skipped tablespaces after opening the database with the RESETLOGS option. The SKIP FOREVER clause causes RMAN to take the datafiles offline with the DROP option. Only use SKIP FOREVER when the specified tablespaces will be dropped after opening the database.</p>
<i>untilClause</i>	<p>Specifies a past time, SCN, or log sequence number for termination of the RECOVER command. When used with one or more tablespaces, indicates a TSPITR operation for the named tablespaces. It cannot be used with RECOVER DATAFILE. It should not be used for RECOVER DATABASE (see "Restrictions and Usage Notes" for details). After DBPITR, you must open the database with the RESETLOGS option.</p> <p>See Also: "untilClause" on page 2-227</p>

dbObject

Syntax Element	Description
DATABASE	Specifies that the entire database is to be recovered. By default, RMAN performs complete recovery. For incomplete recovery, specify an <i>untilClause</i> .

Syntax Element	Description
DATAFILE <i>datafileSpec</i>	<p>Specifies a list of one or more datafiles to recover. Specify datafiles by either filename (by using a quoted string) or absolute datafile number (by using an integer).</p> <p>If you are using the control file as the exclusive repository for RMAN metadata, then the filename must be the name of the datafile as recorded in the control file.</p> <p>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. The instance then fails before you can 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>See Also: "datafileSpec" on page 2-95</p>
TABLESPACE ' <i>tablespace_name</i> '	Specifies tablespaces by tablespace name.

recoverOptionList

Syntax Element	Description
<i>recoverOptionList</i>	Specifies various recovery options.
ARCHIVELOG TAG = <i>tag_name</i>	Specifies the tag for an archived log backup to be used during recovery. If the tagged backup does not contain all the necessary logs for recovery, RMAN uses logs or incremental backups as needed from whatever is available. Note that tag names are not case sensitive and display in all uppercase.
AUXILIARY DESTINATION [=] ' <i>location</i> '	<p>Can only be used when performing TSPITR. Used to automate the management of auxiliary set files during TSPITR. Specifies a location where auxiliary set datafiles, control files and online 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 logs before executing the RECOVER TABLESPACE... UNTIL... command. Otherwise, TSPITR will fail.</p> <p>See also: The chapter on TSPITR in <i>Oracle Database Backup and Recovery Advanced 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, it logs the block in the alert.log and server session trace file.</p> <p>The MAXCORRUPT setting represents the total number of physical and logical corruptions permitted on a file. By default, MAXCORRUPT is 0, so that if there are any corrupt blocks, 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>
CHECK READONLY	Checks the headers of read-only files to ensure that they are current before omitting them from the recovery.

Syntax Element	Description
DELETE ARCHIVELOG [MAXSIZE <i>integer</i> [K M G]]	<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.</p> <p>If you do not specify MAXSIZE, then RMAN deletes restored archived logs as they are applied. If you specify MAXSIZE, then RMAN will not use more than <i>integer</i> amount of disk space for restored archived 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> <p>Note: If archived redo logs are restored to the flash recovery area then the DELETE ARCHIVELOG option is enabled by default.</p>
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. Note that tag names are not case sensitive and display in all uppercase.</p> <p>See Also: "BACKUP" on page 2-22 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>
NOREDO	<p>Suppresses the application of redo logs during recovery. Only incremental backups are applied.</p> <p>One use of this option is in updating full backups of NOARCHIVELOG databases by using incremental backups. If you do not specify NOREDO when recovering a NOARCHIVELOG database, then the database terminates 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 in updating standby or duplicate databases. Incremental backups created with the BACKUP INCREMENTAL FROM SCN command can be applied at the standby or duplicate database. This process is described in <i>Oracle Database Backup and Recovery Advanced User's Guide</i>.</p>
TEST	<p>Use the TEST clause to conduct a trial recovery. A trial recovery is useful if a normal recovery procedure has encountered some problem. It lets you look ahead into the redo stream to detect possible additional 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	<p>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, 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, use this clause to provide a list of tablespaces with undo segments at the target time.</p>

Syntax Element	Description
<code>ALLOW <i>integer</i> CORRUPTION</code>	<p>The <code>ALLOW <i>integer</i> CORRUPTION</code> clause lets you specify, in the event of logfile corruption, the number of corrupt blocks that can be tolerated while allowing recovery to proceed.</p> <p>When you use this clause during trial recovery (that is, in conjunction with the <code>TEST</code> clause), <i>integer</i> can exceed 1. When using this clause during normal recovery, <i>integer</i> can only be 0 or 1.</p>
<code>NOPARALLEL</code>	Specifies not to perform recovery in parallel. Parallel execution is the default for <code>RECOVER</code> .
<code>PARALLEL [<i>integer</i>]</code>	<p>Specifies to perform recovery in parallel (the default).</p> <p>By default, the database determines the optimal degree of parallelism, which is the number of parallel threads used in the recovery operation. It is generally not necessary for you to specify the degree of parallelism. However, you can specify the degree of parallelism directly by including the <i>integer</i> in the <code>PARALLEL</code> clause. Each parallel thread may use one or two parallel execution servers.</p> <p>For more complete information on <code>PARALLEL</code>, see the description of the <code>PARALLEL</code> clause in the discussion of <code>CREATE TABLE</code> in <i>Oracle Database SQL Reference</i>.</p>

Examples

Recovering a Tablespace in an Open Database: Example The following example takes tablespace `tools` offline, uses automatic channels to restore and recover it (deleting the logs that it restored from tape), then brings it back online:

```
SQL "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE";
RESTORE TABLESPACE tbs_1;
# restore only 2M of logs at a time, then delete them
RECOVER TABLESPACE tools DELETE ARCHIVELOG MAXSIZE 2M;
SQL "ALTER TABLESPACE tools ONLINE";
```

Recovering Datafiles Restored to New Locations: Example The following 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:

```
RMAN> RUN
{
  ALLOCATE CHANNEL dev2 DEVICE TYPE sbt;
  SQL "ALTER TABLESPACE users OFFLINE IMMEDIATE";
  SET NEWNAME FOR DATAFILE '?/oradata/trgt/users01.dbf'
    TO '/tmp/users01.dbf';
  RESTORE TABLESPACE users;
  SWITCH DATAFILE ALL;
  RECOVER TABLESPACE users;
  SQL "ALTER TABLESPACE users ONLINE";
}
```

Performing DBPITR with a Backup Control File and Recovery Catalog: Example

Assume that all datafiles and control files as well as archived redo log 40 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 `history` because it has not changed since log 40. After connecting to the target and recovery catalog, follow the example shown here:

```
RMAN> STARTUP FORCE NOMOUNT;
RMAN> RUN
{
```

```
SET UNTIL SEQUENCE 40 THREAD 1;      # Recover database until log sequence 40
RESTORE CONTROLFILE;
ALTER DATABASE MOUNT;
RESTORE DATABASE SKIP TABLESPACE temp, history;
RECOVER DATABASE SKIP FOREVER TABLESPACE temp;
}
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

If the database uses locally-managed temporary tablespaces, and you restore a backup control file, and you do not have a recovery catalog, then you must add tempfiles to these tablespaces afterwards, using the SQL `ALTER TABLESPACE... ADD TEMPFILE` command:

```
RMAN> SQL "ALTER TABLESPACE temp ADD TEMPFILE '"/oradata/trgt/temp01.dbf"' REUSE";
```

RECOVER an Image Copy of a Datafile to a Point in Time: Example Assume that you have an image copy backup of a datafile and you want to roll it forward in time using incremental backups. Use `RECOVER DATAFILECOPY` with the `UNTIL TIME` option. For example, run the following command:

```
RMAN> RECOVER DATAFILECOPY '/disk1/3img.df' UNTIL TIME 'SYSDATE-7';
```

Available incremental backups are applied to the datafile copy to recover it to the desired point in time. Redo from the archived redo logs is not applied by this command.

REGISTER

Syntax

register::=

→ REGISTER DATABASE **;**

Purpose

To register the target database in the recovery catalog so that RMAN can access it. RMAN obtains all information it needs to register the target database from the target database itself.

Note: If you perform a RESETLOGS operation on a database and later register it in the recovery catalog, the catalog records the DB_NAME for the old incarnations as UNKNOWN because the old incarnations were not previously registered. You should not try to remove these records.

See Also: *Oracle Database Backup and Recovery Advanced User's Guide*, and ["CREATE CATALOG"](#) on page 2-89

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt.
- You must be connected to the target database and recovery catalog database.
- You can register multiple databases in the same recovery catalog, but the database identifiers of the databases must be unique.
- You can only register a database once in a given recovery catalog.
- The target database must be mounted or open.
- You should not register a standby database.
- 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 standalone DBNEWID utility.

Note: If you are using RMAN with different target databases that have the same database name and DBID, 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

Example

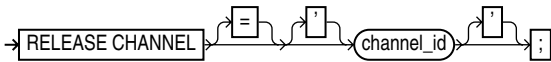
Registering a Database: Example The following commands register a new target database, catalogs an existing datafile copy, then opens the database for use:

```
% rman TARGET / CATALOG rman/rman@catdb
RMAN> STARTUP FORCE MOUNT;
RMAN> REGISTER DATABASE;
RMAN> CATALOG DATAFILECOPY '?/oradata/system01.cpy';
RMAN> ALTER DATABASE OPEN;
```


RELEASE CHANNEL

Syntax

release::=



Purpose

To release a channel while maintaining the connection to the target database instance. Specify the channel name with the same identifier used in the [ALLOCATE CHANNEL](#) command. This command is optional because RMAN automatically releases all channels allocated when a [RUN](#) block terminates.

Restrictions and Usage Notes

Execute this command only within a [RUN](#) block. See "[releaseForMaint](#)" on page 2-166 for the form of `RELEASE CHANNEL` used for maintenance channels.

Keywords and Parameters

Syntax Element	Description
<i>channel_id</i>	The case-sensitive channel ID used in the ALLOCATE CHANNEL command.

Examples

Releasing a Channel: Example This example makes three identical backup sets of datafiles 1 to 4 to tape with channel `ch1`, then releases it. RMAN then makes three identical backups of datafiles 5 to 7 to tape with channel `ch2` and then releases it:

```
RUN {
  SET BACKUP COPIES = 3;
  ALLOCATE CHANNEL ch1 DEVICE TYPE sbt FORMAT 'bkup_%U';
  ALLOCATE CHANNEL ch2 DEVICE TYPE sbt MAXPIECESIZE = 5M;
  BACKUP CHANNEL ch1 DATAFILE 1,2,3,4;
  RELEASE CHANNEL ch1;
  BACKUP DATAFILE 5,6,7;
}
```

releaseForMaint

Syntax

releaseForMaint::=

→ RELEASE CHANNEL **→** ;

Purpose

To release a sequential I/O device specified in an [ALLOCATE CHANNEL FOR MAINTENANCE](#) command. Note that maintenance channels are unaffected by [ALLOCATE CHANNEL](#) and [RELEASE CHANNEL](#) command issued within a [RUN](#) command.

Requirements

- Execute this command only at the RMAN prompt, not within a RUN block.
- You must have a maintenance channel allocated in order to release it.

Examples

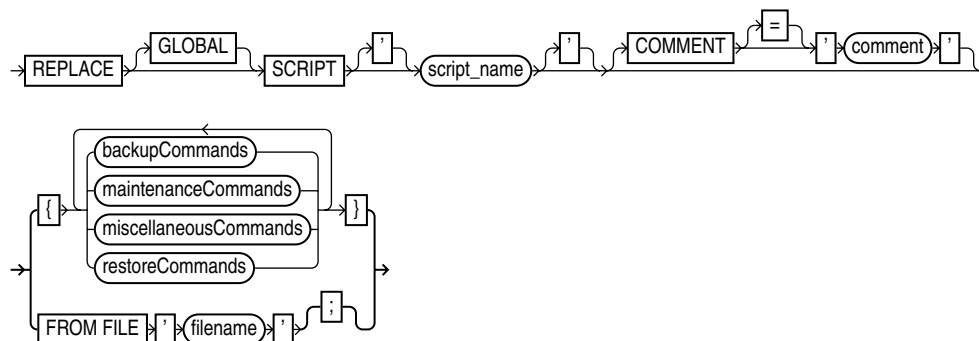
Releasing a Maintenance Channel After a Delete Operation: Example This example allocates and then releases a maintenance channel to the media manager:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;  
DELETE NOPROMPT BACKUPPIECE 100;  
RUN  
{  
    ALLOCATE CHANNEL ch1 DEVICE TYPE DISK;  
    BACKUP DATAFILE 1;  
    RELEASE CHANNEL ch1; # releases RUN channel but not maintenance channel  
}  
RELEASE CHANNEL; # releases maintenance channel
```

REPLACE SCRIPT

Syntax

replaceScript::=



Purpose

To replace an existing script stored in the recovery catalog. If the script does not exist, then `REPLACE SCRIPT` creates it.

A stored script is a sequence of RMAN commands, given a name and stored in the recovery catalog for later execution. A stored script may be local (that is, associated with one target database) or global (available for use with any database registered in the recovery catalog).

For more information about stored scripts and commands used to create, update, delete and execute stored scripts, see ["CREATE SCRIPT"](#) on page 2-91.

Restrictions and Usage Notes

- Execute `REPLACE SCRIPT` only at the RMAN prompt.
- You must be connected to a target database and recovery catalog. 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.
- When using `REPLACE SCRIPT`, RMAN must be connected to a recovery catalog, and the catalog database must be open.
- If no script by the specified name exists, one will be created.
- The `@` and `@@` commands do not work within `REPLACE SCRIPT`.

Keywords and Parameters

Syntax Element	Description
<code>GLOBAL</code>	Restricts RMAN to replacing or creating a global stored script, instead of a local one. If omitted, RMAN looks for a local stored script <i>script_name</i> defined on the current target database. If one is found, RMAN updates the local script with the new script contents. If no local script is found, RMAN looks for a global stored script <i>script_name</i> . If a global script <i>script_name</i> is found, RMAN updates it with the new script contents.
<i>'script_name'</i>	Identifies the local or global script being replaced.

Syntax Element	Description
COMMENT = 'comment'	An explanatory comment, used in the output of LIST SCRIPT NAMES..
FROM FILE 'filename'	The new contents of the script are to be read from this file. The first line of the file must begin with a "{", the last line must contain a "}" and the commands must be valid within a RUN block.
{ backupCommands maintenanceCommands miscellaneousCommands restoreCommands }	Commands valid in a stored script. These are the same commands supported within a RUN block. See " RUN " on page 2-194 for more details about backupCommands, restoreCommands, maintenanceCommands, and miscellaneousCommands.

Example

Replacing a Recovery Catalog Script: Example This example updates a stored script called backup_full:

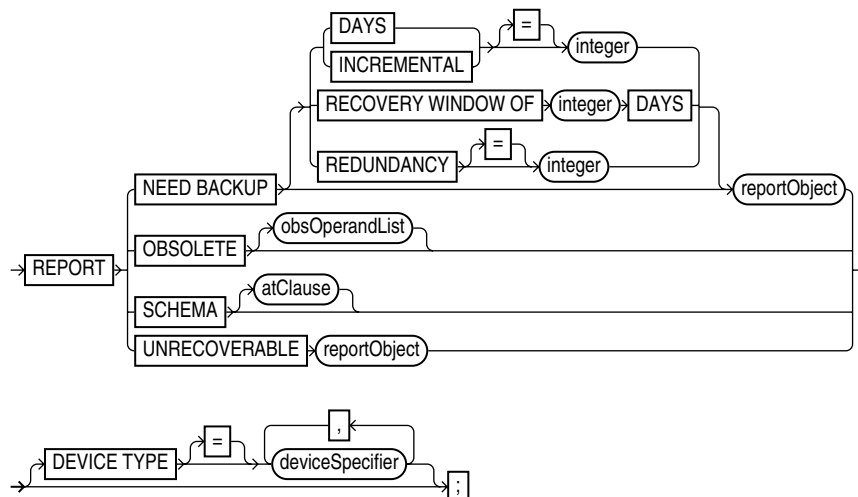
```
REPLACE SCRIPT backup_full
COMMENT 'Run this to back up the database'
{
    # uses configured channel for default device type
    BACKUP DATABASE;
}
```

If there is a local script backup_full, it is updated. If there is no local script backup_full but there is a global script backup_full, the global script is updated.

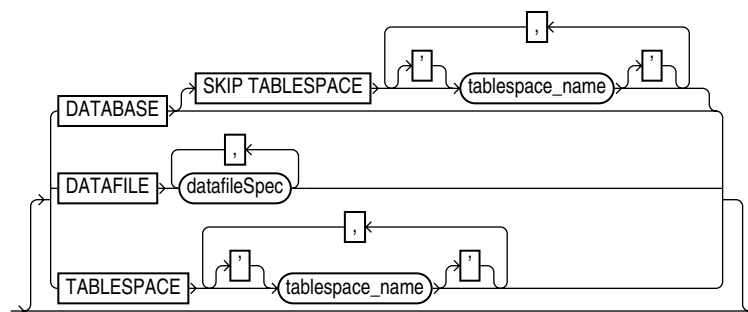
REPORT

Syntax

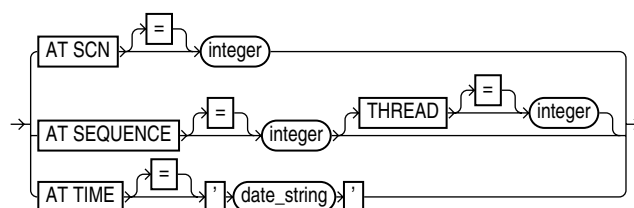
report::=



reportObject::=



atClause::=



Purpose

To perform detailed analyses of the RMAN repository. The Database writes the output from the REPORT command to standard output or the message log file.

Use the REPORT command to answer questions such as the following:

- Which files need a backup?
- Which files have not had a backup for some time?
- Which files are not recoverable due to unrecoverable operations?

- Which backup files can be deleted?
- What was the physical schema of the database at a previous time?

See Also: *Oracle Database Backup and Recovery Basics* to learn how to use RMAN's reporting functionality

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt.
- You must connect to a recovery catalog when issuing a REPORT SCHEMA command with the AT TIME, AT SCN, or AT SEQUENCE options. Otherwise, a recovery catalog is not required for the REPORT command.

Keywords and Parameters

report

Syntax Element	Description
NEED BACKUP	Lists all datafiles in need of a new backup. The report assumes that you will use the most recent backup for restore operations. 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), RMAN generates an error.
DAYS = <i>integer</i>	<p>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.</p> <p>If the target database control file is mounted and current, then RMAN makes the following optimizations to this report:</p> <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.
INCREMENTAL = <i>integer</i>	<p>Specifies a threshold number of incremental backups required for recovery. If complete recovery of a datafile requires more than <i>integer</i> incremental backups, then the datafile requires a new full backup.</p> <p>Note: Files for which no backups exist will not appear in this list: issue the REPORT NEED BACKUP REDUNDANCY command to display them.</p>
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> .
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.

Syntax Element	Description
OBSOLETE obsOperandList	<p>Lists full backups, datafile copies, and archived logs recorded in the RMAN repository that can be deleted because they are no longer needed. The subclause obsOperandList describes the criteria that RMAN uses to determine what is obsolete. If you do not specify parameters in obsOperandList, then RMAN uses the options specified in CONFIGURE RETENTION POLICY. If you use this option in conjunction with DEVICE TYPE, then RMAN only considers backups and copies created on the specified device.</p> <p>Note: Even if you use a KEEP UNTIL clause to specify a time after which a backup is to be considered obsolete, RMAN does not consider that backup obsolete if it is needed to satisfy the specified retention policy. A KEEPUNTIL clause can cause a backup to be kept longer than the retention policy requires, but never a shorter time.</p>
SCHEMA	Lists the names of all datafiles and tablespaces at the specified point in time.
UNRECOVERABLE	<p>Lists all unrecoverable datafiles. A datafile is considered unrecoverable if an unrecoverable operation has been performed against an object residing in the datafile since the last backup of the datafile.</p> <p>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.</p>
DEVICE TYPE deviceSpecifier	Specifies the type of storage device. RMAN only considers backups and copies available on the specified device for its report.

reportObject

Syntax Element	Description
<i>reportObject</i>	<p>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.</p> <p>Note: RMAN includes objects from prior incarnations.</p>
DATAFILE datafileSpec	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.
DATABASE	Lists backups or datafile copies of all files in the current database. Specify SKIP TABLESPACE <i>tablespace_name</i> to exclude the specified tablespace from the DATABASE specification.

atClause

Syntax Element	Description
<i>atClause</i>	Specifies a point in time as a time, an SCN, or a log sequence number.
AT TIME = ' <i>date_string</i> '	Specifies a date. The NLS_LANG and NLS_DATE_FORMAT environment variables specify the format for the time.
AT SCN = <i>integer</i>	Specifies an SCN.
AT SEQUENCE = <i>integer</i> THREAD = <i>integer</i>	Specifies a log sequence number for a specified redo THREAD number. The integer indicates the time when the specified log and thread were first opened.

Report Output

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

- [Table 2–21, " Report of Database Schema"](#)
- [Table 2–22, " Report of Obsolete Backups and Copies"](#)
- [Table 2–23, " Report of Files that Need Backup Due to Unrecoverable Operations"](#)
- [Table 2–24, " Report of Files with Fewer Than n Redundant Backups"](#)
- [Table 2–25, " Report of Files Whose Recovery Needs More Than n Days of Archived Logs"](#)
- [Table 2–26, " Report of Files That Need More than n Incrementals During Recovery"](#)

Table 2–21 Report of Database Schema

Column	Indicates
File	The absolute datafile number.
K-bytes	The size of the file in kilobytes.
Tablespace	The tablespace name.
RB segs	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 datafiles only. The filename of the datafile.
Maxsize (MB) Name	For tempfiles only. The maximum size of the tempfile.
Tempfile Name	For tempfiles only. The filename of the tempfile.

Table 2–22 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 2–23 Report of Files that Need Backup Due to Unrecoverable Operations

Column	Indicates
File	The absolute number of the datafile that needs a new backup due to unrecoverable operations.
Type Of Backup Required	FULL or INCREMENTAL, depending on which type of backup is necessary to ensure the recoverability of all of the data in this file. If FULL, then create a full backup, level 0 backup, or a datafile copy. If INCREMENTAL, then a full or incremental backup will also suffice.
Name	The name of the datafile.

Table 2–24 Report of Files with Fewer Than n Redundant Backups

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

Table 2–25 Report of Files Whose Recovery Needs More Than n Days of Archived Logs

Column	Indicates
File	The absolute file number of a datafile that requires more than n days of archived redo logs for recovery.
Days	The number of days of archived redo data required for recovery.
Name	The name of the datafile.

Table 2–26 Report of Files That Need More than n Incrementals During Recovery

Column	Indicates
File	The absolute file number of a datafile that requires more than n incrementals for complete recovery.
Incrementals	The number of incremental backups required for complete recovery.
Name	The name of the datafile.

Examples

Reporting Database Schema: Example This example, which requires a recovery catalog, reports the names of all datafiles and tablespaces one week ago:

```
REPORT SCHEMA AT TIME 'SYSDATE-7';
```

```
Report of database schema
File K-bytes    Tablespace          RB segs Datafile Name
-----
1      307200 SYSTEM                ***    /oracle/oradata/trgt/system01.dbf
2       20480 UNDOTBS                ***    /oracle/oradata/trgt/undotbs01.dbf
3       10240 CWWMLITE                ***    /oracle/oradata/trgt/cwwmlite01.dbf
4       10240 DRSYS                  ***    /oracle/oradata/trgt/drsys01.dbf
5       10240 EXAMPLE                 ***    /oracle/oradata/trgt/example01.dbf
6       10240 INDX                    ***    /oracle/oradata/trgt/indx01.dbf
7       10240 TOOLS                   ***    /oracle/oradata/trgt/tools01.dbf
8       10240 USERS                    ***    /oracle/oradata/trgt/users01.dbf
```

Reporting Datafiles Needing Incremental Backups: Example This example reports all datafiles in the database that require the application of five or more incremental backups to be recovered to their current state:

```
REPORT NEED BACKUP INCREMENTAL 5 DATABASE;
```

```
Report of files that need more than 5 incrementals during recovery
File Incrementals Name
-----
2      9          /oracle/oradata/trgt/undotbs01.dbf
3      9          /oracle/oradata/trgt/cwwmlite01.dbf
```

4 9 /oracle/oradata/trgt/drsys01.dbf

Reporting Datafiles Needing Backups: Example The following example reports all datafiles from tablespace SYSTEM that will need more than two days of archived redo logs to be applied during recovery after being restored from the most recent backup:

```
REPORT NEED BACKUP DAYS 2 TABLESPACE SYSTEM;
```

Report of files whose recovery needs more than 2 days of archived logs

File	Days	Name
------	------	------

1	3	/oracle/oradata/trgt/drsys01.dbf.f
---	---	------------------------------------

Reporting Unrecoverable Datafiles: Example The following example reports all datafiles that cannot be recovered from existing backups because redo may be missing:

```
REPORT UNRECOVERABLE;
```

Report of files that need backup due to unrecoverable operations

File	Type	of Backup	Required	Name
------	------	-----------	----------	------

1	full			/oracle/oradata/trgt/system01.dbf
---	------	--	--	-----------------------------------

Reporting Obsolete Backups and Copies: Example The following example reports obsolete backups and copies with a redundancy of 1:

```
REPORT OBSOLETE;
```

Report of obsolete backups and copies

Type	Key	Completion Time	Filename/Handle
------	-----	-----------------	-----------------

Backup Set	1	OCT 30 2001 15:54:56	
Backup Piece	1	OCT 30 2001 15:54:56	/oracle/dbs/01d7t0t9_1_1
Archive Log	1	OCT 30 2001 04:52:17	/oracle/oradata/trgt/arch/archive1_21.dbf

RESET DATABASE

Syntax

reset::=

→ RESET DATABASE TO INCARNATION > {primaryKey} > ;

Purpose

To reset the incarnation of the target database in the RMAN repository, which means to do either of the following actions:

- Inform RMAN that the SQL statement `ALTER DATABASE OPEN RESETLOGS` has been executed and that a new incarnation of the target database has been created. Note that if you run the RMAN command `ALTER DATABASE OPEN RESETLOGS` (not the SQL statement with the same keywords), then RMAN resets the target database automatically so that you do not have to run `RESET DATABASE`. By resetting the database, RMAN considers the new incarnation as the current incarnation of the database.
- To reset the database to a previous incarnation. Typically, you would reset the incarnation when performing incomplete recovery to a point before a `RESETLOGS` operation, or when attempting to undo the affects of a `RESETLOGS` by restoring backups taken before a `RESETLOGS`.

Restrictions and Usage Notes

- Execute `RESET DATABASE` only at the RMAN prompt.
- You must be connected to the target database.
- A recovery catalog connection is optional. Unlike in catalog mode, `RESET DATABASE` in nocatalog mode changes the incarnation only for the current RMAN session.
- You must issue a `RESET DATABASE` command before you can use RMAN with a target database that has been opened with the SQL statement `ALTER DATABASE OPEN RESETLOGS` option. If you do not, then RMAN refuses to access the recovery catalog because it cannot distinguish between a `RESETLOGS` operation and an accidental restore of an old control file. The `RESET DATABASE` command informs RMAN that you issued a `RESETLOGS` command.
- If RMAN is connected `NOCATALOG`, then you can only specify `TO INCARNATION` if the database is mounted and the control file contains a record of the prior incarnation. If you do not run `RESET DATABASE`, RMAN recovers to the last incarnation recorded in the control file.
- If RMAN is connected in `CATALOG` mode, then you can specify `TO INCARNATION` when the database is mounted. If database is mounted, however, then the control file must have a record of the prior incarnation.

Keywords and Parameters

Syntax Element	Description
TO INCARNATION <i>primary_key</i>	Changes the current incarnation to an older incarnation. Specify the primary key of the DBINC record for the database incarnation. Run LIST INCARNATION OF DATABASE to obtain possible key values. After you issue RESET DATABASE TO INCARNATION, then you can run RMAN commands such as RESTORE and RECOVER.

Examples

Resetting RMAN to a Previous Incarnation in NOCATALOG Mode: Example In NOCATALOG mode, you must mount a control file that knows about the incarnation that you want to recover. The following scenario makes an old incarnation of database trgt current again:

```
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 2001 16:02:12
1	116	TRGT	1334358386	CURRENT	154877	OCT 30 2001 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 2001 16:02:12
1	116	TRGT	1334358386	PARENT	154877	OCT 30 2001 16:37:39
1	311	TRGT	1334358386	CURRENT	154877	AUG 13 2002 17:17:03

Resetting the Database After Incomplete Recovery: Example This example assumes that an incomplete recovery or recovery with a backup control file was performed in NOCATALOG mode. Later, RMAN is started in CATALOG mode, but the RESYNC command fails because the incarnation has not been reset in the catalog.

```
% rman target / catalog rman/rman@catdb
```

```
Recovery Manager: Release 10.1.0.2.0 - Production
```

Copyright (c) 1995, 2003, Oracle. All rights reserved.

connected to target database: TRGT (DBID=1334531173)
connected to recovery catalog database

RMAN> RESYNC CATALOG;

RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03009: failure of resync command on default channel at 11/01/2001 12:00:43
RMAN-20003: target database incarnation not found in recovery catalog

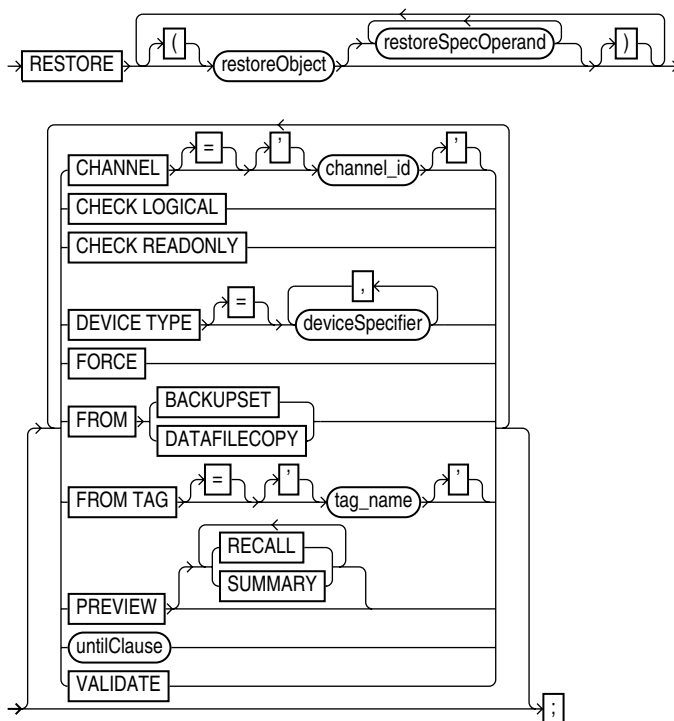
RMAN> RESET DATABASE;

new incarnation of database registered in recovery catalog
starting full resync of recovery catalog
full resync complete

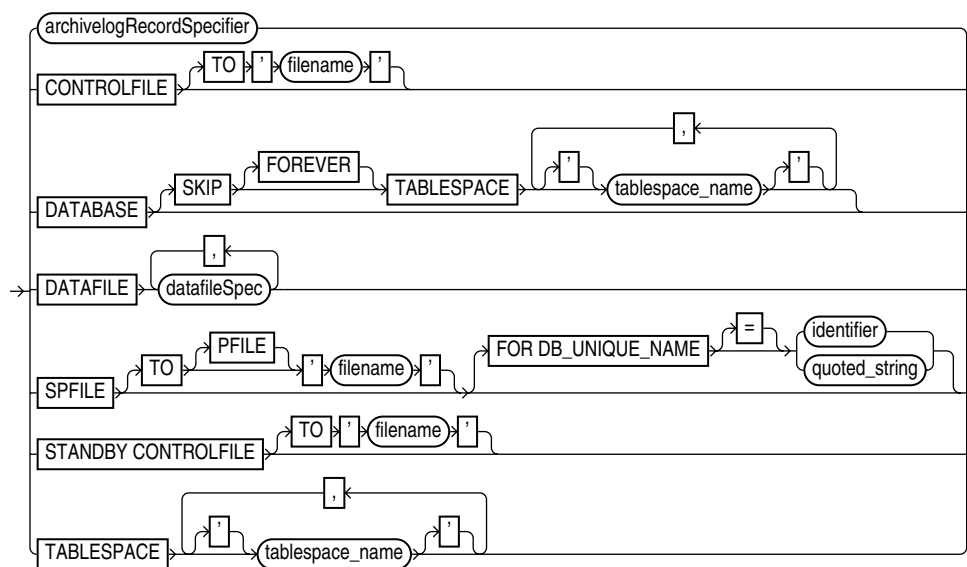
RESTORE

Syntax

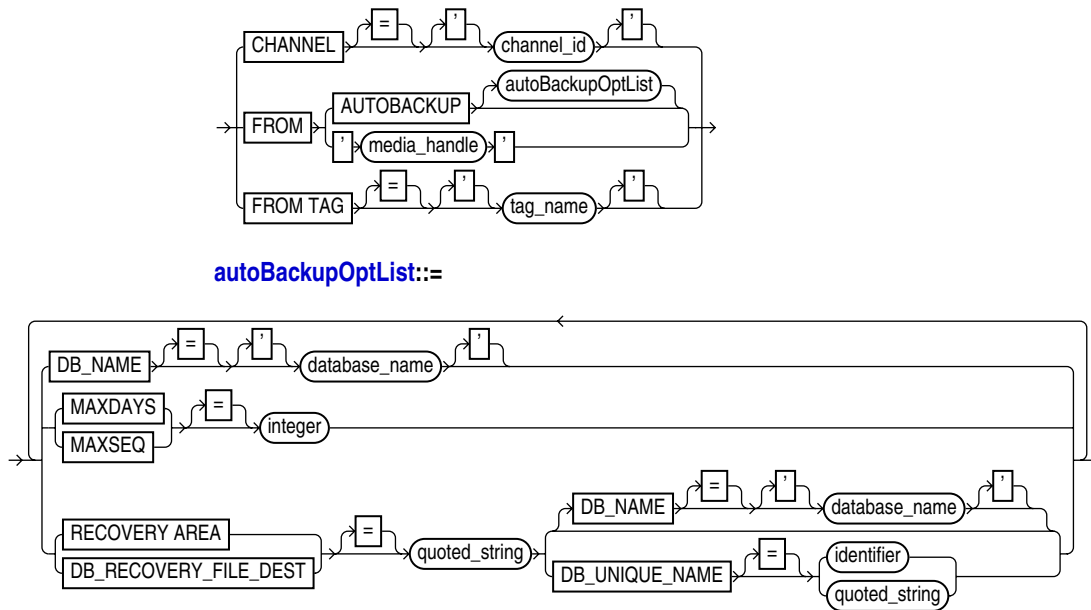
restore::=



restoreObject::=



restoreSpecOperand::=



Purpose

The primary use of RESTORE is to restore files from backups or image copies. Typically, you restore when a media failure has damaged a current datafile, control file, or archived log or prior to performing a point-in-time recovery.

There are other uses of RESTORE that do not actually restore files from backup:

- RESTORE... VALIDATE causes RMAN to select existing backups that it would use to perform a RESTORE operation, and scan them all to ensure that they are present and free of corruption.
- RESTORE... PREVIEW identifies the backups which RMAN will use to perform any RESTORE operation. Output from a RESTORE... PREVIEW is in the same format as the output of the [LIST](#) command.
- RESTORE... PREVIEW RECALL is used to request the recall of backups needed for a RESTORE operation that are stored on remote storage.

RMAN chooses which backups to restore based on the criteria that you specify. For example, you can limit the restore to backups before a given point of time (within the current incarnation) with the [untilClause](#).

Locations of Restored Files

If you restore to the default location (that is, you do not run [SET NEWNAME](#)), then RMAN overwrites files with the same filenames. If you restore to a new location, then issue [SET NEWNAME](#) commands to rename the files and issue a [SWITCH](#) command to make the restored files current. If you do not issue [SWITCH](#) commands, then RMAN considers the restored files as valid copies for use in future restore operations.

If you do not run [SET NEWNAME](#) and RMAN detects that the default filename cannot be used (for example, because the filename is in use by another database that shares the storage), and if the file is an Oracle Managed File or is on an Automatic Storage Management disk group, then RMAN attempts to create a new file in the same location or disk group.

Note: By default, RMAN does not restore a datafile if the file is in the correct place and its header contains the expected data (RMAN does not scan the datafile body for corrupt blocks). The `FORCE` option overrides this behavior and restores the requested files unconditionally.

Channel Allocation in a Restore

If you do not manually allocate channels, then RMAN allocates all automatic channels possibly needed by the `RESTORE` command, subject to any restrictions imposed by the use of the `DEVICE TYPE` option.

For example, assume you configure 3 separate `sbt` channels (each with different `PARMS`) and then configure parallelism for `DISK` and `sbt` as follows:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;  
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;  
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

If you run `RESTORE` in this configuration, then RMAN allocates three `sbt` channels and the two preconfigured `DISK` channels.

Restore Failover

If a backup piece, image copy or proxy copy is inaccessible (for instance, deleted from the device) or if a block is corrupted, then the `RESTORE` command automatically looks for another usable copy of this backup piece or image copy on both the same device and other devices. If no usable copies are available, then RMAN searches for prior backups. RMAN continuously searches for prior usable backups until it has exhausted all possibilities.

Restore failover also occurs when restoring archivelogs for use in `RECOVER`, `BLOCKRECOVER` and `FLASHBACK` operations. RMAN records messages about failover due to block corruption in the alert log and trace files.

See *Oracle Database Backup and Recovery Advanced User's Guide* for details on restore failover.

Restoring from Encrypted Backup Sets

How RMAN handles encrypted backup sets during restore operations depends upon the encryption mode with which the backup was created.

- For transparent-mode encrypted backups, the required passwords must be available in the database wallet. This means that the same wallet that was used when creating the backup must be open, and available, when restoring it. No `SET DECRYPTION` command is required.
- For password-mode encrypted backups, the required passwords must be provided using `SET DECRYPTION`.
- For dual-mode encrypted backups, the required passwords must be available in the database wallet or provided using `SET DECRYPTION`.

For an overview of the encrypted backups mechanism, a guide to its use and information on choosing among the different modes of encryption, see "[Encryption of Backup Sets](#)" on page 2-28 and the extended discussion in *Oracle Database Backup and Recovery Advanced User's Guide*.

The RMAN [CONFIGURE](#), [SET](#) and [SHOW](#) commands are used to manage the RMAN backup encryption settings for your database. See the reference entries for those commands for more details.

Restoring Files in a Real Application Cluster Configuration

In a Real Application Clusters 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 1 connected to instance 1 can read log 1000 from its tape drive, but channel 2 connected to instance 2 cannot read the same log from its tape drive, then channel 1 restores the log. Autolocation is automatically enabled when the channels meet any of the following criteria:

- Different PARMS settings
- Different CONNECT strings

See Also: *Oracle Database Backup and Recovery Basics* to learn how to restore files

Restrictions and Usage Notes

- 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 the database is started but not mounted, then it is recommended that you only restore the control file, if necessary (refer to "[Restrictions and Usage Notes for RESTORE CONTROLFILE](#)" on page 2-182). To restore other files, mount the database and then continue.

Note: When performing a database validation by using `RESTORE . . . VALIDATE`, the database can be open.

- To restore to a new location, run `SET NEWNAME` commands to rename the datafiles and [SWITCH](#) commands to make them the current database files. If you do not use `SWITCH`, then the repository lists restored datafiles as datafile copies.
- If you use the `FROM DATAFILECOPY` option, then the allocated channels must be of `DEVICE TYPE DISK`.
- If you use the `FROM BACKUPSET` clause, then channels for the appropriate type of storage devices must be allocated for the backup sets that need to be restored. For example, if needed backups are only available on tape, and no SBT channels have been allocated, then RMAN cannot find a candidate backup set to restore, and the `RESTORE` command fails.
- RMAN only restores backups that were created on the same type of channels that are allocated for the `RESTORE` command.

For example, if you made some backups of a datafile to `DISK` channels and others to `SBT` channels, and only a `DISK` channel is allocated for the `RESTORE` command, RMAN will not restore backups that were created on `SBT` channels.
- If there are no backups available for a lost datafile, RMAN will create an empty datafile with the checkpoint change as creation SCN. During recovery, all archived logs back to the creation of the datafile will be restored, and all changes during the history of the datafile will be re-applied to re-create its contents.
- If datafile names are symbolic links, that is, files pointing to other files, 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.

- If the database is started but not mounted in NOCATALOG mode, then the RESTORE SPFILE command requires the FROM AUTOBACKUP clause.
- If you are restoring the server parameter file and the control file in a disaster recovery situation, you cannot run RESTORE CONTROLFILE FROM AUTOBACKUP, mount this control file, and then run RESTORE SPFILE *without* the FROM AUTOBACKUP clause.
- Do not specify a datafile more than once in a restore job. For example, the following command is illegal because `datafile 1` is both specified explicitly and implied by the `SYSTEM` tablespace:

```
RESTORE TABLESPACE SYSTEM DATAFILE 1;
```

- You must have already configured a device type by using CONFIGURE (except for DISK, which is preconfigured) before specifying the DEVICE TYPE option.
- You cannot manually allocate channels and then run RESTORE DEVICE TYPE.
- When you run RESTORE with a backup control file while connected to a recovery catalog, RMAN automatically updates the control file to reflect the structure of the restored database, based on the information in the recovery catalog.
- RMAN can back up and restore dictionary-managed temporary tablespaces, but it cannot back up locally managed temporary tablespaces. It can, however, re-create locally managed temporary tablespaces after a database restore.
- RMAN does not support backup and recovery of the change tracking file. Note that database restore and recovery has no user-visible effect on change tracking. The change tracking file is re-created upon database restore and recovery, and the next incremental backup after any recovery is able to use the change-tracking file.
- If no suitable backups are available in the current incarnation of the database, then you can force RMAN to use backups from a previous incarnation. Using the [CHANGE... UNAVAILABLE](#) command, you can make all backups since the RESETLOGS that ended the incarnation unavailable. Run [LIST RECOVERABLE](#) to see valid parent incarnations.

Restrictions and Usage Notes for RESTORE CONTROLFILE

- After you restore a backup control file, you must run RECOVER DATABASE and then open the database with the RESETLOGS option.

[Table 2-27](#) indicates the restrictions that apply in different situations involving the RESTORE CONTROLFILE command.

Table 2–27 RESTORE CONTROLFILE Scenarios

	RESTORE CONTROLFILE;	RESTORE CONTROLFILE FROM AUTOBACKUP;	RESTORE CONTROLFILE ... TO 'filename';	RESTORE CONTROLFILE ... FROM 'media_handle' or TAG 'user_tag';
No catalog, target started in NOMOUNT state	Error. Must specify FROM AUTOBACKUP.	First run SET DBID. Restores to CONTROL_FILES locations.	First run SET DBID. 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 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 started in NOMOUNT state	Restores to CONTROL_FILES locations. Run SET DBID only if DB_NAME not unique in catalog.	Only use with catalog for testing purposes.	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 mounted or open	Error. Must use TO <i>'filename'</i> , where <i>filename</i> is not in CONTROL_FILES list.	Do not use with catalog.	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.

Restrictions and Usage Notes on RESTORE... PREVIEW

Backup media may be stored remotely, whether in a backup vaulting scenario where media is shipped to an off-site location, or in a tape library where media is on-site but not immediately accessible. Some media managers maintain information about whether the media containing specific backups is stored remotely, and make this information available to RMAN.

Backups stored remotely are marked as AVAILABLE in the RMAN repository, even though the media must be retrieved from storage before the backup can be restored. If RMAN attempts to use a remotely stored backup in a RESTORE operation, the RESTORE operation fails.

You can use RESTORE... PREVIEW to identify backups RMAN needs for a RESTORE operation that are stored on media that requires retrieval. The output of RESTORE... PREVIEW indicates whether backups are stored remotely. If a needed backup is stored remotely, your options are:

- Use CHANGE... UNAVAILABLE to prevent RMAN from selecting the needed remote backups, and attempt the RESTORE... PREVIEW operation again to see if RMAN selects another remote backup. When RMAN does not select any remote backups, you can perform your RESTORE operation.
- Use RESTORE... PREVIEW with the RECALL option. If your media manager supports this functionality, RESTORE... PREVIEW RECALL automatically requests the retrieval of the remotely stored backup media. Use RESTORE... PREVIEW again, periodically, to monitor whether the needed backups are stored locally again.

Keywords and Parameters

restore

Syntax Element	Description
CHANNEL 'channel_id'	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, it logs the block in the <code>alert.log</code> 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>MAXCORRUPT</code> setting, 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, the command terminates without populating the views.</p> <p>If the initialization parameter <code>DB_BLOCK_CHECKSUM=TRUE</code>, and if <code>MAXCORRUPT</code> and <code>NOCHECKSUM</code> are 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>
CHECK READONLY	Checks the read-only datafiles to make sure they exist, are readable, and have the appropriate checkpoint. If any of these conditions is not met, then RMAN restores the files—whether or not they are read-only. By default, RMAN does not restore read-only files when you issue the <code>RESTORE DATABASE</code> command.
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.</p> <p>See Also: "<i>deviceSpecifier</i>" on page 2-101</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 DATAFILECOPY]	Specifies whether RMAN should restore from a <code>DATAFILECOPY</code> on disk or a <code>BACKUPSET</code> . By default <code>RESTORE</code> chooses the most recent backup set or file copy, that is, the file copy or backup set that needs the least media recovery.
FROM TAG = 'tag_name'	Refer to the <i>restoreSpecOperand</i> clause.
PREVIEW [SUMMARY RECALL]	<p>When <code>PREVIEW</code> or <code>PREVIEW SUMMARY</code> is appended to any <code>RESTORE</code> command, instead of performing the restore RMAN will report the backups (on disk or sequential media) it will use during the restore. The output will be in the same format as is generated by the <code>LIST BACKUPS</code> and <code>LIST BACKUPS... SUMMARY</code> commands.</p> <p>Some media managers provide status information to RMAN about which backups are vaulted, that is, stored in a remote location such as a secure storage facility, and which therefore cannot be used without retrieving media. The output of <code>RESTORE</code> identifies such backups, if the media manager provides the necessary information.</p> <p>When used with the <code>RECALL</code> option, <code>RESTORE... PREVIEW</code> initiates a request for the retrieval of backup media needed for the specified restore operation from remote storage.</p> <p>See Also: "<code>LIST</code>" on page 2-127, and specifically the <code>BACKUPS</code> and <code>SUMMARY</code> options.</p>

Syntax Element	Description
untilClause	Limits the selection to backup sets or file copies that are suitable for a point-in-time recovery to the specified time. 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 untilClause must fall within the current incarnation. See Also: " untilClause " on page 2-227
VALIDATE	Lets RMAN decide which backup sets, datafile copies, and archived logs need to be restored, and then scans them to verify their contents. No files are restored. Use VALIDATE to verify that the backups required to restore the specified files are intact and usable.

restoreObject

Syntax Element	Description
<i>restoreObject</i>	Specifies the objects to be restored. The RESTORE command restores full backups, incremental backups (level 0 only), or copies of datafiles, control files, and archived redo logs.
archivelogRecordSpecifier	Restores the specified range of archived redo logs. 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 parameters of the target database. These parameters combine in a port-specific fashion to derive the name of the restored log. You can override the default location with the SET ARCHIVELOG DESTINATION command. Because the RECOVER command automatically restores archived logs as needed, you should seldom need to restore logs manually. Possible reasons for manually restoring archived logs are to speed up recovery or to stage the logs to multiple destinations. See Also: " archivelogRecordSpecifier " on page 2-18 Note: The database can be started, mounted, or open for this operation.
CONTROLFILE	Restores the current control file for a primary database. See Also: Table 2-27 for restrictions and usage notes. Note: You must always run the RECOVER command after restoring a control file, and must also always open the database with the RESETLOGS option.
DATABASE [SKIP [FOREVER] TABLESPACE tablespace_name]	Restores all datafiles in the database except those that are offline or read-only. 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. If you specify the CHECK READONLY option, then RMAN examines the headers of all read-only datafiles and restores any that need restoring. Use an optional SKIPTABLESPACE 'tablespace_name' argument to avoid restoring specified tablespaces, which is useful when you want to avoid restoring tablespaces containing temporary data. If you specify SKIP FOREVER TABLESPACE, 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. In other words, FOREVER indicates that RMAN never intends to do anything with the skipped tablespaces again.

Syntax Element	Description
DATAFILE datafileSpec	<p>Restores the datafiles specified by filename or absolute datafile number.</p> <p>See Also: "datafileSpec" on page 2-95</p>
SPFILE [TO [PFILE] 'filename']	<p>Restores a primary or standby server parameter file to the location from which it was backed up (default), or to a different location specified by the TO clause. RMAN cannot overwrite a server parameter file currently in use by the target database.</p> <p>Specify UNTIL or TAG options of RESTORE to restore older versions of the server parameter file. By default RMAN restores the most current server parameter file.</p> <p>If the server parameter file is lost, connect to the target (and catalog if used) and then run SET DBID. Run STARTUP FORCE NOMOUNT before running RESTORE SPFILE (with FROM AUTOBACKUP if in NOCATALOG mode). Then run STARTUP FORCE to restart the database with the restored server parameter file.</p> <p>See Also: "Restrictions and Usage Notes for the SET DBID Command" on page 2-199</p>
[FOR DB_UNIQUE_NAME 'DB_UNIQUE_NAME']	<p>The FOR DB_UNIQUE_NAME clause for RESTORE SPFILE is only useful in a Data Guard environment.</p> <p>In a Data Guard environment, the primary and standby nodes 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 using the value of the DB_UNIQUE_NAME initialization parameter for the primary and standby databases.</p> <p>RMAN cannot determine the DB_UNIQUE_NAME initialization parameter for the target database if the SPFILE of the target database is not available, as is the case when restoring the SPFILE from backup. Therefore, RMAN cannot choose the correct channel configurations to use when restoring the SPFILE from backup at the target node. In such a circumstance, RMAN returns an RMAN-6758 error.</p> <p>The FOR DB_UNIQUE_NAME clause of the RESTORE command is used to specify the DB_UNIQUE_NAME in this situation. With the DB_UNIQUE_NAME, RMAN is able to locate the correct configuration settings for the node on which the SPFILE is being restored, and can use them to access backup devices.</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 SPFILE in a Data Guard environment.</p>
STANDBY CONTROLFILE [TO 'filename']	<p>Restores the current control file for a standby database.</p> <p>See Also: Table 2-27 for restrictions and usage notes.</p> <p>Note: You must always run the RECOVER command after restoring a control file, and must also always open the database with the RESETLOGS option.</p>
TABSPACE 'tablespace_name'	<p>Restores all datafiles in the specified tablespaces.</p> <p>The 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>

restoreSpecOperand

Syntax Element	Description
<i>restoreSpecOperand</i>	Specifies options for the <i>restoreObject</i> clause. Note: These parameters override the parameters with the same name at the RESTORE command level.
CHANNEL 'channel_id'	Specifies the case-sensitive name of a channel to use for this restore operation. If you do not specify a channel, then RESTORE uses any available channel allocated with the correct device type.
FROM AUTOBACKUP [autoBackupOptionList]	Restores a control file autobackup. You can only specify this option 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, 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 2-27 for restrictions and usage notes.
FROM 'media_handle'	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 2-27 for restrictions and usage notes.
FROM TAG [=] 'tag_name'	Overrides the default selection of the most recent backups or file copy available. The tag restricts the automatic selection to backup sets or file copies that were created with the specified tag. If multiple backup sets or file copies have a matching tag, then RMAN selects the most recent one. Note that tag names are not case sensitive. See Also: "BACKUP" on page 2-22 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

Syntax Element	Description
<i>autoBackupOptList</i>	Parameters that control the search for a control file autobackup.
DB_NAME = 'database_name'	Provides a DB_NAME to be used in searching for control file autobackups.
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 DB_RECOVERY_FILE_DEST) = <i>password</i>	Specifies path to flash recovery area to search for autobackups. RECOVERY AREA and DB_RECOVERY_FILE_DEST are synonyms.
DB_NAME = <i>database_name</i>	Provides the DB_NAME of the database in the specified flash recovery area that is the target of the restore operation.

Syntax Element	Description
DB_UNIQUE_NAME = { <i>identifier</i> <i>quoted-string</i> }	Provides the DB_UNIQUE_NAME of the database in the specified flash recovery area that is the target of the restore operation.

Examples

Restoring a Tablespace: Example 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";
```

Restoring the Control File When Using a Recovery Catalog: Example This example restores the control file to its default location, replicates it automatically to all CONTROL_FILES locations, and mounts the database:

```
RUN
{ # SET DBID is not necessary when connected to a recovery catalog
  STARTUP FORCE NOMOUNT;
  RESTORE CONTROLFILE;
  ALTER DATABASE MOUNT;
}
```

Restoring the Control File with a Tag: Example This NOCATALOG example restores the control file specified by a tag, and then mounts the database:

```
CONNECT TARGET /
STARTUP NOMOUNT;
SET DBID 320066378; # required when restoring control file in NOCATALOG mode
RESTORE CONTROLFILE FROM TAG 'monday_cf_backup';
ALTER DATABASE MOUNT;
```

Restoring the Database with a Backup Control File: Example This example restores the control file to a temporary location, replicates it to all control file locations specified in the CONTROL_FILES initialization parameter, and then restores and recovers the database:

```
CONNECT TARGET /
STARTUP NOMOUNT;
SET DBID 320066378; # 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;
```

Restoring Archived Redo Logs to a New Location: Example This example restores all archived redo logs to the /oracle/temp_restore directory:

```
RMAN> RUN
{
  SET ARCHIVELOG DESTINATION TO '/oracle/temp_restore';
  RESTORE ARCHIVELOG ALL;
}
```


Restoring a Control File Autobackup to a Nondefault Location: Example This example restores the latest control file autobackup made on or before June 23, 2000 with a nondefault format of PROD_CF_AUTOBACKUP_%. It starts searching for backups with a sequence number of 20, and searches backward for 5 months:

```

RMAN> SET DBID 320066378; # required when restoring control file in NOCATALOG mode
RMAN> RUN
{
  SET UNTIL TIME '23-JUN-2001 00:00:00';
  SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'prod_cf_autobackup_%.F';
  ALLOCATE CHANNEL CHANNEL_1 DEVICE TYPE sbt;
  RESTORE CONTROLFILE TO '/tmp/autobackup_20001002.dbf' FROM AUTOBACKUP
    MAXSEQ 20 MAXDAYS 150;
}

```

Restoring the Server Parameter File to Current Location: Example The following series of commands restores the current server parameter file in NOCATALOG mode:

```

rman TARGET /
RMAN> SET DBID 1447326980 # set dbid to dbid of target database
RMAN> STARTUP FORCE NOMOUNT; # start instance with dummy SPFILE
RMAN> RUN
{
  ALLOCATE CHANNEL c1 DEVICE TYPE sbt;
  RESTORE SPFILE FROM AUTOBACKUP; # FROM AUTOBACKUP needed in NOCATALOG mode
}
RMAN> STARTUP FORCE; # start with restored SPFILE and open database

```

Identifying Needed Backups with RESTORE... PREVIEW: Example The following example shows the results of a RESTORE PREVIEW, which identifies the backupsets RMAN selects for use in restoring archive logs from backup. (Note that in this output, one of the archive log backups is stored remotely.)

```
RMAN> restore archivelog all preview;
```

```

Starting restore at 10-JUN-05
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-05
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-05	222548	06-JUN-05
1	2	222548	06-JUN-05	222554	06-JUN-05
1	3	222554	06-JUN-05	222591	06-JUN-05
1	4	222591	06-JUN-05	246629	07-JUN-05
1	5	246629	07-JUN-05	262451	10-JUN-05

BS Key	Size	Device Type	Elapsed Time	Completion Time
32	256.00K	SBT_TAPE	00:00:01	10-JUN-05
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-05	262547	10-JUN-05
1	7	262547	10-JUN-05	262565	10-JUN-05

List of remote backup files

=====

Handle: 15gmknbs Media: /v1,15gmknbs

Recalling Vaulted Backups from Remote Storage with RESTORE... PREVIEW

RECALL: Example When used with a media manager that reports information about remote storage of backups and supports recalling vaulted backups, the following command requests that any media needed in the restore of archive logs from backup be recalled from remote storage:

```

RMAN> RESTORE ARCHIVELOG ALL PREVIEW RECALL;
Starting restore at 10-JUN-05
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-05
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-05	222548	06-JUN-05
1	2	222548	06-JUN-05	222554	06-JUN-05
1	3	222554	06-JUN-05	222591	06-JUN-05
1	4	222591	06-JUN-05	246629	07-JUN-05
1	5	246629	07-JUN-05	262451	10-JUN-05

BS Key	Size	Device Type	Elapsed Time	Completion Time
32	256.00K	SBT_TAPE	00:00:01	10-JUN-05
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-05	262547	10-JUN-05
1	7	262547	10-JUN-05	262565	10-JUN-05

Initiated recall for the following list of remote backup files

=====

Handle: 15gmknbs Media: /v1,15gmknbs

Finished restore at 10-JUN-05

Validating a Restore with RESTORE VALIDATE: Example The following example illustrates using RESTORE . . . VALIDATE to confirm that backups required for the restore of datafile 1 are present on disk or tape, readable, and not corrupted:

```
RMAN> RESTORE DATAFILE 1 VALIDATE;
```

```
Starting restore at 10-OCT-03  
using channel ORA_DISK_1
```

```
channel ORA_DISK_1: starting validation of datafile backupset  
channel ORA_DISK_1: restored backup piece 1  
piece handle=/disk1/oracle/work/v1/0pf3hr3o_1 tag=FOURTH_INC  
channel ORA_DISK_1: validation complete  
Finished restore at 10-OCT-03
```

RESYNC

Syntax

resync::=



Purpose

To perform a full resynchronization of the recovery catalog. You can also use `RESYNC CONTROLFILE` to resynchronize the current control file with the RMAN repository in a control file copy.

Resynchronizations can be full or partial. When full, RMAN updates all changed records for the physical schema: datafiles, tablespaces, redo threads, and online redo logs. If the database is open, RMAN also obtains data about rollback segments. When partial, RMAN reads the current control file to update data, but does not resynchronize metadata about the physical schema or rollback segments.

When you run `RESYNC CATALOG`, 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 classes or records described in the following table.

Record Type	Description
Log history	Records that are created whenever a log switch occurs. Note that log history records describe an online log switch, not a log archival.
Archived redo logs	Records associated with archived logs that were created by archiving an online redo log, copying an existing archived redo log, or restoring backups of archived redo logs.
Backups	Records associated with backup sets, backup pieces, proxy copies, and image copies.
Physical schema	Records associated with datafiles and tablespaces. If the target database is open, then rollback segment information is also updated.

RMAN automatically executes a full or partial resynchronization of the recovery catalog as needed when you execute RMAN commands, so long as the control file is mounted and the recovery catalog database is available at command execution. RMAN reads the current control file and resynchronizes metadata about the physical schema if it determines that this information has changed. If RMAN does detect a change, then it performs a full resynchronization.

Use `RESYNC CATALOG` to perform manual full resynchronizations when:

- The recovery catalog is unavailable when you issue any of the commands that automatically perform a resynchronization.
- You are running in `ARCHIVELOG` mode, because the catalog is *not* updated automatically when a log switch occurs or when an online redo log is archived.
- You have made changes to the physical structure of the target database such as adding or dropping a tablespace. As with archive operations, the recovery catalog is *not* updated automatically when the physical schema changes.

The primary use for RESYNC CONTROLFILE occurs when you re-create the control file (for example, to change the database name), which causes you to lose RMAN records. You can then resynchronize the newly created control file with an old copy.

Restrictions and Usage Notes

- You must be connected to a recovery catalog when running RESYNC CATALOG, but a catalog connection is not required for RESYNC CONTROLFILE.
- RMAN updates physical schema information in the recovery catalog only when the target database has the current control file mounted. If the target database has mounted a backup control file, a freshly created control file, or a control file that is less current than a control file that was used previously, then RMAN does not update physical schema information in the recovery catalog.

Keywords and Parameters

Syntax Element	Description
CATALOG	Updates the recovery recovery catalog with RMAN metadata in the current control file (default) or a control file copy.
CONTROLFILE	Updates the current control file (and recovery catalog, if RMAN is connected to one) with RMAN metadata from a control file copy.
FROM CONTROLFILECOPY ' <i>filename</i> '	Specifies the name of the control file copy to use for resynchronization. Physical schema information is not updated when you use this option. Note: The control file copy can either be in the current database incarnation, or created in a prior incarnation (that is, prior to the most recent OPEN RESETLOGS).

Examples

Resynchronizing the Recovery Catalog in ARCHIVELOG Mode: Example This example performs a full resynchronization after archiving all unarchived redo logs:

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

Resynchronizing the Current Control File from a Backup: Example This example updates the RMAN repository in the current control file with metadata from a backup control file:

```
CONNECT TARGET / NOCATALOG
RESYNC CONTROLFILE FROM CONTROLFILECOPY '/tmp/cfile.dbf';
```

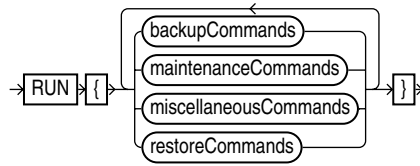
Resynchronizing the Recovery Catalog After a Structural Change: Example This example adds a datafile to tablespace users and then resynchronizes the catalog:

```
#!/usr/bin/tcsh
# connect in nocatalog mode and add datafile
rman TARGET / NOCATALOG <<EOF
SQL "ALTER TABLESPACE users ADD DATAFILE ''?/oradata/trgt/users03.dbf''
    SIZE 1M AUTOEXTEND ON NEXT 10K MAXSIZE 10M";
EXIT
EOF
# connect in catalog mode and resynchronize
rman TARGET / CATALOG rman/rman@catdb <<EOF
RESYNC CATALOG;
EOF
```

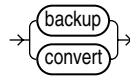
RUN

Syntax

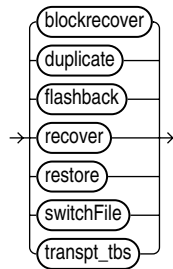
run::=



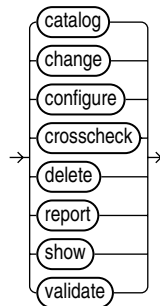
backupCommands::=



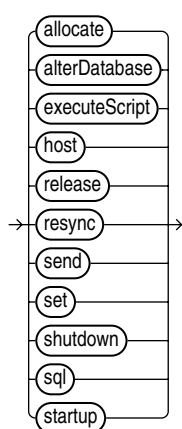
restoreCommands::=



maintenanceCommands::=



miscellaneousCommands::=



Purpose

The RUN command lets you group a series RMAN commands into a block to be executed sequentially. It also creates a scope within which a script can override default configured channels for a task using the `ALLOCATE CHANNEL` and `RELEASE CHANNEL` commands, and other parameters using the `SET` command with appropriate arguments. On completing the execution of the commands listed in the RUN block, the channels allocated within the RUN block are released and settings returned to their values.

Upon 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.

Restrictions and Usage Notes

- Execute this command only at the RMAN prompt.
- You must precede and follow the list of job commands with an opening and closing brace.

Keywords and Parameters

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

Examples

Making a Backup: Example This example backs up a database by using a single manually allocated channel to perform the backup:

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

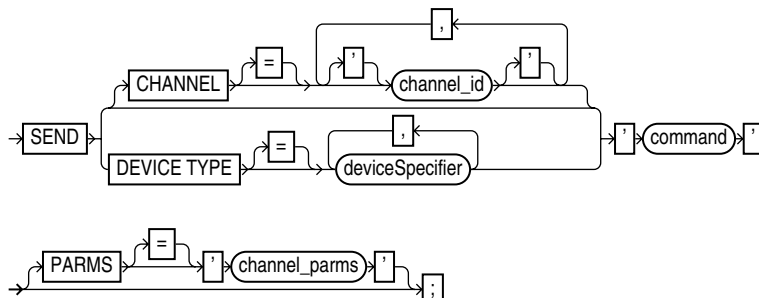
Restoring and Recovering a Tablespace: Example This example takes tablespace `tools` offline, restores it, then performs complete media recovery:

```
RUN
{
  SQL "ALTER TABLESPACE tools OFFLINE IMMEDIATE";
  RESTORE TABLESPACE tools;
  RECOVER TABLESPACE tools;
  SQL "ALTER TABLESPACE tools ONLINE";
}
```

Executing an RMAN Script: Example This example executes the stored script `backup_db`:

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


send::=



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.

- 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.

Syntax Element	Description
CHANNEL <i>'channel_id'</i>	Specifies which channel to use. If you do not specify DEVICE TYPE or CHANNEL, then RMAN uses all allocated channels. You must specify a case-sensitive channel ID, which is the name of the channel, after the CHANNEL keyword. the database uses the channel ID to report I/O errors.
DEVICE TYPE <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 2-101
<i>'command'</i>	Specifies a vendor-specific media management command. See Also: Your media management documentation to determine which commands are supported
PARMS = <i>'channel_parms'</i>	Parameters for the channel communicating with the media manager.

Sending a String to the Media Manager: Example This example sends vendor-specific commands to a media manager:

```

RMAN> SEND 'VAR=a82';
sent command to channel: ORA_SBT_TAPE_1

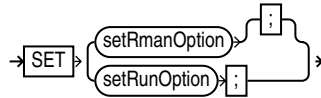
RMAN> BACKUP DATAFILE 2;

```

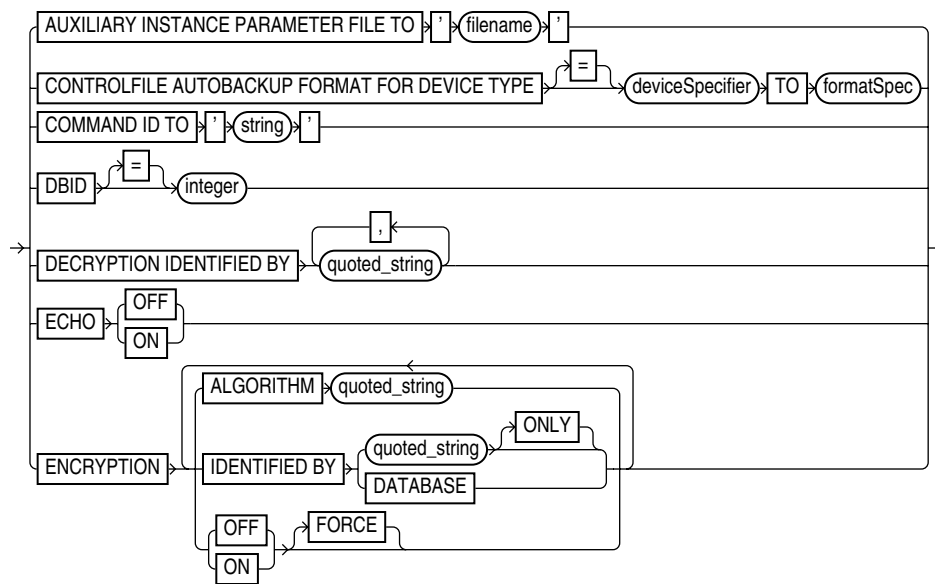
SET

Syntax

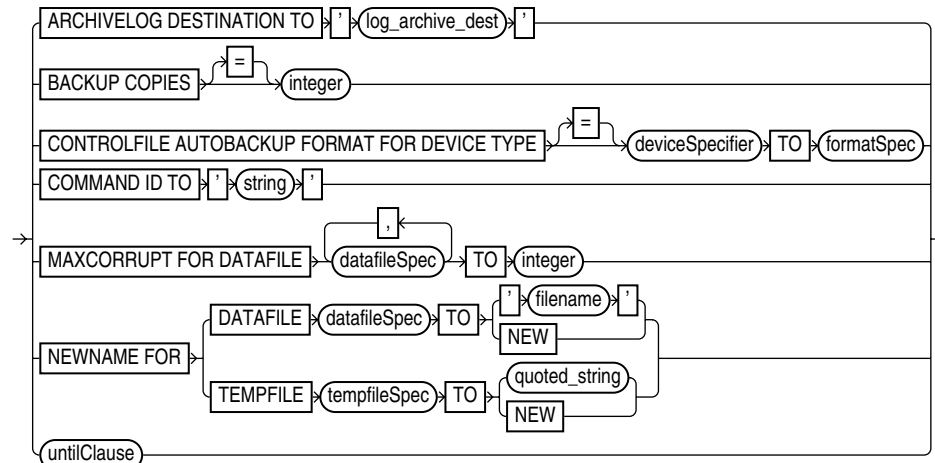
set::=



setRmanOption::=



setRunOption::=



Purpose

You can use the **SET** command either at the RMAN prompt or within a **RUN** block.

When used outside a **RUN** block, attributes changed by **SET** remain in effect until you exit the RMAN client.

Outside of a RUN block, the SET command supports the attributes shown in "setRmanOption" on page 2-200. Use this form of the SET command to:

- Cause RMAN to echo any commands executed in the message log
- Specify the location to search for control file autobackups during RESTORE operations
- Specify the database identifier (DBID) of a database, when restoring a control file or server parameter file from autobackup
- Specify a command ID for an RMAN command, so that rows in V\$SESSION corresponding to sessions related to that command have V\$SESSION.CLIENT_INFO set to this ID
- Specify settings related to encryption and decryption of backup sets for the duration of an RMAN session
- Specify the location to use for auxiliary instance datafiles, when performing TSPITR or creating transportable tablespaces sets from backup

Within a RUN block, the SET command supports the attributes shown in "setRunOption" on page 2-202. Changes made with SET within a RUN block persist until the end of the RUN block, or the next SET command that changes the value of the same attribute.

Use SET specified within a RUN block to:

- Specify new filenames for restored datafiles
- Specify the filenames for the auxiliary database during TSPITR or database duplication.
- Specify a limit for the number of permissible block corruptions.
- Override default archived redo log destinations.
- Set an end time, SCN, or log sequence number for recovery.
- Specify that backup sets should be duplexed, that is, multiple copies should be created of each backup piece in the backup set.
- Determine which server session corresponds to which channel.
- Turn RMAN's automatic location feature on or off.
- Override the default format for control file autobackups at the session level.

Restrictions and Usage Notes for SET Command Within RUN

The following restrictions apply to SET when issued within a RUN command:

- The SET BACKUP COPIES command affects all backups in the RUN block after issuing the command and is in effect until explicitly disabled or changed. The SET BACKUP COPIES command does not affect previous backups.
- SET BACKUP COPIES does not apply to the BACKUP AS COPY command.
- You must include the %F substitution variable in the autobackup format.
- You cannot use SET NEWNAME TO NEW when creating a duplicate or standby database or performing RMAN TSPITR.

Restrictions and Usage Notes for the SET DBID Command

You should only run the SET DBID command in the following specialized circumstances:

- You are not connected to a recovery catalog and want to restore the control file or server parameter file.
- You are connected to a recovery catalog and want to restore the control file, but the database name is not unique in the recovery catalog.
- The server parameter file is lost and you want to restore it.

See Also: [Table 2–27](#) for RESTORE CONTROLFILE usage notes.

Keywords and Parameters

setRmanOption

Syntax Element	Description
AUXILIARY INSTANCE PARAMETER FILE TO <i>'filename'</i>	<p>For use in customizing TSPITR with an automatic auxiliary instance, or in online tablespace cloning with RMAN. Specifies the path (on the host running the RMAN client) to the parameter file to use in starting the instance.</p> <p>Note: This is a client-side location, not a location on the database node.</p> <p>See Also: <i>Oracle Database Reference</i> for more on <code>V\$SESSION.CLIENT_INFO</code></p>
COMMAND ID TO <i>'string'</i>	<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 deviceSpecifier TO formatSpec	<p>Overrides the default filename format for the control file autobackup on the specified device type. The override occurs at the session level only. You can run 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>Note that the <code>%F</code> substitution variable is required to be in the new <code>formatSpec</code>, and that no other substitution variable is legal in a control file autobackup <code>formatSpec</code>.</p> <p>See Also: "formatSpec" on page 2-121 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. 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>See Also: "Restrictions and Usage Notes for the SET DBID Command" on page 2-199</p>

Syntax Element	Description
<pre>DECRYPTION IDENTIFIED BY password [, password...]</pre>	<p>Specifies one or more decryption passwords to be used when reading dual-mode or password-encrypted backups. When RMAN reads encrypted backup pieces, it tries each password in the list until it finds the correct one to decrypt that backup piece. An error is signaled if none of the specified keys work.</p> <p>See Also: "Encryption of Backup Sets" on page 2-28 for more details on encryption of backup sets</p>
<pre>ECHO {ON OFF}</pre>	<p>Controls whether RMAN commands are displayed in the message log. When reading commands from a command file, RMAN automatically echoes those commands to the message log. When reading commands from standard input, RMAN does not echo those 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.</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/rman@rcat < in_file > out_file</pre> <p>By running <code>SET ECHO ON</code>, you enable the commands contained in <code>in_file</code> to be visible in <code>out_file</code>.</p>
<pre>ENCRYPTION</pre>	<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 2-28 for more details on encryption of backup sets</p>
<pre>ALGORITHM algorithm_name</pre>	<p>Specifies the algorithm used during this RMAN session. Overrides the configured default encryption algorithm. Possible values are listed in <code>V\$RMAN_ENCRYPTION_ALGORITHMS</code>.</p>
<pre>IDENTIFIED BY password [ONLY]</pre>	<p>Specifies whether to use a user-specified password in encryption.</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 "Encryption of Backup Sets" on page 2-28 for details on the different encryption modes.</p> <p>Note: If the password provided is not surrounded by quotes, then it is translated internally into upper case. Thus, the following are all synonyms for <code>IDENTIFIED BY "FOO"</code>:</p> <ul style="list-style-type: none"> ■ <code>IDENTIFIED BY foo</code> ■ <code>IDENTIFIED BY Foo</code> ■ <code>IDENTIFIED BY fOo</code>
<pre>{ OFF ON } [FOR ALL TABLESPACES]</pre>	<p>If <code>SET... ENCRYPTION ON</code> or <code>OFF</code> is used without <code>FOR ALL TABLESPACES</code>, then this setting turns on or off encryption of backups for tablespaces where <code>CONFIGURE ENCRYPTION FOR TABLESPACE tablespace_name</code> <code>ON</code> or <code>OFF</code> has not been used to specify encryption behavior.</p> <p>If <code>SET... ENCRYPTION ON</code> or <code>OFF FOR ALL TABLESPACES</code> is used, this command turns on or off encryption for all tablespaces, overriding any <code>CONFIGURE ENCRYPTION FOR TABLESPACE tablespace_name</code> <code>ON</code> or <code>OFF</code> setting.</p> <p>If <code>OFF</code>, then the default is to not encrypt backup sets.</p> <p>If <code>ON</code>, then the default is to encrypt backup sets.</p>

setRunOption

Syntax Element	Description
NEWNAME FOR DATAFILE datafileSpec TO	<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 functions as a datafile copy and is recorded as such in the repository.</p> <p>Note: The SET NEWNAME command supports Automatic Storage Management disk groups.</p> <p>See Also: "datafileSpec" on page 2-95</p>
NEWNAME FOR TEMPFILE tempfileSpec TO <i>password</i> NEW	<p>Sets the new tempfile name for a subsequent SWITCH command that renames the specified tempfile to the specified name. If a filename is passed, that will be the new name of the tempfile after the database is recovered and opened. If NEW is passed the new name will be an OMF filename. The tempfile is created in DB_CREATE_FILE_DEST when the database is opened.</p> <p>Note: The SET NEWNAME command supports Automatic Storage Management disk groups.</p> <p>See Also: "tempfileSpec" on page 2-219</p>
' <i>filename</i> '	<p>Specifies a user-defined filename or Automatic Storage Management disk group for the restored datafile. If you set the NEWNAME to a disk group and run a RESTORE, then RMAN restores the file to the disk group.</p>
NEW	<p>Creates an Oracle-managed file in the directory specified in DB_CREATE_FILE_DEST. 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.</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>
MAXCORRUPT FOR DATAFILE datafileSpec TO <i>integer</i>	<p>Sets a limit on the number of previously undetected block corruptions that the database will allow in a specified datafile or group of datafiles. If a BACKUP or CREATE CATALOG command detects more than the specified number of corruptions, then the command terminates. The default limit is zero, meaning that RMAN tolerates no corrupt blocks.</p> <p>Note: If you specify CHECK LOGICAL, then the MAXCORRUPT limit applies to the sum of logical and physical corruptions detected. Otherwise, MAXCORRUPT only applies to the number of physical block corruptions.</p> <p>See Also: "datafileSpec" on page 2-95</p>
ARCHIVELOG DESTINATION TO ' <i>log_archive_dest</i> '	<p>Overrides the LOG_ARCHIVE_DEST_1 initialization parameter in the target database when forming names for restored archive logs during subsequent RESTORE and RECOVER commands. RMAN restores the logs to the destination specified in '<i>log_archive_dest</i>'. Use this parameter to restore archived redo logs that are not already on disk.</p> <p>Use this command to stage many archived logs to different locations while a database restore is occurring. RMAN knows where to find the newly restored archive logs; it does not require them to be in the destination specified by LOG_ARCHIVE_DEST_1. For example, if you specify a different destination from the one in the parameter file and restore archived redo log backups, subsequent restore and recovery operations will detect this new location. RMAN always looks for archived redo logs on disk first before restoring them from backup sets.</p>

Syntax Element	Description
<i>untilClause</i>	<p>Specifies an end time, SCN, or log sequence number for a subsequent RESTORE or RECOVER command.</p> <p>See Also: "untilClause" on page 2-227</p>
BACKUP COPIES = <i>integer</i>	<p>Specifies the number of copies of each backup piece that the channels should create: 1, 2, 3, or 4. The SET BACKUP COPIES command, which affects only the BACKUP 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> <ul style="list-style-type: none"> ■ BACKUP COPIES ■ SET BACKUP COPIES ■ CONFIGURE . . . BACKUP COPIES <p>The names of the backup pieces are dependent on the FORMAT clause in the BACKUP command. You can specify up to four FORMAT strings. RMAN uses the second, third, and fourth values only when BACKUP COPIES, SET BACKUP COPIES, or CONFIGURE . . . BACKUP COPIES is in effect. When choosing which format to use for each backup piece, RMAN uses the first format value for copy 1, the second format value for copy 2, and so on. If the number of format values exceeds the number of copies, then the extra formats are not used. If the number of format values is less than the number of copies, RMAN reuses the format values, starting with the first one.</p> <p>Note: BACKUP COPIES 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>
COMMAND ID TO ' <i>string</i> '	<p>Enters the specified string into the V\$SESSION.CLIENT_INFO column of all channels. Use this information to determine which database server sessions correspond to which RMAN channels. The SET COMMAND ID command applies only to channels that are already allocated.</p> <p>The V\$SESSION.CLIENT_INFO column contains information for each RMAN server session. The data appears in one of the following formats:</p> <ul style="list-style-type: none"> ■ id=<i>string</i> ■ id=<i>string</i>,ch=<i>channel_id</i> <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 V\$SESSION.CLIENT_INFO column will be cleared.</p> <p>See Also: <i>Oracle Database Reference</i> for more on V\$SESSION.CLIENT_INFO</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. The override occurs at the session level only. You can run this command either in RUN or at the RMAN prompt. The order of precedence is as follows:</p> <ol style="list-style-type: none"> 1. SET CONTROLFILE AUTOBACKUP executed within a RUN block 2. SET CONTROLFILE AUTOBACKUP executed at the RMAN prompt 3. CONFIGURE CONTROLFILE AUTOBACKUP FORMAT <p>Note that the %F substitution variable is required to be in the new <i>formatSpec</i>, and that no other substitution variable is legal in a control file autobackup <i>formatSpec</i>.</p> <p>See Also: "formatSpec" on page 2-121 for the semantics of the %F substitution variable.</p>

Examples

Restoring the Control File When Databases Share the Same Name: Example The following example uses the DBID to restore the control file because multiple target databases share the same DB_NAME in the catalog. After you have restored the target control file, you can mount the database to restore the rest of the database:

```
rman TARGET / CATALOG rman/rman@catdb
RMAN> STARTUP FORCE NOMOUNT;
RMAN> SET DBID = 862893450
; # needed to distinguish target from others with same DB_NAME
RMAN> RESTORE CONTROLFILE; # assuming catalog has automatic channel allocation information
RMAN> ALTER DATABASE MOUNT;
```

Setting the Command ID: Example This example sets the command ID, backs up the users tablespace, then archives the online redo logs:

```
RUN
{
  ALLOCATE CHANNEL t1 DEVICE TYPE DISK FORMAT '/disk1/%U';
  ALLOCATE CHANNEL t2 DEVICE TYPE DISK FORMAT '/disk2/%U';
  SET COMMAND ID TO 'rman';
  BACKUP INCREMENTAL LEVEL 0 MAXSETSIZE 5M TABLESPACE users;
  SQL 'ALTER SYSTEM ARCHIVE LOG ALL';
}
```

Duplexing a Backup Set: Example Assume that you have used the CONFIGURE command to set duplexing as follows:

```
CONFIGURE ARCHIVELOG COPIES FOR DEVICE TYPE sbt TO 4;
CONFIGURE DATAFILE COPIES FOR DEVICE TYPE sbt TO 3;
```

The following example overrides these configurations and makes two copies of each datafile and archived log in the backup:

```
RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE sbt;
  SET BACKUP COPIES = 2;
  BACKUP DATAFILE 1,2,3,4,5;
  BACKUP ARCHIVELOG ALL;
}
```

Overriding the Autobackup Format During a Restore: Example This example sets the DBID, sets a boundary time for the restore, then restores a control file autobackup with a nondefault format. First start RMAN and then run:

```
CONNECT TARGET / NOCATALOG
STARTUP FORCE NOMOUNT
SET DBID 676549873;
RUN
{
  SET UNTIL TIME '10/10/2001 13:45:00';
  SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '?/oradata/cf_%F.bak';
  RESTORE CONTROLFILE FROM AUTOBACKUP MAXSEQ 100;
}
ALTER DATABASE MOUNT;
```

Restoring the Server Parameter File: Example This example restores a lost server parameter file:

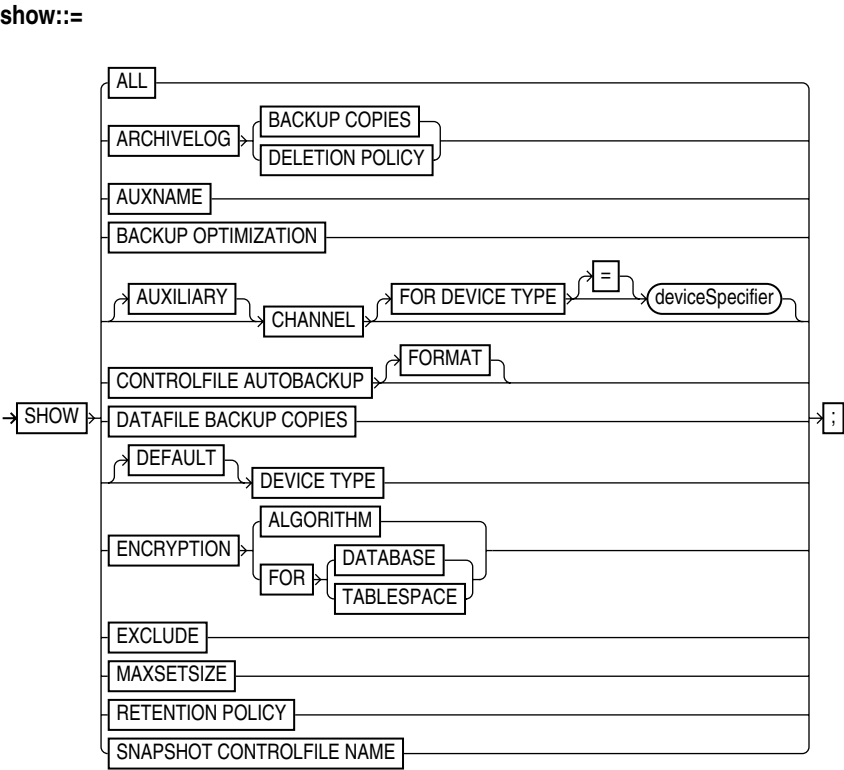
```
CONNECT TARGET / CATALOG rman/rman@catdb
SET DBID 676549873; # set dbid so rman knows the database name
```



```
STARTUP FORCE NOMOUNT # rman starts database with a dummy server parameter file
RESTORE SPFILE;
STARTUP FORCE; # needed so that RMAN restarts database with restored server parameter file
```

SHOW

Syntax



Purpose

To display the current **CONFIGURE** command settings. The output of **SHOW** consists of the **CONFIGURE** commands used to set the configuration. RMAN default configurations are suffixed with `#default`.

Restrictions and Usage Notes

- Execute this command at the RMAN prompt, not in a RUN block.
- If **SHOW ALL** is executed when connected to a target database, only node-specific configurations and database configurations are displayed. Thus, in a standby configuration, **SHOW ALL** on the primary database or the standby database shows the node-specific configuration for the primary or standby databases, for all values except retention policy, tablespace exclude and auxiliary names.

Keywords and Parameters

Syntax Element	Description
ALL	Displays all user-entered CONFIGURE commands as well as default configurations.
ARCHIVELOG BACKUP COPIES	Shows the currently configured degree of duplexing for archived redo log backups.

Syntax Element	Description
ARCHIVELOG DELETION POLICY	Shows the currently configured archived redo log deletion policy.
AUXNAME	Displays the CONFIGURE AUXNAME settings.
BACKUP OPTIMIZATION	Displays the CONFIGURE BACKUP OPTIMIZATION settings: ON or OFF (default).
[AUXILIARY] CHANNEL	Displays the CONFIGURE CHANNEL settings. You can specify a normal channel or an AUXILIARY channel.
FOR DEVICE TYPE deviceSpecifier	Specifies the device type of the channel. For example, SHOW CHANNEL FOR DEVICE TYPE DISK displays only channel settings for disk channels.
CONTROLFILE AUTOBACKUP	Displays the CONFIGURE CONTROLFILE AUTOBACKUP settings: ON or OFF.
FORMAT	Displays the format for the control file autobackup file for configured devices.
{DATAFILE ARCHIVELOG} BACKUP COPIES	Displays the CONFIGURE . . . BACKUP COPIES setting for datafiles and archived redo logs: 1, 2, 3, or 4.
[DEFAULT] DEVICE TYPE	Displays the configured device types and parallelism settings. If DEFAULT is specified, then SHOW displays the default device type and settings.
ENCRYPTION	Shows currently configured encryption settings for the database or tablespaces within the database, when used with ALGORITHM or FOR {DATABASE TABLESPACE }.
ALGORITHM	Displays the configured default algorithm to use for encryption, when writing encrypted backup sets. Possible values are listed in V\$RMAN_ENCRYPTION_ALGORITHMS.
FOR { DATABASE TABLESPACE }	Displays current encryption settings for the database or for each tablespace.
EXCLUDE	Displays only the tablespaces that you have specified should be excluded.
MAXSETSIZE	Displays the CONFIGURE MAXSETSIZE settings.
RETENTION POLICY	Displays the settings for CONFIGURE RETENTION POLICY for the current target database.
SNAPSHOT CONTROLFILE NAME	Displays the CONFIGURE SNAPSHOT CONTROLFILE settings.

Examples

Showing Channel Configurations: Example This example shows commands relevant for displaying automatic channel configurations:

```

RMAN> SHOW CHANNEL;
RMAN> SHOW DEVICE TYPE;
RMAN> SHOW DEFAULT DEVICE TYPE;
RMAN> SHOW MAXSETSIZE;

```

Showing All Configurations: Example This example shows all persistent configurations for the target database (and includes sample output):

```

RMAN> SHOW ALL;

RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE DEVICE TYPE "SBT" PARALLELISM 1;
CONFIGURE DEVICE TYPE DISK PARALLELISM 1; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DISK TO 2;

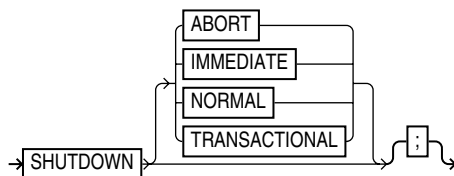
```

```
CONFIGURE DATAFILE BACKUP COPIES FOR SBT TO 1; #default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR SBT TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DISK TO 1; # default
CONFIGURE MAXSETSIZE TO 2097152K;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/oracle/dbs/cf_snap.f';
```

SHUTDOWN

Syntax

shutdown::=



Purpose

To shut down the target database without exiting RMAN. This command is equivalent to using 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

Restrictions and 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.
- The NORMAL, TRANSACTIONAL, and IMMEDIATE options all perform a clean close of the database. The ABORT option does not cleanly close the database; the database will perform instance recovery at startup.
- If the database operates in NOARCHIVELOG mode, then you must shut down the database cleanly and then issue a [STARTUP MOUNT](#) before making a backup.

Keywords and Parameters

Syntax Element	Description
ABORT	Shuts down 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. ■ Crash recovery will be performed on the database at next startup.
IMMEDIATE	Shuts down the target database immediately, 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	Shuts down the 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	Shuts down 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

Shutting Down a Database by Using the Immediate Option: Example This example waits for current SQL transactions to be processed before shutting down, then mounts the database:

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

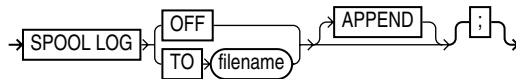
Shutting Down a Database in NOARCHIVELOG Mode: Example This example backs up a database running in NOARCHIVELOG mode:

```
STARTUP FORCE DBA;  
SHUTDOWN IMMEDIATE;  
STARTUP MOUNT;  
# executing the preceding commands ensures that database is in proper state  
# for NOARCHIVELOG backups  
BACKUP COPIES 2 DATABASE;  
ALTER DATABASE OPEN;
```

SPOOL

Syntax

spool::=



Purpose

To write RMAN output to a log file.

If the file does not already exist, then RMAN creates it. If the file does exist, then RMAN overwrites the file by default. If you specify **APPEND**, RMAN will append its output to the end of the file.

If the specified file cannot be opened for writing. Instead, RMAN turns **SPOOL** to **OFF** and continues execution.

See Also: ["cmdLine"](#) on page 2-57 for a description of LOG files

Restrictions and Usage Notes

Execute the **SPOOL** command outside of a RUN block.

Keywords and Parameters

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.
APPEND	Specifies that RMAN should append its output to the end of the existing log.

Examples

Spooling RMAN Output to a File: Example This example directs RMAN output to standard output for the backup of datafile 1, then directs output to a log file for the backup of datafile 2, then directs output to a different log file for the whole database backup:

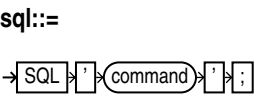
```

BACKUP DATAFILE 1;
SPOOL LOG TO '/tmp/df2log.f';
BACKUP DATAFILE 2;
SPOOL LOG OFF;
SPOOL LOG TO '/tmp/dblog.f';
BACKUP DATABASE;
SPOOL LOG OFF;

```

SQL

Syntax



Purpose

To execute a SQL statement or a PL/SQL stored procedure from within Recovery Manager.

Restrictions and Usage Notes

- If the string that RMAN passes to PL/SQL contains a filename, then the filename must be enclosed in duplicate *single* quotes and the entire string following the *SQL* keyword must be enclosed in *double* quotes. For example, use the following syntax:

```
SQL "CREATE TABLESPACE temp1 DATAFILE ' '?oradata/trgt/temp1.dbf' '
    SIZE 10M TEMPORARY";
```

If you attempt to use single quotes for the string following the SQL keyword or use only one set of single quotes for the filename, then the command fails.

- You cannot execute *SELECT* statements.

See Also: For valid SQL syntax, see the *Oracle Database SQL Reference*

Keywords and Parameters

Syntax Element	Description
' <i>command</i> '	<p>Specifies a SQL statement for execution. For example, issue the following at the RMAN prompt to archive the online redo logs:</p> <pre>SQL 'ALTER SYSTEM ARCHIVE LOG ALL';</pre> <p>Because <i>EXECUTE</i> is a SQL*Plus command, you cannot execute a PL/SQL command by specifying <i>EXECUTE</i> within the RMAN <i>SQL</i> command. Instead, you must use the <i>BEGIN</i> and <i>END</i> keywords. For example, to execute a PL/SQL procedure named <i>rman.rman_purge</i> through the RMAN <i>SQL</i> command, issue the following:</p> <pre>SQL 'BEGIN rman.rman_purge; END;';</pre>

Examples

Archiving the Unarchived Online Logs: Example This example backs up a tablespace and then archives all unarchived online logs:

```
BACKUP TABLESPACE users;
SQL "ALTER SYSTEM ARCHIVE LOG CURRENT";
```

Specifying a Filename within a Quoted String: Example This example specifies a filename by using duplicate single quotes within the context of a double-quoted string:

```
SQL "ALTER TABLESPACE tbs_1 ADD DATAFILE ' '/oracle/dbs/tbs_7.f' ' NEXT 10K MAXSIZE 100k;"
```

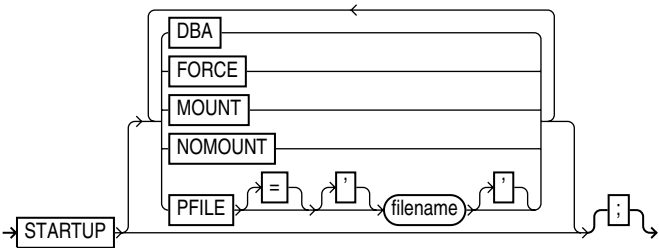
Executing a PL/SQL Stored Procedure Within RMAN: Example This example issues a PL/SQL stored procedure called `scott.update_log`:

```
RUN
{
  SQL ' BEGIN scott.update_log; END; ';
}
```

STARTUP

Syntax

startup::=



Purpose

To start the target database from within the RMAN environment. This command is equivalent to using the SQL*Plus `STARTUP` command. You can:

- Start the instance without mounting a database.
- Start the instance and mount the database, but leave it closed.
- Start the instance, and mount and open the database in:
 - unrestricted mode (accessible to all users)
 - restricted mode (accessible to DBAs only)

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

Restrictions and Usage Notes

- You cannot use the RMAN `STARTUP` command to open the recovery catalog database: execute a `STARTUP` statement in a SQL*Plus session instead.

Keywords and Parameters

If you do not specify any options, then RMAN mounts and opens the database with the default server parameter file.

Syntax Element	Description
STARTUP	If you specify only <code>STARTUP</code> with no other options, then the instance starts, then mounts and open 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 SHUTDOWN ABORT 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

Syntax Element	Description
NOMOUNT	Starts the instance without mounting the database. If no parameter file exists, then RMAN starts the instance with a "dummy" parameter file. You can then run RESTORE SPFILE to restore a backup server parameter file.
PFILE = ' <i>filename</i> '	Specifies the filename of the <code>init.ora</code> file for the target database. If this parameter is not specified, then the default <code>init.ora</code> filename is used.

Examples

Opening the Database by Using the Default Parameter File: Example This example starts and opens the database:

```
STARTUP;
```

Mounting the Database While Specifying the Parameter File: Example This example forces a SHUTDOWN ABORT and then mounts the database with restricted access, specifying a nondefault parameter file location:

```
STARTUP FORCE MOUNT DBA PFILE=/tmp/initTRGT.ora;
```

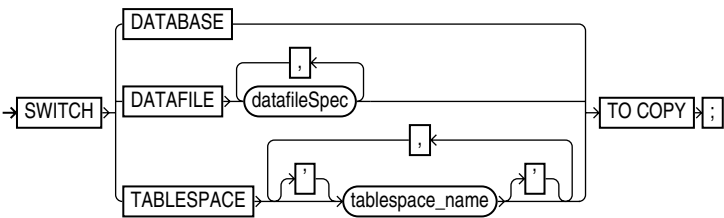
Starting an Instance Without a Parameter File: Example The following example starts an instance without using a parameter file, then runs RESTORE SPFILE:

```
SET DBID 1447326980;
STARTUP FORCE NOMOUNT; # RMAN starts instance with dummy parameter file
RESTORE SPFILE FROM AUTOBACKUP; # restore a server parameter file
STARTUP FORCE; # restart instance with restored server parameter file
```

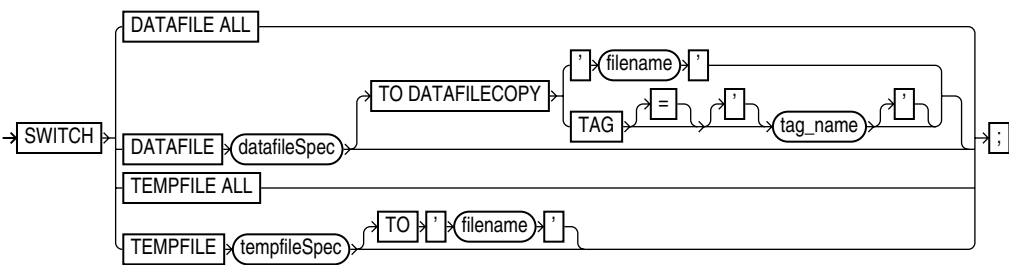
SWITCH

Syntax

switch::=



switchFile::=



Purpose

To specify that a datafile copy is now the **current datafile**, that is, the datafile pointed to by the control file. A SWITCH is equivalent to using the PL/SQL ALTER DATABASE RENAME FILE statement: the names of the files in the RMAN repository are updated, but the database does not actually rename the files at the operating system level. Note that this command deletes the RMAN repository records for the datafile copy from the recovery catalog and updates the control file records to status DELETED.

Restrictions and Usage Notes

- Execute the forms of SWITCH in the switch syntax diagram outside of a RUN block. Execute the forms of SWITCH in the switchFile syntax diagram within a RUN block.
- If RMAN is connected to a recovery catalog, and the database is using a control file restored from backup, SWITCH updates the control file with records of any datafiles known to the recovery catalog but missing from the control file.

Keywords and Parameters

switch

Syntax Element	Description
DATABASE TO COPY	<p>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 file.</p> <p>After a database switch, RMAN considers the previous database files as datafile copies.</p>

Syntax Element	Description
DATAFILE <i>datafileSpec</i> TO COPY	Specifies the datafile that you wish to rename. As with DATABASE TO COPY, specifies to switch this datafile to the latest image copy. After the switch, the control file no longer considers the specified file as current.
TABLESPACE ' <i>tablespace_name</i> ' TO COPY	Switches all datafiles within the tablespace, as with SWITCH DATAFILE . . . TO COPY.

switchFile

Syntax Element	Description
DATAFILE ALL	Specifies that all datafiles for which a SET NEWNAME FOR DATAFILE command has been issued in this job are switched to their new name.
DATAFILE <i>datafileSpec</i> TO DATAFILECOPY { ' <i>filename</i> ' TAG = ' <i>tag_name</i> ' }	Specifies the datafile that you wish to rename. After the switch, the control file no longer considers the specified file as current. For example, this command points the control file from tbs_1.f to cp1.f: SWITCH DATAFILE '?/dbs/tbs_1.f' TO DATAFILECOPY '?/dbs/copies/cp1.f'; If you do not specify a TO option, then RMAN uses the filename specified on a prior SET NEWNAME command in the RUN block for this file number as the switch target. The filename or tag provided in the TO DATAFILECOPY clause specifies the input copy file for the switch, that is, the datafile copy that you wish to rename. Specify the file by filename or tag. For example, the following command sets df2.copy as the filename for datafile 2: SWITCH DATAFILE 2 TO DATAFILECOPY '?/dbs/df2.copy'; Note that if you specify a tag and more than one copy uses this tag name, then RMAN uses the most current copy, that is, the one needing the least recovery. The following command switches datafile 3 to the most recently created datafile copy tagged mondayPMcopy: SWITCH DATAFILE 3 TO DATAFILECOPY TAG mondayPMcopy;
TEMPFILE ALL	Specifies that all tempfiles for which a SET NEWNAME FOR TEMPFILE command has been issued in this job are switched to their new name.
TEMPFILE <i>tempFileSpec</i> TO { <i>quoted_string</i> NEW	With <i>quoted_string</i> , renames the specified tempfile to the name specified by <i>quoted_string</i> . With NEW, renames the specified tempfile to an OMF filename in DB_CREATE_FILE_DEST. The tempfile is created when the database is opened, if it does not exist.

Examples

Switching Datafile Filenames After a Restore: Example This example allocates one disk device and one tape device to allow RMAN to restore from disk and tape.

```

RUN
{
  ALLOCATE CHANNEL dev1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL dev2 DEVICE TYPE sbt;
  SQL "ALTER TABLESPACE tbs_1 OFFLINE IMMEDIATE";
  SET NEWNAME FOR DATAFILE '/disk7/oracle/tbs11.f'
    TO '/disk9/oracle/tbs11.f';
  RESTORE TABLESPACE tbs_1;
  SWITCH DATAFILE ALL;
  RECOVER TABLESPACE tbs_1;
  SQL "ALTER TABLESPACE tbs_1 ONLINE";
}

```

Renaming a Tempfile Directly Using SWITCH TEMPFILE: Example This example renames a tempfile without using the SET NEWNAME command to specify the new name for the file. Note that the database must not be open for this operation.

```
RUN
{
  SWITCH TEMPFILE 1 TO '/newdisk/dbs/temp1.f';
}
```

Renaming a Tempfile Using SET NEWNAME and SWITCH TEMPFILE: Example This example demonstrates using SET NEWNAME to specify a new name for a tempfile, and SWITCH to actually rename the tempfile to the specified name. Note that the database is not open at the start of this procedure. The tempfile is re-created when the database is opened.

```
RUN
{
  SET NEWNAME FOR TEMPFILE 1 TO '/newdisk/dbs/temp1.f';
  SWITCH TEMPFILE 1;
  RESTORE DATABASE;
  RECOVER DATABASE;
  ALTER DATABASE OPEN;
}
```

Renaming Tempfiles Using SET NEWNAME and SWITCH TEMPFILE ALL: Example This example demonstrates using SET NEWNAME to specify new names for several tempfiles, and SWITCH TEMPFILE ALL to actually rename the tempfiles to the specified names. The database must be offline at the beginning of this procedure. The tempfiles are re-created at the new locations when the database is opened.

```
RUN
{
  SET NEWNAME FOR TEMPFILE 1 TO '/newdisk/dbs/temp1.f';
  SET NEWNAME FOR TEMPFILE 2 TO '/newdisk/dbs/temp2.f';
  SET NEWNAME FOR TEMPFILE 3 TO NEW; #use OMF for this one
  SWITCH TEMPFILE ALL;
  RESTORE DATABASE;
  RECOVER DATABASE;
  ALTER DATABASE OPEN;
}
```

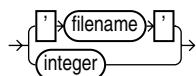
Renaming Tempfiles Using SET NEWNAME and SWITCH TEMPFILE ALL: Example This example illustrates specifying names for tempfiles when duplicating a database for standby.

```
RUN
{
  SET NEWNAME FOR TEMPFILE 1 TO '/newdisk/dbs/temp1.f';
  SET NEWNAME FOR TEMPFILE 2 TO '/newdisk/dbs/temp2.f';
  SET NEWNAME FOR TEMPFILE 3 TO '/newdisk/dbs/temp3.f';
  DUPLICATE DATABASE FOR STANDBY;
}
```

tempfileSpec

Syntax

tempFileSpec::=



Purpose

A subclause that specifies a tempfile by name or absolute file number.

Restrictions and Usage Notes

- 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 ? to represent the Oracle home and @ for the Oracle SID.

See Also: ["Placeholders"](#) on page 1-2 to learn about the difference between single and double quotes, as well as the behavior of environment variables in RMAN quoted strings

- You can use the `REPORT SCHEMA` command or the `V$TEMPFILE` control file view to get information about the current tempfiles.

Keywords and Parameters

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

Examples

Specifying a Tempfile by Filename: Example This example renames tempfile `?/oradata/trgt/tmp1.f` to `/newdisk/tmp1.f`, specifying it by filename:

```

RUN {
  SWITCH TEMPFILE ?/oradata/trgt/tmp1.f TO /newdisk/tmp1.f;
}

```

Note that the database must not be open when performing this example.

Specifying a Tempfile by Absolute File Number: Example This example renames tempfile 1 to `/newdisk/tmp1.f`, specifying it by absolute file number:

```

RUN {
  SWITCH TEMPFILE 1 TO /newdisk/tmp1.f;
}

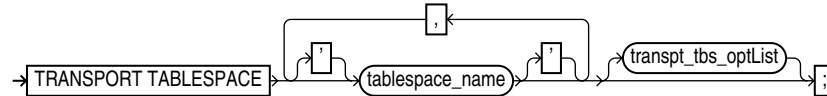
```

Note that the database must not be open when performing this example.

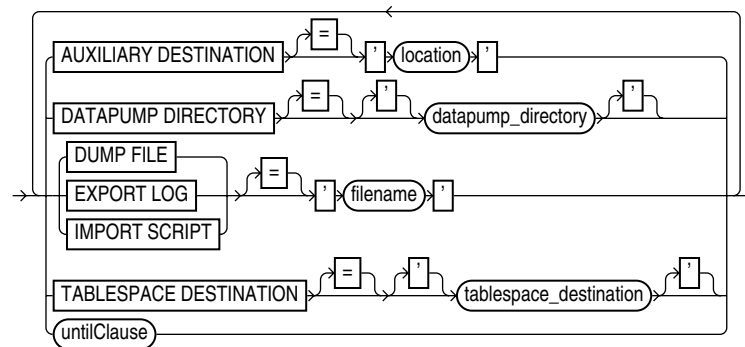
TRANSPORT TABLESPACE

Syntax

transpt_tbs::=



transpt_tbs_optlist::=



Purpose

To create transportable tablespace sets from RMAN backups, instead of the live datafiles of the source database. Also, to create transportable tablespace sets that are recovered to a point in time in the past instead of the present time.

Restrictions and Usage Notes

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 perform endian format conversion. If the target platform has a different endian format, then you must use the RMAN [CONVERT](#) command to perform the separate step of converting the endian format of the datafiles in the transportable set.

See Also: ["CONVERT"](#) on page 2-79 to learn how to convert a tablespace for transport to a target platform with a different endian format.

There are also limitations specific to creating a transportable tablespace set using RMAN:

- 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 point in time for the TRANSPORT TABLESPACE operation.

Note: If RMAN is not part of the backup strategy for your database, you can still use RMAN `TRANSPORT TABLESPACE`, as long as the needed datafile copies and archived redo logs are available on disk. Use the RMAN `CATALOG` command to record the datafile copies and archived logs in the RMAN repository. You can then use `TRANSPORT TABLESPACE`. See *Oracle Database Backup and Recovery Basics* for details on using `CATALOG`.

You also have the option of using RMAN to back up your database specifically to create backups for use in creating a transportable tablespace set from backup.

- Because the RMAN process for creating transportable tablespaces from backup uses the Data Pump Export and Import utilities, you cannot use this process if the tablespaces to be transported use XMLTypes. In such a case, you must use the process documented in *Oracle Database Administrator's Guide*.
- 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 you drop a tablespace, then you cannot later use `TRANSPORT TABLESPACE` to include that 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, you cannot use `TRANSPORT TABLESPACE` to create a transportable tablespace set as of a point in time before the tablespace was renamed. (RMAN has no information about the previous name of the tablespace.)
- You cannot `TRANSPORT` tables without their associated constraints, or constraints without their associated tables.
- Neither the transportable set nor the auxiliary set datafiles can contain any of the following:
 - Replicated master tables
 - Partial tables
 - Tables with VARRAY columns, nested tables, or external files
 - Snapshot logs and snapshot tables
 - Tablespaces containing undo or rollback segments
 - Tablespaces that contain objects owned by SYS, including rollback segments

If you are performing `TRANSPORT TABLESPACE` without a recovery catalog, the following additional restrictions apply:

- If not using a recovery catalog and transporting tablespaces as of a point in time in the past, then the set of tablespaces with undo segments at the time `TRANSPORT TABLESPACE` is executed must be the same as the set of tablespaces with undo segments at the time selected for transport.

Tablespaces including undo segments as of the target SCN for `TRANSPORT TABLESPACE` must be part of the auxiliary set. The RMAN repository in 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 target SCN, then `TRANSPORT TABLESPACE` fails.

- If the database has re-used the control file records for the RMAN repository that contained information about backups required for the TRANSPORT TABLESPACE process, then the process fails because RMAN cannot locate the required 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.

Keywords and Parameters

Syntax Element	Description
<i>tablespace_name</i>	Specifies the name of each tablespace to transport.

transpt_tbs_oplist

Syntax Element	Description
AUXILIARY DESTINATION	Optional argument that specifies the location for files for the auxiliary instance. The SET NEWNAME and CONFIGURE AUXNAME can be used to override this argument for individual files, and the DB_FILE_NAME_CONVERT and LOG_FILE_NAME_CONVERT initialization parameters can be used instead of AUXILIARY DESTINATION, if using your own initialization parameter file to customize the auxiliary instance. See <i>Oracle Database Backup and Recovery Advanced User's Guide</i> for details on the interactions among the different methods of naming the auxiliary instance files.
DATAPUMP DIRECTORY	Optional argument that specifies a database directory object where Data Pump Export outputs are created. If not specified, files will be created in the location specified by TABLESPACE DESTINATION. See <i>Oracle Database Utilities</i> for more details on Data Pump Export and database directory objects.
DUMP FILE	Optional argument that specifies where to create the 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.
EXPORT LOG	Optional argument that 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.
IMPORT SCRIPT	Optional argument that 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.
TABLESPACE DESTINATION	Argument that specifies the directory where the datafiles for the transported tablespaces are left at the end of the tablespace transport operation.
<i>untilClause</i>	Optional argument that specifies a past point in time. If specified, RMAN restores and recovers the tablespaces at the auxiliary instance to their contents at that past point in time before export.

Examples

Creating Transportable Tablespaces from Backup with TRANSPORT TABLESPACE:

Example In this example, the tablespaces for the transportable set are `tbs_2` and `tbs_3`, the transportable set files are to be stored at `/disk1/transport_dest`, and the transportable tablespaces are to be recovered to the present time:

```
transport tablespace tbs_2, tbs_3
```

```
tablespace destination '/disk1/transportdest'  
auxiliary destination '/disk1/auxdest'  
;
```

Using TRANSPORT TABLESPACE with a Past Point in Time: Example In this example, the tablespaces for the transportable set are tbs_2 and tbs_3, the transportable set files are to be stored at /disk1/transport_dest, and the transportable tablespaces are to be recovered to a specific SCN:

```
transport tablespace tbs_2, tbs_3  
  tablespace destination '/disk1/transportdest'  
  auxiliary destination '/disk1/auxdest'  
  UNTIL SCN 251982;  
;
```

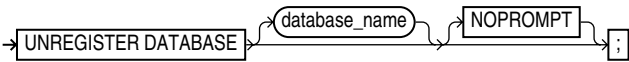
Using TRANSPORT TABLESPACE with Custom File Locations: Example This example illustrates the use of the optional arguments of TRANSPORT TABLESPACE that control the locations of the Data Pump-related files such as the dump file, as well as the DATAPUMP DIRECTORY which in this case references a directory object mypumpdir:

```
TRANSPORT TABLESPACE tbs_2  
  TABLESPACE DESTINATION '/disk1/transportdest'  
  AUXILIARY DESTINATION '/disk1/auxdest'  
  DATAPUMP DIRECTORY mypumpdir  
  DUMP FILE 'mydumpfile.dmp'  
  IMPORT SCRIPT 'myimportscript.sql'  
  EXPORT LOG 'myexportlog.log';
```

UNREGISTER DATABASE

Syntax

unregister::=



Purpose

To unregister a database from the recovery catalog.

See Also: ["DROP DATABASE"](#) on page 2-103 to learn how to delete a database and unregister it with one command.

Restrictions and Usage Notes

- Execute only at the RMAN prompt.
- RMAN must be connected to the recovery catalog in which the target database is registered.
- You can identify the database to unregister in one of three ways:
 - Connect RMAN to the target database
 - Provide the `database_name` argument to identify the database to unregister, if the database name is unique;
 - Use [SET DBID](#) to identify the database if RMAN is not connected to the target database and the `database_name` is not unique in the recovery catalog.

Keywords and Parameters

Syntax Element	Description
<code>database_name</code>	Specifies the name of the target database that you are unregistering. You do not have to specify <code>db_name</code> if RMAN is connected to the target database.
<code>NOPROMPT</code>	Specifies that RMAN should not prompt for confirmation before unregistering the database.

Example

Unregistering a Database: Example In this example, you connect to the target database `test1` and then unregister it:

```
rman TARGET SYS/oracle@test1 CATALOG rman/rman@catdb

RMAN> UNREGISTER DATABASE NOPROMPT;
```

Unregistering a Database That is Not Unique in Catalog: Example The following UNIX shell script unregisters database `testdb` from the recovery catalog. Because multiple databases called `testdb` are registered in the recovery catalog, and because RMAN is not connected to the target database (which has already been deleted from the file system), you must run `SET DBID`:

```
rman CATALOG rman/rman@catdb

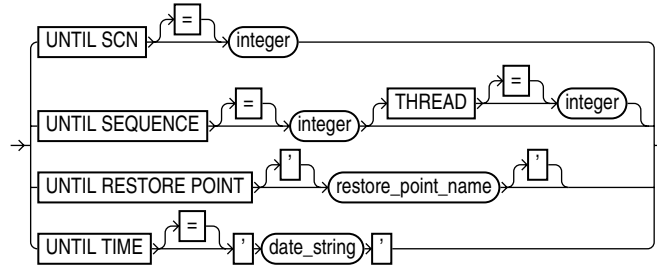
RMAN> RUN
```

```
{  
  SET DBID 1334531173;  # specifies test database by DBID  
  UNREGISTER DATABASE testdb NOPROMPT;  
}
```

untilClause

Syntax

untilClause::=



Purpose

A subclause that specifies an upper limit by time, SCN, restore point or log sequence number for various RMAN operations.

See Also: *Oracle Database Backup and Recovery Basics* to learn how to set the date format used by RMAN

Restrictions and Usage Notes

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

Following are examples of typical date settings in RMAN commands:

```

BACKUP ARCHIVELOG FROM TIME 'SYSDATE-31' UNTIL TIME 'SYSDATE-14';
RESTORE DATABASE UNTIL TIME "TO_DATE('09/20/00', 'MM/DD/YY')";

```

- SET UNTIL RESTORE POINT can only be used when the database is mounted, because the defined restore points are recorded in the control file. For example, you cannot use SET UNTIL RESTORE POINT to specify the target point in time for a RESTORE CONTROLFILE operation.

Keywords and Parameters

Syntax Element	Description
<code>UNTIL SCN = integer</code>	Specifies an SCN as an upper limit. RMAN selects only files that can be used to restore or recover up to but not including the specified SCN. For example, <code>RESTORE DATABASE UNTIL SCN 1000</code> chooses only backups that could be used to recover to SCN 1000.
<code>UNTIL RESTORE POINT restore_point_name</code>	Specifies a restore point, so that the SCN at which the restore point was created is the upper limit. RMAN selects only files that can be used to restore or recover up to but not including the corresponding SCN. Note: The database must be mounted when using <code>SET UNTIL RESTORE POINT</code> .

Syntax Element	Description
UNTIL SEQUENCE = <i>integer</i> THREAD = <i>integer</i>	Specifies a redo log sequence number and thread as an upper limit. 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 THREAD 1</code> reports only backups that could be used to recover through log sequence 7999.
UNTIL TIME = ' <i>date_string</i> '	Specifies a time as an upper limit. 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.

Examples

Performing Incomplete Recovery Until a Log Sequence Number: Example This example assumes that log sequence 1234 was lost due to a disk failure and the database needs to be recovered by using available archived redo logs.

```
RUN
{
  SET UNTIL SEQUENCE 1234 THREAD 1;
  RESTORE CONTROLFILE ;
  ALTER DATABASE MOUNT;
  RESTORE DATABASE;
  RECOVER DATABASE; # recovers through log 1233
  ALTER DATABASE OPEN RESETLOGS;
}
```

Performing Incomplete Recovery to a Specified SCN: Example This example (which assumes a mounted database) recovers the database until a specified SCN:

```
RUN
{
  ALLOCATE CHANNEL ch1 TYPE sbt;
  RESTORE DATABASE;
  RECOVER DATABASE UNTIL SCN 1000; # recovers through SCN 999
  ALTER DATABASE OPEN RESETLOGS;
}
```

Performing Incomplete Recovery to a Restore Point: Example This example (which assumes that the database starts out not mounted) recovers the database until a specified restore point:

```
STARTUP MOUNT;
RUN
{
  SET UNTIL RESTORE POINT 'before_batch';
  RESTORE DATABASE;
  RECOVER DATABASE;
  ALTER DATABASE OPEN RESETLOGS;
}
```

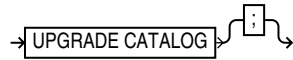
Reporting Obsolete Backups: Example This example assumes that you want to be able to recover to any point within the last week. It considers as obsolete all backups that could be used to recover the database to a point one week ago:

```
REPORT OBSOLETE UNTIL TIME 'SYSDATE-7';
```


UPGRADE CATALOG

Syntax

upgradeCatalog::=



Purpose

To upgrade the recovery catalog schema from an older version to the version required by the RMAN executable. For example, if you use a release 8.0 recovery catalog with a release 8.1 version of RMAN, then you must upgrade the catalog.

Note that UPGRADE CATALOG does not run scripts to perform the 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 forth.

Restrictions and Usage Notes

- You must be connected to the catalog database, and the catalog database must be open. You do not have to be connected to the target database.
- You must enter the UPGRADE command twice in a row to confirm the upgrade.
- You will receive an error if the recovery catalog is already at a version greater than needed by the RMAN executable. RMAN permits the command to be run if the recovery catalog is already current, however, so that the packages can be re-created if necessary. RMAN displays all error messages generated during the upgrade in the message log.

Keywords and Parameters

None.

Example

Upgrading a Recovery Catalog: Example This example connects to recovery catalog database `recdb` at the operating system command line and then upgrades it to a more current version:

```
% rman CATALOG rcat/rcat@recdb

connected to recovery catalog database
PL/SQL package rcat.DBMS_RCVCAT version 08.00.04 in RCVCAT database is too old

RMAN> UPGRADE CATALOG

recovery catalog owner is rcat
enter UPGRADE CATALOG command again to confirm catalog upgrade

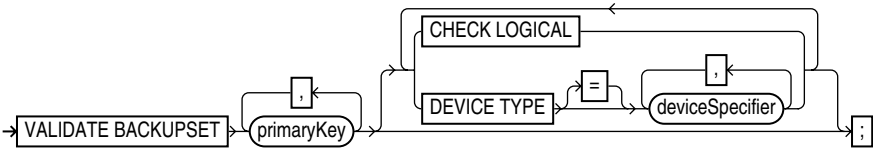
RMAN> UPGRADE CATALOG

recovery catalog upgraded to version 09.00.01
DBMS_RCVMAN package upgraded to version 09.00.01
DBMS_RCVCAT package upgraded to version 09.00.01
```

VALIDATE

Syntax

validate::=



Purpose

To examine a backup set and report whether it can be restored. RMAN scans all of the backup pieces in the specified backup sets and looks at the checksums to verify that the contents are intact so that the backup can be successfully restored if necessary.

Note: The `VALIDATE BACKUPSET` command tests whether the backup sets can be restored, whereas `CROSSCHECK` examines the headers of the specified files if they are on disk or queries the media management catalog if they are on tape.

Use this command when you suspect that one or more backup pieces in a backup set are missing or have been damaged. Use `VALIDATE BACKUPSET` to specify which backups to test; use the `VALIDATE` option of the `RESTORE` command to let RMAN choose which backups to validate. For validating image copies, run `RESTORE VALIDATE FROM DATAFILECOPY`.

Restrictions and Usage Notes

- If you do not have automatic channels configured, manually allocate at least one channel before executing a `VALIDATE BACKUPSET` statement.
- The target instance must be started.

Keywords and Parameters

Syntax Element	Description
<i>primary_key</i>	Specifies the backup sets to be validated by <i>primary_key</i> . Obtain the primary keys of backup sets by executing a LIST statement or, if you use a recovery catalog, by querying the <code>RC_BACKUP_SET</code> recovery catalog view.
CHECK LOGICAL	Tests data and index blocks in the backup set 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 <code>alert.log</code> 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> .

Syntax Element	Description
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 <code>VALIDATE . . . DEVICE TYPE DISK</code> , RMAN allocates only disk channels. See Also: " deviceSpecifier " on page 2-101

Example

Validating a Backup Set: Example This example validates the status of the backup set whose primary key is 218:

```
VALIDATE BACKUPSET 218;
# As the output indicates, RMAN determines whether it is possible to restore the
# specified backup set.

allocated channel: ORA_SBT_TAPE_1
channel ORA_SBT_TAPE_1: sid=14 devtype=SBT_TAPE
using channel ORA_DISK_1
channel ORA_SBT_TAPE_1: starting validation of datafile backupset
channel ORA_SBT_TAPE_1: restored backup piece 1
piece handle=09dg9kkl_1_1 tag=TAG20020208T125443 params=NULL
channel ORA_SBT_TAPE_1: validation complete
```


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 3-1.

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 3–1 *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

Table 3–1 (Cont.) Recovery Catalog Views

Recovery Catalog View	Corresponding V\$ View	Catalog View Describes ...
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
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 SPFILE backups for Enterprise Manager
RC_BACKUP_SPFILE_SUMMARY	V\$BACKUP_SPFILE_SUMMARY	Summary of information about SPFILE 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_RMAN_OUTPUT	V\$RMAN_OUTPUT	Output from RMAN commands for use in Enterprise Manager
RC_PROXY_ARCHIVEDLOG	V\$PROXY_ARCHIVEDLOG	Archived log backups taken with the proxy copy functionality

Table 3–1 (Cont.) Recovery Catalog Views

Recovery Catalog View	Corresponding V\$ View	Catalog View Describes ...
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
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_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_STATUS	V\$RMAN_STATUS	Historical status information about RMAN operations.
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, 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 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 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, 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.

Column	Datatype	Description
STATUS	VARCHAR2 (1)	The status of the archived redo log: A (available), U (unavailable), D (deleted), or X (expired).
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, 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.

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 backupsets, 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.
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

RC_BACKUP_ARCHIVELOG_SUMMARY

RC_BACKUP_ARCHIVELOG_SUMMARY contains summary information about backups of archived redo log files.

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.
NUM_DISTINCT_FILES_BACKED	NUMBER	The number of distinct archived redo log files backed up, where redo logs are distinguished by unique log sequence number, thread number, and resetlogs branch.
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 nM, nG, nT, nP.

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 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.
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, 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.
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 backup 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.

Column	Datatype	Description
AUTOBACKUP_DATE	DATE	The date of the control file autobackup.
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, 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.
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.
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i>

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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
OUTPUT_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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.
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 COPY 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 (10)	The KEEP options specified for this datafile copy. Possible values are LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (this backup has no KEEP options, and becomes obsolete based on the 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 nM, nG, nT, nP.

RC_BACKUP_COPY_SUMMARY

RC_BACKUP_COPY_SUMMARY contains summary information all AVAILABLE control file and datafile 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_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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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 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\$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.
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, 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 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.
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, 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.
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 DATAFILE_BLOCKS because for full backups, blocks that have never been used are not included in the backup, and for incremental backups, blocks that have not changed are not included in the backup. This value is never greater than DATAFILE_BLOCKS.
BLOCK_SIZE	NUMBER	The size of the data blocks in bytes.
STATUS	VARCHAR2 (1)	The status of the backup set: A (all pieces available), D (all pieces deleted), O (some pieces are available but others are not, so the backup set is unusable).
BS_LEVEL	NUMBER	The incremental level (NULL, 0, or 1) specified when this backup was created. This value can be different from the INCREMENTAL_LEVEL column because if you run, for example, a level 1 incremental backup, but no previous level 0 backup exists for some files, a level 0 backup is automatically taken for these files. In this case, BS_LEVEL is 1 and INCREMENTAL_LEVEL is 0.
PIECES	NUMBER	The number of backup pieces in the backup set that contains this backup datafile.

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, 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.
SESSION_KEY	NUMBER	Session identifier. Use in joins with RC_RMAN_OUTPUT.
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.
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 nM, nG, nT, nP.

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.
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
OUTPUT_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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.

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 retention policy. Values are YES or NO.
KEEP_UNTIL	DATE	Date after which this backup is considered obsolete.
KEEP_OPTIONS	VARCHAR2(13)	"Keep attributes" affecting retention for this backup. Possible values are: <ul style="list-style-type: none"> ■ LOGS (retain all redo logs needed to recover this backup as long as the backup is retained), ■ NOLOGS (redo logs may be considered obsolete even if this backup must be retained), and ■ NULL (let the configured retention policy apply to this backup).
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 row with 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. (Valid only if FILE_TYPE is PIECE or COPY.)
MEDIA	VARCHAR2(80)	Media ID for the media on which the backup is stored. Valid only if BACKUP_TYPE is "BACKUP SET" and FILE_TYPE is "PIECE".
RECID	NUMBER	Recid 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. Valid 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. Valid only if FILE_TYPE is PIECE or COPY.
COMPRESSED	VARCHAR2(3)	Whether the backup piece represented by this row is compressed. Valid only if file-type is PIECE (since image copies cannot be compressed, by definition).
OBSOLETE	VARCHAR2(3)	Whether this backup piece or copy is obsolete. Possible value: YES, NO. Valid only if FILE_TYPE is PIECE or COPY

Column	Datatype	Description
BYTES	NUMBER	Size of file described by this row. If FILE_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. Valid only if FILE_TYPE is BACKUP SET.
BS_COUNT	NUMBER	Backup set count. Valid only if FILE_TYPE is BACKUP SET.
BS_STAMP	NUMBER	Backup set timestamp. Valid only if FILE_TYPE is BACKUP SET.
BS_TYPE	VARCHAR2(32)	Type of backup set contents (datafiles or archived redo logs). Valid only if FILE_TYPE is BACKUP SET.
BS_INCR_TYPE	VARCHAR(32)	Backup set incremental type (full or not). Valid only if FILE_TYPE is BACKUP SET.
BS_PIECES	NUMBER	Number of pieces in backup set. Valid only if FILE_TYPE is BACKUP SET.
BS_COPIES	NUMBER	Number of copies of this backup set. Valid only if FILE_TYPE is BACKUP SET.
BS_COMPLETION_TIME	DATE	Completion time of the backup set. Valid only if FILE_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..) Valid only if FILE_TYPE is BACKUP SET.
BS_BYTES	NUMBER	Sum of the sizes of all backup pieces in the backup set. Valid only if FILE_TYPE is BACKUP SET.
BS_COMPRESSED	VARCHAR2(3)	Whether the backup pieces of this backup set are compressed. Valid only if FILE_TYPE is BACKUP SET.
BS_TAG	VARCHAR2(1024)	Tag or tags of the backup pieces of this backup set. If pieces have different tags, BS_TAGS will contain a comma-separated list of all tags for pieces in the backup set. Valid only if FILE_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, this field will contain a comma-separated 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". Valid only if FILE_TYPE is BACKUP SET.
BP_PIECE#	NUMBER	Number of backup pieces that make up this backup set. Valid only if FILE_TYPE is BACKUP SET.
BP_COPY#	NUMBER	Number of copies of this backup set. Valid only if FILE_TYPE is BACKUP SET.
DF_FILE#	NUMBER	File number of the datafile described by this row. Valid only if FILE_TYPE is DATAFILE.
DF_TABLESPACE	VARCHAR2(30)	Tablespace name for the datafile described by this row. Valid only if FILE_TYPE is DATAFILE.
DF_RESETLOGS_CHANGE#	NUMBER	Resetlogs change of the datafile described by this row. Valid only if FILE_TYPE is "DATAFILE".
DF_CREATION_CHANGE#	NUMBER	Creation change number of the datafile described by this row. Valid only if FILE_TYPE is "DATAFILE".
DF_CHECKPOINT_CHANGE#	NUMBER	Checkpoint change number of the datafile described by this row. Valid 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".

Column	Datatype	Description
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 the target database. 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, 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.
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 in Oracle® Database 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.

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.
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.
DBID	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, 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.
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. See also: V\$BACKUP_DEVICE
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 associated with this backup piece.
START_TIME	DATE	The time when RMAN started to write the backup piece.

Column	Datatype	Description
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.
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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 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.
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, 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.
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.

Column	Datatype	Description
PIECES	NUMBER	The number of pieces in the backup set.

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 the target database. 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, this value appears in the KEY column of the output.
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.
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_OPTIONS	VARCHAR2 (10)	The KEEP options specified for this backup set. Options can be LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (the backup has no KEEP options and is subject to the default retention policy).

Column	Datatype	Description
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.

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.
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.
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	NUMBER	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.
KEEP_OPTIONS	VARCHAR2 (10)	The KEEP options specified for this backup set. Options can be LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (the backup has no KEEP options and is subject to the default retention policy).

Column	Datatype	Description
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), this column contains "*". For backups on disk, DISK. Otherwise...
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. Otherwise...
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
OUTPUT_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
ORIGINAL_INPRATE_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as ORIGINAL_INPRATE_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
OUTPUT_RATE_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_RATE_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
TIME_TAKEN_DISPLAY	VARCHAR2 (4000)	Same value as ELAPSED_SECONDS, but converted to a user-displayable format in hours, minutes and seconds.

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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
ORIGINAL_INPRATE_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as ORIGINAL_INPRATE_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
OUTPUT_RATE_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_RATE_BYTES, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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 <code>LIST</code> command while connected to the recovery catalog, this value appears in the <code>KEY</code> column of the output.
RECID	NUMBER	The record identifier propagated from <code>V\$BACKUP_SPFILE</code> . <code>RECID</code> and <code>STAMP</code> form a concatenated primary key that uniquely identifies this record in the target database control file.
STAMP	NUMBER	The stamp from <code>V\$BACKUP_SPFILE</code> . <code>RECID</code> and <code>STAMP</code> 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 <code>RC_BACKUP_SET</code> .
SET_STAMP	NUMBER	The <code>SET_STAMP</code> value from <code>V\$BACKUP_SET</code> . <code>SET_STAMP</code> and <code>SET_COUNT</code> 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 <code>SET_COUNT</code> value from <code>V\$BACKUP_SET</code> . <code>SET_STAMP</code> and <code>SET_COUNT</code> 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 <code>RECID</code> value from <code>V\$BACKUP_SET</code> .
BS_STAMP	NUMBER	The <code>STAMP</code> value from <code>V\$BACKUP_SET</code> . Note that <code>BS_STAMP</code> is different from <code>SET_STAMP</code> . <code>BS_STAMP</code> is the stamp of the backup set record when created in the control file, whereas <code>SET_STAMP</code> joins with <code>SET_COUNT</code> 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.

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.
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.
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

RC_BACKUP_SPFILE_SUMMARY

RC_BACKUP_SPFILE_SUMMARY provides summary information about SPFILE 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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

RC_CHECKPOINT

This view is deprecated. See [RC_RESYNC](#) on page 3-58 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, 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.
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 (10)	The KEEP options specified for this control file copy. Options can be LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (the backup has no KEEP options and is subject to the default retention policy).
IS_RECOVERY_DEST_FILE	VARCHAR2 (3)	This copy is located in the flash recovery area: YES or NO.

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, 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 most recent RESETLOGS operation when the record was created.
RESETLOGS_TIME	DATE	The time stamp of the most recent RESETLOGS operation when the record was created.

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)	<p>The type of block corruption in the datafile. Possible values are:</p> <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.
DB_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 (8)	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.

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.

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, 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.
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).
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.

Column	Datatype	Description
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 (10)	The KEEP options specified for this datafile copy. Options can be LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (the backup has no KEEP options and will be made 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.

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_RMAN_OUTPUT

RC_RMAN_OUTPUT corresponds to the control file view V\$RMAN_OUTPUT.

This view is primarily intended to be used internally 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.
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 (129)	RMAN output text.

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, 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 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.
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.

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 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.
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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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 <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .

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, 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.
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 (10)	The KEEP options specified for this control file backup. Options can be LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (the backup has no KEEP options and will be made 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).

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.
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 (10)	The KEEP options specified for this backup. Options can be LOGS (RMAN keeps the logs needed to recover this backup), NOLOGS (RMAN does not keep the logs needed to recover this backup), or NULL (the backup has no KEEP options and will be made 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 nM, nG, nT, nP.

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 nM, nG, nT, nP.

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, 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.
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.
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.

Column	Datatype	Description
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 (10)	The KEEP options specified for this backup. Options can be LOGS (RMAN keeps logs needed to recover this backup), NOLOGS (RMAN does not keep logs needed to recover this backup), or NULL (the backup has no KEEP options and will be obsolete based on retention policy).

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 combination 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.

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_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 individual RMAN backup sessions, combining all operations within the session. Details for individual types of operation performed during the 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	Session identifier. Use in joins with RC_RMAN_OUTPUT.
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 job
END_TIME	DATE	End time of the last backup command 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)	'DISK', 'SBT_TAPE' or '*'. A '*' indicates a backup job that wrote its output to more than one device type.
AUTOBACKUP_COUNT	NUMBER	Number of autobackups performed by this job.
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'
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 nM, nG, nT, nP.
OUTPUT_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example nM, nG, nT, nP.

Column	Datatype	Description
INPUT_BYTES_PER_SEC_DISPLAY	VARCHAR2 (4000)	Same value as INPUT_BYTES_PER_SEC, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
OUTPUT_BYTES_PER_SEC_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_BYTES_PER_SEC, but converted to a user-displayable format, for example <i>nM</i> , <i>nG</i> , <i>nT</i> , <i>nP</i> .
TIME_TAKEN_DISPLAY	VARCHAR2 (4000)	Total time for job, converted to a user-displayable format, such as <i>hhh:mmm:sss</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.
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
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.
OBJECT_TYPE_WEIGHT	NUMBER	Used internally by Enterprise Manager.
OPTIMIZED_WEIGHT	NUMBER	Used internally by Enterprise Manager.
OUTPUT_DEVICE_TYPE	VARCHAR2 (17)	'DISK', 'SBT_TAPE' or '*'. A '*' indicates a backup job that wrote its output to more than one device type.
AUTOBACKUP_DONE	VARCHAR2 (3)	'YES' or 'NO', depending upon whether a control file autobackup was done as part of this job.
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 OPERATION=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 nM, nG, nT, nP.
OUTPUT_BYTES_DISPLAY	VARCHAR2 (4000)	Same value as OUTPUT_BYTES, but converted to a user-displayable format, for example nM, nG, nT, nP.

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.

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
SID	NUMBER	The session ID of the session that executed this RMAN operation.
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.)
PARENT_RECID	VARCHAR2 (40)	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 parent row of this row.
PARENT_STAMP	VARCHAR2 (40)	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 parent row of this row.
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.
ROW_LEVEL	NUMBER	<p>This is the level of this row.</p> <ul style="list-style-type: none"> ■ If ROW_LEVEL=0 then this is a session row (that is, ROW_TYPE=SESSION and the row represents an invocation of the RMAN client). ■ If ROW_LEVEL=1 then this row represents a command entered in the RMAN client and executed. (ROW_TYPE=COMMAND for rows at level 1.) ■ IF ROW_LEVEL>1 then this row represents a recursive operation, a sub-operation 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.)
ROW_TYPE	VARCHAR2 (19)	<p>This is the type of operation represented by this row. Possible values are:</p> <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
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.
OPERATION	VARCHAR2 (33)	This is the name of the operation presented by this row. For SESSION operations, this column is set to 'RMAN'. For COMMAND operations, it describes the command executed, such as "BACKUP", "RESTORE", "CONFIGURE", "REPORT" and so on.

Column	Datatype	Description
STATUS	VARCHAR2 (23)	The status of the operation described by this row. Possible values are: COMPLETED, COMPLETED WITH WARNINGS, COMPLETED WITH ERRORS, and FAILED.
MBYTES_PROCESSED	NUMBER	If the operation represented by this row performed some data transfer (such as backing up or restoring data) then this column contains the number of megabytes processed in the operation. Otherwise, this row contains NULL.
START_TIME	DATE	The start time for the operation represented by this row.
END_TIME	DATE	The end time for the operation represented by this row.
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.
INPUT_MBYTES	NUMBER	Sum of sizes of all input files backed up by this job, in megabytes.
OUTPUT_MBYTES	NUMBER	Sum of sizes of all output files produced by this job, in megabytes.
OPTIMIZED_ID	NUMBER	Used internally by Enterprise Manager.
OBJECT_TYPE_ID	NUMBER	Used internally by Enterprise Manager.
OUTPUT_DEVICE_TYPE_ID	NUMBER	Used internally by Enterprise Manager.
OPTIMIZED	VARCHAR2 (3)	YES if backup optimization was applied during the backup job. Otherwise, NO.
OBJECT_TYPE	VARCHAR2 (50)	Contains one of the following values: DATABASE FULL, RECOVERY AREA, DATABASE INCR, DATAFILE FULL, DATAFILE INCR, ARCHIVELOG, CONTROLFILE, SPFILE.
OUTPUT_DEVICE_TYPE	VARCHAR2 (4)	'DISK', 'SBT_TAPE' or '*'. A '*' indicates that output was written to more than one device type.
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.

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.

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.

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.
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.

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.
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 Oracle8i, and the CONFIGURE AUXNAME command replaced the SET AUXNAME command in Oracle9i, but 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
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	CONFIGURE CHANNEL ... PARMS="BLKSIZE"	n/a
10.0.1	COPY	BACKUP AS COPY
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	Now 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
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

Table A–1 (Cont.) Deprecated RMAN Syntax

Deprecated in Release	Deprecated Syntax	Preferred Current Syntax
9.0.1	SET LIMIT CHANNEL . . .	ALLOCATE CHANNEL . . . CONFIGURE CHANNEL . . .
9.0.1	SET SNAPSHOT	CONFIGURE SNAPSHOT
9.0.1	UNTIL LOGSEQ (see " untilClause ")	UNTIL SEQUENCE (see " untilClause ")
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 " cmdLine ")	AUXILIARY (see " cmdLine ")
8.1.5	CONFIGURE CLONE	CONFIGURE AUXILIARY
8.1.5	MSGLOG (see " cmdLine ")	LOG (see " cmdLine ")
8.1.5	RCVCAT (see " cmdLine ")	CATALOG (see " cmdLine ")

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](#)
- [RMAN Compatibility Matrix](#)
- [RMAN Compatibility: Scenario](#)

About RMAN Compatibility

The following table describes the components of an RMAN environment. Each component has a release number associated with it.

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 duplicate database
- A release 8.1.7 recovery catalog database whose catalog tables were created with RMAN release 9.0.1

Determination of 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 `rcver` catalog table. For example, run this query:

```
SQL> SELECT * FROM rcver;
```

```
VERSION
-----
08.01.05.00
09.00.01.00
10.02.01.00
```

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 catalog schema version is 09.00.01.00 and the previous version was 08.01.05.00.

Note that for releases 10.2 and later, the last two digits 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 8.X or 9.X RMAN catalog schema in any Oracle database release 8.1.X (or higher) and Release 10g RMAN 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.
- 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-1](#)). The RMAN client cannot be of a greater version than the target or auxiliary database.
- While backing up a Release 10g database using the 9.X RMAN client, you cannot include a control file that was created using compatible=10.0.0 in a datafile backupset. The workaround is to turn control file autobackup ON.

[Table B-1](#) shows version requirements for RMAN components. Note the following conventions when interpreting this table:

- ">=8.1.7" means 8.1.7, 9.0.1, 9.2.0 and 10.1.0 and their patches
- ">=8.0.6" means 8.0.6, 8.1.7, 9.0.1, 9.2.0 and 10.1.0 and their patches

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.

Table B-1 RMAN Compatibility Table

Target/Auxiliary Database	RMAN client	Catalog Database	Catalog Schema
8.0.6	8.0.6	>=8.1.7	>=8.0.6
8.1.7	8.0.6.1	>=8.1.7	>=8.1.7
8.1.7	8.1.7	>=8.1.7	>=RMAN client
8.1.7.4	8.1.7.4	>=8.1.7	8.1.7.4
8.1.7.4	8.1.7.4	>=8.1.7	>= 9.0.1.4
9.0.1	9.0.1	>=8.1.7	>= RMAN client

Table B–1 (Cont.) RMAN Compatibility Table

Target/Auxiliary Database	RMAN client	Catalog Database	Catalog Schema
9.2.0	>=9.0.1.3 and <= target database executable	>=8.1.7	>= RMAN client
10.1.0	>=9.0.1.3 and <=target database executable	>=9.0.1	>= RMAN client
10.2.0	>=9.0.1.3 and <=target database executable	>=9.0.1	>= RMAN client

Cross-Version Compatibility of Recovery Catalog Exports and Imports

Exports of the recovery catalog are often used as a way to backup its contents. When planning to use Oracle export utilities to back up the recovery catalog, refer to *Oracle Database Utilities* for details on compatibility issues surrounding the use of database exports across versions of Oracle. Exports from a later version of Oracle cannot be imported into databases running under earlier versions of Oracle. You must export your recovery catalog data using the export utility from the earliest version of Oracle 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:

- 8.1.7
- 9.0.1
- 9.2.0
- 10.1.0
- 10.2.0

You want to record RMAN repository data about these databases in a single recovery catalog database. According to [Table B–1](#) on page B-2, you can use a single 9.2.0 recovery catalog database with a 10.2.0 catalog schema for all target databases.

The solution for this combination of target databases is to do the following:

- Use a single 9.2.0 catalog database.
- Use a single 10.2.0 catalog schema for all databases.
- Ensure that the version of the RMAN client used to back up each target database meets the requirements specified in [Table B–1](#) on page B-2.

Index

Symbols

? symbol
 in RMAN quoted strings, 1-3
@ command, 2-5
 in RMAN quoted strings, 1-3
@@ command, 2-6

A

ALLOCATE CHANNEL command, 2-7, 2-13
 and shared server, 2-7, 2-10
 FOR MAINTENANCE option, 2-10
ALTER DATABASE command, 2-16
archivelogRecordSpecifier clause, 2-18
autobackups
 control file, 2-70

B

BACKUP
 INCREMENTAL
 FROM SCN, 2-27
BACKUP command, 2-22
backup sets
 and Backup Unused Space Compression, 2-28
 and unused block compression, 2-28
 binary compression, 2-28
Backup Unused Space Compression, 2-28
backups
 binary compression, 2-28
 datafiles
 Backup Unused Space compression, 2-28
 unused block compression, 2-28
backups,vaulted, 2-183
binary compression of backup sets, 2-28
BLOCKRECOVER command, 2-48

C

CATALOG command, 2-51
CHANGE command, 2-54
channels
 allocating to shared server sessions, 2-7, 2-10
code examples
 description of RMAN, 1-5
command line

 arguments for RMAN, 2-57
commands, Recovery Manager
 @, 2-5
 @@, 2-6
 ALLOCATE CHANNEL, 2-7, 2-13
 ALLOCATE CHANNEL FOR
 MAINTENANCE, 2-10
 ALTER DATABASE, 2-16
 archivelogRecordSpecifier clause, 2-18
 BACKUP, 2-22
 BLOCKRECOVER, 2-48
 CATALOG, 2-51
 CHANGE, 2-54
 completedTimeSpec clause, 2-61
 CONFIGURE, 2-63
 CONNECT, 2-75
 connectStringSpec clause, 2-77
 CREATE CATALOG, 2-89
 CREATE SCRIPT, 2-91
 CROSSCHECK, 2-93
 DELETE, 2-96
 DELETE SCRIPT, 2-100
 deprecated, A-1
 DROP CATALOG, 2-102
 DROP DATABASE, 2-103
 DUPLICATE, 2-104
 EXECUTE SCRIPT, 2-112
 EXIT, 2-113
 fileNameConversionSpec clause, 2-114
 FLASHBACK, 2-116
 HOST, 2-124
 LIST, 2-127
 listObjList clause, 2-143
 PRINT SCRIPT, 2-150
 QUIT, 2-151
 recordSpec, 2-152
 RECOVER, 2-154
 REGISTER, 2-163
 RELEASE CHANNEL, 2-165, 2-166
 REPLACE SCRIPT, 2-167
 REPORT, 2-169
 RESET DATABASE, 2-175
 RESTORE, 2-178
 RESYNC CATALOG, 2-192
 RUN, 2-194
 SEND, 2-197

SET, 2-198
SHOW, 2-206
SHUTDOWN, 2-209
SPOOL, 2-211
SQL, 2-212
STARTUP, 2-214
summary, 2-1, 3-1
SWITCH, 2-216
TRANSPORT TABLESPACE, 2-221
UNREGISTER DATABASE, 2-225
untilClause, 2-227
UPGRADE CATALOG, 2-229
VALIDATE, 2-230
comments in RMAN commands, 1-1
compatibility
 recovery catalog, B-1
 Recovery Manager, B-1
completedTimeSpec clause, 2-61
CONFIGURE command, 2-63
CONNECT command, 2-75
connectStringSpec clause, 2-77
control files
 automatic backups, 2-70
corrupt datafile blocks
 maximum acceptable number, 2-202
corruption detection
 using SET MAXCORRUPT command, 2-202
CREATE CATALOG command, 2-89
CREATE SCRIPT command, 2-91
CROSSCHECK command, 2-93

D

datafiles
 backing up
 Backup Unused Space Compression, 2-28
 unused block compression, 2-28
dates
 specifying in RMAN commands, 2-227
DELETE command, 2-96
DELETE SCRIPT command, 2-100
deprecated commands
 Recovery Manager, A-1
DROP CATALOG command, 2-102
DROP DATABASE command, 2-103
DUPLICATE command, 2-104
duplicate database
 synchronizing
 using incremental backups, 2-27

E

environment variables
 in RMAN strings, 1-3
EXECUTE SCRIPT command, 2-112
EXIT command, 2-113

F

fileNameConversionSpec clause, 2-114
FLASHBACK command, 2-116

H

HOST command, 2-124

I

Incremental Roll Forward of Database Copy, 2-27

K

keywords
 in syntax diagrams, 1-2

L

LIST command, 2-127
listObjList clause, 2-143

M

MAXCORRUPT parameter
 SET command, 2-202
MAXSETSIZE parameter
 BACKUP command, 2-35

P

parameters
 in syntax diagrams, 1-2
PL/SQL stored procedures
 executing within RMAN, 2-213
PRINT SCRIPT command, 2-150

Q

QUIT command, 2-151
quoted strings
 environment variables, 1-3

R

RC_ARCHIVED_LOG view, 2-18, 3-4
RC_BACKUP_CONTROLFILE view, 3-8
RC_BACKUP_CORRUPTION view, 3-14
RC_BACKUP_DATAFILE view, 3-15
RC_BACKUP_PIECE view, 3-22
RC_BACKUP_REDOLOG view, 3-25
RC_BACKUP_SET view, 3-27
RC_BACKUP_SPFILE view, 3-32
RC_CHECKPOINT view, 3-35
RC_CONTROLFILE_COPY view, 3-36
RC_COPY_CORRUPTION view, 3-37
RC_DATABASE recovery catalog view, 3-38
RC_DATABASE view, 2-200
RC_DATABASE_BLOCK_CORRUPTION
 view, 3-39
RC_DATABASE_INCARNATION view, 2-200, 3-40
RC_DATAFILE view, 3-41
RC_DATAFILE_COPY view, 3-42
RC_LOG_HISTORY view, 3-44
RC_OFFLINE_RANGE view, 3-45
RC_PROXY_CONTROLFILE view, 3-50

- RC_PROXY_DATAFILE view, 3-54
- RC_REDO_LOG view, 3-56, 3-64
- RC_REDO_THREAD view, 3-57
- RC_RESYNC view, 3-58
- RC_RMAN_CONFIGURATION view, 3-63
- RC_STORED_SCRIPT view, 3-66
- RC_STORED_SCRIPT_LINE view, 3-67
- RC_TABLESPACE view, 3-68, 3-69
- RECALL, and RESTORE... PREVIEW, 2-183
- recordSpec clause, 2-152
- RECOVER command, 2-154
- recovery catalog
 - views, 3-1
- Recovery Manager
 - backups
 - control file autobackups, 2-70
 - commands
 - @, 2-5
 - @@, 2-6
 - ALLOCATE CHANNEL, 2-7, 2-13
 - ALLOCATE CHANNEL FOR MAINTENANCE, 2-10
 - ALTER DATABASE, 2-16
 - archiveLogRecordSpecifier clause, 2-18
 - BACKUP, 2-22
 - BLOCKRECOVER, 2-48
 - CATALOG, 2-51
 - CHANGE, 2-54
 - completedTimeSpec, 2-61
 - CONFIGURE, 2-63
 - CONNECT, 2-75
 - connectStringSpec, 2-77
 - CREATE CATALOG, 2-89
 - CREATE SCRIPT, 2-91
 - CROSSCHECK, 2-93
 - DELETE, 2-96
 - DELETE SCRIPT, 2-100
 - DROP CATALOG, 2-102
 - DROP DATABASE, 2-103
 - DUPLICATE, 2-104
 - EXECUTE SCRIPT, 2-112
 - EXIT, 2-113
 - fileNameConversionSpec, 2-114
 - FLASHBACK, 2-116
 - HOST, 2-124
 - LIST, 2-127
 - listObjList, 2-143
 - PRINT SCRIPT, 2-150
 - QUIT, 2-151
 - recordSpec, 2-152
 - RECOVER, 2-154
 - REGISTER, 2-163
 - RELEASE CHANNEL, 2-165
 - REPLACE SCRIPT, 2-167
 - REPORT, 2-169
 - RESET DATABASE, 2-175
 - RESTORE, 2-178
 - RESYNC, 2-192
 - RUN, 2-194
 - SEND, 2-197

- SET, 2-198
- SHOW, 2-206
- SHUTDOWN, 2-209
- SPOOL, 2-211
- SQL, 2-212
- STARTUP, 2-214
- SWITCH, 2-216
- TRANSPORT TABLESPACE, 2-221
- UNREGISTER DATABASE, 2-225
- untilClause, 2-227
- UPGRADE CATALOG, 2-229
- VALIDATE, 2-230
- compatibility, B-1
- dates in commands, 2-227
- symbolic links for filenames, 2-181
- syntax conventions, 1-1
- REGISTER command, 2-163
- RELEASE CHANNEL command (RMAN), 2-165
 - releasing a maintenance channel, 2-166
- REPLACE SCRIPT command, 2-167
- REPORT command, 2-169
- RESET DATABASE command, 2-175
- RESTORE command, 2-178
- RESTORE... PREVIEW
 - RECALL, 2-183
- RESYNC CATALOG command, 2-192
- RUN command, 2-194

S

- SEND command, 2-197
- SET command, 2-198
- shared server
 - allocating channels, 2-7, 2-10
- SHOW command, 2-206
- SHUTDOWN command, 2-209
- SPOOL command, 2-211
- SQL command, 2-212
- STARTUP command, 2-214
- stored procedures
 - executing within RMAN, 2-213
- SWITCH command, 2-216
- symbolic links
 - and RMAN, 2-181
- synchronizing duplicate database
 - using incremental backups, 2-27
- syntax conventions
 - Recovery Manager, 1-1
- syntax diagrams
 - explanation of, 1-1
 - keywords, 1-2
 - parameters, 1-2

T

- TRANSPORT TABLESPACE command, 2-221

U

- UNREGISTER DATABASE command, 2-225
- untilClause, 2-227

unused block compression, 2-28
UPGRADE CATALOG command, 2-229

V

V\$ARCHIVED_LOG view, 2-18, 2-20, 2-138
V\$BACKUP_CORRUPTION view, 2-32
V\$BACKUP_DEVICE view, 2-8, 2-11
V\$BACKUP_PIECE view, 2-131, 2-133, 2-135
V\$BACKUP_SET view, 2-131, 2-134, 2-135, 2-136
V\$DATABASE view, 2-200
V\$DATABASE_BLOCK_CORRUPTION view, 2-48,
2-49, 2-50, 2-159, 2-184, 2-230
V\$DATAFILE view, 2-95, 2-219
V\$DATAFILE_COPY view, 2-95, 2-137, 2-138
V\$DATAFILE_HEADER view, 2-95
V\$PROXY_DATAFILE view, 2-133
V\$SESSION view, 2-200, 2-203
VALIDATE command, 2-230
vaulted backups, 2-183
views
 recovery catalog, 3-1