

CHAPTER 3

RMAN Setup and Configuration



Let's get started with this RMAN thing, shall we? Let's just reach down, pull on the handle, I said, pull on the handle...and, it doesn't start. We first need to set up RMAN and our database for backup and recovery operations before we can actually do anything. In this chapter, we look at initial RMAN setup requirements and options. First, we dive into Oracle redo logs a little deeper than we did in Chapter 1.

These are critical structures in the Oracle database for recovery. Building on that discussion, we look at putting the database in ARCHIVELOG mode, in case you want to do online backups. We then look at the basic RMAN interface, so that you can get into RMAN itself. Next, we discuss configuring RMAN for database backup operations. Finally, we discuss the RMAN recovery catalog, including why you might want to use it and how to configure it.

Configuring Your Database to Run in ARCHIVELOG Mode

Now that you have learned about ARCHIVELOG mode and NOARCHIVELOG mode in Chapter 1 and learned how important redo is to your database, you probably understand why many DBAs run their databases in ARCHIVELOG mode. If you are content with running in NOARCHIVELOG mode, then much of this section's discussion will not apply to you. If you are going to run in ARCHIVELOG mode, you will need to do some basic configuration, which is the topic of this section.

When running in ARCHIVELOG mode, you have two choices in configuring where the archived redo logs are copied. In fact, you can choose to use both choices. The first choice is to configure for ARCHIVELOG destination directories, and the second is to configure the Oracle flash recovery area (FRA). We will discuss those two topics next. Afterward, we will discuss actually putting the database in ARCHIVELOG mode.

ARCHIVELOG Destination Directories

When configuring ARCHIVELOG mode, you will need to decide where you want Oracle to create archived redo logs. The option that has been available for the longest is to use archive log destination directories. To use archive log destination directories, you set some specific parameters in Oracle to configure this option. First, you use the LOG_ARCHIVE_DEST_ *n* (where *n* is a number in the range of 1 to 10) parameter to define up to ten different archive log destinations. These destinations can be local directories, network directories (for example, NT folders), NAS (network-attached storage), or even a defined database service name if you are using standby database/Data Guard. Note that there is no default location defined for LOG_ARCHIVE_DEST_ *n*.

If you are using SPFILES, you use the **alter system** command to set the LOG_ARCHIVE_DEST_ *n* parameter as seen here:

```
alter system set log_archive_dest_1='location=c:\oracle\oraarc\beta1';
```

NOTE

Setting the LOG_ARCHIVE_DEST directory to a directory location that does not exist, or that Oracle cannot write to, is a common mistake.

*Just make sure that after you set the parameter and put the database in ARCHIVELOG mode, you issue an **alter system switch logfile** command to make sure that ARCH is writing the archived redo logs properly.*

Each LOG_ARCHIVE_DEST_ *n* location can be defined as either a mandatory or optional location. By default, all LOG_ARCHIVE_DEST_ *n* locations are optional in Oracle Database 11g.

Mandatory locations mean just that—the archived redo logs have to be written to that location. Failure of the ARCH process to write to mandatory locations will result in suspension of database activities fairly quickly (after you have cycled through all the online redo logs). Optional locations will have no impact on database operations.

```
alter system set log_archive_dest_1='location=c:\oracle\oraarc\beta1 mandatory';
```

In Oracle Database 11g, all LOG_ARCHIVE_DEST_ *n* locations are optional by default (though one location must always succeed since the minimum setting of LOG_ARCHIVE_MIN_SUCCEED_DEST is 1). The parameter LOG_ARCHIVE_MIN_SUCCEED_DEST indicates how many archive log destination directories must have successful copies for an online redo log to be considered successfully archived. The default setting for LOG_ARCHIVE_MIN_SUCCEED_DEST is 1, and this is the minimum setting for this parameter. Here is an example of setting this parameter to a value of 2:

```
alter system set log_archive_min_succeed_dest=2;
```

Other parameters are related to archived redo logs, the ARCH process, and the LOG_ARCHIVE_DEST series of parameters:

- **LOG_ARCHIVE_STATE_ *n*** Defines one of two different states for each archive log destination. If set to ENABLE, the ARCH process will consider the destination associated with this state as a valid archive log destination. If set to DEFER, the ARCH process will not archive logs to the related LOG_ARCHIVE_DEST_ *n* location.
- **LOG_ARCHIVE_FORMAT** Provides a template for Oracle to use when naming archived redo logs. As Oracle creates the archived redo logs, it renames them in such a way that each of the archived redo logs has a unique name assigned to it. Using the LOG_ARCHIVE_FORMAT parameter, you can manipulate the default naming standard as you require. This parameter has no effect for archived redo logs being created in the FRA.
- **LOG_ARCHIVE_START** This parameter is obsolete in Oracle Database 10g and later versions. Oracle will now start the ARCH process for you automatically.
- **LOG_ARCHIVE_MAX_PROCESSES** This parameter defines the number of ARCH processes that Oracle initially starts when the database is started.

NOTE

If you are running Oracle Database 9i or earlier, you will need to make sure you set the LOG_ARCHIVE_START parameter to TRUE when configuring your database for ARCHIVELOG mode. This is no longer required in Oracle Database 10g and later.

Each of the different parameters mentioned thus far is defined in the *Oracle Database 11g Reference Manual* (which is part of the overall Oracle documentation), should you need further information on them.

In the following example, we have a database we want to put in ARCHIVELOG mode. We will create three different archive log destination directories, including one to a service name that supports an Oracle standby database. We will also enforce the requirements that at least two of these destinations must be written to in order for the movement of the archived redo log to be considered complete, and that the standby database must be one of those two locations. Here is

an example of the use of the various database parameter file parameters related to ARCHIVELOG mode operations:

```
log_archive_dest_1='location=d:\oracle\oraarc\robt mandatory'
log_archive_dest_2='location=z:\oracle\oraarc\robt optional'
log_archive_dest_3='service=recover1 mandatory'
log_archive_min_succeed_dest=2
log_archive_format="robt_%s_%t.arc"
```

In this example, our first archive log destination goes to d:\oracle\oraarc\robt. The second archive log destination is to a secondary location on the Z: drive. We have made this an optional archiving location because it is a networking device (which may not be all that reliable). The third destination is to an Oracle Net service (probably a standby database) called recover1. This will cause Oracle to send the archived redo logs through Oracle Net as they are generated.

Proceeding through the example, by using the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter, we have indicated that the archived redo logs must be successfully copied to at least two different locations. The format of the archived redo log is defined with the LOG_ARCHIVE_FORMAT parameter.

The Flash Recovery Area

The flash recovery area (FRA) allows you to centralize storage of all recovery-related files. The FRA can use locally attached storage, the Oracle Cluster File System (OCFS), or Automatic Storage Management (ASM) features. Table 3-1 lists the file types that can be contained within the FRA. The FRA helps with the management of overall disk space allocation and provides a centralized storage area for all recovery-related files.

Retention of files in the FRA is determined by the RMAN retention policy. This is set via the RMAN **configure retention policy** command. This command and RMAN retention will be discussed in much more detail later in this chapter. If a file does not have a retention policy associated with it or it's a permanent file, then it will never be deleted. If a file is not yet obsolete under the RMAN retention policy, then it will not be deleted. Finally, archived logs are eligible for deletion once they are obsolete.

The FRA is created in a specific location defined by the parameter DB_RECOVERY_FILE_DEST. This location can be a file system or an ASM volume. You define the quota of space allocated to the database's FRA by using the DB_RECOVERY_FILE_DEST_SIZE parameter. This is a logical, database-specific, Oracle-controlled file space limitation, independent of the overall space available in the file system itself. Oracle monitors the space available in the FRA used by the database, and once the amount of available space in the FRA starts to diminish to unsafe levels, Oracle generates a warning in the alert log. In Oracle Database 11g, these warnings occur when reclaimable space is less than 15 percent of the DB_RECOVERY_FILE_DEST_SIZE value. A critical alert is also signaled when the database is at less than 3 percent of reclaimable space. These alerts appear in OEM, in the alert log, or you can review the DBA_OUTSTANDING_ALERTS table. Several views are available for you to reference when trying to determine if your flash recovery area is running out of space. We address those views later in this chapter in the section titled "Flash Recovery Area Views."

NOTE

Running out of space in the FRA can be troublesome if that area is your only archive log destination, as this can cause your database to eventually halt. If the FRA is going to be your only archive log destination, monitor space availability carefully.

File Type	Notes
Archived redo logs	Archived redo logs will be stored in the FRA.
Control file	One copy of the control file is created in the FRA when the database is created.
Control file autobackups	The default location for the RMAN control file autobackups will be the FRA, if it is defined.
Flashback logs	Flashback logs (discussed later in this chapter) will be stored in the FRA, if it is defined.
Redo log	One copy of each redo log group member can be stored in the FRA.
RMAN datafile copies	The default location for the RMAN datafile copies will be the FRA, if it is defined.
RMAN backup and other related files	The default location for the RMAN files in general (backup set pieces, etc.) will be the FRA, if it is defined.

TABLE 3-1 *File Types Found in the Flash Recovery Area*

Figuring out how much space to allocate to the FRA can be a bit challenging. Typically you have to monitor space usage and adjust the size of the FRA as required. If the database already exists, you can review the amount of archive log space consumed by checking the DBA_HIST_LOG view. This view derives its data from Oracle's AWR infrastructure. It will tell you the average size of the archived redo logs and the time of the log switch. Here is an example of a query against the DBA_HIST_LOG view:

```
SQL> alter session set nls_date_format='mm/dd/yyyy hh24:mi:ss';
Session altered.
```

```
SQL> select sequence#, first_time Log_started
2  ,lead(first_time, 1,NULL) over (order by first_time) Log_ended
3  from (select distinct sequence#, first_time
4  from dba_hist_log
5  where archived='YES'
6  and sequence#!=0
7  order by first_time)
8  order by sequence#;
```

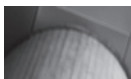
SEQUENCE#	LOG_STARTED	LOG_ENDED
82	05/18/2009 22:02:56	05/19/2009 12:52:06
83	05/19/2009 12:52:06	05/19/2009 22:01:24
84	05/19/2009 22:01:24	05/20/2009 15:25:39
85	05/20/2009 15:25:39	05/21/2009 10:00:58
86	05/21/2009 10:00:58	05/21/2009 17:02:00
87	05/21/2009 17:02:00	05/21/2009 22:01:28
88	05/21/2009 22:01:28	

7 rows selected.

Many large systems run more than one Oracle database. Each of these databases will use archive log storage space, and typically these archived redo logs will all be stored on the same file system. On occasion, a single database can consume all the space on that storage device. This can impact all databases using that storage device, since they will no longer be able to create archived redo logs. One benefit of the FRA is that it provides the ability to allocate a specific space quota to each database. Thus, with an FRA, you can reduce the risk that one database will consume all archive log space and negatively impact other databases.

If you find that the FRA has run out of space, you can respond to the problem as follows:

1. If the problem is one of insufficient space allocation via the parameter `DB_RECOVERY_FILE_DEST_SIZE` and sufficient physical disk space exists to increase the space allocated to the FRA, increase the size of the parameter. This will immediately add space to the FRA. Of course, you cannot increase this parameter to a value that is greater than the amount of space that is physically available on the file system.
2. If you need more physical space, allocate additional physical space to the file system, and then increase the size of the `DB_RECOVERY_FILE_DEST_SIZE` parameter.
3. If additional space is not available, you can move the FRA to another file system where more space is available.
4. You can also make room in the FRA by using the RMAN **backup recovery area** command to move the contents of the FRA to another location. We will cover the **backup recovery area** command and its limitations during discussions on performing RMAN backups.
5. As a last-ditch effort, physically remove older backup set pieces and/or archived redo logs from the FRA, and then use the RMAN **crosscheck** command to get the database to recognize that the files have been removed.



NOTE

If you find yourself queasy at the idea of removing physical files from the FRA, then your gut instincts are good. Essentially this means either that your retention policy is not correct or that you have not allocated enough space to support the retention policy established for your database. Also, removing files potentially compromises the recoverability of your database, so exercise extreme caution when removing files.

Setting Up the Flash Recovery Area

To set up the FRA, you will want to configure the following parameters:

Parameter	<code>DB_RECOVERY_FILE_DEST_SIZE</code>	<code>DB_RECOVERY_FILE_DEST</code>
Example	<code>Alter system set db_recovery_file_dest_size=20G scope=both;</code>	<code>Alter system set db_recovery_file_dest='/u01/oracle/flash_recovery' scope=both;</code>
Purpose	Sets the allocated size of the FRA, in bytes, and must be defined in order to enable the FRA. This allows you to control how much disk space is allocated to the FRA. You should not set this value to a size greater than the total amount of available disk space.	Specifies the location of the FRA. This can be a file system, an ASM disk location, or an OMF location.

Note that you must specify the `DB_RECOVERY_FILE_DEST_SIZE` parameter before you specify the `DB_RECOVERY_FILE_DEST` parameter. Failure to do so will result in an ORA-32001 error message. In a similar fashion, you must disable the `DB_RECOVERY_FILE_DEST` parameter before you reset the `DB_RECOVERY_FILE_DEST_SIZE` parameter. Leaving `DB_RECOVERY_FILE_DEST` empty will disable the FRA. Here is an example of disabling the FRA by resetting the `DB_RECOVERY_FILE_DEST` parameter:

```
alter system set db_recovery_file_dest=' ' scope=both;
```

Oracle allows you to archive to both the FRA and to one or more additional locations through the use of the `LOG_ARCHIVE_DEST_n` parameters. One case when you would want to do this is if you were configuring standby databases and you still wanted to take advantage of the features of the FRA.

To configure both FRA and archive log destination directories, you set the standard FRA parameter `DB_RECOVERY_FILE_DEST`, defining the location of the FRA. You will also define the various `LOG_ARCHIVE_DEST_n` parameters that are required. By default, when a `LOG_ARCHIVE_DEST_n` parameter is defined, that location will be used instead of the FRA. To get Oracle to use the FRA when a `LOG_ARCHIVE_DEST_n` parameter is set, you need to define an additional `LOG_ARCHIVE_DEST_n` parameter for the FRA. Typically, this will be `LOG_ARCHIVE_DEST_10`, and you will use the Oracle-supplied constant `USE_DB_RECOVERY_FILE_DEST` to indicate that this destination is the FRA. Here is an example where we configure Oracle to use the FRA and a regular archive log destination directory:

```
alter system set log_archive_dest_10='LOCATION=USE_DB_RECOVERY_FILE_DEST';
alter system set log_archive_dest_1='location=c:\oracle\oraarc\beta1 mandatory';
```

In this example, the ARCH process will now create archived redo logs in both `LOG_ARCHIVE_DEST_1` and `LOG_ARCHIVE_DEST_10`, which is the FRA.

Flash Recovery Area Views

Several views are available to help you manage the FRA. These views include the following:

- `DBA_OUTSTANDING_ALERTS`
- `V$RECOVERY_FILE_DEST`
- `V$FLASH_RECOVERY_AREA_USAGE`

Also, columns are available in several other views that help you to manage the FRA. Let's look at each of these views and columns in more detail.

The `DBA_OUTSTANDING_ALERTS` View As files are added or removed from the FRA, records of these events are logged in the database alert log. You can check the new DBA view, `DBA_OUTSTANDING_ALERTS`, for information on outstanding issues with the FRA. Note that there is somewhat of a lag between the time a space-related issue occurs and when the warning appears in the `DBA_OUTSTANDING_ALERTS` view.

The following is an example where the FRA has run out of space and is posting an alert to the `DBA_OUTSTANDING_ALERTS` view. You would need to deal with this situation quickly or risk

the database coming to a complete halt. In this case, we used the **alter system** command to increase the amount of space allocated to the FRA.

```
SQL> select reason from dba_outstanding_alerts;
REASON
-----
db_recovery_file_dest_size of 524288000 bytes is 100.00% used
and has 0 remaining bytes available.

SQL> alter system set db_recovery_file_dest_size=800m;
```

The V\$RECOVERY_FILE_DEST View The V\$RECOVERY_FILE_DEST view provides an overview of the FRA that is defined in your database. It provides the size that the FRA is configured for, the amount of space used, how much space can be reclaimed, and the number of files in the FRA. In the following example, we can see that the increase in space to the FRA to 800MB has been recorded (SPACE_LIMIT). However, we still have used too much space (SPACE_USED), and the FRA is still full.

```
SQL> select * from v$recovery_file_dest;
NAME
-----
SPACE_LIMIT          SPACE_USED  SPACE_RECLAIMABLE NUMBER_OF_FILES
-----
c:\oracle\product\10.2.0\flash_recovery_area
838,860,800          1,057,116,672  338,081,280          11
```

One nice thing about Oracle is that it manages the FRA space for us as much as it can, and if there is reclaimable space available, it will free it as required. Note that in the previous query, Oracle indicated we were out of FRA space. Did you notice the SPACE_RECLAIMABLE column, though? This column indicates that there is reclaimable space available. This is space that is taken up by archived redo logs or backup set pieces that are no longer needed by virtue of whatever retention criteria we have selected (we will discuss retention criteria and setting those criteria later in this chapter). When Oracle needs space in the FRA (say, for example, we force a log switch), it will remove any files that are reclaimable and free up space. In the next query, we can see that this has occurred. After we ran the previous query that indicated we were out of FRA space, we forced a log switch. This caused Oracle to reclaim space from the FRA for reuse, and it then was able to write out the archived redo log. We can query the V\$RECOVERY_FILE_DEST view and see that this has indeed occurred:

```
SQL> alter system switch logfile;
System altered.
SQL> select * from v$recovery_file_dest;
NAME
-----
SPACE_LIMIT          SPACE_USED  SPACE_RECLAIMABLE NUMBER_OF_FILES
-----
c:\oracle\product\10.2.0\flash_recovery_area
838,860,800          719,412,736   64,000              7
```

The V\$FLASH_RECOVERY_AREA_USAGE View The V\$FLASH_RECOVERY_AREA_USAGE view provides more detailed information on which types of files are occupying space in the FRA. This view groups the file types and then provides the percentage of space that is used by each file type, the percentage of the total FRA reclaimable space that comes from that group, and the number of files in the FRA that come from that group. Here is a query of the V\$FLASH_RECOVERY_AREA_USAGE view:

```
SQL> SELECT * FROM V$FLASH_RECOVERY_AREA_USAGE;
```

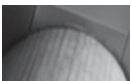
FILE_TYPE	PERCENT_SPACE_USED	PERCENT_SPACE_RECLAIMABLE	NUMBER_OF_FILES
CONTROLFILE	0	0	0
ONLINELOG	0	0	0
ARCHIVELOG	17.14	17.09	7
BACKUPPIECE	108.88	23.22	4
IMAGECOPY	0	0	0
FLASHBACKLOG	0	0	0

In this example, we notice a few things:

- We are over our defined space allocation (the PERCENT_SPACE_USED of all the rows exceeds 100 percent). This is probably because the size of the FRA was recently changed and Oracle has not yet reclaimed enough space to bring the total used below 100 percent.
- The backup set pieces are consuming most of that space, and 23.22 percent of that space is reclaimable.
- The archived redo logs consume only 17 percent of the space allocated to the FRA, and even if we were to remove all of the archived redo logs, we would not free up enough space to bring the FRA under the amount of space allocated to it.

Other Views with FRA Columns The column IS_RECOVERY_DEST_FILE can be found in a number of Oracle Database V\$ views such as V\$CONTROLFILE, V\$LOGFILE, V\$ARCHIVED_LOG, V\$DATAFILE_COPY, V\$DATAFILE, and V\$BACKUP_PIECE. This column is a Boolean that indicates whether the file is in the FRA.

Another column, BYTES, can be found in the V\$BACKUP_PIECE and RC_BACKUP_PIECE (an RMAN recovery catalog view) views. This column indicates the size of the backup set piece in bytes.



NOTE

*Manually removing fixed files from the FRA can have unexpected consequences. Oracle does not immediately detect the removal of these files, and thus the space is not reclaimed. If you end up manually removing files (or lose a disk perhaps), use the RMAN **crosscheck** command along with the **delete** command to cause Oracle to update the current control file information on the FRA. The folks at Oracle recommend that you not manually remove files managed by Oracle if at all possible.*

Other Flash Recovery Area Features

The **alter database add logfile** and **alter database add standby logfile** commands create an online redo log member in the FRA if the OMF-related `DB_CREATE_ONLINE_LOG_DEST_n` parameter is not set. The **alter database drop logfile** and **alter database rename file** commands also support files in the FRA. The nice thing about using these OMF-related features is that Oracle will manage the physical files for you. Thus, if you drop an online redo log group, and the physical files of that group were created by Oracle based on the setting of `DB_CREATE_ONLINE_LOG_DEST_n`, then Oracle will remove those physical files for you.

During database creation, Oracle can use the FRA to store the database control file and online redo logs. If the OMF-related parameter `DB_CREATE_ONLINE_LOG_DEST_n` is defined, then the control file and redo logs will be created in those locations, but will not be created in the FRA, even if the FRA is defined. If `DB_CREATE_ONLINE_LOG_DEST_n` is not defined, but `CREATE_FILE_DEST` is defined, then the control file and online redo logs will be created in the location defined by `CREATE_FILE_DEST`. If `DB_RECOVERY_FILE_DEST` is also defined, then a copy of the control file and online redo logs will get created there as well. The result is a multiplexed online redo log. Finally, if only `DB_RECOVERY_FILE_DEST` is defined, then the control file will get created in that location. If none of these parameters is defined, then the control file and online redo logs will be created to a default location, which is OS specific.

An additional use of the FRA has to do with Flashback Database–related features. We discuss Oracle’s Flashback Database features in more detail in Chapter 15.

The FRA and ASM

RMAN supports the use of Automatic Storage Management (ASM) for the storage of RMAN backups. What is ASM? ASM is a disk management tool that eliminates the need for the DBA to manage the physical files associated with a given database. ASM is somewhat like the logical volume groups you might be used to in Unix. ASM uses *ASM disk groups*, which are logical units of storage. Physical disks are assigned to an ASM disk group, providing the overall storage capability of that ASM disk group. ASM disk groups can exist on previously allocated file systems or on raw disks. Combined with OCFS, clustered servers can share ASM disks in RAC configurations. Having configured ASM and having defined the various disk groups, you can then assign datafiles, control files, online redo logs, and various RMAN backup files to the ASM disk groups.

ASM offers a number of features including load balancing, data redundancy, and easy addition and removal of new disks to the ASM disk groups. It is beyond the scope of this book to discuss configuration of ASM in general. However, be aware that RMAN does support ASM disk groups should you wish to use them. We are not necessarily recommending ASM in this book. Most non-RAC sites probably will find little value in an ASM implementation. However, if you are a RAC site, you might want to consider ASM coupled with OCFS as an alternative to other clustering options, depending on your platform.

If you are using ASM, you can configure the FRA such that it will be created in the ASM file system, as shown in this example:

```
alter system set db_recovery_file_dest='+ASMV01';
```

In this case, Oracle will use the ASM disk volume `ASMV01` for the FRA. We can then use RMAN to back up to the FRA. We will discuss backups in Chapter 11.

Should You Use the FRA?

We think the idea behind the FRA is a good one. We also like to copy those backups to some other media, such as tape, so we can send them offsite for disaster recovery purposes (nothing like a good flood, bomb, or tornado to make your disaster recovery planning seem really important).

We also like the FRA for the archived redo logs, but we also like the idea of copying archived redo logs to more than one location (and more specifically, to more than one disk). Keep in mind that the archived redo logs are critical to database recovery, and if you lose one, all the others after that one are pretty much worthless. So, we tend to configure our databases using FRA and at least one other archive log destination that is on a different disk. This means that we use the `LOG_ARCHIVE_DEST_n` parameters to configure the database to use both the FRA and another, separate file system to store our archived redo logs.

Another benefit of the FRA we like is the implementation of space quotas. Many database servers these days run more than one database. We have seen cases where one database has consumed all of the physical disk space with archived redo logs. This caused problems not only for the database that filled up the archived redo log destination directory, but also for all of the other databases on the system. By using a quota system, you can limit one database's ability to impact others.

We could go beyond this and tell you how much we like things such as standby databases and the like, but that's not what this book is about. The bottom line is that you need to protect the data in your charge, because there is no worse feeling than coming into work on Monday morning and finding out that the system crashed over the weekend and that the entire database is lost...along with all your backups.

Switching Between ARCHIVELOG Modes

Once you have configured the database to run in ARCHIVELOG mode, you can switch it between NOARCHIVELOG and ARCHIVELOG mode quite easily. To put the database in ARCHIVELOG mode, you must first shut down the database in a consistent state using one of these commands: **shutdown**, **shutdown immediate**, or **shutdown transactional**. Once the database has been cleanly shut down, mount the database by issuing the **startup mount** command. Once the database is mounted, issue the command **alter database archivelog** to put the database in ARCHIVELOG mode. You can then open the database with the **alter database open** command.

If you wish to take the database out of ARCHIVELOG mode, reverse the process. First shut down the database. Once the database has been shut down, mount the database by issuing the **startup mount** command. Once the database is mounted, issue the command **alter database noarchivelog** to put the database in NOARCHIVELOG mode. You can then open the database with the **alter database open** command.

If You Created Your Database with the Oracle Database Configuration Assistant

If you created your database with the Oracle Database Configuration Assistant (ODBCA), it is likely that Oracle has configured much of RMAN for you. ODBCAs will configure the database in ARCHIVELOG mode, configure the FRA, and even offer you the chance to schedule RMAN

backups. For smaller installations, this may well be all that is needed, and you will not need to worry about any other basic RMAN configuration issues. Still, it's a good idea to be aware of all the options that RMAN offers. For example, encryption of backups is not enabled when you create a database with the ODBCA, and you might want to enable that feature.

RMAN Workshop: *Put the Database in ARCHIVELOG Mode*

Workshop Notes

For this workshop, you need an installation of the Oracle software, and a database that is up and running in NOARCHIVELOG mode. Before you start the workshop, determine where you want the flash recovery area to reside. You will also need to decide where a second archive log destination directory will be, as this workshop will have you archiving to two locations.

Step 1. Configure both the FRA and a separate archive log destination for the archived redo logs. First, set the FRA parameters `DB_RECOVERY_FILE_DEST_SIZE` and `DB_RECOVERY_FILE_DEST`:

```
SQL> alter system set db_recovery_file_dest_size=2G;
System altered.
SQL> alter system set
db_recovery_file_dest='c:\oracle\product\10.2.0\flash_recovery_area';
System altered.
```

Step 2. Now, define two archive log destination directories, one of which will be the FRA. Set the database parameter file, and set the `LOG_ARCHIVE_DEST_1` parameter so that it is pointing to a predefined file system that will be our first archive log directory. Since we are configuring `LOG_ARCHIVE_DEST_1` and we want to use the FRA, we need to set the `LOG_ARCHIVE_DEST_10` parameter to point to the FRA by using the parameter `USE_DB_RECOVERY_FILE_DEST`. Use the **show parameter** command to verify that the settings are correct:

```
SQL> alter system set log_archive_dest_1='location=d:\archive\rob10R2';
System altered.
SQL> alter system set
log_archive_dest_10='LOCATION=USE_DB_RECOVERY_FILE_DEST';
SQL> show parameter log_archive_dest
NAME                                TYPE                                VALUE
-----                                -
log_archive_dest_1                  string                               location=d:\archive\rob10R2
log_archive_dest_10                 string                               LOCATION=USE_DB_RECOVERY_FILE_DEST
```

Step 3. Shut down the database:

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Step 4. Mount the database:

```
SQL> startup mount
ORACLE instance started.
Total System Global Area      84700976 bytes
Fixed Size                    282416 bytes
Variable Size                  71303168 bytes
Database Buffers              12582912 bytes
Redo Buffers                   532480 bytes
Database mounted.
```

Step 5. Put the database in ARCHIVELOG mode:

```
SQL> alter database archivelog ;
Database altered.
```

Step 6. Open the database:

```
SQL> alter database open;
Database altered.
```

Although it is not part of the workshop, the process of taking the database out of ARCHIVELOG mode is as simple as reversing the process described in the workshop. Shut down the database, restart the database instance by issuing the **startup mount** command, and put the database in NOARCHIVELOG mode by issuing the command **alter database noarchivelog**. Note that you are not required to shut down the database in a consistent manner when moving from ARCHIVELOG mode to NOARCHIVELOG mode. Here is an example of switching back into NOARCHIVELOG mode:

```
SQL> shutdown
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.
Total System Global Area      84700976 bytes
Fixed Size                    282416 bytes
Variable Size                  71303168 bytes
Database Buffers              12582912 bytes
Redo Buffers                   532480 bytes
Database mounted.
SQL> alter database noarchivelog;
Database altered.
SQL> alter database open;
Database altered.
```

Finally, you should do a backup of the database once you have completed either task.

The Oracle Database 11g Fault Diagnosability Infrastructure

One of the major new features in Oracle Database 11g is the new Fault Diagnosability Infrastructure. We will cover various features associated with the new Fault Diagnosability throughout this book. This infrastructure is designed to help prevent, detect, diagnose, and resolve

problems such as database bugs and various forms of corruption. This new infrastructure changes some things, such as where the alert log is generated, and adds a great deal of new functionality to the Oracle Database. Throughout this text we will discuss the Fault Diagnosability Infrastructure in more detail. In Chapter x, we will discuss the Support Workbench, which provides for automated responses to database problems. Chapter 13 will also discuss the health checkers, another new component associated with the Oracle Automatic Diagnostic Repository (ADR). In Chapters 12 and 13, we will talk about using the Recovery Advisor, closely linked to the ADR, during database restores.

For the purposes of setting up the Fault Diagnosability Infrastructure for an Oracle Database, what we are concerned with in this chapter is the setting of the new parameter `DIAGNOSTIC_DEST`. The new `DIAGNOSTIC_DEST` parameter defines the root of the ADR and deprecates several parameters including `USER_DUMP_DEST`, `CORE_DUMP_DEST`, and `BACKGROUND_DUMP_DEST`. As a result, if you create a new Oracle Database 11g database with the DBCA, you will not find the alert log or user trace files where you previously would have expected them.

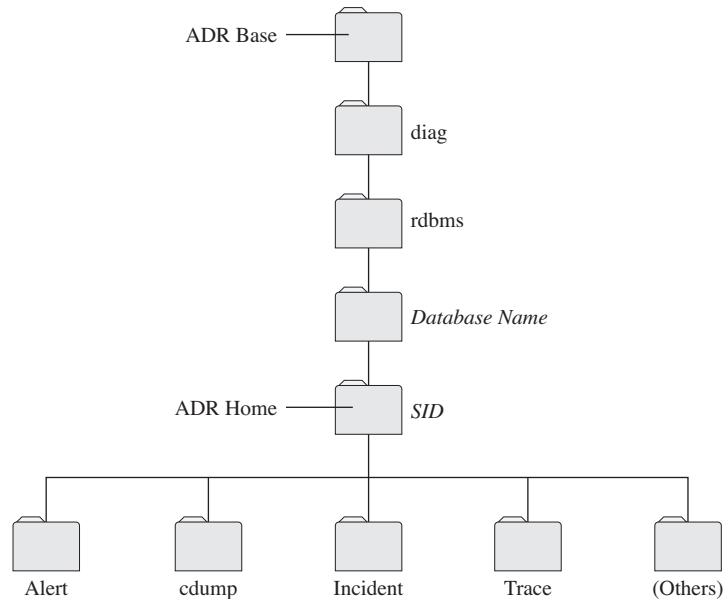
By default, the `DIAGNOSTIC_DEST` parameter is set to `$ORACLE_BASE`. If `$ORACLE_BASE` is not set, then it is set to the value of `$ORACLE_HOME`. The root directory of the ADR directory structure starts with a directory called `diag`, under which is a subdirectory that references the product type. For example, for the database, the product is called `rdbms`. Under `rdbms` is a directory for each database, followed by a directory for each individual instance.

For example, if `$ORACLE_BASE` is `/u01/oracle`, the database name is `mydb`, and the database instance is `mydb1`, then the structure of the ADR directory for that database will be `/u01/oracle/diag/rdbms/mydb/mydb1`. This directory structure is called the ADR home, and each instance has its own ADR home. If you are using RAC, you can use shared storage for ADR, or individual storage on each node. We would recommend shared storage in a RAC environment since you can see the aggregate diagnostic data from any node. Also, a shared ADR allows for more robust recovery options for the Data Recovery Advisor.

Under this directory structure will be a number of other directories. Some of the most common directories include the following:

- **Alert** This is the location of the XML-formatted alert log.
- **cdump** This is the location of the core dumps for the database.
- **Trace** This contains trace files generated by the system, as well as a text copy of the alert log.
- **Incident** This directory contains multiple subdirectories, one for each incident.

Here is diagram of the ADR Base structure:



A new view, `V$DIAG_INFO`, provides information on the various ADR locations, as well as information related to ADR, such as active incidents. Here is an example of a query against the `V$DIAG_INFO` view:

```
SQL> select * from v$diag_info;
```

INST_ID	NAME	VALUE
1	Diag Enabled	TRUE
1	ADR Base	C:\ORACLE\PRODUCT
1	ADR Home	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4
1	Diag Trace	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4\trace
1	Diag Alert	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4>alert
1	Diag Incident	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4\incident
1	Diag Cdump	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4\cdump
1	Health Monitor	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4\hm
1	Default Trace File	C:\ORACLE\PRODUCT\diag\rdbms\rob11gr4\rob11gr4\trace\rob11gr4_ora_7832.trc
1	Active Problem Count	1
1	Active Incident Count	1

11 rows selected.

The RMAN Command Line

Now that the database is in ARCHIVELOG mode (if you are going to do online backups), you are ready to configure RMAN and your database for backups. Before you can do that, it would be nice to actually know how to use the RMAN executable. So, let's take a slight detour in our setup discussion to look at the RMAN command-line interface (CLI) and how to use it.

There are two different ways to get to RMAN. The first is from the command line, and the second is by using OEM (Oracle Enterprise Manager). We will deal with the OEM interface in more detail in Chapter 11. Most of the examples you will see in this book, however, will be done using the CLI. We figure that if you can do it from the command line, you can do it from anywhere. In the next sections, we will look at how to connect to databases with the RMAN command line and also how to use the **connect** command.

Connecting via the RMAN Command Line

You can start RMAN from the OS prompt simply by typing the command **rman**. Once you have started the RMAN command interpreter, you can perform whatever operations you might need to perform. Often, it's much easier to get some of the preliminary work done by using command-line parameters. Thus, when we start RMAN, we can pass several command-line parameters. You can use the command-line parameters to connect RMAN to the database you are going to back up (known as the *target database*), to the recovery catalog, or for a number of other tasks. Table 3-2 provides a list of the command-line parameters, the data type for the argument of the parameter (if there is one), and the purpose of the parameter.

RMAN Command-Line Parameter	Parameter Argument Type	Purpose
target	Character string	Defines the username, password, and service name of the target database to connect to.
catalog	Character string	Defines the username, password, and service name of the recovery catalog.
nocatalog	No arguments	Indicates that no recovery catalog is going to be used by this session. This parameter is the default parameter in Oracle8i and Oracle9i.
cmdfile	Character string	Indicates the name of a command file script to execute.
log	Character string	Indicates that the RMAN session should be logged. The log file will take the name of the argument to this parameter. Also causes all RMAN messages to the screen to be suppressed (except the RMAN prompt).
trace	Character string	Indicates that the RMAN session should be traced. The trace file will take the name of the argument to this parameter.

TABLE 3-2 RMAN Command-Line Parameters

RMAN Command-Line Parameter	Parameter Argument Type	Purpose
append	No arguments	Indicates that the log file (defined by the log parameter) should be appended to.
debug	Various arguments	Indicates that RMAN should be started in debug mode.
msgno	No arguments	Indicates that the RMAN- prefix should be shown with each error message. If this option is not selected, then certain non-error messages will not include a message number with them.
send	Character string	Sends the character string message to the media management layer.
pipe	String	Invokes the RMAN pipe interface.
timeout	Integer	Indicates the number of seconds to wait for pipe input.
auxiliary	Character string	Defines the username, password, and service name of the auxiliary database to connect to.
checksyntax	None	Checks the command file listed for syntax errors.
slxdebug	None	Checks for command line and RMAN prompt parsing errors.

TABLE 3-2 RMAN Command-Line Parameters (continued)

Here are some examples of starting RMAN with some command-line parameters (and you will see others later):

```

RMAN target=system/manager@robt nocatalog
RMAN target='sys/robert as sysdba@robt' nocatalog
RMAN target=system/manager@robt
catalog=system/manager@catalog log="RMAN.log"
RMAN target system/manager@robt nocatalog log "RMAN.log"

```

NOTE

The = sign between the command-line parameter and the value of that parameter is optional. Also, if you are running Oracle Database 11g Real Application Clusters, you can connect to only one instance of that cluster.

Note that RMAN *always* connects as SYSDBA to the target database. This is good to know because it implies that the account we connect to has to have the SYSDBA privileges.

If you forget the command-line arguments to RMAN (and somehow manage to leave this book and your documentation at home), there is a way to get RMAN to display the valid command-line parameters. Simply start RMAN with an invalid parameter. As you can see in

the following example, RMAN will return an error, but will also provide you with a list of valid command-line parameters (we removed some of the errors at the bottom of the listing for brevity):

```
C:\Documents and Settings\Robert>rman help
Argument      Value          Description
-----
target        quoted-string  connect-string for target database
catalog       quoted-string  connect-string for recovery catalog
nocatalog     none          if specified, then no recovery catalog
cmdfile       quoted-string  name of input command file
log           quoted-string  name of output message log file
trace         quoted-string  name of output debugging message log file
append        none          if specified, log is opened in append mode
debug         optional-args  activate debugging
msgno         none          show RMAN-nnnn prefix for all messages
send          quoted-string  send a command to the media manager
pipe          string         building block for pipe names
timeout       integer       number of seconds to wait for pipe input
checksyntax   none          check the command file for syntax errors
-----
```

Both single and double quotes (' or ") are accepted for a quoted-string. Quotes are not required unless the string contains embedded white-space.

RMAN offers the **checksyntax** parameter, which allows you to check the RMAN commands you want to issue for errors. Here is an example of the use of the **checksyntax** parameter:

```
C:\Documents and Settings\Robert>rman checksyntax
Recovery Manager: Release 11.2.0.1.0 - Production on Thu Nov 5 04:03:03 2009
Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.
RMAN> backup database pls archivelog;
RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-00558: error encountered while parsing input commands
RMAN-01009: syntax error: found "identifier": expecting one of: "archivelog,
auxiliary, backupset, backup, channel, controlfilecopy, copy, current, database,
datafilecopy, datafile, db_recovery_file_dest, delete, diskratio, filesperset,
force, format, from, include, keep, maxsetsize, noexclude, nokeep, not, plus,
pool, recovery, reuse, section, skip readonly, skip, spfile, tablespace, tag, to,
(, ;"
RMAN-01008: the bad identifier was: pls
RMAN-01007: at line 1 column 17 file: standard input

RMAN> backup database plus archivelog;
The command has no syntax errors
```

Note that a lot can be divined from RMAN error messages. Often, within the message, you can see that RMAN was expecting a particular keyword or phrase.

RMAN Client Compatibility

When using the RMAN client, you will want to consider the compatibility of that client with the target database that you are connecting to. You will also need to consider the compatibility of the recovery catalog, which we will discuss in more detail in Chapter 9. In general, the version of the RMAN client should be the same or higher than the version of the target database that you will be connecting to. Here is a table that provides guidelines on RMAN compatibility between the target and auxiliary databases and the RMAN client:

Target/Auxiliary Database	Client Version Requirement
8.0.6	8.0.6 only
8.1.7	8.0.6.1 or 8.1.7 only
8.1.7.4	8.1.7.4 only
9.0.1	9.0.1 only
9.2.0	>= 9.0.1.3 and <= target database executable version
10.1.0	>= 9.0.1.3 and <= target database executable version
10.2.0	>= 9.0.1.3 and <= target database executable version
11.1.0	>= 9.0.1.3 and <= target database executable version
11.2.0	>= 9.0.1.3 and <= target database executable version

RMAN is stored in the \$ORACLE_HOME/bin directory, and this directory should be in the PATH on the OS Oracle is running on. If you have several ORACLE_HOME directories, then you will want to be cautious. Make sure that the OS PATH is pointing to the correct ORACLE_HOME before you start RMAN. If you do not set the PATH correctly, you could be using the wrong ORACLE_HOME directory. Also be cautious that there is not some other rman executable in the path before the RMAN executable. For example, an rman command in some versions of Unix sometimes ends up running instead of RMAN because it comes first in the path. In cases like this, you will need to adjust the path, or you may need to change to the \$ORACLE_HOME/bin directory and run rman directly from that location.

Typically, RMAN will generate an error if you are using an incompatible client version. You can use OS-level utilities (such as oraenv) to determine if you are using the correct ORACLE_HOME, or you can check the banner of the RMAN client.

Using the RMAN connect Command

If you start RMAN and realize that you either have not connected to the correct database or wish to connect to a different database (target, catalog, or auxiliary), you can use the **connect** command to change which database RMAN is connected to. To change to another target database, use the **connect target** command. To change to a different recovery catalog, use the **connect catalog** command. To connect to a different auxiliary database, use the **connect auxiliary** command. Here are some examples of the use of the **connect** command:

```
connect target sys/password@testdb;
connect catalog rcat_user/password@rodb;
```

Exiting the RMAN Client

When you are done with RMAN, it's time to get out of the client. RMAN offers two commands, **quit** and **exit**. These commands will return you to the OS prompt. RMAN also allows you to shell out to the OS with the **host** command. Here are some examples:

```
C:\>rman target=/
Recovery Manager: Release 11.2.0.1.0-Production on Wed Oct 4 22:49:14 2006
Copyright (c) 1982, 2005, Oracle. All rights reserved.
connected to target database: ORCL2 (DBID=582838926)

RMAN> host;
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>exit
host command complete
RMAN> exit
Recovery Manager complete.
```

Configuring the Database for RMAN Operations

Now that you know how to start RMAN, we need to deal with some configuration issues. While it is possible to just fire up RMAN and do a backup, it's a better idea to deal with some configuration questions before you do so. First, you need to set up the database user that RMAN will be using. Next, you can configure RMAN to use several settings by default, so we will look at those settings as well.

Setting Up the Database User

By default, you can use RMAN with the SYS account (as sysdba) without any configuration required. Of course, that's probably not the best account to use when you are doing production backups. We recommend, before you use RMAN to do a backup, that you create a separate account that is designated for RMAN backups. The following workshop helps you do just that.

RMAN Workshop: Create the Target Database RMAN Backup Account

Workshop Notes

For this workshop, you need an installation of the Oracle software and a database that is up and running. You need administrative privileges on this database.

Step 1. Determine the user account name that you want to use, and create it with the database **create user** command:

```
CREATE USER backup_admin IDENTIFIED BY backupuserpassword
DEFAULT TABLESPACE users;
```

Step 2. Grant the sysdba privilege to the BACKUP_ADMIN user. You need to grant this privilege because RMAN always connects to the database by using the sysdba login. Here is an example of granting the sysdba privilege to the BACKUP_ADMIN account:

```
GRANT sysdba TO backup_admin;
```

**NOTE**

*If you created your database with the **dbca**, you were offered an option to set up automated daily backups. If you selected this option, Oracle will do some initial RMAN configuration for you (it will configure the FRA, for example). While this RMAN configuration is sufficient for databases that are not of consequence, if you are managing databases that are mission critical, you should still follow the steps outlined in this chapter and ensure that your database is properly configured for RMAN operations.*

So, what happens if you try to connect RMAN to an account that is not properly created? The following error will occur:

```
D:\oracle\oradata\robt>RMAN target=backup/backup@robt
Recovery Manager: Release 11.2.0.1.0
Production on Tue Aug 22 21:40:51 2006
Copyright (c) 1982, 2005, Oracle. All rights reserved.
RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-00554: initialization of internal recovery manager package failed
RMAN-04005: error from target database:
ORA-01031: insufficient privileges
```

Now that we have created the user and granted it the privileges it will need, we are a step closer to being ready to use RMAN. Still, we have some RMAN default settings we need to configure, so let's look at those next.

Setting Up Database Security

We need to discuss briefly the differences between connecting to RMAN on the local server and connecting to it via Oracle Net. When you start RMAN, you might be logged onto the same server as the database. In this case, if you are logged on using a privileged OS user account, you do not need to do anything beyond the two steps in the preceding RMAN Workshop. How do you know whether your user account is a privileged one? It depends on the OS you are using. If you are using Unix, there is generally a Unix group called `dba` (though it may be called something else) that is created when the Oracle-owning account (usually called Oracle) is created. If your Unix user account is assigned to this group, then you will be able to connect to a target database without any additional work. If you are using Windows platforms, then the privileged users are assigned to an NT group, generally called `ORA_DBA`.

If you are not logging onto the local server using a privileged account, or if you are connecting to the target database using Oracle Net from a client workstation (for example, you are connecting using `system/manager@testdb`), then you need to configure your database to use a password file. To do so, you first need to create the password file, and then need to configure the database so that it knows to use it. Let's look at each of these steps in detail.

Create the Password File

To create the database password file, you use the Oracle utility **orapwd**. This command takes three parameters:

- **file** The password filename

- **password** The password for the sys user
- **entries** Any number of entries to reserve for additional privileged Oracle user accounts

By default, the Oracle database (on NT) will expect the password file to take on the naming standard `PWDsid.ora`, where *sid* is your database name. Here is an example of the creation of a password file:

```
orapwd file=PWDrobt.ora password=robert entries=20
```

So, now that we have created the password file, we need to configure the database to use it, and thus to allow us to do remote backups via Oracle Net.

Configure the Database to Use the Password File

By default, an Oracle database is not configured to use the password file (unless you have used the ODBC to create your database). To configure the database, edit the parameter file (`init.ora`) in your favorite editor. The parameter we are interested in is `REMOTE_LOGIN_PASSWORDFILE`. This parameter can be set to one of three values in Oracle Database 11g:

- **none** The default value. In this case, Oracle will ignore the password file, and only local privileged logins will be recognized for `sysdba` access.
- **shared** This parameter indicates that multiple databases can use the same password file. When in this mode, only the `SYS` user account password can be stored.
- **exclusive** This parameter indicates that the password file is used by only one database. In this mode, the password file can contain passwords for several privileged Oracle accounts. This is the recommend mode of operation, particularly when running RMAN. If you wish to connect RMAN to your database from a remote client, you must use this parameter setting.

If you are using an SPFILE instead of a text-based parameter file, then use the **alter system** command to modify this parameter setting:

```
alter system set REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE scope=spfile;
```

Finally, the `REMOTE_LOGIN_PASSWORDFILE` parameter is not dynamic, so you cannot change it with the database up and running. Instead you will have to change the SPFILE (using the **scope=spfile** parameter of the **alter system** command) and then shut down the database and restart it.

Setting the CONTROL_FILE_RECORD_KEEP_TIME Parameter

When configuring your database for RMAN, you should consider how long you wish backup records to be stored in the control file. This includes records of full database backups and of specific datafile, control file, parameter file, and archive log backups. The database parameter `CONTROL_FILE_RECORD_KEEP_TIME` is defined in days (the default is 7). Thus, by default, Oracle will maintain RMAN backup and recovery records for seven days. You can set this parameter to any value between 0 and 365 days.

This parameter can have a number of operational database impacts. First, it directly impacts the size of the database control file, because as RMAN backups occur, records relating to these backups are stored in the control file. As records are saved in the control file, the control file might well run out of space. In this case, Oracle will expand the control file to accommodate the storage of the required number of backup records. Setting this parameter to **0** will disallow any control file growth, but has the negative effect of making the RMAN backup history retention period uncertain.

We suggest that you set `CONTROL_FILE_RECORD_KEEP_TIME` to a value no less than your selected database backup retention period. Otherwise, you risk having database backups available on your backup media without related backup records available in the control file. This can cause serious complications if you need to recover these older backups for some reason!



CAUTION

There are a number of places where incorrectly set file retention can cause your backup's retention strategy to fail. These include incorrectly setting `CONTROL_FILE_RECORD_KEEP_TIME`, RMAN retention policies, or retention policies on your tape vendor products. Make sure all retention policies are aligned so you don't wake up someday and find you are unable to restore your backups.

Configuring RMAN Default Settings

RMAN allows you to perform automated database backup and recovery, as you will see in later chapters. To support this feature, RMAN allows you to define default values for a number of settings, such as channel configuration. In this section, we look at the configuration of default RMAN settings. Of course, if you can configure something, you will want to be able to change that configuration, and even to remove it completely if required. We will look at that, too. So, what will be the benefit of all of this configuration work? It will make the process of actually doing backups much easier in the end. First, we will quickly examine the **configure** command in RMAN and all that it provides us. Then, we will look at several of the different defaults you might want to configure by using the **configure** command.

Throughout this section, we use a number of terms that you might not yet be familiar with because they are covered in later chapters. Many of the terms were introduced in Chapter 2, though others may not be quite clear to you yet. That's okay, because to use RMAN, none of the default configuration options are really required. We suggest that you skim this section to get a feel for the various default values that you can set, and then, after you have read later chapters, return here and reread this section. At that point, you will be ready to decide what defaults you want to apply to your Oracle database.

Introducing the **configure** Command

RMAN provides the **configure** command, which allows you to define default values to be applied when executing backup and recovery sessions. Using the **configure** command, RMAN allows you to make changes to the default values of the various parameters that are persistent until cleared or changed again. The ability to customize default configuration settings allows you to execute

automated RMAN operations. The following are several of the different settings that you can configure:

- A default device type, such as disk or SBT (system backup tape), to use for RMAN jobs.
- The number of channels that are automatically allocated when performing automated backup and restore jobs.
- A tablespace exclusion policy to configure specific tablespaces to be excluded during full database backup operations.
- The maximum size for any given backup piece and the size of any backup set when doing an automated backup.
- Backup optimization to default to ON or OFF. Backup optimization eliminates duplicate backups of identical datafiles (for example, those associated with read-only tablespaces) and archived redo logs.
- The default filename for the snapshot control file (refer to Chapter 2 for more details on the snapshot control file).
- The default for automated backups of the control file to ON or OFF, as well as the default format for the control file backup output files and the default device on which to create these backups.
- The default filenames for files of an auxiliary database.
- A default retention policy, which determines which backups and copies are eligible for deletion because they are no longer needed.
- The default encryption value and the associated encryption algorithm.
- The default compression algorithm to use if compression is to be used.
- A deletion policy for archived redo logs.

Each configurable setting has a default value assigned to it. The defaults are stored in the database control file (as are any configured values). This is true even if you are connecting to a recovery catalog. You can see the currently configured values for the various RMAN parameters by using the **show** command. Any nondefault RMAN-configured settings are also listed in the `V$RMAN_CONFIGURATION` database view. Here are some examples of the **show** command's use:

```
show default device type;
show maxsets size;
show retention policy;
show all;
```

Configuring Various RMAN Default Settings

This section looks at setting RMAN defaults. First, let's look at configuration of channel default settings. You can configure channels in different ways. You can configure defaults for all channels with the **configure channel device type** command, or configure defaults for specific default channels with the **configure channel *n* device type** command.

You can clear channel defaults for all channels with the **configure channel device type clear** command, and clear channel defaults for specific default channels with the **configure channel n device type clear** command.

When you allocate a channel with the **allocate channel** command, you can specify the assigned names to the channels that you allocate. For example, the **allocate channel d1 device type disk** command will create a channel called d1. When automated channels are allocated, Oracle assigns default names to these channels. These default names depend on the type of default device used. The following table provides an example of the default name format that will be used.

Device Type	Default Name Format	Example
Disk	ORA_DISK_#	ORA_DISK_1 ORA_DISK_2
Tape	ORA_SBT_TAPE_#	ORA_SBT_TAPE_1 ORA_SBT_TAPE_2

The number of channels that are automatically allocated depends on the default level of parallelism defined (which we will discuss later in this chapter).

When you issue the **configure** command, Oracle displays the previous configuration settings, followed by the new configuration setting. Now, let's look at some of the ways that you can use the **configure** command to automate the backup and restore process with RMAN.

Examples of Using the configure Command

This section presents some examples of using the **configure** command to define default values. In this section, we cover a number of topics revolving around the **configure** command, including:

- Configuring channel default settings
- Using the format string
- Configuring default automated backups of the control file and the SPFILE
- Configuring default retention policies
- Configuring default levels of encryption
- Configuring archive log deletion policies

Configuring Channel Default Settings

Let's start with an example of configuring the default backup/restore device to tape or to disk. In this case, all channels assigned to backups will be allocated to disk:

```
CONFIGURE DEFAULT DEVICE TYPE TO SBT;
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
```

When default device types are configured, Oracle will use that default channel unless you override the default using the **backup device type** parameter. Maintenance channels for **delete** commands and auxiliary channels for duplicate operations will also be automatically allocated.

Once we have configured a default device type, we can configure defaults for the specific type of backup that should occur when that device is used. For example, when doing backups to disk, we can opt to have Oracle back up the database by default using the standard Oracle backup set methodology, or we can have it default to using copies (which can only go to disk). You can also indicate that backup sets should be compressed by default and indicate the degree of parallelism (which represents the number of channels that will be allocated for that backup). Here are examples of configuring for these different options:

```
CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET;
CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET;
CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY;
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
```

One word about compression, which was a new feature of RMAN in Oracle Database 10g. Compression provides real compression of your Oracle backup sets, not unlike zip compression. This can make your backup sets much smaller. Of course, the compression itself consumes resources and will make the backups take longer to complete or restore.

Now, let's look at an example of configuring the number of channels to be allocated during an automated backup or recovery operation. Also in this example, we have set the default level of parallelism for disk operations to two. Thus, if you start an automated backup, two channels will be allocated to perform the backup in parallel.

```
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT 'd:\backup\robt\backup_%U';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT 'e:\backup\robt\backup_%U';
```

NOTE

Generally, when setting the default level of parallelism, you should set it to the number of tape drives you will be backing up to. When using disks, some trial and error might be called for. Since disks have multiple heads and may be striped, it may be that multiple channels will result in better throughput. Test parallelism to your disks and act accordingly on the results.

Several options are available when configuring channels. With the **maxpiecesize** parameter, you can control the size of a backup set piece. You can control the maximum number of files that RMAN can open at one time with the **maxopenfiles** parameter. The **rate** parameter allows you to throttle RMAN and to control the rate at which a backup occurs in either bytes, kilobytes, megabytes, or gigabytes per second.

In this example, we put all these options to use. We limit channel 1 to creating each individual backup piece at a maximum size of 100MB, and we limit RMAN to opening a maximum of eight files on this channel. Finally, we have constrained the channel such that it cannot have a throughput of more than 100MB.

```
CONFIGURE CHANNEL 1 DEVICE TYPE DISK MAXPIECESIZE 100m maxopenfiles 8
rate 100MB;
```

**NOTE**

*Don't get confused about the difference between the **maxpiecesize** parameter and the **maxsetsize** parameter: **maxpiecesize** limits the size of the individual backup set pieces and has no impact on the overall cumulative size of the backup. The **maxsetsize** parameter, on the other hand, can and will limit the overall size of your backup, so use it carefully!*

If we had wished to limit all channels, we could have issued the command slightly differently:

```
CONFIGURE CHANNEL DEVICE TYPE DISK MAXPIECESIZE 100m;
```

So, why might we want to change the maximum size that a given backup set piece can be? First, we might have some specific file size limitations that we have to deal with. Tapes can only handle so much data, and some disk file systems have limits on how large a given datafile can be.

We might also want to set a tape device as the default device for all channels, along with some specific parameter settings. In this case, our **configure** command might look like this:

```
-- Note that we could have used the = sign after the PARMS clause if
-- we preferred like this:
-- PARMS='ENV=(NB_ORA_CLASS=RMAN_rs100_tape)'.
-- This is true with many parameters.
CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 100m
PARMS 'ENV=(NB_ORA_CLASS=RMAN_rs100_tape)';
```

When using the **configure** command, you may find that you need to clear a given configuration so that you can use the default. To do this, use the **configure** command with the **clear** option. In this example, we are clearing out the default options set for default channel 1:

```
CONFIGURE CHANNEL 1 DEVICE TYPE DISK CLEAR;
```

Configuring Backup Set–Related Settings

You may wish to configure a default maximum size for an entire backup set, in which case you would use this slightly modified syntax (it is followed by an example of resetting this value back to the default, which is unlimited):

```
CONFIGURE MAXSETSIZE TO 7500K;
CONFIGURE MAXSETSIZE CLEAR;
```

**CAUTION**

*Be careful when using **maxsetsize** to limit the size of the entire backup that is being created. While your database might be smaller than the **maxsetsize** defined initially, it could quickly grow beyond the **maxsetsize**, causing your database backups to fail.*

As you will see in later chapters, you can configure the backup process to create duplexed backups; in other words, multiple copies of the backup can be created at different locations. You can also configure database default settings such that automatic backups will be duplexed using the **configure** command. Here is an example where we have defined that all backups to disk by default will be duplexed, with two copies:

```
configure datafile backup copies for device type disk to 2;
```

You may wish to exclude specific tablespaces during an automated backup, which Oracle allows you to do with the **configure** command. Here is an example of excluding a tablespace by default:

```
configure exclude for tablespace old_data;
```

The **configure** command allows you to enable or disable backup optimization. When enabled, *backup optimization* will cause Oracle to skip backups of files that already have identical backups on the device being backed up to. Here is an example of configuring backup optimization:

```
configure backup optimization on;
```

Note that for optimization to occur, you must have enabled it. In addition, you must issue the **backup database** or **backup archivelog** command with the **like** or **all** option. Alternatively, you can use the **backup backupset all** command (more information on these types of backups is provided in later chapters). Finally, you can disable the setting for backup optimization by using the **force** parameter of the **backup** command.

Configuring Snapshot Control File Settings

We discussed the snapshot control file in Chapter 2. This file is a point-in-time copy of the database control file that is taken during RMAN backup operations. The snapshot control file ensures that the backup is consistent to a given point in time. Thus, if you add a tablespace or datafile to a database after the backup has started (assuming an online backup, of course), that tablespace or datafile will not be included in the backup. Sometimes it is desirable to have RMAN create the backup control file in a location other than the default location. In this event, you can use the **configure** command to define a new default location for the snapshot control file:

```
configure snapshot controlfile name to 'd:\oracle\backup\scontrolf_mydb';
```

Note that Oracle does not create the snapshot control file in the FRA even if the FRA is configured. Also note in this example, we include the name of the database (or database instance if running RAC) to ensure the snapshot control filenames are unique.

Using the Format String

Note in previous examples that in several places we defined one or more disk locations and filename formats. This is known as the *format string specification*. You will see the format string specification used a great deal in this book, and you will often use it when working with RMAN unless you are using the FRA. The FRA uses Oracle's own file-naming conventions, so using a format string when backing up to the FRA is not recommended or required (and can cause problems with file maintenance). Since the FRA is the default location for backups, there is no need to configure a backup device to point to the FRA. You may need to configure channels for other reasons, but do not configure them such that they have a format string pointing to the FRA.

The format string is platform independent (though directory structures will be platform specific). A format string on Windows will look pretty much the same on Unix or on any other platform. For example, if we were using a Unix system, our format string might look like this:

```
CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT
'/u01/opt/oracle/backup/robt/backup_%U';
CONFIGURE CHANNEL 2 DEVICE TYPE DISK FORMAT
'/u01/opt/oracle/backup/robt/backup_%U';
```

NOTE

*Oracle will not manage your backup files if you use the **format** parameter, even if you are backing up to the FRA, because the backup is not managed by Oracle. If the **format** parameter is used, then the retention policy will have to remove the formatted backups. If **format** is not used, then OMF names are used, and the files are created in the FRA. Do not use the **format** option when backing up to the FRA.*

The format string is used a lot in the **configure** command. You will also see it in other RMAN commands such as the **backup**, **restore**, and **allocate channel** commands. RMAN offers several *syntax elements* associated with the format string specification. These elements are placeholders that will cause RMAN to replace the format string with the associated defined values. For example, the %U syntax element in the previous example tells RMAN to substitute a system-generated unique identifier for the filename. %U then keeps each backup filename unique. Table 3-3 lists the valid syntax elements and gives a quick description of their use.

Element	Description
%a	Indicates that the activation ID of the database should be substituted.
%b	Specifies the filename without any directory paths. This can only be used with the set newname command or for creating a backup using image copies.
%c	Specifies that the copy number of the backup piece within a set of duplexed backup pieces, with a maximum value of 256, should be substituted. This number will be 1 for nonduplexed backup sets and 0 for proxy copies.
%d	Indicates that the name of the database should be substituted.
%D	Indicates that the current day of the month from the Gregorian calendar in the format DD should be substituted.
%e	Indicates that the archived log sequence number should be substituted.
%f	Indicates that the absolute file number should be substituted.
%F	Provides a unique and repeatable name that combines the database ID (DBID), day, month, year, and sequence.
%h	Indicates that the archived redo log thread number should be substituted.
%l	Indicates that the DBID should be substituted.

TABLE 3-3 *Format String Specification Descriptions*

Element	Description
%M	Indicates that the month in the Gregorian calendar in the format MM should be substituted.
%N	Indicates that the tablespace name should be substituted.
%n	Indicates that the name of the database, padded on the right with x characters to a total length of eight characters, should be substituted. For example, if ROBDB is the database name, then the padded name is ROBDBxxx.
%p	Indicates that the piece number within the backup set should be substituted. This value starts at 1 for each backup set and is incremented by 1 as each backup piece is created.
%s	Indicates that the backup set number should be substituted. This number is a counter in the control file that is incremented for each backup set. The counter value starts at 1. This number will be unique for the lifetime of the control file (thus, it is reset at RESETLOGS or when the control file is restored or re-created).
%t	Indicates that the backup set timestamp, which is a 4-byte value derived as the number of seconds elapsed since a fixed reference time, should be substituted. %s and %t combined can be used to form a unique name for the backup set.
%T	Indicates that the year, month, and day from the Gregorian calendar in the format YYYYMMDD should be substituted.
%u	Indicates that an eight-character name, consisting of compressed representations of the backup set or image copy number and the time the backup set or image copy was created, should be substituted.
%U	<p>This is the default file-naming pattern and provides a system-generated unique filename for RMAN-related files. The meaning of this substitution string differs when dealing with image copies or backup pieces.</p> <p>When using backup set pieces, %U specifies a convenient shorthand for %u_%p_%c that guarantees uniqueness in generated backup filenames. The meaning differs when using image copies, and depending on the type of image copy.</p> <p>Meaning when used with an image copy of datafiles: data-D-%d_id-%I_TS-%N_FNO-%f_%u</p> <p>Meaning when used with an image copy of an archived redo log: arch-D_%d-id-%I_S-%e_T-%h_A-%a_%u</p> <p>Meaning when used with an image copy of a control file: cf-D_%d-id-%I_%u</p>
%Y	Indicates that the year in the format YYYY should be substituted.
%%	Indicates that you wish to actually use the % character; for example, %%Y.

TABLE 3-3 *Format String Specification Descriptions (continued)*

Configuring Default Automated Backups of the Control File and the SPFILE

RMAN in Oracle Database 10g and later offers the ability to back up the control file and the database parameter file, and you can configure these backups to take place by default. Again, you can use the **configure** command to configure this automated backup process to happen automatically during a backup. Here is an example of configuring automated backups of these important database files, and an example of turning off the default configuration:

```
configure controlfile autobackup on;
configure controlfile autobackup off;
```

When autobackup of the control and parameter files is configured, the following rules apply:

- The control file and the server parameter file will be automatically backed up with each RMAN **backup** or **copy** command issued that is not included in a **run** block.
- If a **run** block is used, then the control files and parameter files will be backed up at the end of the **run** block if the last command is not **backup** or **copy**.

NOTE

If you are not going to use a recovery catalog and the following conditions apply:

- *You wish to be able to recover your control file after an automated control file backup.*
- *You are not using the FRA.*

In addition to the last two types of automated control file backups, a special type of control file backup can be configured to occur as a direct result of database changes such as adding new tablespaces, adding datafiles, adding online redo logs, and so on. This type of automatic backup can only happen to disk. A special option of the **configure controlfile autobackup** command can be used to facilitate this backup. Here is an example:

```
RMAN> configure controlfile autobackup format for device type
disk to 'd:\backup\conf\robt_%F'
```

When this option is used, the Oracle RDBMS will automatically back up the control file during database structure changes that impact the control file. These changes might include 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. Note that this automated backup can only be to disk, because tape is not supported. These backups can get a bit large (since the control file contains a history of many of the past backups), so make sure you allocate enough disk space to the backup directory. In spite of the additional space that will be required, these backups can be incredibly handy to have for recovery. Finally, be aware that if the backup fails for any reason, the database operation itself will not fail.

**NOTE**

You must know the DBID of the database. You should, as a part of your initial setup and configuration of RMAN, note the DBIDs of the databases that you will be backing up and save that list somewhere safe. The DBID of the database is available from the V\$DATABASE view in the DBID column. The DBID of the database is also displayed when you start RMAN and connect to a target database, as shown in this example:

```
[oracle@robertgfreeman ~]$ rman target=/
Recovery Manager: Release 11.2.0.1.0 - Production on Thu Nov 5 04:04:06 2009
Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.
connected to target database: RO1 (DBID=1854903786)
```

Configuring Default Retention Policies

So, how long do you want to keep your database backups? RMAN enables you to configure a backup retention policy by using the **configure retention policy** command. If you configure a retention policy and are using the FRA, then RMAN and Oracle will take care of automatically removing backups when they become obsolete. If you are not using the FRA, then configuring a retention policy will not cause backups to be deleted automatically, but will cause expired backup sets to appear when the **report obsolete** command is executed. See Chapter 15 for more on **report obsolete**.

There are really three kinds of retention policies in Oracle: recovery window–based, redundancy-based, and none. Let’s look at each of these in more detail next.

Recovery Window–Based Retention Policies The recovery window–based retention policy is designed to ensure that your database can be recovered to a specific point in time. For example, if you wanted to make sure that you could recover your database back to any point in time up to three days ago (assuming you were running in ARCHIVELOG mode, of course), you would set a recovery window–based retention policy of three days. The command to configure such a retention policy would be as follows:

```
RMAN> configure retention policy to recovery window of 3 days;
old RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 3 DAYS;
new RMAN configuration parameters are successfully stored
```

Note that the recovery window–based retention criteria can result in backups actually being maintained longer than the stated recovery window. For example, if your recovery window is three days, but your last full backup was five days ago, then that backup will remain valid until it is no longer needed to restore your database. Even if you back up your database today, five days later, the backup that is five days ago is still needed because it is the only source of recovery back to day 3. Figure 3-1 provides a graphical demonstration of this.

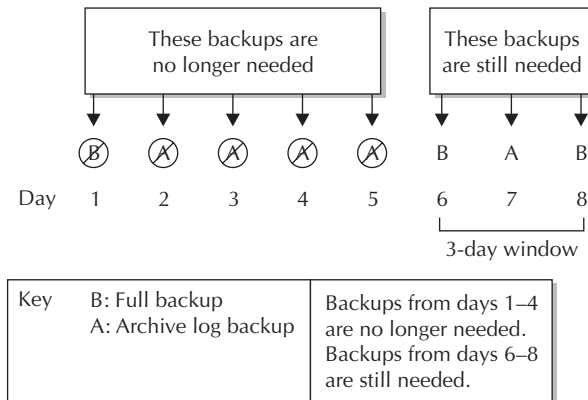
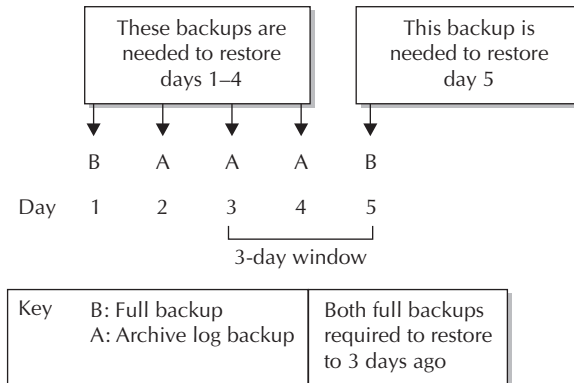


Figure 3-1 Recovery window maintaining older backups

Now that we have configured our retention policy, let's see which previous backups are reported to be obsolete:

```
RMAN> report obsolete;
RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 3 days
Report of obsolete backups and copies
```

Type	Key	Completion Time	Filename/Handle
Archive Log	12	08-SEP-09	/oracle/app/oracle/flash_recovery_area/ROB1/archivelog/2009_09_08/o1_mf_1_9_5bd8qv45_.arc
Backup Set	24	08-SEP-09	
Backup Piece	34	08-SEP-09	/oracle/app/oracle/flash_recovery_area/ROB1/backupset/2009_09_08/o1_mf_annnn_TAG20090908T202600_5bg4kr90_.bkp
Backup Set	25	08-SEP-09	
Backup Piece	35	08-SEP-09	/oracle/app/oracle/flash_recovery_area/ROB1/backupset/2009_09_08/o1_mf_nnnd0_TAG20090908T202601_5bg4kttk1_.bkp

In this example, we have two backup sets and two related backup pieces that are obsolete based on our backup retention policy. Additionally, we have an archived redo log that is ready to be removed as well. If these backups are in a defined FRA (which these are), Oracle will remove them as required. If you are not using an FRA, or if these backups were created before you converted to using an FRA, you will need to use the **delete obsolete** command to remove them. More information on the **delete obsolete** command can be found in Chapter 15, and an example is provided here, too:

```

RMAN> delete obsolete;
RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 3 days
using channel ORA_DISK_1
using channel ORA_DISK_2
Deleting the following obsolete backups and copies:
Type                Key      Completion Time   Filename/Handle
-----
Archive Log         12      08-SEP-09
/oracle/app/oracle/flash_recovery_area/ROB1/archivelog/2009_09_08/
o1_mf_1_9_5bd8qv45_.arc
Backup Set          24      08-SEP-09
Backup Piece       34      08-SEP-09
/oracle/app/oracle/flash_recovery_area/ROB1/backupset/2009_09_08/
o1_mf_annnn_TAG20090908T202600_5bg4kr90_.bkp
Backup Set          25      08-SEP-09
Backup Piece       35      08-SEP-09
/oracle/app/oracle/flash_recovery_area/ROB1/backupset/2009_09_08/
o1_mf_nnnd0_TAG20090908T202601_5bg4kttk1_.bkp
Do you really want to delete the above objects (enter YES or NO)? yes

```

Note in the preceding example that the system will ask you to confirm that you really want to remove the objects that are slated to be removed. If any of the listed objects are not available to be removed, then you will need to run the **crosscheck** command (discussed in Chapter 14). Otherwise, each item listed as deleted in the **delete obsolete** output will be deleted by Oracle.

Redundancy-Based Retention Policies This kind of retention policy is based on the total number of backups maintained by RMAN and is more typically used if you are backing up your database infrequently. This is the default retention policy, with a default value of 1. If you were to set this value to 3, then Oracle would consider the last three backups as current, and any other

backups would be considered obsolete. Here is an example of configuring a redundancy retention policy of 3:

```

RMAN> configure retention policy to redundancy 3;
old RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO REDUNDANCY 3;
new RMAN configuration parameters are successfully stored

```

Note in the output that RMAN displays both the old and new settings for the retention policy.

No Retention Policy If you want to disable the retention policy, you use the command **configure retention policy to none**, and no retention policy will be applicable. Use the **configure retention policy clear** command to reset the retention policy to the default value, which is a redundancy of 1.



NOTE

If you are using a tape management system, it may have its own retention policy. If the tape management system's retention policy conflicts with the backup retention policy that you have defined in RMAN, the tape management system's retention policy will take precedence, and your ability to recover a backup will be in jeopardy.

Configuring Default Levels of Encryption

RMAN can create encrypted backups starting with Oracle Database 10g Release 2 and later. During the backup, the backup sets are encrypted as they are created. When the backups are restored, Oracle will decrypt the backup sets. In this section, we discuss the types of encryption that are available and then look at how to configure RMAN so that it can use encryption.

Oracle offers three different encryption modes:

- **Transparent mode** Transparent mode encryption requires no DBA interaction. To use this mode, you must have configured the Oracle Encryption Wallet.
- **Password mode** Password mode encryption requires that a password be supplied when creating backups to be encrypted or when restoring backups that were encrypted when they were created. The password is supplied by using the command **set encryption on identified by password only** in your RMAN backup scripts. This is the encryption mode we will use in this text.
- **Dual mode** Dual mode backups can be restored either by password or by the presence of the Oracle Encryption Wallet. This makes offsite restores of backups easier, since the install of the Oracle Encryption Wallet is not required. To create a dual mode encrypted backup, you use the **set encryption on identified by password** command (note that the **only** keyword is missing).

Use the **configure** command to configure various persistent settings related to RMAN encryption of backups. You can use the RMAN **configure** command to indicate the following:

- Whether all database files should be encrypted

- Whether specific tablespaces should be encrypted
- Which of the available encryption algorithms should be used to encrypt your backups

If you are using Oracle Encryption Wallet–based security, then you only need to set the persistent RMAN settings required by the **configure** command. If you wish to use password mode encryption or dual mode encryption, you need to configure the persistent security defaults with the **configure** command, and then use the **set** command when starting your backups to set the correct password for the backup. RMAN does not persistently set the backup password, so it must be entered for each RMAN backup or recovery session. The **set** command, and how to use it during backups, is covered in much more detail in Chapter 9. In the following command, we configure and enable backup encryption for the entire database. Notice that if we have not configured the Oracle Encryption Wallet, any subsequent backups will fail unless we use the **set** command to establish an encryption password for the session (we are jumping the gun just a bit, but we provide an example of using the **set** command to set the backup password in appropriate context).

```
-- Configures default encryption.
-- Uses transparent mode encryption by default.
RMAN> CONFIGURE ENCRYPTION FOR DATABASE ON;
-- For this session, we want password mode encryption,
-- so we have to set the
-- password. This is good only for this session, until we exit RMAN or
-- issue another connect command.
RMAN> SET ENCRYPTION ON IDENTIFIED BY robert ONLY;
-- Way ahead of ourselves, but this backs up the database!
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

Archived redo log backups are backed up using encryption if the following are true:

- The **set encryption on** command is in effect at the time that the backup of the archived redo logs is occurring.
- Encryption has been configured for the entire database, or for at least one tablespace of the database.

The **configure** command also provides the ability to determine the encryption algorithm you wish to use. The available algorithms can be seen in the `V$RMAN_ENCRYPTION_ALGORITHMS` view as seen in this example:

```
SQL> select algorithm_name from V$RMAN_ENCRYPTION_ALGORITHMS;
ALGORITHM_NAME
-----
AES128
AES192
AES256
```

Knowing the algorithms available, we can now configure a default encryption algorithm we wish to use, as seen here:

```

RMAN> Configure encryption algorithm 'AES128';
using target database control file instead of recovery catalog
new RMAN configuration parameters:
CONFIGURE ENCRYPTION ALGORITHM 'AES128';
new RMAN configuration parameters are successfully stored

```

Configuring Archive Log Deletion Policies

You can configure RMAN to manage your archived redo log deletion policy for you. By default, Oracle applies the configured backup retention policy to the archived redo logs. In Oracle Database 11g, you can also configure a separate deletion policy for archived redo logs. This policy will get applied to archived redo logs in both the FRA and in those stored outside the FRA. Only those in the FRA will be removed by Oracle, however. If logs are in the FRA, Oracle will try to keep them as long as possible, only removing them when additional space is required. If you are using a non-FRA location, then you will need to use the **delete obsolete** or **delete archivelog** command to remove archived redo logs marked as obsolete. In this example we use the **configure** command to configure an archive log deletion policy. In this case, all archived redo logs that are backed up three times will be eligible for removal:

```

RMAN> Configure archivelog deletion policy to backed up 3 times to device type disk;
new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO BACKED UP 3 TIMES TO DISK;
new RMAN configuration parameters are successfully stored

```

In versions of Oracle before Oracle Database 11g, the archived redo log deletion policy applied only to archived redo logs being applied on a standby database. In these versions, you could configure RMAN to mark archived redo logs as eligible for removal after they have been applied to a mandatory standby database by using the **configure archivelog deletion policy to applied on standby** command. In this case, once the archived redo log has been successfully applied to a mandatory standby database location, it is eligible for removal from the FRA by Oracle. This functionality remains in Oracle Database 11g and later.

If You Are Using Shared Servers

If you are using Oracle's Shared Servers option (known as Multi-Threaded Server, or MTS, in previous Oracle versions), then you have to configure a dedicated server for use with RMAN because RMAN cannot use a Shared Servers session to connect to the database. If you are using a Shared Servers architecture, refer to Chapter 5 of the *Oracle Database Backup and Recovery Advanced Users Guide (11g Release 2)* for more information on how to configure RMAN for use with the Oracle Database 11g Shared Servers option.

Essentially, you must configure a dedicated connection in Oracle Net for your server by using the **SERVER=dedicated** syntax, as shown in this example (note that Oracle Net configurations vary greatly, so what may be required of you might differ):

```

Rob1_ded =
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=robpc)(port1521))
  (CONNECT_DATA=(SERVICE_NAME=rob1_ded)(SERVER=dedicated)))

```

Summary of RMAN Configuration Tasks

We have thrown a great deal of information at you in this chapter. The following summarizes the tasks that you will need to perform to get set up to do database backups with RMAN. We also provide a few suggestions along the way. Each of the steps listed here has detailed instructions included earlier in this chapter:

1. Determine whether you wish to run the database in ARCHIVELOG mode or NOARCHIVELOG mode. Configure the database accordingly. In most cases, we would recommend ARCHIVELOG mode since it provides a large number of recovery options.
2. We recommend that you configure and use the FRA.
3. Set up a separate database user account (not sys) for use with RMAN.
4. In the database parameter file, set the `CONTROL_FILE_RECORD_KEEP_TIME` parameter to a number of days equivalent to or greater than the number of days you wish to retain database backups. Retention is one area within RMAN that can have “gotchas.” Make sure that your retention policies in RMAN, the `CONTROL_FILE_RECORD_KEEP_TIME` parameter, and any retention policies established by your tape administrators (if you are backing up to tape) are aligned.
5. If you are using shared servers, set up a dedicated server address for RMAN to connect to.
6. Using RMAN, connect to the target database to ensure that the database is set up correctly (error messages will appear if your RMAN account is not correctly set up).
7. Use the **configure** command to establish your default RMAN values. In particular, consider configuring the following:
 - Configure the default degree of parallelism for tape or disk backups. Set it to a default value equivalent to the number of disks or tape drives that you will be backing up to. If you are backing up to a SAN with many disk drives, consider using parallel channels to back up to those disk devices.
 - Configure automatic channels and device types. The number of channels that you configure should equal the degree of parallelism that you have configured. Configure as many channels as you have individual devices, and configure the same number of channels as you have.
 - Configure automated control file/database parameter file autobackups.
 - If you own ASO (Advanced Security Option, which is the license required to use encryption), then configure automated database backup encryption.
8. Configure the retention policy as required. Make sure this retention policy is in sync with any other retention policies, such as those associated with tape management systems. Also, if required, consider retention criteria for your archived redo logs.
9. Configure RMAN for control file and SPFILE automatic backups.
10. Before you use it for production database backups, test your RMAN configuration by doing a backup and recovery as demonstrated in later chapters.

Other Backup and Recovery Setup and Configuration Considerations

Finally, let's consider the other backup and recovery implications of your database. RMAN will not back up certain things that you need to consider as a part of your overall backup and recovery strategy planning. These include such things as the base Oracle RDBMS software and the parameter files (`tnsnames.ora`, `names.ora`, `sqlnet.ora`, and so on). You need to make plans to back up and recover these files as a part of your overall backup and recovery planning.

You also need to consider your disaster planning with regard to RMAN and non-RMAN backups. How will you protect these backups from flood, fire, and earthquake? In advance is a very good time to consider these questions, not when the fire is burning two flights below!

Summary

Whew! We have covered a great deal of ground in this chapter, and, indeed, there are several things you need to do before you start using RMAN. First, we described how to set up the database in ARCHIVELOG mode if that is what you wish to do. Next, we looked at the RMAN command line and at how to configure your database for use with RMAN, including setting up the password file and configuring a user account for use with RMAN. We also looked at configuring RMAN default settings. We strongly suggest you take advantage of this feature in RMAN, because it can make your life much easier. Finally, we provided you with a summary of RMAN configuration tasks and talked about other backup and recovery considerations.