Oracle9*i*

Recovery Manager Quick Reference

Release 2 (9.2)

March 2002

Part No. A96564-01

About This Quick Reference

This Quick Reference shows the basic uses and syntax for Recovery Manager (RMAN). For detailed concepts and procedures, refer to the *Oracle9i Recovery Manager User's Guide*. For detailed syntax information, refer to the *Oracle9i Recovery Manager Reference*.

This Quick Reference discusses the following topics:

Conventions for Command Syntax

The following table shows the syntax convention used in this manual.

Convention	Meaning
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.
|	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.
`...`	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.
`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.
`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.
`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.

Overview of the RMAN Environment

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

RMAN uses database server sessions to perform backup and recovery. It stores metadata about its operations in the control file of the target database and, optionally, in a recovery catalog schema in an Oracle database.

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

- RMAN executable
- Target database
- Recovery catalog database (optional)
- Media management software (optional)

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

About the RMAN Executable

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

About the Target Database

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

About the RMAN Repository

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

- Backup sets and pieces
- Image copies
- Proxy copies
- Archived redo logs
- The target database schema
- Persistent configuration settings

You can access this metadata by issuing `LIST`, `REPORT`, and `SHOW` commands in the RMAN interface, or by using `SELECT` statements on the catalog views (only if you use a recovery catalog). You can either create a recovery catalog in which to store the repository, or let RMAN store the repository exclusively in the target database control file.

About the RMAN Media Management Interface

To store backups on tape, RMAN requires a third-party media manager. A media manager is a software program that loads, labels, and unloads sequential media such as tape drives used to back up and recover data. If you choose not to install a media manager, then you can back up to disk.

Starting and Exiting RMAN

RMAN is a client executable that is installed with the Oracle database server. You can connect it to a **target database**, and then use server sessions on the target database to back up, restore, and recover database files.

You have the following options for starting RMAN:

- Specify the database connection information at the command line:

  ```
  % rman TARGET SYS/pwd@target_str # connects in NOCATALOG mode
  % rman TARGET / CATALOG cat_usr/pwd@cat_str
  % rman TARGET / CATALOG cat_usr/pwd@cat_str AUXILIARY SYS/pwd@aux_str
  ```

- Omit the database connection at the command line, and use the `CONNECT` command in your RMAN scripts:

  ```
  % rman
  ```

In both cases, RMAN displays an `RMAN>` prompt at which you can enter commands or run a command file. For example:

```
RMAN> @some_command_file.rcv  # runs specified command file
```

RMAN can connect to the following types of databases.

Database	Explanation
Target database	The database that you will use RMAN to back up and restore. RMAN connects to the target database as `SYSDBA`. If you do not have this privilege, then the connection fails. You can connect with the `SYSDBA` privilege by using a password file or using operating system authentication.
Recovery catalog database	This database is *optional*: you can use RMAN with the default `NOCATALOG` option, in which case RMAN uses the control file as the exclusive repository of metadata. A recovery catalog is a set of tables containing RMAN metadata. RMAN obtains the metadata from the target database control file. If you choose to use a recovery catalog, refer to *Oracle9i Recovery Manager User's Guide* for complete instructions.
Auxiliary database	The auxiliary instance is only used for duplicate and standby database created with the `DUPLICATE` command and for tablespace point-in-time recovery.

The following syntax shows the most common RMAN command-line options:

```
RMAN
[ TARGET [=] connectStringSpec
| { CATALOG [=] connectStringSpec | NOCATALOG }
| AUXILIARY [=] connectStringSpec
| LOG [=] ['] filename [']
.
.
.
]...

connectStringSpec::=
['] [userid] [/ [password]] [@net_service_name] [']
```

The following examples show various command-line options:

```
% rman TARGET SYS/oracle@prod1 @/scripts/b_whole.rcv # runs command file
% rman TARGET / LOG $ORACLE_HOME/dbs/log/msglog.f APPEND
```

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

```
RMAN> EXIT
```

Configuring the RMAN Environment

You can configure persistent settings in the RMAN environment. The configuration setting is done once and is used by RMAN to perform all subsequent operations. Display the preconfigured settings as follows:

```
SHOW ALL;
```

Configuring Disk Devices and Channels By default, RMAN sends all backups to an operating system specific directory on disk. To configure RMAN to make backups to other media, see *Oracle9i Recovery Manager User's Guide*.

RMAN **channels** (that is, connections to server sessions on the target database) perform all RMAN operations. By default, RMAN allocates one disk channel for all operations.

The following command configures RMAN to write disk backups to the `/backup` directory (refer to "Backing Up and Copying Database Files" on page 9). The format specifier `%t` is replaced with a four byte time stamp, `%s` with the backup set number, and `%p` with the backup piece number:

```
CONFIGURE CHANNEL DEVICE TYPE DISK FORMAT '/tmp/ora_df%t_s%s_s%p';
```

Configuring Tape Devices and Channels After configuring an `sbt` (that is, tape or media management) device according to the instructions in your media management vendor documentation, you can make the media manager the default device:

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt; # change sbt to DISK to make disk default
```

If the media manager documentation indicates that RMAN requires a `PARMS` string, then configure it as follows:

```
CONFIGURE CHANNEL DEVICE TYPE sbt PARMS='ENV=media_manager_env_settings';
```

This command configures two `sbt` channels for use in RMAN jobs:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
```

Configuring a Retention Policy The following command configures the retention policy to a recovery window of 30 days, which ensures that RMAN retains all backups needed to recover the database to any point in time in the last 30 days:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 30 DAYS;
```

You can use the `DELETE OBSOLETE` command to delete backups that are no longer required by the retention policy. To exclude a backup from consideration by the policy, use `KEEP` option of the `BACKUP` command.

Configuring Control File Autobackups This command configures RMAN to back up the control file to a default location after each backup or copy:

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

The following command configures RMAN to write control file autobackups to the `?/oradata` directory instead of the default location:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '?/oradata/cf%F';
```

Clearing a Configuration You can return any `CONFIGURE` command to its default setting by running the command with the `CLEAR` option, as in:

```
CONFIGURE CHANNEL DEVICE TYPE sbt CLEAR;
CONFIGURE RETENTION POLICY CLEAR;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK CLEAR;
```

Backing Up and Copying Database Files

Use the BACKUP command to back up files to the configured default device by using the configured channels. For example, the following command backs up the database and all archived logs:

```
BACKUP DATABASE PLUS ARCHIVELOG;
```

RMAN stores its backups in **backup sets**. A backup set is a logical structure that contains one or more **backup pieces**, which are the physical files containing the data. A backup set usually contains only one backup piece. Only RMAN can create and restore backup sets.

Backing Up Individual Files You can also back up individual tablespaces, database files, server parameter files, and backup sets with various options, as in these examples:

```
BACKUP ARCHIVELOG TIME BETWEEN ’SYSDATE-31’ AND ’SYSDATE-7’;
BACKUP TABLESPACE system, users, tools, undotbs;
BACKUP DATAFILE ’?/oradata/trgt/users01.dbf’, ’?/oradata/trgt/tools01.dbf’;
BACKUP CURRENT CONTROLFILE TO ’/backup/curr_cf.copy’;
BACKUP SPFILE;
BACKUP BACKUPSET ALL;
```

Backup Options RMAN has a number of BACKUP command options that control all aspects of backup set generation.

Parameter	Example	Explanation
FORMAT	FORMAT ’/tmp/%U’	Specifies a nondefault location and name for backup pieces. You must use substitution variables.
FILESPERSET	FILESPERSET 20	Limits the number of database files or archived logs placed in a backup set.
MAXSETSIZE	MAXSETSIZE 5G	Specifies the maximum byte size of the backup set.
COPIES	COPIES 2	Specifies the number of identical copies of each backup piece.
TAG	TAG ’monday_bak’	Specifies a user-defined string as a label for the backup. If you do not specify a tag for backups and copies, RMAN assigns a default tag with the date and time.

The following BACKUP commands illustrate these options:

```
BACKUP TABLESPACE tools, indx, undotbs FORMAT '?/oradata/%U';
BACKUP FILESPERSET 20 FORMAT='AL_%d/%t/%s/%p' ARCHIVELOG LIKE ’%arc_dest%’;
BACKUP TAG ’weekly_full_db_bkup’ DATABASE MAXSETSIZE 10M;
BACKUP COPIES 2 DEVICE TYPE sbt BACKUPSET ALL;
```

Incremental Backups In an incremental backup strategy, you first create a **level 0** backup, which is a whole backup of the database. For example:

```
BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

Later, you can create backups at a higher "level." In a cumulative incremental backup, RMAN only backs up those blocks that are different between the specified level *n* and the most recent level *n*-1. The following level 1 backup only copies blocks changed since the last level 0 backup:

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

Testing Backups You can run a test RMAN backup that does not generate any output. The test checks datafiles for physical and logical corruption and that all database files exist and are in the correct locations. For example:

```
BACKUP VALIDATE DATABASE ARCHIVELOG ALL;
```

Copying Files The `COPY` command makes image copies of datafiles, control files, and archived logs, as well as copies of these files. The image copies are *not* in an RMAN-specific format and can be restore with user-managed methods. For example:

```
COPY DATAFILE 1 TO '/tmp/df1.cpy', CURRENT CONTROLFILE TO '/tmp/cf.cpy';
```

Restoring and Recovering Database Files

The `RESTORE` and `RECOVER` commands manage all aspects of RMAN restore and recovery.

Recovering the Whole Database Use the `RESTORE DATABASE` and `RECOVER DATABASE` commands on the whole database. Only run these commands when the database is mounted. For example:

```
STARTUP FORCE MOUNT;
RESTORE DATABASE;
RECOVER DATABASE;
ALTER DATABASE OPEN;
```

Recovering Individual Tablespaces Use the `RESTORE TABLESPACE` and `RECOVER TABLESPACE` commands on individual tablespaces when the database is open. Take the tablespace that needs recovery offline, restore and then recover the tablespace, and bring the recovered tablespace online. This example recovers tablespace `users`:

```
RUN
{
  SQL 'ALTER TABLESPACE users OFFLINE';
  # To restore to a different location, uncomment the following commands.
  # SET NEWNAME FOR DATAFILE 8 TO '/newdir/new_filename_for_8.f';
  RESTORE TABLESPACE users;
  # If you restored to different locations, uncomment the following line.
  # SWITCH DATAFILE ALL;
  RECOVER TABLESPACE users;
  SQL 'ALTER TABLESPACE users ONLINE';
}
```

Recovering Individual Datafiles Use the `RESTORE DATAFILE` and `RECOVER DATAFILE` commands on individual datafiles when the database is open. Take the datafile that needs recovery offline, restore and recover the datafile, and bring the datafile online. For example, to restore and recover datafile `7`:

```
RUN
{
  SQL 'ALTER DATABASE DATAFILE 7 OFFLINE';
  # To restore to a different location, uncomment the following command.
  # SET NEWNAME FOR DATAFILE 7 TO '/newdir/new_filename.f';
  RESTORE DATAFILE 7;
  # If you restored to different locations, uncomment the following line.
  # SWITCH DATAFILE ALL;
  RECOVER DATAFILE 7;
  SQL 'ALTER DATABASE DATAFILE 7 ONLINE';
}
```

Recovering Individual Data Blocks You can use RMAN to recover one or more corrupted data blocks. Typically, the corruption is reported in alert logs, trace files or results of SQL queries. For example:

```
BLOCKRECOVER DATAFILE 7 BLOCK 233, 235 DATAFILE 4 BLOCK 101;
```

You can also use BLOCKRECOVER to repair all corrupted blocks listed in V$BACKUP_CORRUPTION and V$COPY_CORRUPTION. These views are filled whenever RMAN performs a complete scan of a file for a backup. Run this command to repair blocks listed in the views:

```
BLOCKRECOVER CORRUPTION LIST;
```

Reporting on RMAN Operations

Within RMAN, use the LIST and REPORT commands to obtain metadata from the RMAN repository, and use SHOW ALL to display the current RMAN configuration. You can also query the views described in "Backup and Recovery Views" on page 27.

Listing Backups and Copies Run the LIST BACKUP and LIST COPY commands to displays all RMAN backups and copies. You can also display specific objects as in the following examples:

```
LIST BACKUP OF DATABASE;
LIST COPY OF DATAFILE 1, 2, 3;
LIST BACKUP OF ARCHIVELOG FROM SEQUENCE 1437;
LIST CONTROLFILECOPY "/tmp/cf.cpy";
LIST BACKUPSET 193, 207, 242;
```

For backups, you can control the format of LIST output with these options:

Parameter	Example	Explanation
BY BACKUP	LIST BACKUP OF DATABASE BY BACKUP	Organizes the output by backup set. This is the default mode of presentation.
BY FILE	LIST BACKUP BY FILE	Lists the backups according to which file was backed up.
SUMMARY	LIST BACKUP SUMMARY	Displays reduced output. By default, the output is VERBOSE.

For both backups and copies you have the following additional options:

Parameter	Example	Explanation
EXPIRED	LIST EXPIRED COPY	Displays files that the CROSSCHECK command showed as inaccessible.
RECOVERABLE	LIST BACKUP RECOVERABLE	Specifies datafile backups or copies that are available and that can be restored and recovered in the current database incarnation.

Reporting on Database Files and Backups The REPORT command performs more complex analysis than LIST. Some of the main options are:

Parameter	Example	Explanation
NEED BACKUP	REPORT NEED BACKUP DATABASE	Shows which files need backups based on configured retention policy. Alternatively, use REDUNDANCY and RECOVERY WINDOW parameters to specify criteria.
OBSOLETE	REPORT OBSOLETE	Lists the backups and copies that are obsolete under the configured retention policy. Alternatively, use REDUNDANCY and RECOVERY WINDOW parameters to specify criteria.
UNRECOVERABLE	REPORT UNRECOVERABLE	Lists all datafiles for which an unrecoverable operation has been performed against an object in the datafile since the last backup of the datafile.
SCHEMA	REPORT SCHEMA AT TIME 'SYSDATE-30'	Reports the tablespaces and datafiles in the database at the current time (default) or a different time.

Monitoring RMAN Through V$ Views To correlate a channel with a process, run the following query in SQL*Plus while the RMAN job is executing:

```
COLUMN CLIENT_INFO FORMAT a30
COLUMN SID FORMAT 999
COLUMN SPID FORMAT 9999

SELECT s.SID, p.SPID, s.CLIENT_INFO
FROM V$PROCESS p, V$SESSION s
WHERE p.ADDR = s.PADDR
AND CLIENT_INFO LIKE 'rman%';
```

To calculate the progress of an RMAN job, run the following query in SQL*Plus while the RMAN job is executing:

```
SELECT SID, SERIAL#, CONTEXT, SOFAR, TOTALWORK,
       ROUND(SOFAR/TOTALWORK*100,2) "% COMPLETE"
FROM V$SESSION_LONGOPS
WHERE OPNAME LIKE 'RMAN%'
AND OPNAME NOT LIKE '%aggregate%'
AND TOTALWORK != 0
AND SOFAR <> TOTALWORK;
```

Managing the RMAN Repository

RMAN always stores all of its metadata in the target database control file. Optionally, you can create a recovery catalog schema in a separate database, which obtains its metadata from the control file.

Monitoring Control File Records If you do not use a recovery catalog, then eventually RMAN control file records are overwritten. Set this initialization parameter in the target database to determine how long records are kept:

```
CONTROL_FILE_RECORD_KEEP_TIME = number_of_days_to_keep
```

Creating a Recovery Catalog Create a user and schema for the recovery catalog, and grant the necessary privileges. For example, in SQL*Plus enter:

```
CREATE USER rman IDENTIFIED BY rman TEMPORARY TABLESPACE temp
DEFAULT TABLESPACE cattbs QUOTA UNLIMITED ON cattbs;
GRANT RECOVERY_CATALOG_OWNER, CONNECT, RESOURCE TO rman;
```

Then, connect to the recovery catalog database as the catalog owner. For example:

```
% rman CATALOG rman/rman@rcat
```

In the RMAN shell, run the following command:

```
CREATE CATALOG;
```

Registering the Target Database in the Recovery Catalog To register the target database in the recovery catalog, connect to the target and catalog databases, and run this command:

```
REGISTER DATABASE; # target must be mounted or open; catalog must be open
```

Storing RMAN Scripts in the Recovery Catalog You can create scripts in RMAN and then store them in the recovery catalog. For example:

```
CREATE script b_whole_l0
{
    BACKUP INCREMENTAL LEVEL 0 TAG mon_bkup DATABASE;
    BACKUP ARCHIVELOG ALL DELETE ALL INPUT;
}
```

Execute stored scripts within a `RUN` command as follows:

```
RUN { EXECUTE script mon_bkup; } # executes script mon_bkup
```

Use `DELETE SCRIPT` to delete a script, `REPLACE SCRIPT` to replace a script (or create it if does not exist), and `PRINT SCRIPT` to display a stored script.

Crosschecking Backups and Copies The CROSSCHECK command checks whether RMAN backups and copies exist and are readable. Assuming that you have configured automatic channels, you can run these commands:

```
CROSSCHECK BACKUP;  # checks all RMAN backups on configured devices
CROSSCHECK COPY;  # checks all RMAN copies on configured devices
```

If automatic sbt channels are not configured, then you must allocate a maintenance channel before running CROSSCHECK and DELETE commands on sbt objects:

```
ALLOCATE CHANNEL FOR MAINTENANCE DEVICE TYPE sbt;  # not needed for disk
```

Deleting Backups and Copies The DELETE command removes RMAN backups and copies from DISK and sbt devices, marks the objects as DELETED in the control file, and removes the records from the recovery catalog (if you use a catalog). For example:

```
DELETE BACKUPSET 101, 102, 103;
DELETE CONTROLFILECOPY '/tmp/cf.cpy';
DELETE NOPROMPT ARCHIVELOG UNTIL SEQUENCE = 7300;
DELETE BACKUP OF SPFILE TABLESPACE users DEVICE TYPE sbt; # deletes tape backups
DELETE BACKUP OF DATABASE LIKE '/tmp%'; # LIKE specifies name of the backup piece
DELETE ARCHIVELOG ALL BACKED UP 2 TIMES TO DEVICE TYPE sbt;
```

The following options of the DELETE command are also useful:

Parameter	Example	Explanation
EXPIRED	DELETE EXPIRED	Deletes the backups and copies marked as EXPIRED (that is, "not found") by the CROSSCHECK command.
OBSOLETE	DELETE OBSOLETE	Deletes the backups and copies that are obsolete (that is, "not needed") under the configured retention policy. Alternatively, use REDUNDANCY and RECOVERY WINDOW parameters to specify criteria.
NOPROMPT	DELETE NOPROMPT OBSOLETE	Specifies that you do not want to be prompted to confirm the list of files to be deleted.

Cataloging and Uncataloging Backups and Copies If you made copies of datafiles, control files, or archived logs with operating system utilities, then you can run CATALOG to add metadata about them to the RMAN repository. Run CHANGE ... UNCATALOG to remove metadata about backups and copies. For example:

```
CATALOG DATAFILECOPY '/backup/users01.bak'; # users01.bak is a user-managed copy
CHANGE CONTROLFILECOPY '/tmp/cf.cpy' UNCATALOG;
CHANGE BACKUPSET 121,122,127,203,300 UNCATALOG;
```

RMAN Quick Syntax Reference

The most common RMAN commands are the following:

@

BACKUP

CHANGE

CONFIGURE

COPY

CROSSCHECK

DELETE

LIST

RECOVER

REPORT

RESTORE

RUN

SET

SHOW

The following subclauses are shared by several commands:

- archivelogRecordSpecifier
- completedTimeSpec
- datafileSpec
- deviceSpecifier
- maintQualifier
- untilClause

The syntax descriptions use vertical ellipses to indicate parameters and options that are not documented in this *Quick Reference*. Refer to the *Oracle9i Recovery Manager Reference* for complete documentation of RMAN syntax.

@

Runs a text file containing RMAN commands.

```
@filename
```

archivelogRecordSpecifier

This subclause specifies a range of archived redo logs.

```
ARCHIVELOG
{ ALL
| LIKE 'string_pattern'
| archlogRange [LIKE 'string_pattern' [THREAD [=] integer]]
}

{ { { UNTIL TIME | FROM TIME } [=] 'date_string'
  | { TIME BETWEEN 'date_string' AND
    | FROM TIME [=] 'date_string' UNTIL TIME [=]
    }
    'date_string'
  | UNTIL SCN [=] integer
  | SCN BETWEEN integer AND integer
  | FROM SCN [=] integer [UNTIL SCN [=] integer]
  }
  [THREAD [=] integer]
| { UNTIL SEQUENCE [=] integer
  | FROM SEQUENCE [=] integer [UNTIL SEQUENCE [=] integer]
  | SEQUENCE [BETWEEN integer AND] integer
  }
  [THREAD [=] integer]
}
```

BACKUP

Backs up database files, archived logs, backups, and copies.

```
BACKUP [ FULL | INCREMENTAL LEVEL [=] integer ]
[backupOperand [backupOperand]...] backupSpec [backupSpec]...
[PLUS ARCHIVELOG [backupSpecOperand [backupSpecOperand]...]];

backupOperand::=
{ FORMAT [=] 'format_string' [, 'format_string']...
| CHANNEL ['] channel_id [']
| CUMULATIVE
| MAXSETSIZE [=] integer [ K | M | G ]
| FILESPERSET [=] integer
| TAG [=] ['] tag_name [']
| keepOption
| SKIP { OFFLINE | READONLY | INACCESSIBLE }
| VALIDATE
| NOT BACKED UP [SINCE TIME [=] 'date_string']
| COPIES [=] integer
| DEVICE TYPE deviceSpecifier
.
.
.
}

backupSpec::=
[(]
{ BACKUPSET
  { { ALL | completedTimeSpec }
  | primary_key [, primary_key]...
  }
| DATAFILE datafileSpec [, datafileSpec]...
| DATAFILECOPY 'filename' [, 'filename']...
| DATAFILECOPY TAG [=] ['] tag_name ['] [, ['] tag_name [']]...
| TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name [']]...
| DATABASE
| archivelogRecordSpecifier
| CURRENT CONTROLFILE [FOR STANDBY]
| CONTROLFILECOPY 'filename'
| SPFILE
}
[backupSpecOperand [backupSpecOperand]...]
[)]

backupSpecOperand::=
{ FORMAT [=] 'format_string' [, 'format_string']...
| CHANNEL ['] channel_id [']
| MAXSETSIZE [=] integer [ K | M | G ]
| FILESPERSET [=] integer
| TAG [=] ['] tag_name [']
| keepOption
| SKIP { OFFLINE | READONLY | INACCESSIBLE }
| NOT BACKED UP [ SINCE TIME [=] 'date_string'
                | integer TIMES
                ]
| DELETE [ALL] INPUT
.
.
.
}
```

CHANGE

Alters the status of a backup or copy in the RMAN repository.

```
CHANGE
{ { BACKUP | COPY } [OF listObjList] [ maintQualifier [maintQualifier]...]
| recordSpec [DEVICE TYPE deviceSpecifier [, deviceSpecifier]...]
}
{ AVAILABLE | UNAVAILABLE | UNCATALOG | keepOption }
[DEVICE TYPE deviceSpecifier [, deviceSpecifier]...];
```

completedTimeSpec

```
COMPLETED
{ AFTER [=]
| BETWEEN 'date_string' AND | BEFORE [=] } 'date_string'
```

CONFIGURE

Creates a persistent RMAN configuration.

```
CONFIGURE
{ deviceConf
| backupConf
| { AUXNAME FOR DATAFILE datafileSpec
  | SNAPSHOT CONTROLFILE NAME
  }
  { TO 'filename' | CLEAR }
| cfauConf
};

deviceCon::=
{ DEFAULT DEVICE TYPE { TO deviceSpecifier | CLEAR }
| DEVICE TYPE deviceSpecifier { PARALLELISM integer | CLEAR }
| [AUXILIARY] CHANNEL [integer] DEVICE TYPE deviceSpecifier
  { allocOperandList | CLEAR }
}

allocOperandList::=
{ PARMS [=] 'channel_parms'
| FORMAT [=] 'format_string' [, 'format_string']...
| { MAXPIECESIZE [=] integer | RATE [=] integer } [ K | M | G ]
.
.
.
}...
connectStringSpec::=
['] [userid] [/ [password]] [@net_service_name] [']

backupConf::=
{ RETENTION POLICY { TO { RECOVERY WINDOW OF integer DAYS
                           | REDUNDANCY [=] integer
                           | NONE
                         }
                      | CLEAR
                    }
| MAXSETSIZE { TO { integer [ K | M | G ]
                    | UNLIMITED
                   }
               | CLEAR
              }
| { ARCHIVELOG | DATAFILE }
  BACKUP COPIES FOR DEVICE TYPE deviceSpecifier
  { TO integer | CLEAR }
| BACKUP OPTIMIZATION { ON | OFF | CLEAR }
| EXCLUDE FOR TABLESPACE tablespace_name [CLEAR]
}

cfauConf::==
CONTROLFILE AUTOBACKUP
{ ON
| OFF
| CLEAR
| FORMAT FOR DEVICE TYPE deviceSpecifier { TO 'format string' | CLEAR }
}
```

COPY

Makes image copies of database files, archived logs, backups, and copies.

```
COPY [copy_option [copy_option]...]
{ copy_inputfile TO
  { AUXNAME | 'filename' [copy_option [copy_option]...] }
  [, copy_inputfile TO
   { AUXNAME | 'filename' [copy_option [copy_option]...] }
  ]...
};
```

```
copy_option::=
{ TAG [=] ['] tag_name [']
| LEVEL [=] integer
| keepOption
.
.
.
}

copy_inputfile::=
{ DATAFILE datafileSpec
| DATAFILECOPY { 'filename' | TAG [=] ['] tag_name ['] }
| ARCHIVELOG 'filename'
| CURRENT CONTROLFILE [FOR STANDBY]
| CONTROLFILECOPY { 'filename' | TAG [=] ['] tag_name ['] }
}
```

CROSSCHECK

Checks backup pieces, proxy copies, and disk copies to determine whether they exist.

```
CROSSCHECK
{
  { BACKUP [OF listObjList]
  | COPY [OF listObjList]
  | archivelogRecordSpecifier
  } [maintQualifier [maintQualifier]...]
| recordSpec [DEVICE TYPE deviceSpecifier [, deviceSpecifier]...]
};

listObjList::=
[ DATAFILE datafileSpec [, datafileSpec]...
| TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name [']]...
| archivelogRecordSpecifier
| DATABASE [SKIP TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name
[']]...]
| CONTROLFILE
| SPFILE
]...

recordSpec::=
{ { BACKUPPIECE | PROXY }
  { 'media_handle' [, 'media_handle']...
  | primary_key [, primary_key]...
  | TAG [=] ['] tag_name [']
  }
| BACKUPSET primary_key [, primary_key]...
| { CONTROLFILECOPY | DATAFILECOPY }
  { { primary_key [, primary_key]...
    | 'filename' [, 'filename']...
    }
  | TAG [=] ['] tag_name ['] [, ['] tag_name [']]...
  }
| ARCHIVELOG
  { primary_key [, primary_key]...
  | 'filename' [, 'filename']...
  }
}
```

datafileSpec

```
datafileSpec::=
{ 'filename'| integer }
```

DELETE

Deletes backups and copies, updates their metadata records to status `DELETED` in the control file, and removes their metadata from the recovery catalog (if you use a catalog).

```
DELETE [FORCE] [NOPROMPT]
{ [EXPIRED]
  {
    { BACKUP [OF listObjList]
    | COPY   [OF listObjList]
    | archivelogRecordSpecifier
    } [maintQualifier [maintQualifier]...]
  | recordSpec [DEVICE TYPE deviceSpecifier [, deviceSpecifier]...]
  }
| OBSOLETE [obsOperandList]
  [DEVICE TYPE (deviceSpecifier [, deviceSpecifier]...]
};

obsOperandList::=
[ REDUNDANCY [=] integer | RECOVERY WINDOW OF integer DAYS | ORPHAN ]...
```

deviceSpecifier

```
deviceSpecifier::=
{ DISK | ['] media_device ['] }
```

LIST

Lists the backups and copies recorded in the repository.

```
LIST
{ INCARNATION [OF DATABASE [['] database_name [']]]
| [EXPIRED]
  { listObjectSpec
    [ maintQualifier | RECOVERABLE [untilClause] ]...
  | recordSpec
  }
};
```

```
listObjectSpec::=
{ BACKUP [OF listObjList] [listBackupOption]
| COPY [OF listObjList]
| archivelogRecordSpecifier
}

listObjectList::=
[ DATAFILE datafileSpec [, datafileSpec]...
| TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name [']]...
| archivelogRecordSpecifier
| DATABASE [SKIP TABLESPACE ['] tablespace_name [']
  [, ['] tablespace_name [']]...]
| CONTROLFILE
| SPFILE
]...

listBackupOption::=
[ [BY BACKUP] [VERBOSE]
| SUMMARY
| BY { BACKUP SUMMARY | FILE }
]
```

maintQualifier

```
{ TAG [=] ['] tag_name [']
| completedTimeSpec
| LIKE 'string_pattern'
| DEVICE TYPE deviceSpecifier [, deviceSpecifier]...
| BACKED UP integer TIMES TO DEVICE TYPE deviceSpecifier
}
```

RECOVER

Performs media recovery on RMAN backups and copies.

```
RECOVER [DEVICE TYPE deviceSpecifier [, deviceSpecifier]...]
recoverObject [recoverOptionList];

recoverObject::=
{ DATABASE
  [ untilClause
  | [untilClause] SKIP [FOREVER] TABLESPACE
    ['] tablespace_name ['] [, ['] tablespace_name [']]...
  ]
| TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name [']]...
| DATAFILE datafileSpec [, datafileSpec]...
}

recoverOptionList::=
{ DELETE ARCHIVELOG [MAXSIZE {integer [K | M | G]}]
| CHECK READONLY
| NOREDO
| { FROM TAG | ARCHIVELOG TAG } [=] ['] tag_name [']
.
.
.
}...
```

REPORT

Reports on various RMAN features: which files need backups, which backups are obsolete or unrecoverable, and which files are in the database.

```
REPORT
{ { NEED BACKUP [ { INCREMENTAL | DAYS } [=] integer
                | REDUNDANCY [=] integer
                | RECOVERY WINDOW OF integer DAYS)
                ]
  | UNRECOVERABLE
  }
  reportObject
| SCHEMA [atClause]
| OBSOLETE [obsOperandList]
}
[ DEVICE TYPE deviceSpecifier [,deviceSpecifier]... ]

reportObject::=
[ DATAFILE datafileSpec [, datafileSpec]...
| TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name [']]...
| DATABASE [SKIP TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name
[']]...]
]

atClause::=
{ AT TIME [=] 'date_string'
| AT SCN [=] integer
| AT SEQUENCE [=] integer THREAD [=] integer
}

obsOperandList::=
[ REDUNDANCY [=] integer | RECOVERY WINDOW OF integer DAYS | ORPHAN ]...
```

RESTORE

Restores RMAN backups and copies.

```
RESTORE
[(] restoreObject [(restoreSpecOperand [restoreSpecOperand]...] [)]...
[ CHANNEL ['] channel_id [']
| PARMS [=] 'channel_parms'
| FROM { BACKUPSET | DATAFILECOPY }
| untilClause
| FROM TAG [=] ['] tag_name [']
| VALIDATE
| DEVICE TYPE deviceSpecifier [, deviceSpecifier]...
.
.
.
]...;

restoreObject::=
{ CONTROLFILE [TO 'filename']
| DATABASE
  [SKIP [FOREVER] TABLESPACE
   ['] tablespace_name ['] [, ['] tablespace_name [']]...
  ]
| DATAFILE datafileSpec [, datafileSpec]...
| TABLESPACE ['] tablespace_name ['] [, ['] tablespace_name [']]...
| archivelogRecordSpecifier
| SPFILE [TO [PFILE] 'filename']
}

restoreSpecOperand::=
{ CHANNEL ['] channel_id [']
| FROM TAG [=] ['] tag_name [']
| PARMS [=] 'channel_parms'
| FROM
  { AUTOBACKUP
    [{ MAXSEQ | MAXDAYS } [=] integer)]...
  | 'media_handle'
  }
}
```

RUN

Although you typically run commands at the RMAN prompt, you can also include commands within a RUN command. This method is essential when you want to run ALLOCATE CHANNEL or for some uses of the SET command.

```
RUN
{
  .
  .
  .
}
```

SET

Creates settings that apply only to the current RMAN session.

```
SET { set_rman_option [;] | set_run_option; }

set_rman_option::=
{ ECHO { ON | OFF }
| DBID [=] integer
| CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE deviceSpecifier TO 'frmt_string'

set_run_option::=
{ NEWNAME FOR DATAFILE datafileSpec TO { 'filename' | NEW }
| ARCHIVELOG DESTINATION TO 'log_archive_dest'
| untilClause
| COMMAND ID TO 'string'
| CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE deviceSpecifier TO 'frmt_string'
.
.
.
}
```

SHOW

Displays the currently enabled `CONFIGURE` commands.

```
SHOW
{ RETENTION POLICY
| [DEFAULT] DEVICE TYPE
| [AUXILIARY] CHANNEL [FOR DEVICE TYPE deviceSpecifier]
| MAXSETSIZE
| { DATAFILE | ARCHIVELOG } BACKUP COPIES
| BACKUP OPTIMIZATION
| SNAPSHOT CONTROLFILE NAME
| AUXNAME
| EXCLUDE
| CONTROLFILE AUTOBACKUP [FORMAT]
| ALL
};
```

untilClause

```
{ UNTIL TIME [=] 'date_string'
| UNTIL SCN [=] integer
| UNTIL SEQUENCE [=] integer THREAD [=] integer
}
```

Backup and Recovery Views

You should refer to the *Oracle9i Recovery Manager Reference* for documentation of the recovery catalog views and *Oracle9i Database Reference* for documentation of the V$ views.

Recovery Catalog View	Corresponding V$ View	Catalog View Describes
RC_ARCHIVED_LOG	V$ARCHIVED_LOG	Archived and unarchived redo logs
RC_BACKUP_CONTROLFILE	V$BACKUP_DATAFILE	Control files in backup sets
RC_BACKUP_CORRUPTION	V$BACKUP_CORRUPTION	Corrupt block ranges in datafile backups
RC_BACKUP_DATAFILE	V$BACKUP_DATAFILE	Datafiles in backup sets
RC_BACKUP_PIECE	V$BACKUP_PIECE	Backup pieces
RC_BACKUP_REDOLOG	V$BACKUP_REDOLOG	Archived logs in backups
RC_BACKUP_SET	V$BACKUP_SET	Backup sets for all database incarnations
RC_BACKUP_SPFILE	V$BACKUP_SPFILE	Server parameter files in backups
RC_CONTROLFILE_COPY	V$DATAFILE_COPY	Control file copies on disk
RC_COPY_CORRUPTION	V$COPY_CORRUPTION	Control file copies on disk
RC_DATABASE	V$DATABASE	Databases registered in the recovery catalog
RC_DATABASE_BLOCK_CORRUPTION	V$DATABASE_BLOCK_CORRUPTION	Database blocks marked as corrupt in the most recent RMAN backup or copy
RC_DATABASE_INCARNATION	V$DATABASE_INCARNATION	All database incarnations registered in the catalog
RC_DATAFILE	V$DATAFILE	All datafiles registered in the recovery catalog
RC_DATAFILE_COPY	V$DATAFILE_COPY	Datafile copies on disk
RC_LOG_HISTORY	V$LOG_HISTORY	Historical information about online redo logs
RC_OFFLINE_RANGE	V$OFFLINE_RANGE	Offline ranges for datafiles
RC_PROXY_CONTROLFILE	V$PROXY_DATAFILE	Control file backups created by proxy copy
RC_PROXY_DATAFILE	V$PROXY_DATAFILE	Datafile backups created by proxy copy

Recovery Catalog View	Corresponding V$ View	Catalog View Describes
RC_REDO_LOG	V$LOG and V$LOGFILE	Online redo logs for all incarnations of the database since the last catalog resynchronization
RC_REDO_THREAD	V$THREAD	All redo threads for all incarnations of the database since the last catalog resynchronization
RC_RESYNC	n/a	Recovery catalog resynchronizations
RC_RMAN_CONFIGURATION	V$RMAN_ CONFIGURATION	RMAN persistent configuration settings
RC_STORED_SCRIPTS	n/a	Names of stored scripts
RC_STORED_SCRIPT_LINE	n/a	Lines of stored scripts
RC_TABLESPACE	V$TABLESPACE	All tablespaces registered in the recovery catalog, all dropped tablespaces, and tablespaces that belong to old incarnations

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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