Back Up and Recovery Guide

Author: Geoff Chapman

Date: 3rd December 2007

Version: 0.6

Location of Document:

DOCUMENT HISTORY

Version	Date	Changed By:	Remarks
0.1	18/10/04	G S Chapman	Initial generic version
0.2	22/11/04	G S Chapman	Addition of Tivoli TDP information, and information upon Recovery Catalog considerations.
0.3	15/12/04	G S Chapman	Addition of recovery scenarios when recovery catalogue and controlfiles are lost.
0.4	19/01/05	G S Chapman	Information about controlfile autobackup.
0.5	18/07/05	G S Chapman	Duplicate database information included.
0.6	03/12/07	G S Chapman	Minor changes. Backup recommendations.

DOCUMENT DISTRIBUTION

Copy No	Name	Role	Organisation

DOCUMENT REFERENCES

Document Name	Originator	Part Number	Version	Date
Rman recovery without a recovery catalog or controlfiles	Bonnie J Bizzaro Alliance Data Systems	White Paper		Dec 2004
How to Extract Controlfiles, Datafiles, and Archived Logs from RMAN Backupsets	Oracle Corporation	Note 60545.1		Feb 2003
Top 10 Backup and Recovery best practices.	Oracle Corporation	Note 388422.1		Dec-2006
Performing OS Backups of an Oracle9i and 10g Database on MS Windows	Oracle Corporation	Note 345069.1		Jan 2007

TABLE OF CONTENTS

1 Operating System Backups 1
1.1 Windows File system backups 1
1.2 Unix file system backups 2
2 RMAN database backups 5
2.1 General 5
2.2 Process 5
2.3 Database recovery 6
2.3.1 Database open, datafile deleted 7
2.3.2 Complete restore (lost online redo) and rollforward - database closed 7
2.3.3 Restore of a subset of datafiles, complete recovery 8
3 RMAN Controlfile considerations 9
3.1 Automatic Controlfile backup generation 9
3.1.1 Restoring a Control File Autobackup to a Nondefault Location: Example 10
4 Recovery Catalog Considerations 11
4.1 Using the Recovery Catalog 11
4.2 Backup and Recovery of Recovery Catalog 11
4.3 Use the Appropriate Method for Physical Backups 11
4.4 Store the Recovery Catalog appropriately 12
4.5 Make Logical Backups 12
4.6 Recovering the Recovery Catalog 12
4.6.1 Recovering the Catalog Using Operating System Methods 12
4.6.2 Re-Creating the Recovery Catalog 13
5 Oracle Media Management API 14
5.1 Tivoli TDP 14
5.2 TDP Configuration and Setup 14
5.3 TSM API Configuration and Environment setup 16
5.4 Known 9i issues: 18
5.4.1 ORACLE 9i is not able to load the libobk library 18
5.4.2 SBTTEST fails to work complaining about an invalid 64 bit object. 18
6 Recovery using PL/SQL 20
6.1 Recovery from backupsets without RMAN 20
6.1.1 How RMAN works with PL/SQL 20
6.1.2 Prerequisites 21
6.1.3 About the controlfile 22
6.1.4 Extracting the controlfile from a backupset 23
6.1.5 Extracting datafiles from a backupset 24
6.1.6 Applying incrementals 25
6.1.7 Extracting archivelogs from a backupset 26
6.1.8 A typical scenario - outline 28
6.1.9 Possible Errors that may be encounted 28
6.1.10 Processes and scripts used (case study) 29
6.1.10.1 Restore example #1 29
6.1.10.2 Restore example #2 30
6.1.10.3 Restore example #3 31
6.1.11 Possible exhancements 32
6.2 Summary 32
7 Using RMAN to duplicate a database. 33

7.1 Duplicate Database using RMAN in Oracle9i 33
7.2 Alternate client restore with RMAN and Tivoli 34
7.2.1 Preparing the Auxiliary Instance for Duplication: 34
7.2.1.1 Basic Steps 34
8 RMAN and standby databases 41
9 RMAN Catalog maintenance and Tivoli TSM. 43
10 Top Backup and Recovery best practices 45

TABLE OF FIGURES

Figure 1 - Windows Backup Utility 1
Figure 2 - Backup File and Directory Selection 2

PURPOSE OF DOCUMENT

The aim of this document is to provide a generic description of the back up and recovery processes. It specifically describes how a back up of system files is achieved with Windows 2000 and Solaris 9, although is generally applicable for all Unix systems.

For database back up, the document focuses on the Oracle RMAN utility and recommends a particular method for backing up data.

The back up and recovery process must remain necessarily generic because there are no requirements that specify what software utilities are to be employed.

Backup and Recovery

Backup of the system fall into two categories: the backup of the base system itself and the backup of the database information. A full file system backup assumes that there are no running databases on the system at the time of the backup. This is often referred to as a '*cold*' backup. In contrast a backup whilst the databases are running is known as a '*hot*' backup. Hot backups of the database are performed using the '*rman*' utility.

1 Operating System Backups

1.1 Windows File system backups

The Windows operating system comes with a backup/restore utility that can be used for backing up the files. Unfortunately it does not have the ability to backup 'live' files. i.e. Files that are being accessed or updated whilst the backup is running. Third party software is required for this capability.

For a cold back up, all database files need to be 'quiesced' whilst the backup is running and the simplest way to achieve this is to stop the databases. An alternative is to use the Oracle supplied RMAN utility to generate flat files that can then be backed up to tape whilst the database is still running. It is these flat files, which are copied to tape not the actual database files themselves. The flat files can be used to restore the database if necessary.

From the My Computer -> C drive ->Properties -> Tools -> Backup menu the following screen is displayed:

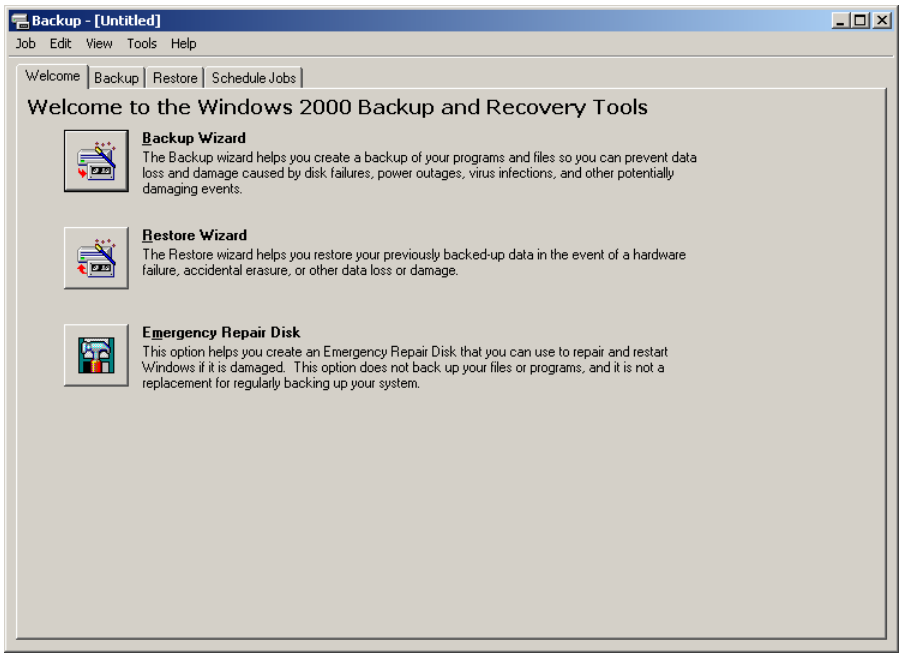


Figure 1 - Windows Backup Utility

Choosing the Backup tab display, presents a new window displaying the drives mounted on the computer. The number and names of the drives will vary depending upon the computer.

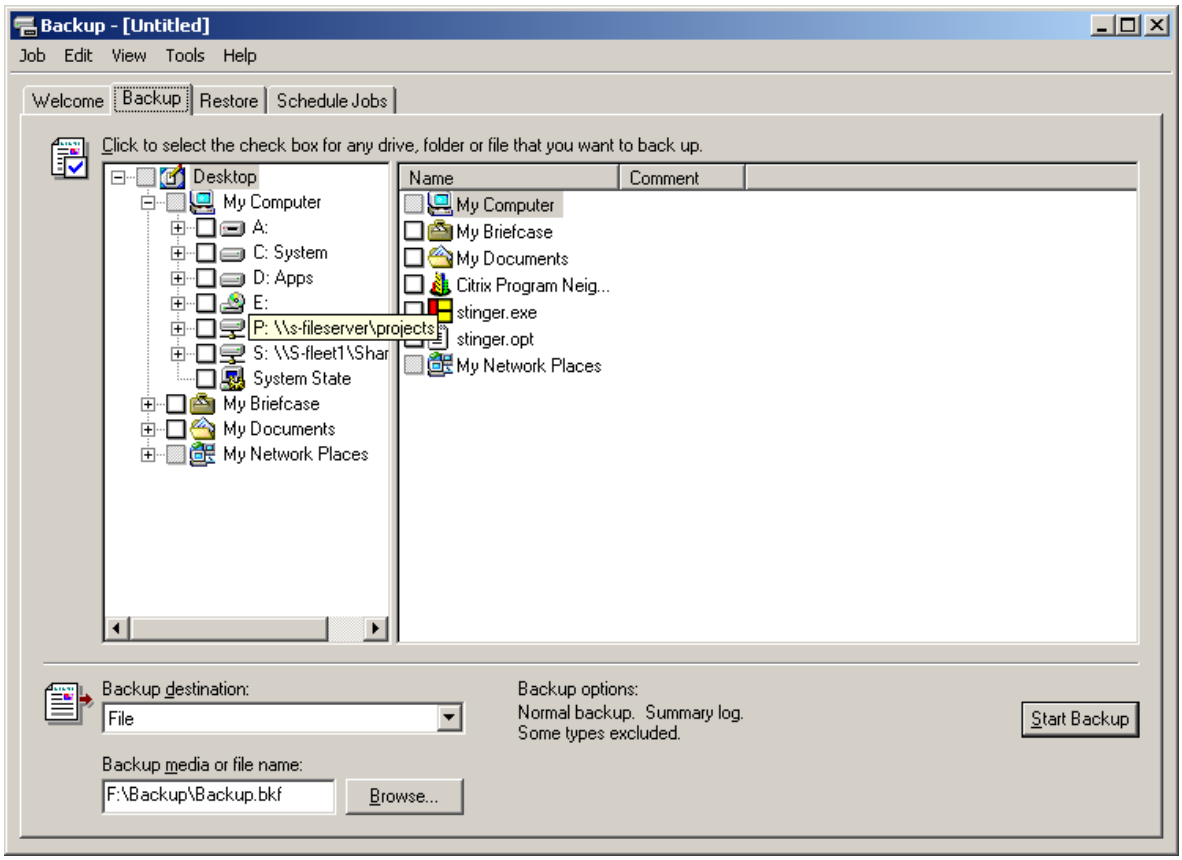


Figure 2 - Backup File and Directory Selection

The backup destination tape will show any attached tape drives if any are present. If a DLT tape drive is attached to the database server and will show itself (for example) as a 'DLT drive' (To backup up the databases or an application server on another machine, the disks attached to it need to be shared with the (main) database server, and backed up to tape from the latter).

All Oracle databases should be configured using OFA (Oracle Flexible Architecture) and hence all active files will then be located in a directory of the form:

Drive *X*:\oradata\<*Database SID*>*filename*.dbf (or .ctl or .log depending upon file type.)

In the restoration process, it is possible to restore files to a different location from that which they originated.

1.2 Unix file system backups

Unix backups and in this section Solaris will be used to illustrate the technique, are performed to locally attached tape devices using '*ufsdump*' utility, whilst a restore is performed using '*ufsrestore*'.

A basic shell script is provided below for a tape attached to a Solaris database machine. [Attached to /dev/rmt/0]

```
#!/usr/bin/ksh
#
# Script to dump all file systems to an attached tape drive.
#
# G S Chapman 2nd Sept 2003
#
#  Assumes databases are closed down at this point.
```

```
#
# Now dump /
#
ufsdump 0fu /dev/rmt/0n /dev/rdsk/c1t0d0s0
#
# Now dump /export/home0
#
ufsdump 0fu /dev/rmt/0n /dev/dsk/c1t1d0s7
#
# Now dump /u01
#
ufsdump 0fu /dev/rmt/0n /dev/dsk/c3t10d0s7
#
# Now dump /u02
#
ufsdump 0fu /dev/rmt/0n /dev/dsk/c3t11d0s7
#
# Now dump /u03
#
ufsdump 0fu /dev/rmt/0n /dev/dsk/c3t12d0s7
mt –f /dev/rmt/0n weof
mt –f /dev/rmt/0n weof
mt –f /dev/rmt/0 rewind
```

Note that the backup media will consist of a number of separate files, each one a complete file system, separated by end of file markers, with two end of file markers at the end. This information enables the media to be positioned at the correct location prior to the invocation of the restore process.

An example of finding the files on the /u03 file generated as a result of running the above script is shown below using a database named as 'crmdbs':

```
# ufsrestore ifs /dev/rmt/0n 5
ufsrestore > l
.:
 lost+found/   oradata/
ufsrestore > cd oradata
ufsrestore > ls
./oradata:
 crmdbs/
ufsrestore > cd crmdbs
ufsrestore > ls
./oradata/crmdbs:
 control03.ctl   temp01.dbf
ufsrestore > quit
```

From a separate Unix terminal window a check can be made against the actual file system:

```
$ pwd
/u03/oradata/crmdbs
$ ls –la
total 85972
drwxr–xr–x   2 oracle   dba          512 Sep  1 16:47 .
drwxr–xr–x   3 oracle   dba          512 Sep  1 16:46 ..
–rw–r–––––   1 oracle   dba      2023424 Sep  2 09:12 control03.ctl
–rw–r––r––   1 oracle   dba     41951232 Sep  1 16:47 temp01.dbf
$
```

See the *ufsdump* and *ufsrestore* documentation for full details of the utilities and their use.

From a remote machine (Application server or another database machine) the script is only slightly different, the difference being the access of the a remote attached tape drive. [The example below assumes the setting up of a symbolic link between /usr/bin/rsh and a host name 'wsdbs', and the remote tape drive will be /dev/rmt/0.]

```
#/usr/bin/ksh
#
# Script to dump all file systems to a remotely attached tape drive.
#
# G S Chapman 2nd Sept 2003
#
#  Assumes databases are closed down at this point.
# wsdbs is a symbolic link to /usr/bin/rsh
#
PATH=$PATH:/usr/local/bin
#
# Now dump /
#
ufsdump 0fu wsdbs:/dev/rmt/0n /dev/rdsk/c1t0d0s0
#
# Now dump /export/home0
#
ufsdump 0fu wsdbs:/dev/rmt/0n /dev/dsk/c1t1d0s7
wsdbs mt -f /dev/rmt/0n weof
wsdbs mt -f /dev/rmt/0n weof
wsdbs mt -f /dev/rmt/0 rewind
```

To restore, for example, the second file the following command is used:

```
ufsrestore -ifs wsdbs:/dev/rmt/0n 2
```

2 RMAN database backups

All Oracle databases come with the RMAN utility, which can be used to backup and restore the databases. Although there are a number of ways in which the utility can be used to back up the databases, the following method has been widely used and is strongly recommended.

RMAN backups by default (out of the box installation) are configured to write to a disk file system. RMAN can be configured to write to Media Management Libraries such as Tivoli. See the section below on Tivoli TDP configuration. The differences between the scripts used is minor, but the section that follows assumes that backup to disk is being used.

2.1 General

The process makes use of the database control files for the saving of backup and restore information. (An alternative using a database catalog is documented in a later section.) A batch script (shown below) carries out the tasks of placing the database into a hot backup mode and then backing up the database contents to a specified location. The database is placed back into normal mode once the backup is complete. The file produced is then backed up to the tape media (i.e. DLT) using the standard, supplied backup utility in the operating system. The examples are for an Oracle 9 system, but are very similar fo Oracle 8i or greater.

2.2 Process

For each database:

- Place the database in a closed mode so that archive logging can be turned on. This carried out by connecting to the database as a '*sysdba*' user and closing and then restarting the database in *nomount* mode.
- Issue the command:

```
ALTER DATABASE ARCHIVELOG;
```

- Open the database using:

```
ALTER DATABASE OPEN;
```

- Establish the location of the archive log using:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST =
'C:\Oracle\admin\<DBSID>\archivelogs';
```

Where <DBSID> is the name of the database instance identified. The location will be slightly different for the Solaris databases but follows the same OFA (Oracle Flexible Architecture) conventions.

- Start the archiving process:

```
ALTER SYSTEM ARCHIVE LOG START;
```

For this to happen automatically after each server restart, the initialisation file must be changed and this is achieved by adding the parameter 'log_archive_start=true' to the ini.ora file. This is a multi—step process, and the ini.ora file has to be generated from the binary spfile as follows:

```
CREATE PFILE FROM SPFILE;
```

Edit the generated ini.ora file, located in the $ORACLE_HOME/dbs or $ORACLE_HOME/database directory, dependent upon the platform, and add the following line to the end of the file:

```
*.log_archive_start=true
```

The database will then have to be shutdown and restarted from the ini file. Once restarted, the spfile can be regenerated from the pfile.

```
CREATE SPFILE FROM PFILE;
```

This will then ensure that any further startups will have archiving enabled).

- Prepare the databases for use by the RMAN utility.
  - The Oracle user running the RMAN utility requires some database privileges and these are issued from an SQL session by the 'SYS' user as follows:

```
GRANT SYSDBA TO SYSTEM;
GRANT SYSOPER TO SYSTEM;
```

Where a system user runs the database backups.

- Start RMAN using:

```
rman target=system/{password}@<DBSID> nocatalog
```

- There is a one time setup for RMAN that involves setting the redundancy value:

```
configure retention policy to redundancy 3;
```

This will ensure that three copies of the backups are retained and it also allows the command '`delete obsolete`' to remove extra copies in the normal run script.

The run script is as follows (for Windows 2000):

```
run {
allocate channel dev1 type disk;
backup full tag 'Database <DBSID> full backup'
   format 'D:\backup_dir\<DBSID>_t%t_s_%s_p%p'
(database include current controlfile);
backup tag '<DBSID> Archivelog backup'
   format 'D:\backup_dir\<DBSID>_t%t_s_%s_p%p' archivelog all delete
input;
release channel dev1;
}
report obsolete device type disk;
delete obsolete;
exit;
```

Where <*DBSID*> is the name of the database instance being backed up.

The script backs up the full database in '*hot backup mode*'. So the database is still available for full usage by any users. There will be a slight impact upon performance. The active control file is also backed up at the same time. The archivelogs are then also backed up to another file, before being deleted from the server. Finally all '*obsolete*' backups (ie. any more than three) are listed and then removed from the control files.

- Execute the 'Run' script using:

```
@script
```

where the *script* name is the fully qualified patch and name given to the script, eg. *C:\backup.scr*.

2.3 Database recovery

Database recovery is very dependent upon the situation for which recovery is being performed. Situations vary from individual datafiles, log files, controlfiles, tablespaces or the complete database itself. These variations make it virtually impossible to cover every single situation but

the following indicate a few of the more common scenarios. As with all database recovery situations, these need to be carried out by a suitably trained Database administrator (DBA) to ensure that the process is performed smoothly and efficiently.

2.3.1 Database open, datafile deleted

Datafile has been deleted from a running database. There are two methods of open database recovery: restore the datafile and then recover the datafile, or restore the tablespace and then recover the tablespace. The next two examples typically show both methods:

(a) Datafile recovery

```
RMAN> run {
2> allocate channel dev1 type disk;
3> sql "alter tablespace users offline immediate";
4> restore datafile {datafile};
5> recover datafile {datafile};
6> sql "alter tablespace users online";
7> release channel dev1;
8> }
```

(b) Tablespace 'Users' recovery

```
RMAN> run {
2> allocate channel dev1 type disk;
3> sql "alter tablespace {users} offline immediate";
4> restore tablespace {users};
5> recover tablespace {users};
6> sql "alter tablespace {users} online";
7> release channel dev1;
8> }
```

Note that if it is the system tablespace datafiles to be restored, the database must be closed. It is not possible to offline the system tablespace.

2.3.2 Complete restore (lost online redo) and rollforward - database closed

```
RMAN> run {
2> allocate channel dev1 type disk;
3> set until logseq={105} thread=1;
4> restore controlfile to {'/oracle/dbs/ctrltargdb.ctl'};
5> replicate controlfile from {'/oracle/dbs/ctrltargdb.ctl'};
6> restore database;
7> sql "alter database mount";
8> recover database;
9> sql "alter database open resetlogs";
10> release channel dev1;
11> }
RMAN> reset database;
```

Notes:

The 'set until' command dictates at which log sequence recovery will stop . It is critical that this command is issued BEFORE datafiles are restored, otherwise RMAN will attempt to restore the most recent set of datafiles, which could be ahead of the specified log - The 'replicate controlfile' copies the restored controlfile to the controlfiles referenced in init.ora - Because the database is opened with resetlogs, it is necessary to register the new incarnation of the database with the RESET DATABASE command. It is important to take a full backup of the database immediately after a resetlogs

2.3.3 Restore of a subset of datafiles, complete recovery

```
RMAN> run {
2> allocate channel dev1 type disk;
3> sql "alter database mount";
4> restore datafile {datafile2};
5> restore datafile {datafile3};
6> restore archivelog all;
7> recover database;
8> sql "alter database open";
9> release channel dev1;
10> }
```

3 RMAN Controlfile considerations

The recovery catalog is dependent on the target database control file for information. Consequently, the currency of the information in the control file determines the effectiveness of the recovery catalog.

The size of the target database's control file grows depending on the number of:

- Backups that are performed.
- Archived redo logs that Oracle generates.
- Days that this information is stored in the control file.

The CONTROL_FILE_RECORD_KEEP_TIME parameter can be used to specify the minimum number of days that Oracle keeps this information in the control file. Entries older than the number of days specified are candidates for overwrites by newer information. The larger the CONTROL_FILE_RECORD_KEEP_TIME setting, the larger the control file.

Note:
At a minimum, resynchronize the recovery catalog at intervals less than the CONTROL_FILE_RECORD_KEEP_TIME setting, because after the number of days specified in this parameter, Oracle overwrites the information in the control file with the most recently created information. If there has not been a resynchronization of the recovery catalog, and Oracle has overwritten the information, then this information cannot be propagated to the recovery catalog.

If no recovery catalog is being used, the backup information is ONLY stored within the controlfile.

3.1 Automatic Controlfile backup generation.

Oracle 9i has a feature called controlfile autobackup. Once this feature is enabled, the controlfile is automatically backed up at the end of every backup and will contain all of the information about the backup. This is activated using the configure command:

```
RMAN>configure controlfile autobackup on;
```

An example of this working is as follows:

When the database structure is changed, as when a tablespace is dropped, the alert log reports that an automated backup is done:

```
DROP TABLESPACE HAM INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINTS
Thu Dec 16 09:55:07 2004
Deleted file /t02/oradata/obsd040/ppp
Starting control autobackup
Control autobackup written to DISK device
handle '/t01/app/oracle/product/9.2/dbs/c-1245553587-20041216-02'
```

The above shows the controlfile autobackup set up to go to disk. [This file could then be included in the normal backup mechanism for the machine.] This could be configured to go to tape (i.e. Tivoli disk), if it is desired.

If a restore is required, one could use something like: 'set controlfile autobackup format ...'

```
run {
SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE disk
to '/t01/app/oracle/product/9.2/dbs/%F'
restore controlfile from autobackup;
}
```

The configuration can be temporarily changed to accommodate this:

configure controlfile autobackup format for device type disk to

'/t01/app/oracle/product/9.2/dbs /%F'

By default, the format of the autobackup file for all configured devices is the substitution variable %F.

IN the above example it is: `c-1245553587-20041216-02`

The %F variable format translates into c-IIIIIIIIII-YYYYMMDD-QQ, where:

- IIIIIIIIII stands for the DBID. The DBID is a unique numerical identifier for the database. The DBID is printed in decimal so that it can be easily associated with the target database.
- YYYYMMDD is a time stamp in the Gregorian calendar of the day the backup is generated
- QQ is the sequence in hexadecimal number that starts with 00 and has a maximum of FF (256)

It is possible to change the default format by using the following command, where deviceSpecifier is any valid device such as DISK or sbt, and 'string' contains the variable %F and is a valid handle for the specified device:

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE deviceSpecifier TO 'string';

It is also possible to override the configured location for the control file backup in an RMAN session by using the SET command to specify the directory and possibly a prefix and suffix. For example, the following could be set:

```
SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'controlfile_%F';
RUN { SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '/tmp/%F.bck'; }
```

All backups will go to $ORACLE_HOME/dbs if a location is not specified.

3.1.1 Restoring a Control File Autobackup to a Nondefault Location: Example

This example restores the latest control file autobackup made on or before June 23, 2000 with a nondefault format of PROD_CF_AUTOBACKUP_%F. It starts searching for backups with a sequence number of 20, and searches backward for 5 months:

```
SET DBID 320066378; # required when restoring control file in NOCATALOG mode
RUN
{
SET UNTIL TIME '23-JUN-2001 00:00:00';
SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE sbt TO 'prod_cf_autobackup_%F';
ALLOCATE CHANNEL CHANNEL_1 DEVICE TYPE sbt;
RESTORE CONTROLFILE TO '/tmp/autobackup_20001002.dbf' FROM AUTOBACKUP
MAXSEQ 20 MAXDAYS 150;
}
```

4 Recovery Catalog Considerations

RMAN backup information can be stored within the database controlfile, or most usefully when multiple database are to be backed up within a RMAN database catalog. If a database catalog is used then some further decisions need to be taken.

4.1 Using the Recovery Catalog

The script illustrated above need to be slightly modified to make use of a database recovery catalog.

The command to start RMAN using a recovery catalog will become:

```
rman target=system/{password}@<DBSID> rcvcat=rman/rmanpw@CATALOG
```

Where CATALOG is the SID of the recovery catalog database, and rmanpw is the rman password within that recovery database. It is also assumed that 'rman' is the owner of the recovery catalog.

The RMAN allocate channel command would be modified as follows:

```
allocate channel dev1 type sbt_tape;
```

4.2 Backup and Recovery of Recovery Catalog

Include the recovery catalog in the backup and recovery strategy. If there is no backup of the recover catalog and a disk crash occurs, then some or all of your data may be lost. Avoid this unpleasant scenario by deciding how to back up and recover the recovery catalog.

Make Logical Backups

Make Regular Backups

Back up the recovery catalog with the same frequency that the target database is backed up. For example, if there is a weekly whole database backup of the target database, then back up the recovery catalog immediately after each target database backup. The backed up catalog will have a record of the target backup preceding it, so if there is a need to restore the catalog if can also be uses to restore the target database backup.

4.3 Use the Appropriate Method for Physical Backups

When backing up the recovery catalog, there is a choice of making operating system or RMAN backups. The advantage of making operating system backups is that they can be restored using operating system methods without relying on RMAN or a recovery catalog.

If RMAN is used to back up the recovery catalog database, there has to be a consideration of how the recovery catalog will be restored in case of failure.

For example, one can:

Back up the recovery catalog database using RMAN, but start RMAN with the nocatalog option so that the backup repository for the recovery catalog is the control file in the catalog database. Make frequent backups of this control file using SQL. Remember the CONTROL_FILE_RECORD_KEEP_TIME initialization parameter must be set to a value that is high enough to store an adequate amount of historical backup data for the catalog.

Create a second recovery catalog in a separate database. The main catalog is kept synchronized as normal, while the secondary catalog is synchronized manually by periodically issuing the resync catalog command.

4.4 Store the Recovery Catalog appropriately

Never store a recovery catalog containing the RMAN repository for a database in the same database as the target database. For example, do not store the catalog for database PROD1 in database PROD1. A recovery catalog for database PROD1 is only effective if it is separated from the data that it is designed to protect. If database PROD1 suffers a total media failure, and the recovery catalog data for database PROD1 is also stored in database PROD1, then there is no catalog to aid in recovery.

This rule is especially important when the recovery catalog database has to be backed up. Take a case in which database RCAT contains the recovery catalog repository for target database PROD1. It is decides to use a recovery catalog to back up RCAT, but it is unsure where to store this catalog. If the catalog containing the repository for RCAT is stored in database RCAT itself, then in the event of lose of RCAT due to a media failure there will be difficulty restoring RCAT and that will leave database PROD1 unprotected.

4.5 Make Logical Backups

Use the Export utility to make convenient backups of the recovery catalog data. An export of the catalog allows the most flexibility when the recovery catalog must be restored, because it can be restored to any existing Oracle database.

To make a logical export of the recovery catalog from the command line:

Execute the export operation at the command line, making sure to do the following:

a) Connect as the owner of the recovery catalog.

b) Specify the OWNER option.

c) Specify an output file.

For example, if the owner of the catalog in database PROD1 is RMAN, the following command can be issued at the UNIX command line to export the catalog to file cat.dmp:

```
% exp rman/rman@prod1 file=cat.dmp owner=rman
```

Examine the output to make sure you were successful:

```
Export terminated successfully without warnings.
```

4.6 Recovering the Recovery Catalog

The method used to recover the catalog depends on the method used to back it up. You have these options:

- Recovering the Catalog Using Operating System Methods
- Recovering the Catalog Using RMAN
- Importing a Logical Backup of the Catalog

4.6.1 Recovering the Catalog Using Operating System Methods

If the recovery catalog was backed up using operating system commands, then restore the backup of the catalog using operating system commands and issue SQL*Plus commands to recover it.

If RMAN is used to recover the catalog, then reference relevant restoring and recovering procedures.

If Export was used to make a logical backup of the recovery catalog, then use Import to recover it. If a media failure damages your recovery catalog database, do the following:

- Create a new user in another database. See the manual for the recommended SQL syntax for creating a new user in a recovery catalog database.
- Import the catalog data from the Export file. Execute the import at the command line, making sure to do the following:
    - Connect as the new owner of the recovery catalog.
    - Specify the old owner using the FROMUSER parameter.
    - Specify the new owner using the TOUSER parameter.
    - Specify the import file.

For example:

```
% imp userid=rcat/rcat_pwd@new_cat file=cat.dmp fromuser=rman touser=rcat
```

4.6.2 Re-Creating the Recovery Catalog

If the recovery catalog database is lost or damaged, and recovery of the recovery catalog database through the normal Oracle recovery mechanisms is not possible, then you must re-create the catalog.

You have two options for partially re-creating the contents of the old catalog:

1. Issue catalog commands to re-catalog archived redo logs, backup control files, and datafile copies.
2. Use the resync catalog from controlfilecopy command to extract information from a backup control file and rebuild the recovery catalog from it.

You can re-create information about backup sets only by using the resync catalog from controlfilecopy command, because the catalog command does not support re-cataloging of backup pieces or backup sets. RMAN does not verify that the files being re-cataloged still exist, so the resynchronization may add records for files that no longer exist. Remove such records by issuing change . crosscheck or crosscheck backup commands.

5 Oracle Media Management API

The following information is based upon experience using the IBM Tivoli Data Protection Agent for Oracle. The information can reasonably be used with other Media Vendors products since in the main it is very generic. Items such as locations of the media manager library may vary, but the API is written to the same standard regardless of vendor.

5.1 Tivoli TDP

In Oracle 9i, the SBT_LIBRARY may be used for specifying the library file to be loaded for the backup. This eliminates the need to relink the Oracle libraries when switching between the Oracle (dummy) disk library and a third party media software library such as the TDP for Oracle.

In order to use the dummy SBT library, the allocate for an SBT channel needs to be specified with the parms 'SBT_LIBRARY=oracle.disksbt' option.

If attempting to load the TDP for Oracle libobk.x library file, do not use the SBT_LIBRARY parm as this has been seen to cause a failure for the library to load. Oracle 9i will load the media management libraries (libobk.x) dynamically when a channel is allocated with a type of 'SBT_TAPE". Ensure that the libobk.x (where x is the appropriate extension for the operating system) exists in the $ORACLE_HOME/lib directory and that the $ORACLE_HOME/lib directory is first in the LIBPATH (or LD_LIBRARY_PATH).

In Oracle 9i, Global RMAN parameters can be setup for the Oracle backups. Thus when configuring the TDP, it is possible to set the ENV parms for the backups in this global rman environment. For example:

```
RMAN> show all;
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 15 DAYS;
CONFIGURE BACKUP OPTIMIZATION OFF;                                  # default
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT_TAPE';
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE SBT_TAPE TO '%F'; #default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; #default
CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 4;
CONFIGURE DEVICE TYPE DISK PARALLELISM 1;                           # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1;     # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1;         # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1;   # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1;       # default
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'ENV=(TDPO_OPTFILE=/usr/tivoli/tsm/client/oracle/bin64/tdpoTEST.opt)';
CONFIGURE MAXSETSIZE TO UNLIMITED;                                  # default
```

5.2 TDP Configuration and Setup

The TDP for Oracle client will only look in the TDPO_OPTFILE for the variables that it will use. Note that this environment variable is case sensitive and has to be specified in upper case.

The following are the default TDPO_OPTFILE values

- 32-bit /tivoli/tsm/client/oracle/bin/tdpo.opt

- 64-bit /tivoli/tsm/client/oracle/bin64/tdpo.opt.

This default location can be changed by using the TDPO_OPTFILE

Parameter for the operations involving the TDP for Oracle.

The DSMI_ORC_CONFIG and TDPO_NODE are critical parameters in this tdpo.opt file and should always be specified. Other parameters can be set as needed.

- DSMI_ORC_CONFIG Points to the TSM user options file (dsm.opt).
- DSMI_LOG Should point to a directory (not including a filename) where the Oracle user performing the backup has write permissions.
- TDPO_NODE The name of the TSM node that has been defined on the TSM Server to be used for the Oracle backups.
- TDPO_FS Can be used specify a name for the filespace that will be created on the TSM Server to hold the TDP Oracle backups. Only the name should be specified (no leading slash).
- TDPO_AVG_SIZE can be used to override the size estimate for the TDP backup. If this is not specified or is set to the default value, then the size estimate is obtained/used from Oracle.
- TDPO_OWNER If not specified, the user account that is performing the backup will be the owner of the objects on the TSM Server. If a different Unix account attempts to restore or delete the objects, they may not be able to access them if the owner is different. The TDPO_OWNER is recommended to be used if different Unix accounts will be performing the different TDP Oracle tasks.
- TDPO_MGMT_CLASSx these values are only used for multiple copies of the archive redo logs.
- TDPO_DATE_FMT, TDPO_NUM_FMT, TDPO_TIME_FMT for international format support.

Ensure that the tdpo.opt file has read permissions by the Oracle user performing the backup.

The TDPOCONF utility is included with the DP for Oracle client and is

Used for password setup and configuration assistance. The two commands are:

Tdpoconf password

Tdpoconf showenv

After the tdpo.opt file has been configured, the password for the TSM node must be set. Run the TDPOCONF PASSWORD to set the password for the TDP node (this must be run as root). This will create a TDPO.NodeName file. Ensure this TDPO.NodeName encrypted password file has read permissions by the Oracle user.

The TDPOCONF SHOWENV can be used to check that the TDP Oracle is working correctly. The output would look similar to the following.

```
IBM Tivoli Storage Manager for Databases:
Data Protection for Oracle
Version 5, Release 2, Level 0.0
(C) Copyright IBM Corporation 1997, 2003. All rights reserved.

DATA PROTECTION FOR ORACLE INFORMATION
Version:        5
Release:        2
Level:          0
Sublevel:       0
Platform:       32bit TDP Oracle HP
```

```
TSM SERVER INFORMATION
Server Name:          TSM_ORACLE
Server Address:       123.45.67.89
Server Type:          AIX-RS/6000
Server Port:          1500
Communication Method: TCP/IP

SESSION INFORMATION
Owner Name:           ora32
Node Name:            client_oracle
Node Type:            TDP Oracle HP
DSMI_DIR:             /opt/tivoli/tsm/client/api/bin
DSMI_ORC_CONFIG:      /opt/tivoli/tsm/client/oracle/bin/dsm.opt
TDPO_OPTFILE:         /opt/tivoli/tsm/client/oracle/bin/tdpo.opt
Password Directory:   /opt/tivoli/tsm/client/oracle/bin
Compression:          FALSE
```

If the TDPO.NodeName file does not exist, the TDPOCONF SHOWENV output will indicate that the password is not set instead of showing the Password Directory.

5.3 TSM API Configuration and Environment setup

The TSM API interface is used to send the data to the TSM server. In general there is an application (Oracle/TDP) on the one side, the TSM server on the other side and the API is the data transport mechanism in the middle. It can be compared as the API to being in line at a bucket brigade where you have people passing the water bucket back and forth along the line to the fire. The TSM API is very similar it is just passing the data along, down the line. During a backup, passing the data from the application to the TSM server and on a restore it takes the data from the TSM server and passes it back over to the Application. Note in this little bucket brigade type scenario that the API is doing nothing more than passing that data along. This is a very realistic type of comparison for the API. It does not verify the data nor does it determine if the data is right or wrong.

The TSM API setup for the TDP Oracle is the same general setup that

Would be put in place for any API client. The same environment variables that are used by the TDP when backing up Oracle are the same variables that are utilized by other TSM API applications.

- DSMI_ORC_CONFIG (DSMI_CONFIG) Points to the user options file (dsm.opt).
- DSMI_DIR (is set to the api installation by default) This points to the API installation directory and also is used to find the dsm.sys file (on Unix).
- DSMI_LOG This points to directory where an error log would be written if the API witnesses any errors. If there is an ERRORLOGNAME parameter in the options file (dsm.sys/dsm.opt) then this will take precedence and be used for the error log name instead of the default dsierror.log name. The environment variables for a TDP backup are set in the TDPO_OPTFILE.

For Unix, either a valid dsm.sys file or a symbolic link to the dsm.sys must exist in the TSM API directory. The default location for the dsm.sys file is the /opt/tivoli/tsm/client/api/bin (or bin64) directory. This dsm.sys file must contain a stanza with the same SERVERNAME entry as in the TSM user option file specified by the DSMI_ORC_CONFIG (which is in the tdpo.opt file).

The "dsm.sys" file can be set up so that it contains multiple different stanzas. The use of stanzas in the dsm.sys file allows for specific differences for each client such as Passwordaccess or the

inclexcl (Include/Exclude) files can be tailored specifically for different clients. There is some additional information on multiple stanzas in the Using The Unix Backup/Archive Clients manual.

It is recommended to create a symbolic link to a single dsm.sys file (which usually exists in the backup/archive client directory) so it can be seen from the API directory (/tivoli/tsm/client/api/bin) and the TDP backup can find and use this dsm.sys. In this one dsm.sys file, there will normally be more than one ServerName stanza when using the TDP Oracle on the same machine. One ServerName stanza for the filesystem backups and another ServerName stanza for the TDP Oracle backups (which need passwordaccess prompt). During the TSM API processing, it will find the SERVERNAME stanza for the TDP Oracle backup and read the parameters under this until stanza unit it reaches another ServerName entry or the end of the file.

In general, the dsm.sys file would be similar to the following:

```
servername   TSMbackup
COMMMETHOD   tcpip
TCPServeraddress    xxx.xxx.xxx.xxx
TCPPORT    1500
NODENAME   Client
PASSWORDACCESS    generate

servername   TSMOracle
COMMMETHOD   tcpip
TCPServeraddress    xxx.xxx.xxx.xxx
TCPPORT    1500
PASSWORDACCESS    prompt
```

There would be a dsm.opt file for the Backup/Archive client that would normally be located in the /tivoli/tsm/client/ba/bin directory. Based on the dsm.sys example above, this would contain the line:

```
servername   TSMbackup
```

There would be a second dsm.opt file (which could have a different filename, such as dsmoracle.opt) for the TDP for Oracle. Based on the dsm.sys example this user option file for the Oracle client would contain the line:

```
servername TSMOracle
```

and would have passwordaccess prompt.

When using the TDP for Oracle on Unix, the passwordacess must be set to prompt. Passwordaccess generate cannot be used due to the manner in which the Unix operating system stores the password. On the Unix operating system, the TSM client would normally save the password in the TSM.PWD file when passwordaccess generate is used. The dsmtca module for the TSM client is then utilized to read the password from this encrypted file. For the Oracle backup with the TDP, the dsmtca cannot be launched as this would be a child process for the executable. The TDP is not an executable, so it is not able to have a child process. Thus for Unix, there is no child process capability for the dsmtca module to retrieve the password. Therefore, the TDP Oracle for the Unix Operating Systems must use passwordaccess prompt. This is different than Windows, which uses passwordaccess generate. For Windows, the password is stored in the registry and can be retrieved directly without any dsmtca involvement.

The TSM API must also be the same 32-bit versus 64-bit package to coincide with Oracle and the TDP. The reason that the bit version of the products must all coincide is due to the hand-in-hand library interfaces of the products. If these libraries are not all the same bit version, then when the data is passed, the library calls will be incorrect to handle the data and an error will be witnessed.

Thus, the TDP Oracle and TSM API version that is used, 32-bit versus 64-bit, is dependent on whether Oracle is in 32-bit or 64-bit mode.

It is important to ensure that all the options files have read permissions by the Oracle user that will be performing the backup (or other oracle processing with the TDP client).

5.4 Known 9i issues:

5.4.1 ORACLE 9i is not able to load the libobk library

When trying to run a backup using TDP for Oracle 5.2 on AIX5 with Oracle 9i (9.2.0.5) the following error is returned:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 12/23/2002 11:08:26
ORA-19554: error allocating device, device type: SBT_TAPE, device name:
ORA-27211: Failed to load Media Management Library
Additional information: 8
```

To resolve this,

1) Link the TDP library file directly to the Oracle directory. The Link would look like: $ORACLE_HOME/lib/libobk.a ->/usr/tivoli/tsm/client/oracle/bin64/libobk64.a
2) Change the LIBPATH environment variable to include ORACLE_HOME/lib before /usr/lib. If there is a LD_LIBRARY_PATH, ensure that this has the $ORACLE_HOME/lib before /usr/lib.
3) Ensure the SBT_LIBRARY parameter is not set
4) The Oracle instance may need to be restarted as it has been seen where the LIBPATH value was held in memory.

5.4.2 SBTTEST fails to work complaining about an invalid 64 bit object.

SBTTEST is a program supplied by Oracle to check that the communication between Oracle and the media manager is working correctly. Problems have been encountered where sbttest fails but that RMAN can allocate channels and backup the database. In this situation the failure of sbttest, may be considered cosmetic.

The following shows the type of error encountered:

$ sbttest a -libname /usr/tivoli/tsm/client/oracle/bin/libobk.a

```
/usr/tivoli/tsm/client/oracle/bin/libobk.a could not be loaded.  Check that it is installed
properly, and that LD_LIBRARY_PATH environment variable
(or its equivalent on your platform) includes the directory
where this file can be found.  Here is some additional
information on the cause of this error:
    0509-022 Cannot load module /usr/tivoli/tsm/client/oracle/bin/libobk.a.
    0509-124 The program is a discontinued 64-bit object file.
```

The following points are worth checking:

a) As "ENV=(TDPO_OPTFILE=/.../tdpo.opt)" from the RMAN allocate channel is not visible to sbttest, maybe the option file is not found. Placing it at the default location /usr/tivoli/tsm/client/oracle/bin/tdpo.opt or /usr/tivoli/tsm/client/oracle/bin64/tdpo.opt and giving read permission to the user running sbttest might help. (Don't forget the "x" in the directory path.).

b) DSMI_DIR pointing to the directory containing dsm.sys the binaries and other support files. If not set, default for application clients is /usr/tivoli/tsm/client/api/bin

c) DSMI_CONFIG pointing to the dsm.opt file. If not set, default for application clients is /usr/tivoli/tsm/client/api/bin/dsm.opt

d) Problems have been reported with SBTTEST when the "-libname" parameter is used instead of just allowing the library file to loaded dynamically.

If sbttest is unable to communicate with the media manager library file there is not much that the media manager supplier can do to assist. Since the libobk file is not being loaded, there would be no additional output, so there are no additional diagnostics capabilities that could be provided. This then becomes an Oracle support problem as to why the library is unable to be loaded.

6 Recovery using PL/SQL

6.1 Recovery from backupsets without RMAN

When using RMAN to restore objects (datafiles, controlfiles, or archivelogs) from backupsets, the object restore can be driven from the recovery catalog or the target database controlfile. In some situations, such as the loss of the catalogue and of the controlfiles, this may not be possible In these situations all is not necessarily lost, as there is the possibility of using PL/SQL procedure calls to effectively perdform the tasks that RMAN itself would perform in a normal situation. This section was developed from information obtained from Oracle Metalink and a paper presented at Oracle World 2004. It is hoped that the information contained herein is never required in earnest.

6.1.1 How RMAN works with PL/SQL

RMAN in essence is the program interface that constructs, compiles and executes PL/SQL programs. It provides the easy interface to the PL/SQL commands. RMAN constructs and executes one PL/SQL program for each channel that is allocated. Each channel is one database server session.

RMAN uses the packages DBMS_BACKUP_RESTORE, DBMS_RCVCAT, and DBMS_RCVMAN. The two packages DBMS_RCVCAT and DBMS_RCVMAN are used by RMAN to interact with the recovery catalog and/or controlfile. These packages will not be described any further and the concentration is upon the DBMS_BACKUP_RESTORE package as used by RMAN for backups and restores. It is created by the dbmsbkrs.sql and the prvtbkrs.sql scripts in the $ORACLE_HOME/rdbms/admin directory. These scripts are run by catproc.sql to create this object in every database.

The DBMS_BACKUP_RESTORE procedures are listed below. These programs can be executed whenever the database instance is started but they cannot run in a Multi Threaded Server process. A dedicated server connection is required. The database does not need to be mounted nor opened for these to run. Additional information about DBMS_BACKUP_RESTORE can be found in the script $ORACLE_HOME/rdbms/admin/dbmsbkrs.sql.

- DBMS_BACKUP_RESTORE.deviceAllocate: This specifies a device for restoring (and creating) backups. This allocates a context that will exist until deviceDeallocate is called or the session terminates. Input parameters used most often for restores are:
  - type (Type of device)
  - ident (User supplied device identifier – this can be anything you want)
  - parms (Any parameters that need to be passed to the device)
- DBMS_BACKUP_RESTORE.restoreSetDatafile: This indicates to Oracle that we will be restoring from a database backup set. This begins the restore conversation. The device must be allocated before this procedure is executed. There are no input parameters for this.
- DBMS_BACKUP_RESTORE.restoreControlfileTo: This is used to define the name of the file (including location) that the controlfile is to be restored to. The only input parameter is cfname which is the fully qualified file name.
- DBMS_BACKUP_RESTORE.restoreArchivedLog: This is used to specify the logs to be restored. Input parameters are:
  - thread (Can be found in the RMAN backup output)
  - sequence (Log sequence number)

- DBMS_BACKUP_RESTORE.restoreDatafileTo: This is used to restore a datafile to a particular location. Input parameters are:
  - dfnumber (This can be found in the RMAN backup output log)
  - toname (The name and location to restore the file to)
  - max_corrupt (Maximum number of allowed corruptions in the file)
- DBMS_BACKUP_RESTORE.restoreBackupPiece: This is used to restore files from complete backups. It reads one backup piece and writes its contents to the files in the restore list. Input parameters are:
  - handle (This is the name of the backup piece)
  - done (A boolean object used to indicate that all pieces in the restore list have been restored)
  - parms (Any parameters that need to be passed. These are device and port specific.)
- DBMS_BACKUP_RESTORE.restoreSetArchivedLog: This indicates to Oracle that there is a restoration from an archive log backup set. The database does not need to be mounted for this to work, but the instance must be started. This begins the restore conversation. The device must be allocated before this procedure is executed. Input parameter is:
  - destination (Where to restore the archive log files to)
- DBMS_BACKUP_RESORE.deviceDeallocate: This releases the currently allocated device. This should be done before the end of the recovery session. If there is a need to restore files from more than one media (device) in the same session, then this needs to be exectuted before it is possible to use a new device using the DBMS_BACKUP_RESTORE.deviceAllocate. Input parameter is:
  - parms (Any parameters that need to be passed. These are device and port specific.)

6.1.2 Prerequisites

There is some important information that is needed to restore files using the DBMS_BACKUP_RESTORE package:

- For any type of restore, the name of the backup piece and the device type.
- To restore archive log files, the sequence numbers required.
- To restore individual datafiles, the 'dfnumber' that is assigned that file during the backup process.

It's important to know the name of the backup pieces containing the files required. A backup piece is the physical file that is written to tape or disk. There can be one or more than one backup piece in a backup set. To easily identify the backup piece(s) needed for a restore, it is preferred to use the format clause of the allocate channel command to give them unique and meaningful names. It is not recommended to let RMAN use its default naming convention. For example:

```
run{
allocate channel t1 type 'SBT_TAPE' format
"${ORACLE_SID}_inc_${BACKUP_TYPE}_${DATETIME}_%p.%s.rman" parms
${MML_ENVIRONMENT};
allocate channel t2 type 'SBT_TAPE' format
"${ORACLE_SID}_inc_${BACKUP_TYPE}_${DATETIME}_%p.%s.rman" parms
${MML_ENVIRONMENT};
backup full database;
release channel t1;
```

```
release channel t2;
}
```

The above will create backup pieces named PROD1_inc_0_040226:01:05_1.1747.rman, and PROD1_inc_0_040226:01:05_1.1748.rman, if there are two backup pieces in the backup set and your variables are set as follows:

ORACLE_SID=PROD1

BACKUP_TYPE=0 (for full level 0 backup)

DATETIME=`date +%y%m%d:%H:%M` (and the date is 02/26/04 1:05)

MML_ENVIRONMENT= (any parameters you need for a non-disk device)

Naming the backup pieces using this type of format provides all the information required to identify the backup piece. The name of the backup piece includes the database name, the type of backup, and the date and time.

It is good practice to keep a log of the RMAN backup output and copy the logs to tape. In the event of a server loss, the log files can be restored and the information contained within them can be used to restore the database(s). If there is not a copy of the backup output log available, it is possible to scan the tape(s) and retrieve the backup piece names. In this situation having named the backup pieces uniquely is very important to assist in identifying the piece(s).

To uniquely name the output log of the RMAN backup within a script, the file could be written as follows:

```
$ORACLE_HOME/bin/rman  << EOF > ${LOGFILE_NAME} 2>&1
connect catalog $CUSER_PASS
connect target $DUSER_PASS
run{
allocate channel t1 type 'SBT_TAPE' format
"${ORACLE_SID}_inc_${BACKUP_TYPE}_${DATETIME}_%p.%s.rman" parms
${MML_ENVIRONMENT};
backup  current controlfile;
release channel t1;
}
EOF
```

6.1.3 About the controlfile

A backup of the controlfile should be taken *after* every backup. In this way data about the most current backup will be contained in the controlfile that is written to tape. It's a common misconception that data pertaining to backups is not stored in the controlfile if using a recovery catalog. Backup information is always stored in the controlfile. This data is stored in a circular fashion and is aged out (overwritten) after seven days by default. Additional backup history can be kept in the controlfile by modifying the initialization parameter: CONTROL_FILE_RECORD_KEEP_TIME.

If there is the most recent backup information in your controlfile, it is posible to use the PL/SQL program(s) to restore the controlfile then use traditional RMAN recovery processes to restore the rest of your datafiles. For this reason it is a good practice to backup your controlfile separately after each backup. If the controlfile is backed up as a part of the regular backup there is no guarantee that it will be written in the last backup piece. As a result, it may only contain information about part of the backup.

The following script could be used for a full database backup. Notice that two channels are allocated and the 'backup current controlfile' command has been specified.

```
run{
allocate channel t1 type 'SBT_TAPE' format
"${ORACLE_SID}_inc_${BACKUP_TYPE}_${DATETIME}_%p.%s.rman" parms
${MML_ENVIRONMENT};
allocate channel t2 type 'SBT_TAPE' format
"${ORACLE_SID}_inc_${BACKUP_TYPE}_${DATETIME}_%p.%s.rman" parms
${MML_ENVIRONMENT};
backup full database;
backup current controlfile;
release channel t1;
release channel t2;
}
```

The log from this backup shows that the controlfile was included in the *first* backup piece. Therefore, the copy of the controlfile written to tape will not contain information about the files in the second or subsequent backup piece. For this reason it is recommended that the controlfile is backed up separately after each backup.

<From the logfile>
RMAN-08008: channel t1: starting incremental level 0 datafile backupset
RMAN-08502: set_count=2369 set_stamp=502948690 creation_time=2003-08-25:03:58:10
RMAN-08010: channel t1: specifying datafile(s) in backupset
RMAN-08522: input datafile fno=00009 name=/s1/oradata01/PROD1/prod1_data02.dbf
RMAN-08522: input datafile fno=00007 name=/s1/oradata07/PROD1/prod1_data01.dbf
RMAN-08522: input datafile fno=00001 name=/s1/oradata07/PROD1/system.dbf
RMAN-08011: *including current controlfile in backupset*
RMAN-08013: channel t1: piece 1 created
RMAN-08503: piece handle=PROD1_inc_0_040104:00:15_1.1815.rman comment=API Version 2.0,MMS Version 4.0.0.0

Oracle 9i has a feature called controlfile autobackup. Once this feature is enabled, the controlfile is automatically backed up at the end of every backup and will contain all of the information about the backup. This is activated using the configure command:

```
RMAN>configure controlfile autobackup on;
```

To repeat some of the above, there must be a knowledge of the contents of backupsets i.e. what they contain, when the backups were created, and the type of backups. Ideally there be logs of the RMAN backup sessions that produced the backupsets.

Note that all of the following anonymous PL/SQL blocks are run upon the instance of the database being recovered (the 'target'). The instance must be at least started (once the controlfile has been restored the database can also be mounted). Anonymous blocks can be executed in this manner as long as they call only 'fixed' packages. The DBMS_BACKUP_RESTORE packages are fixed.

IMPORTANT: All the anonymous blocks must be executed by SYS or a user who has execute privilege on SYS.DBMS_BACKUP_RESTORE.

6.1.4 Extracting the controlfile from a backupset

The first stage is to extract the controlfile from a backupset. This is achieved by making use of the following SYS.DBMS_BACKUP_RESTORE packaged functions & procedures:

FUNCTION deviceAllocate	- allocates a device for sequential I/O
PROCEDURE restoreSetDataFile	- begins a restore conversation
PROCEDURE restoreControlfileTo	- specifies the controlfile destination
PROCEDURE restoreBackupPiece	- performs the restore
PROCEDURE deviceDeallocate	- deallocates the I/O device

The following anonymous block can be created and executed to restore a controlfile from a backupset. Before executing it, the block MUST be edited as follows:

a. The filetable PL/SQL table entries must reflect the backuppieces comprising the backupset

b. The v_maxPieces variable must reflect the number of backuppieces comprising the backupset

c. The call to restoreControlfileTo must specify the correct controlfile path & filename

IMPORTANT: The latest backup of the controlfile should be restored. Because recovery (using backup controlfile) will be performed manually, the recovering session will need to start applying redo from the current log sequence AT THE TIME OF THE CONTROLFILE BACKUP.

Thus, to take advantage of incremental backups, restore a controlfile taken along with the incremental backups, and thereby reduce the amount of redo required during recovery.

```
DECLARE
        v_dev           varchar2(50);           -- device type allocated for restore
        v_done          boolean;                -- has the controlfile been fully extracted yet
        type t_file     Table is table of varchar2(255)  index by binary_integer;
        v_fileTable     t_fileTable;      -- Stores the backuppiece names
        v_maxPieces     number:=1;      -- Number of backuppieces in backupset
BEGIN
-- Initialise the filetable & number of backup pieces in the backupset
-- This section of code MUST be edited to reflect the available
-- backupset before the procedure is compiled and run. In this example, the
-- backupset consists of 4 pieces:
        v_fileTable(1):='fulldb_s15_p1';
        v_fileTable(2):='fulldb_s15_p2';
        v_fileTable(3):='fulldb_s15_p3';
        v_fileTable(4):='fulldb_s15_p4';
        v_maxPieces:=4;
-- Allocate a device. In this example, 'sbt_tape' is specified as the media manager
-- is used for reading backuppieces. If the backuppiece is on disk, specify type=>null
        v_dev:=sys.dbms_backup_restore.deviceAllocate(type=>'sbt_tape',
                        ident=>'t1');
-- Begin the restore conversation
        sys.dbms_backup_restore.restoreSetDatafile;
-- Specify where the controlfile is to be recreated
        sys.dbms_backup_restore.restoreControlfileTo(
                cfname=>'/support2/OFA_V804/u1/oradata/dbs/ctrl1V804.ctl');
-- Restore the controlfile
        FOR i IN 1..v_maxPieces LOOP
                sys.dbms_backup_restore.restoreBackupPiece(done=>v_done,
                        handle=>v_fileTable(i),
                        params=>null);
                IF v_done THEN
                        GOTO all_done;
                END IF;
        END LOOP;
<<all_done>>
-- Deallocate the device
        sys.dbms_backup_restore.deviceDeallocate;
END;
/
```

6.1.5 Extracting datafiles from a backupset

The second stage is to extract the datafiles from a backupset. This is achieved by making use of the following SYS.DBMS_BACKUP_RESTORE packaged functions & procedures:

```
        FUNCTION deviceAllocate             - allocates a device for sequential I/O
        PROCEDURE restoreSetDataFile        - begins a restore conversation
        PROCEDURE restoreDataFileTo         - datafile number & destination
        PROCEDURE restoreBackupPiece        - performs the restore
        PROCEDURE deviceDeallocate          - deallocates the I/O device
```

The following anonymous block can be created and executed to restore a datafile from a backupset. Before executing it, the block MUST be edited as follows:

a. The filetable PL/SQL table entries must reflect the backuppieces comprising the backupset

b. The v_maxPieces variable must reflect the number of backuppieces comprising the backupset

c. The call to restoreDataFileTo must specify the correct datafile number, and datafile path & filename

```
DECLARE
        v_dev           varchar2(50);           -- device type allocated for restore
        v_done boolean:=false;                  -- has the datafile been fully extracted yet
        type t_fileTable is table of varchar2(255)  index by binary_integer;
        v_fileTable     t_fileTable;            -- Stores the backuppiece names
        v_maxPieces     number:=1;              -- Number of backuppieces in backupset
BEGIN
-- Initialise the filetable & number of backup pieces in the backupset
-- This section of code MUST be edited to reflect the available
-- backupset before the procedure is compiled and run. In this example, the
-- backupset consists of 4 pieces:
        v_fileTable(1):='fulldb_s15_p1';
        v_fileTable(2):='fulldb_s15_p2';
        v_fileTable(3):='fulldb_s15_p3';
        v_fileTable(4):='fulldb_s15_p4';
        v_maxPieces:=4;
-- Allocate a device. In this example, 'sbt_tape' is specified as the media manager is used
-- to read backuppieces  If the backuppiece is on disk, specify type=>null
        v_dev:=sys.dbms_backup_restore.deviceAllocate(type=>'sbt_tape',
                ident=>'t1');
-- Begin the restore conversation
        sys.dbms_backup_restore.restoreSetDatafile;
-- Specify where the datafile is to be recreated
        sys.dbms_backup_restore.restoreDataFileTo(dfnumber=>1,
                toname=>'/support2/OFA_V804/u1/oradata/dbs/sysV804.dbf');
-- Restore the datafile
        FOR i IN 1..v_maxPieces LOOP
                sys.dbms_backup_restore.restoreBackupPiece(done=>v_done,
                        handle=>v_fileTable(i),
                        params=>null);
                IF v_done THEN
                        GOTO all_done;
                END IF;
        END LOOP;
<<all_done>>
-- Deallocate the device
        sys.dbms_backup_restore.deviceDeallocate;
END;
/
```

6.1.6 Applying incrementals

If incrementals are to be applied, the following anonymous block must be executed for each incremental datafile backup. The following SYS.DBMS_BACKUP_RESTORE packaged functions & procedures are called:

```
        FUNCTION deviceAllocate         - allocates a device for sequential I/O
        PROCEDURE applySetDataFile      - begins a restore conversation
        PROCEDURE applyDataFileTo       - datafile number & destination
        PROCEDURE applyBackupPiece      - performs the restore
        PROCEDURE deviceDeallocate      - deallocates the I/O device
```

The following anonymous block can be created and executed to restore a datafile from a backupset. Before executing it, the block MUST be edited as follows:

a. The filetable PL/SQL table entries must reflect the backuppieces comprising the backupset

b. The v_maxPieces variable must reflect the number of backuppieces comprising the backupset

c. The call to applyDataFileTo must specify the correct datafile number, and datafile path & filename

```
DECLARE
        v_dev           varchar2(50);           -- device type allocated for restore
        v_done          boolean:=false;         -- has the datafile been fully extracted yet
        type            t_fileTable is table of varchar2(255)
                        index by binary_integer;
        v_fileTable     t_fileTable;            -- Stores the backuppiece name
        v_maxPieces     number:=1;              -- Number of backuppieces in backupset
BEGIN
-- Initialise the filetable & number of backup pieces in the backupset
-- This section of code MUST be edited to reflect the available
-- backupset before the procedure is compiled and run. In this example, the
-- backupset consists of 1 piece, a level 2 backupset:

        v_fileTable(1):='fulldb_level2_s18_p1';
        v_maxPieces:=1;

-- Allocate a device. In this example, 'sbt_tape' is specified as the media manager is used
-- to read backuppieces  If the backuppiece is on disk, specify type=>null

        v_dev:=sys.dbms_backup_restore.deviceAllocate(type=>'sbt_tape',
                        ident=>'t1');

-- Begin the restore conversation

        sys.dbms_backup_restore.applySetDataFile;

-- Specify where the datafile is to be recreated

        sys.dbms_backup_restore.applyDataFileTo(dfnumber=>1,
                        toname=>'/support2/OFA_V804/u1/oradata/dbs/sysV804.dbf');
-- Restore the datafile
        FOR i IN 1..v_maxPieces LOOP
                sys.dbms_backup_restore.applyBackupPiece(done=>v_done,
                        handle=>v_fileTable(i),
                        params=>null);
                IF v_done THEN
                        GOTO all_done;
                END IF;
        END LOOP;
<<all_done>>
        -- Deallocate the device
        sys.dbms_backup_restore.deviceDeallocate;
END;
/
```

6.1.7 Extracting archivelogs from a backupset

The last restore stage is to extract the archivelogs from a backupset. This is achieved by making use of the following SYS.DBMS_BACKUP_RESTORE packaged functions & procedures:

```
        FUNCTION deviceAllocate             - allocates a device for sequential I/O
        PROCEDURE restoreSetArchivedLog     - begins a restore conversation
        PROCEDURE restoreArchivedLog        - archivelog sequence & thread numbers
```

```
PROCEDURE restoreBackupPiece        - performs the restore
PROCEDURE deviceDeallocate          - deallocates the I/O device
```

The following anonymous block can be created and executed to restore an archivelog from a backupset. Before executing it, the block MUST be edited as follows:

a. The filetable PL/SQL table entries must reflect the backuppieces comprising the backupset

b. The v_maxPieces variable must reflect the number of backuppieces comprising the backupset

c. The call to restoreSetArchivedLog must specify the destination where the archivelog is to be restored. Ideally the destination string should be the same as init.ora:log_archive_dest

d. The call to restoreArchivedLog must specify the log sequence number and thread number of the archivelog

```
DECLARE
        v_dev           varchar2(50);   -- device type allocated for restore
        v_done          boolean:=false; -- has the log been fully extracted yet
        type t_fileTable is table of varchar2(255) index by binary_integer;
        v_fileTable     t_fileTable;    -- Stores the backuppiece names
        v_maxPieces     number:=1;      -- Number of backuppieces in backupset
BEGIN
-- Initialise the filetable & number of backup pieces in the backupset
-- This section of code MUST be edited to reflect the available
-- backupset before the procedure is compiled and run. In this example, the
-- archivelog backupset consists of 2 pieces:
        v_fileTable(1):='al_s20_p1';
        v_fileTable(2):='al_s20_p2';
        v_maxPieces:=2;
-- Allocate a device. In this example, 'sbt_tape' is specified as the media manager is used
-- to read backuppieces  If the backuppiece is on disk, specify type=>null
        v_dev:=sys.dbms_backup_restore.deviceAllocate(type=>'sbt_tape',
                ident=>'t1');
-- Begin the restore conversation
        sys.dbms_backup_restore.restoreSetArchivedLog(
                destination=>'/support2/OFA_V804/app/oracle/admin/arch/arch_');
-- Specify where the archivelog is to be recreated
        sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                sequence=>100);
-- Restore the archivelog
        FOR i IN 1..v_maxPieces LOOP
                sys.dbms_backup_restore.restoreBackupPiece(done=>v_done,
                                handle=>v_fileTable(i),
                                params=>null);
                IF v_done THEN
                        GOTO all_done;
                END IF;
        END LOOP;
<<all_done>>
-- Deallocate the device
        sys.dbms_backup_restore.deviceDeallocate;
END;
/
```

For restoring multiple archives from a backupset, add a loop around sys.dbms_backup_restore.restoreArchivedLog():

```
        for seq in <min seq#>..<max seq#> loop
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>seq);
        end loop
```

6.1.8 A typical scenario - outline

A database backupsets consisting of:

- an incremental level 0 database backup
- an incremental level 2 database backup
- archivelogs from the time of the level 2 backup to the current time

The target database and recovery catalog have been irretrievably lost.

In this situation, the following steps should be followed (using the above anonymous blocks):

1) Start the target instance (nomount)
2) Restore the latest controlfile, ideally from the same backupset as the last incremental to be restored (make further copies if necessary as per the init.ora)
3) Mount the database
4) Restore the datafiles from the level 0 backupset
5) Restore (apply) the datafiles from the level 2 backupset
6) Restore the archivelogs from the archivelog backupset
7) Using traditional v7 recovery techniques, recover the database (until cancel using backup controlfile)
8) Open the database (resetlogs)
9) Rebuild the recovery catalog & re-register the target database
10) Make backups of the target database and recovery catalog database

6.1.9 Possible Errors that may be encounted

ORA-19615 & ORA-19613 when attempting to extract files

Errorstack:

ORA-19583: conversation terminated due to error

ORA-19615: some files not found in backup set

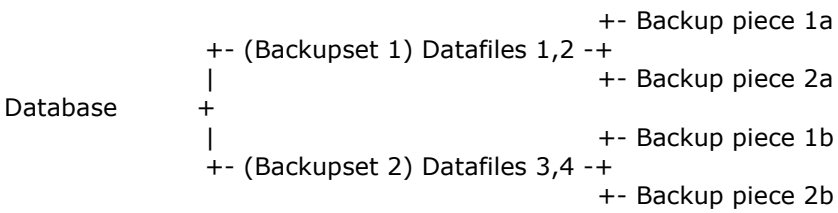
ORA-19613: datafile <file#> not found in backup set

ORA-06512: at "SYS.X$DBMS_BACKUP_RESTORE", line 1043

ORA-06512: at line 40

The problem is that one or more backup pieces specified in the v_fileTable table contain NO blocks for the datafile that you are trying to extract.

For example, an RMAN backup may have been run and allocated 2 channels to backup the (4 datafile) database. This will create 2 backupsets.



Although the backup pieces may contain blocks from all datafiles associated with their backupset, they will not contain blocks from a different backupset i.e. pieces 1a and 1b will NOT contain blocks from datafiles 3 or 4.

If it is desired to restore datafile 1, and include either backup pieces 1b or 2b in v_fileTable, an errorstack as above will be generated.

This is why it is important to know what files are in what backupset. The original RMAN backup log will assist here.

6.1.10 Processes and scripts used (case study)

When everything on a server is lost, there are a few things that need to be restored before it is possible to restore the database files. They are:

- Restore or re-install your Oracle software.
- Restore or recreate the oratab file.
- Restore or recreate the database's initSID.ora file

After these are done, it is possible to continue to restore your database files. The restore examples, which follow, assume that these 3 tasks have already been completed.

6.1.10.1 Restore example #1

If there is a copy of the controlfile on tape, restore the file using DBMS_BACKUP_RESTORE. Then, copy it to all of the controlfile locations. At that time the standard RMAN process can be used to restore the database. Here are the steps in detail:

1. Set the Oracle environment for the SID to be restored by running oraenv.
2. Log in to sqlplus as sysdba (or log in to Server Manager and connect internal for Oracle versions less than 9i). Start the database instance using the 'startup nomount' command.
3. Restore the controlfile using DBMS_BACKUP_RESTORE. Execute a PL/SQL block similar to the one below from within the sqlplus (or svrmgrl) session.

```
spool rest_ctl.out
DECLARE
      devtype varchar2(256);
      done boolean;
BEGIN
-- Allocate a device
      devtype:=sys.dbms_backup_restore.deviceAllocate(
            type=>'sbt_tape',
            ident=>'t1');
-- Begin restore
      sys.dbms_backup_restore.restoreSetDatafile;
-- Specify where controlfie is to be restored
      sys.dbms_backup_restore.restoreControlfileTo(
            cfname=>'/tmp/cont01_PROD1.ctl');
-- Restore the controlfile
      sys.dbms_backup_restore.restoreBackupPiece(
            done=>done,
            handle=>'PROD1_inc_0_20030911:23:31:00_1.730.rman',
            params=>null);
-- Deallocate the device
      sys.dbms_backup_restore.deviceDeallocate;
END;
/
```

4. Shutdown the database instance.
5. Copy the controlfile to all of the controlfile locations listed in the init<SID>.ora file.
6. Log in to sqlplus as sysdba (or log in to Server Manager and connect internal for Oracle versions less than 9i). Start up and mount the database instance using the 'startup mount' command.
7. Restore the rest of the database files using RMAN commands. For example:

```
$rman target userid/password
RMAN> run
{
allocate channel t1 type 'SBT_TAPE';
allocate channel t2 type 'SBT_TAPE';
restore database;
release channel t1;
release channel t2;
}
```

6.1.10.2 Restore example #2

The situation where a database backup has the 'backup current controlfile' command executed in the same run statement as the 'database full backup' command. As a result, the controlfile will be copied to tape as part of the first backup piece and will not contain information about subsequent backup pieces. To restore this database, it is necessary to restore the controlfile first and then restore the datafiles separately using a second PL/SQL program.

Perform steps #1 through # 5 from Restore example #1. This will recover the controlfile from tape, and it can be placed in the appropriate locations.

6. Log in to sqlplus as sysdba (or log in to Server Manager and connect internal for Oracle versions less than 9i). Start the database instance using the 'startup mount' command.
7. Restore the datafiles using DBMS_BACKUP_RESTORE. Execute a PL/SQL block similar to the one below from within the sqlplus (or svrmgrl) session.

```
spool rest_files.out
DECLARE
        devtype varchar2(256);
        done boolean;
BEGIN
-- Allocate a device
        devtype:=sys.dbms_backup_restore.deviceAllocate(
                type=>'sbt_tape',
                ident=>'t1');
-- Begin restore
        sys.dbms_backup_restore.restoreSetDatafile;
-- Specify where datafiles are to be restored
        sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>22,
                toname=>'/u01/oradata/PROD1/batch102.dbf');
        sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>24,
                toname=>'/u01/oradata/PROD1/batch104.dbf');
        sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>26,
                toname=>'/u01/oradata/PROD1/batch106.dbf');
        sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>28,
                toname=>'/u01/oradata/PROD1/batch108.dbf');
        sys.dbms_backup_restore.restoreDatafileTo(dfnumber=>30,
                toname=>'/u01/oradata/PROD1/batch110.dbf');
-- Restore the file
        sys.dbms_backup_restore.restoreBackupPiece(
                done=>done,
                handle=>'PROD1_inc_0_20030912:23:31:00_1.742.rman',
```

```
                params=>null);
-- Deallocate the device
        sys.dbms_backup_restore.deviceDeallocate;
END;
/
```

8. Restore all of the archive logs that are possible. If they were backed up using RMAN it is possible to use RMAN commands, or use PL/SQL shown in Restore example #3.
9. After the datafiles and archive logs have been restored, startup and mount the database.
10. Since the controlfile restored from tape will be older than the archive logs, the most recent archive logs will not be recorded in it. To force Oracle to apply the logs anyway use the command 'recover database using backup controlfile until cancel'. Apply all of the logs that are available. When there are no more logs to apply enter CANCEL at the prompt.
11. Open the database using the command 'alter database open resetlogs;'.
12. Take another backup because you have reset the logs.

6.1.10.3 Restore example #3

It may be necessary to restore the archive logs using PL/SQL. First restore the RMAN backup log file from an operating system backup to tape. The RMAN backup log should contain the information needed to restore the archive log files.

```
<From the RMAN backup log: >
RMAN-08009: channel t1: starting archivelog backupset
RMAN-08502: set_count=1816 set_stamp=514513562 creation_time=2004-01-
04:00:26:02
RMAN-08014: channel t1: specifying archivelog(s) in backup set
RMAN-08504: input archivelog thread=1 sequence=8034 recid=6933 stamp=514080911
RMAN-08504: input archivelog thread=1 sequence=8035 recid=6934 stamp=514090811
RMAN-08504: input archivelog thread=1 sequence=8036 recid=6935 stamp=514112406
RMAN-08504: input archivelog thread=1 sequence=8037 recid=6936 stamp=514113015
RMAN-08504: input archivelog thread=1 sequence=8038 recid=6937 stamp=514115690
RMAN-08504: input archivelog thread=1 sequence=8039 recid=6938 stamp=514117846
```

1. Set the Oracle environment for the SID to be restored by running oraenv.
2. Log in to sqlplus as sysdba (or log in to Server Manager and connect internal for Oracle versions less than 9i). Start the database instance using the 'startup nomount' command. If all of your datafiles have been restored it is possible to mount the database, but that isn't required to run this PL/SQL.
3. Restore the archive logs using DBMS_BACKUP_RESTORE. Execute a PL/SQL block similar to the one below from within the sqlplus (or svrmgrl) session.

```
spool rest_logs.out
DECLARE
        devtype varchar2(256);
        done boolean;
BEGIN
-- Allocate a device
        devtype:=sys.dbms_backup_restore.deviceAllocate(
                type=>'sbt_tape',
                ident=>'t1');
-- Begin the restore converstation
-- Specify where the archivelog is to be recreated
        sys.dbms_backup_restore.restoreSetArchivedLog(
                destination=>'/archive');
        sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                sequence=>8252);
```

```
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>8253);
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>8254);
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>8255);
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>8256);
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>8257);
                sys.dbms_backup_restore.restoreArchivedLog(thread=>1,
                        sequence=>8258);
        -- Restore the archivelog
                sys.dbms_backup_restore.restoreBackupPiece(
                        done=>done,
                        handle=>'PROD1_inc_A_20030912:17:00:00_1.741.rman',
                        params=>null);
        -- Deallocate the device
                sys.dbms_backup_restore.deviceDeallocate;
        END;
        /
```

4. Mount the database if it isn't already mounted.
5. If the controlfile restored from tape is older than these archive logs, force Oracle to apply the logs. Do this using the command 'recover database using backup controlfile until cancel;'. Apply all of the logs that are available. When there are no more logs to apply enter CANCEL at the prompt.
6. Open the database using the command 'alter database open resetlogs;'
7. Take another backup since the logs have been reset.

6.1.11 Possible exhancements

If the procedures fail with an unhandled exception (quite likely, as no exception handlers have been set up), the allocated device does not get deallocated. This is unfriendly (the user must exit & restart the session) and should be addressed.

6.2 Summary

- Become familiar with the package DBMS_BACKUP_RESTORE.
- Name the backup pieces with unique and meaningful names.
- Increase the value of CONTROL_FILE_RECORD_KEEP_TIME if it is desired to have more than seven day's worth of backup history in the controlfile.
- Backup the controlfiles separately after each backup.
- Export the recovery catalog database often so there is no need to use PL/SQL for restores.
- Test the restore and recovery processes regularly!

7 Using RMAN to duplicate a database.

Oracle 9i introduces the RMAN 'duplicate database' feature. All the examples the author has encountered have been of a simple duplication of a database upon the same node. It is a requirement to be able to use this feature across different nodes and hence this section is intended to address this issue with specific reference to Tivoli.

7.1 Duplicate Database using RMAN in Oracle9i

You can use the RMAN DUPLICATE command to create a duplicate database from target database backups while still retaining the original target database. The duplicate database can be either identical to the original database or contain only a subset of the original tablespaces.

To prepare for database duplication, one must first create an auxiliary instance. For the duplication to work correctly there is a need to connect RMAN to both the target (primary) database and an auxiliary instance started in NOMOUNT mode. There must be at least one auxiliary channel allocated on the auxiliary instance.

The principal work of the duplication is performed by the auxiliary channel, which starts a server session on the auxiliary host. This channel then restores the necessary backups of the primary database, uses them to create the duplicate database, and initiates recovery.

As part of the duplicating operation, RMAN manages the following:

1. Restores the target datafiles to the duplicate database and performs incomplete recovery by using all available incremental backups and archived logs.
2. Shuts down and starts the auxiliary database.
3. Opens the duplicate database with the RESETLOGS option after incomplete recovery to create the online redo logs.
4. Generates a new, unique DBID for the auxiliary database.

During duplication, RMAN must perform incomplete recovery because the online redo logs in the target are not backed up and cannot be applied to the auxiliary database. The furthest that RMAN can go in recovery of the duplicate database is the most recent redo log archived by the target database.

When duplicating a database, one can do the following:

1. Run the DUPLICATE command with or without a recovery catalog
2. Skip read-only tablespaces with the SKIP READONLY clause.

   Read-only tablespaces are included by default.

   If they are omitted then they can be added later.
3. Exclude tablespaces from the auxiliary database with the SKIP TABLESPACE clause. It is not possible to skip the SYSTEM tablespace or tablespaces containing rollback or undo segments.
4. Create the auxiliary database in a new host. If the directory structure is the same on the new host, then one can specify the NOFILENAMECHECK option and reuse the target datafile filenames for the auxiliary datafiles.
5. Use the SET UNTIL command or DUPLICATE command with the UNTIL clause when creating the auxiliary database to recover it to a noncurrent time.

By default, the DUPLICATE command creates the database by using the most recent backups of the target database and then performs recovery to the most recent consistent point contained in the incremental backups and archived logs.

Register the auxiliary database in the same recovery catalog as the target database. This option is possible because RMAN gives the duplicate database a new DBID during duplication.

Oracle Note 265595.1 describes how to achieve restores with TIVOLI DATAPROTECTION FOR ORACLE (TDPO) and Oracle Recovery Manager (RMAN) on a different host, than the backup host. The note is for TDPO releases only, which use TDPO_OPTFILE environment. I.e. Release 2.2.0 and above. (NOT for ADSM Connect Agent). There were significant changes at release 2.2.0 so this is not applicable at earlier releases/

7.2 Alternate client restore with RMAN and Tivoli

The alternate client restores functionality means that there is a need to restore backups on a different host like the backup host.

This is needed in several circumstances:

- Creation of a test database with DUPLICATE DATABASE command
- Creation of a standby database using DUPLICATE DATABASE ... FOR STANDBY
- Restoring Database for Point-in-Time-Recovery (PITR)
- Restoring parts of database for Tablespace-Point-in-Time-Recovery (TSPITR)
- Restoring full database after disaster, when original host no longer exists

The note 265595.1 describes the simple situation, and the reader is referred to that note if interested. Some of the set up steps are identical.

The situation that will be described concerns a database called BDAW located on a node called "src_node". The requirement is to duplicate the database to be named EDAW upon a node called "dest_node".

7.2.1 Preparing the Auxiliary Instance for Duplication:

7.2.1.1 Basic Steps

Perform these tasks before performing RMAN duplication:

Task 1: setenv ORACLE_SID AUX

Create an Oracle Password File for the Auxiliary Instance

Task 2: Ensure Oracle Net Connectivity to the Auxiliary Instance

It is necessary to connect to the auxiliary instance with SYSDBA privileges, so a password file must exist.

Task 3: Create an Initialization Parameter File for the Auxiliary Instance. Certain mandatory initialization parameter settings are required for the auxiliary database:

db_block_size = <same size as the target>

DB_NAME=EDAW

compatible = 9.2.0.0 /* should be the same as the target

CONTROL_FILES=xxxxxxxxx

Alternatively, and this is the easiest, copy the existing database pfile and modify the values to suit.

Task 4: Start the Auxiliary Instance NOMOUNT

CONNECT / AS SYSDBA

STARTUP FORCE NOMOUNT

Task 5: Mount or Open the Target Database

Task 6: Make Sure You Have the Necessary Backups and Archived Redo Logs

Task 7: Allocate Auxiliary Channels if Automatic Channels Are Not Configured

Start RMAN with a connection to the target database, the auxiliary database, and (if you use one) the recovery catalog database.

It is possible to start the RMAN executable on any host so long as it can connect to all the instances.

Note that if the auxiliary instance requires a client-side initialization parameter file, then this file must exist on the same host that runs the RMAN executable.

If there are no automatic channels configured, then before issuing the DUPLICATE command, manually allocate at least one auxiliary channel within the same RUN command.

If the backups reside on disk, then the more channels you allocate, the faster the duplication will be.

For tape backups, limit the number of channels to the number of devices available for the operation.

All backups taken with RMAN are stored on the TSM Server together with the node name (TDPO_NODE) of the backup host. This information is stored in the TDPO client configuration file called "TDPO_OPTFILE", which is referenced by RMAN at allocating a tape (SBT_TAPE) channel:

```
allocate channel tivoli type 'SBT_TAPE' parms
'ENV=(TDPO_OPTFILE=/usr/tivoli/tsm/client/oracle/bin64/tdpo.opt)';
```

OR in the persistent channel configuration template:

```
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'ENV=(TDPO_OPTFILE=/usr/tivoli/tsm/client/oracle/bin64/tdpo.opt)';
```

NOTE: TDPO_OPTFILE is case sensitive and must be provided in UPPERCASE only.

Our client configuration file on host "src_node":

```
************************************************************************
* IBM Tivoli Storage Manager for Databases
* Data Protection for Oracle
*
* Sample tdpo.opt for the AIX Data Protection for Oracle 64bit
************************************************************************

*DSMI_ORC_CONFIG    /usr/tivoli/tsm/client/api/bin64/dsm.opt
*DSMI_LOG           <dirname>

TDPO_FS             src_node_db
TDPO_NODE           src_node_oracle
*TDPO_OWNER         <username>
*TDPO_PSWDPATH      /usr/tivoli/tsm/client/oracle/bin64

*TDPO_DATE_FMT      1
```

```
*TDPO_NUM_FMT       1
*TDPO_TIME_FMT      1

*TDPO_MGMT_CLASS_2   mgmtclass2
*TDPO_MGMT_CLASS_3   mgmtclass3
*TDPO_MGMT_CLASS_4   mgmtclass4
```

In this example TDPO_NODE is called "src_node_oracle". The filespace "src_node_db" is used to logically separate the different database backups on this host.

Task 8: Create a separate client configuration, where the TDPO_NODE is named "src_node_oracle" and the TDPO_FS is "src_node_db" upon the node that we want to restore to. In this case "dest_node".

We can copy over the actual tdpo.opt file from 'src_node' and name it to tdpo_src_node.opt on node "dest_node". We also need to create a separate passwordfile for this configuration (or since we are using a standard installation setup, copy the password file from the source node 'src_node'). The password file will be named 'TDPO.src_node_oracle'.

To create a new file use the command:

```
# tdpoconf passw -tdpo_optfile=tdpo_src_node.opt

************************************************************
*       Tivoli Data Protection for Oracle Utility          *
*  Password file initialization/update program             *
*     ROOT privilege needed to update value                *
************************************************************

Please enter current password:

Please enter new password:

Please reenter new password for verification:

ANS1025E (RC137)  Session rejected: Authentication failure
```

The password is node specific on TSM server and thus we have to use password for the src_node_oracle node!!! After using correct TSM password it works:

```
ANU0260I Password successfully changed.
```

You can verify TDPO configuration with this command:

```
# tdpoconf showenv -tdpo_optfile=tdpo_src_node.opt
```

The example uses the following:

- A recovery catalog.
- The target database BDAW is on src_node and contains 11 datafiles.
- The target is to be duplicated to database EDAW on the host dest_node.
- The datafiles for EDAW are to the /u0?/oradata/EDAW/ subdirectories.
- The restore of the duplicate db is to the current time.
- There are to be three online redo logs groups, each with two members of size 25600 KB.
- The channel device is to be sbt_tape.

- The auxiliary instance EDAW has initialization parameter file in the default location (so the PFILE parameter is not necessary on the DUPLICATE command).
- RMAN is started from the EDAW node.

Note: The account that is connecting to the target and auxiliary databases must be able to connect as SYSDBA.

The following is the log generated from the run. It includes the commands issued. [Passwords are deliberately obscured.]

```
Recovery Manager: Release 9.2.0.5.0 - 64bit Production
Copyright (c) 1995, 2002, Oracle Corporation.  All rights reserved.

RMAN> connect target system/xxxx@BDAW
2> connect catalog rman/xxxx@CAT
3> connect auxiliary sys/xxxxx@EDAW
4> run {
5> allocate auxiliary channel tivoli type 'SBT_TAPE' FORMAT '%d_%U_%s' parms
'ENV=(TDPO_OPTFILE=/usr/tivoli/tsm/client/oracle/bin64/tdpo_src_node.opt)';
6>    set newname for datafile 1 to '/u01/oradata/EDAW/system01.dbf';
7>    set newname for datafile 2 to '/u02/oradata/EDAW/undotbs01.dbf';
8>    set newname for datafile 3 to '/u01/oradata/EDAW/CONTHIST_DATA.dbf';
9>    set newname for datafile 4 to '/u02/oradata/EDAW/CONTHIST_INDX.dbf';
10>   set newname for datafile 5 to '/u01/oradata/EDAW/DAW_DATA.dbf';
11>   set newname for datafile 6 to '/u02/oradata/EDAW/DAW_INDX.dbf';
12>   set newname for datafile 7 to '/u01/oradata/EDAW/LARGE_DATA.dbf';
13>   set newname for datafile 8 to '/u02/oradata/EDAW/LARGE_INDX.dbf';
14>   set newname for datafile 9 to '/u01/oradata/EDAW/SMALL_DATA.dbf';
15>   set newname for datafile 10 to '/u02/oradata/EDAW/small_indx01.dbf';
16>   set newname for datafile 11 to '/u01/oradata/EDAW/tools01.dbf';
17>   duplicate target database to EDAW
18>      logfile
19>         group 1(
20>               '/u01/oradata/EDAW/edaw_log1a.rdo'
21>               ,'/u02/oradata/EDAW/edaw_log1b.rdo'
22>         ) size 25600k reuse,
23>         group 2(
24>               '/u01/oradata/EDAW/edaw_log2a.rdo'
25>               ,'/u02/oradata/EDAW/edaw_log2b.rdo'
26>         ) size 25600k reuse,
27>         group 3(
28>               '/u01/oradata/EDAW/edaw_log3a.rdo'
29>               ,'/u02/oradata/EDAW/edaw_log3b.rdo'
30>          ) size 25600k reuse;
31> }
32>
33>
connected to target database: BDAW (DBID=1706142598)
connected to recovery catalog database
connected to auxiliary database: EDAW (not mounted)
allocated channel: tivoli
channel tivoli: sid=10 devtype=SBT_TAPE
channel tivoli: Tivoli Data Protection for Oracle: version 5.2.0.0
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
Starting Duplicate Db at 22-JUN-05
printing stored script: Memory Script
{
   set until scn  19121262;
   set newname for datafile  1 to
 "/u01/oradata/EDAW/system01.dbf";
   set newname for datafile  2 to
 "/u02/oradata/EDAW/undotbs01.dbf";
   set newname for datafile  3 to
 "/u01/oradata/EDAW/CONTHIST_DATA.dbf";
   set newname for datafile  4 to
 "/u02/oradata/EDAW/CONTHIST_INDX.dbf";
   set newname for datafile  5 to
```

```
 "/u01/oradata/EDAW/DAW_DATA.dbf";
   set newname for datafile  6 to
 "/u02/oradata/EDAW/DAW_INDX.dbf";
   set newname for datafile  7 to
 "/u01/oradata/EDAW/LARGE_DATA.dbf";
   set newname for datafile  8 to
 "/u02/oradata/EDAW/LARGE_INDX.dbf";
   set newname for datafile  9 to
 "/u01/oradata/EDAW/SMALL_DATA.dbf";
   set newname for datafile  10 to
 "/u02/oradata/EDAW/small_indx01.dbf";
   set newname for datafile  11 to
 "/u01/oradata/EDAW/tools01.dbf";
   restore
   check readonly
   clone database
   ;
}
executing script: Memory Script
executing command: SET until clause
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
executing command: SET NEWNAME
Starting restore at 22-JUN-05
channel tivoli: starting datafile backupset restore
channel tivoli: specifying datafile(s) to restore from backup set
restoring datafile 00001 to /u01/oradata/EDAW/system01.dbf
restoring datafile 00002 to /u02/oradata/EDAW/undotbs01.dbf
restoring datafile 00003 to /u01/oradata/EDAW/CONTHIST_DATA.dbf
restoring datafile 00004 to /u02/oradata/EDAW/CONTHIST_INDX.dbf
restoring datafile 00005 to /u01/oradata/EDAW/DAW_DATA.dbf
restoring datafile 00006 to /u02/oradata/EDAW/DAW_INDX.dbf
restoring datafile 00007 to /u01/oradata/EDAW/LARGE_DATA.dbf
restoring datafile 00008 to /u02/oradata/EDAW/LARGE_INDX.dbf
restoring datafile 00009 to /u01/oradata/EDAW/SMALL_DATA.dbf
restoring datafile 00010 to /u02/oradata/EDAW/small_indx01.dbf
restoring datafile 00011 to /u01/oradata/EDAW/tools01.dbf
channel tivoli: restored backup piece 1
piece handle=BDAW_g2gni4qj_1_1_1538 tag=TAG20050621T191529 params=NULL
channel tivoli: restore complete
Finished restore at 22-JUN-05
sql statement: CREATE CONTROLFILE REUSE SET DATABASE "EDAW" RESETLOGS ARCHIVELOG
  MAXLOGFILES     10
  MAXLOGMEMBERS      5
  MAXDATAFILES      100
  MAXINSTANCES     1
  MAXLOGHISTORY      226
 LOGFILE
  GROUP  1 ( '/u01/oradata/EDAW/edaw_log1a.rdo', '/u02/oradata/EDAW/edaw_log1b.rdo' ) SIZE   26214400  REUSE,
  GROUP  2 ( '/u01/oradata/EDAW/edaw_log2a.rdo', '/u02/oradata/EDAW/edaw_log2b.rdo' ) SIZE   26214400  REUSE,
  GROUP  3 ( '/u01/oradata/EDAW/edaw_log3a.rdo', '/u02/oradata/EDAW/edaw_log3b.rdo' ) SIZE   26214400  REUSE
 DATAFILE
  '/u01/oradata/EDAW/system01.dbf'
 CHARACTER SET WE8ISO8859P1

printing stored script: Memory Script
{
   switch clone datafile all;
}
executing script: Memory Script

datafile 2 switched to datafile copy
input datafilecopy recid=1 stamp=561649367 filename=/u02/oradata/EDAW/undotbs01.dbf
datafile 3 switched to datafile copy
input datafilecopy recid=2 stamp=561649368 filename=/u01/oradata/EDAW/CONTHIST_DATA.dbf
datafile 4 switched to datafile copy
input datafilecopy recid=3 stamp=561649368 filename=/u02/oradata/EDAW/CONTHIST_INDX.dbf
datafile 5 switched to datafile copy
input datafilecopy recid=4 stamp=561649368 filename=/u01/oradata/EDAW/DAW_DATA.dbf
datafile 6 switched to datafile copy
input datafilecopy recid=5 stamp=561649368 filename=/u02/oradata/EDAW/DAW_INDX.dbf
datafile 7 switched to datafile copy
input datafilecopy recid=6 stamp=561649368 filename=/u01/oradata/EDAW/LARGE_DATA.dbf
datafile 8 switched to datafile copy
```

```
input datafilecopy recid=7 stamp=561649368 filename=/u02/oradata/EDAW/LARGE_INDX.dbf
datafile 9 switched to datafile copy
input datafilecopy recid=8 stamp=561649368 filename=/u01/oradata/EDAW/SMALL_DATA.dbf
datafile 10 switched to datafile copy
input datafilecopy recid=9 stamp=561649368 filename=/u02/oradata/EDAW/small_indx01.dbf
datafile 11 switched to datafile copy
input datafilecopy recid=10 stamp=561649368 filename=/u01/oradata/EDAW/tools01.dbf

printing stored script: Memory Script
{
   set until scn  19121262;
   recover
   clone database
    delete archivelog
   ;
}
executing script: Memory Script
executing command: SET until clause
Starting recover at 22-JUN-05
starting media recovery
channel tivoli: starting archive log restore to default destination
channel tivoli: restoring archive log
archive log thread=1 sequence=1559
channel tivoli: restored backup piece 1
piece handle=BDAW_g3gni4tq_1_1_1539 tag=TAG20050621T191713 params=NULL
channel tivoli: restore complete
archive log filename=/u02/oraarchive/EDAW/edaw_1559.arc thread=1 sequence=1559
channel clone_default: deleting archive log(s)
archive log filename=/u02/oraarchive/EDAW/edaw_1559.arc recid=1 stamp=561649693
media recovery complete
Finished recover at 22-JUN-05

printing stored script: Memory Script
{
   shutdown clone;
   startup clone nomount ;
}
executing script: Memory Script
database dismounted
Oracle instance shut down

connected to auxiliary database (not started)
Oracle instance started

Total System Global Area     202868072 bytes

Fixed Size                   742760 bytes
Variable Size              134217728 bytes
Database Buffers            67108864 bytes
Redo Buffers                798720 bytes
sql statement: CREATE CONTROLFILE REUSE SET DATABASE "EDAW" RESETLOGS ARCHIVELOG
  MAXLOGFILES     10
  MAXLOGMEMBERS      5
  MAXDATAFILES      100
  MAXINSTANCES     1
  MAXLOGHISTORY      226
 LOGFILE
  GROUP  1 ( '/u01/oradata/EDAW/edaw_log1a.rdo', '/u02/oradata/EDAW/edaw_log1b.rdo' ) SIZE   26214400  REUSE,
  GROUP  2 ( '/u01/oradata/EDAW/edaw_log2a.rdo', '/u02/oradata/EDAW/edaw_log2b.rdo' ) SIZE   26214400  REUSE,
  GROUP  3 ( '/u01/oradata/EDAW/edaw_log3a.rdo', '/u02/oradata/EDAW/edaw_log3b.rdo' ) SIZE   26214400  REUSE
 DATAFILE
  '/u01/oradata/EDAW/system01.dbf'
 CHARACTER SET WE8ISO8859P1

printing stored script: Memory Script
{
   catalog clone datafilecopy  "/u02/oradata/EDAW/undotbs01.dbf";
   catalog clone datafilecopy  "/u01/oradata/EDAW/CONTHIST_DATA.dbf";
   catalog clone datafilecopy  "/u02/oradata/EDAW/CONTHIST_INDX.dbf";
   catalog clone datafilecopy  "/u01/oradata/EDAW/DAW_DATA.dbf";
   catalog clone datafilecopy  "/u02/oradata/EDAW/DAW_INDX.dbf";
   catalog clone datafilecopy  "/u01/oradata/EDAW/LARGE_DATA.dbf";
   catalog clone datafilecopy  "/u02/oradata/EDAW/LARGE_INDX.dbf";
   catalog clone datafilecopy  "/u01/oradata/EDAW/SMALL_DATA.dbf";
   catalog clone datafilecopy  "/u02/oradata/EDAW/small_indx01.dbf";
   catalog clone datafilecopy  "/u01/oradata/EDAW/tools01.dbf";
   switch clone datafile all;
}
executing script: Memory Script

cataloged datafile copy
datafile copy filename=/u02/oradata/EDAW/undotbs01.dbf recid=1 stamp=561649705
```

```
cataloged datafile copy
datafile copy filename=/u01/oradata/EDAW/CONTHIST_DATA.dbf recid=2 stamp=561649705
cataloged datafile copy
datafile copy filename=/u02/oradata/EDAW/CONTHIST_INDX.dbf recid=3 stamp=561649706
cataloged datafile copy
datafile copy filename=/u01/oradata/EDAW/DAW_DATA.dbf recid=4 stamp=561649706
cataloged datafile copy
datafile copy filename=/u02/oradata/EDAW/DAW_INDX.dbf recid=5 stamp=561649706
cataloged datafile copy
datafile copy filename=/u01/oradata/EDAW/LARGE_DATA.dbf recid=6 stamp=561649706
cataloged datafile copy
datafile copy filename=/u02/oradata/EDAW/LARGE_INDX.dbf recid=7 stamp=561649706
cataloged datafile copy
datafile copy filename=/u01/oradata/EDAW/SMALL_DATA.dbf recid=8 stamp=561649706
cataloged datafile copy
datafile copy filename=/u02/oradata/EDAW/small_indx01.dbf recid=9 stamp=561649706
cataloged datafile copy
datafile copy filename=/u01/oradata/EDAW/tools01.dbf recid=10 stamp=561649706
datafile 2 switched to datafile copy
input datafilecopy recid=1 stamp=561649705 filename=/u02/oradata/EDAW/undotbs01.dbf
datafile 3 switched to datafile copy
input datafilecopy recid=2 stamp=561649705 filename=/u01/oradata/EDAW/CONTHIST_DATA.dbf
datafile 4 switched to datafile copy
input datafilecopy recid=3 stamp=561649706 filename=/u02/oradata/EDAW/CONTHIST_INDX.dbf
datafile 5 switched to datafile copy
input datafilecopy recid=4 stamp=561649706 filename=/u01/oradata/EDAW/DAW_DATA.dbf
datafile 6 switched to datafile copy
input datafilecopy recid=5 stamp=561649706 filename=/u02/oradata/EDAW/DAW_INDX.dbf
datafile 7 switched to datafile copy
input datafilecopy recid=6 stamp=561649706 filename=/u01/oradata/EDAW/LARGE_DATA.dbf
datafile 8 switched to datafile copy
input datafilecopy recid=7 stamp=561649706 filename=/u02/oradata/EDAW/LARGE_INDX.dbf
datafile 9 switched to datafile copy
input datafilecopy recid=8 stamp=561649706 filename=/u01/oradata/EDAW/SMALL_DATA.dbf
datafile 10 switched to datafile copy
input datafilecopy recid=9 stamp=561649706 filename=/u02/oradata/EDAW/small_indx01.dbf
datafile 11 switched to datafile copy
input datafilecopy recid=10 stamp=561649706 filename=/u01/oradata/EDAW/tools01.dbf
printing stored script: Memory Script
{
  Alter clone database open resetlogs;
}
executing script: Memory Script
database opened
Finished Duplicate Db at 22-JUN-05
Recovery Manager complete.
```

8 RMAN and standby databases

It is possible to use a standby database for backup purposes. The standby database does not need to be registered in recovery catalog.

The following error occurs if one tries to register standby database.

RMAN> register database;

RMAN-00571:
===
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===
RMAN-03002: failure of register command at 07/27/2007 01:02:55
RMAN-08040: full resync skipped, control file is not current or backup

The same recovery catalog can be used to manage the backup of primary and standby database.

On the Primary Database

```
C:\> set ORACLE_SID=ora10gr2db1
C:\>rman target / catalog rman/password@rman
Recovery Manager: Release 10.2.0.3.0 - Production on Fri Jul 27 00:53:55 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
connected to target database: ORA10GR2 (DBID=3930865674)
connected to recovery catalog database
RMAN> register database;
database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
RMAN>
```

On the standby database:

```
C:\> set ORACLE_SID=stdby
C:\>rman target / catalog rman/password@rman
Recovery Manager: Release 10.2.0.3.0 - Production on Fri Jul 27 00:52:34 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
connected to target database: ORA10GR2 (DBID=3930865674, not open)connected to recovery
catalog database
RMAN> backup incremental level 0 database ;
Starting backup at 27-JUL-07
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=159 devtype=DISK
channel ORA_DISK_1: starting incremental level 0 datafile backupset
channel ORA_DISK_1: specifying datafile(s) in backupset
input datafile fno=00001 name=C:\ORADATA\ORASTDBY\SYSTEM01.DBF
input datafile fno=00002 name=C:\ORADATA\ORASTDBY\UNDOTBS01.DBF
input datafile fno=00005 name=C:\ORADATA\ORASTDBY\USERS02.DBF
input datafile fno=00003 name=C:\ORADATA\ORASTDBY\SYSAUX01.DBF
input datafile fno=00004 name=C:\ORADATA\ORASTDBY\USERS01.DBF
channel ORA_DISK_1: starting piece 1 at 27-JUL-07
channel ORA_DISK_1: finished piece 1 at 27-JUL-07
piece
handle=C:\FLASH_RECOVERY_AREA\ORASTDBY\BACKUPSET\2007_07_27\O1_MF_NNND0_TAG20
070727T005247_3BLYJZJQ_.BKP
tag=TAG20070727T005247 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:45
channel ORA_DISK_1: starting incremental level 0 datafile backupset
channel ORA_DISK_1: specifying datafile(s) in backupset
including current control file in backupset
channel ORA_DISK_1: starting piece 1 at 27-JUL-07
```

```
channel ORA_DISK_1: finished piece 1 at 27-JUL-07
piece
handle=C:\FLASH_RECOVERY_AREA\ORASTDBY\BACKUPSET\2007_07_27\O1_MF_NCNN0_TAG20
070727T005247_3BLYLG36_.BKP
tag=TAG20070727T005247 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
Finished backup at 27-JUL-07

RMAN>
```

Notes:

First, a standby database does not have to be in mounted state while being backed up by RMAN, contrary to what some notes in Metalink say. The standby could be any state - managed recovery or open read only, meaning it was in INCONSISTENT state. In that state, the restore simply needs more archive logs to be consistent, just like a simple RMAN or hot backup, nothing special.

Second, only the archived logs of standby can be backed up, not the primary. While recovering the primary, you can simply use the archived logs from the standby, without any problems. Some documentation seems to indicate to the contrary.

In a restore situation you have the following choices.

1. If a data file of primary is to be restored, merely ftp over the data file from standby, rename it if necessary to the primary's name and recover that data file.
2. If the standby data file is gone, too; restore the RMAN backup of the data file to the primary and recover it. Remember this backup was taken at the standby.
3. If archived logs are missing from primary, merely ftp over from the standby or restore directly to primary from backup.
4. If the primary is intact but the standby is broken, instead of restoring the standby data file from tape, place the tablespace in hot backup mode in primary and ftp the file over to the standby and perform a manual recovery. Then place the standby in managed recovery.

9 RMAN Catalog maintenance and Tivoli TSM.

There is a need to periodically cleanse the RMAN catalog of all the unused and unwanted database incarnations and backups. When using a tool such as Tivoli TSM there is an additional need to ensure that the RMAN catalog and the TSM catalog stay in sync.

If this is not done, there is the potential for backups never being deleted from Tivoli, and thus they will remain 'held' upon tape, since there is no easy immediately obvious means to indicate to Tivoli that they are no longer required. The two recommended methods to ensure the two catalogs stay in sync is to ensure deletion via RMAN or for the node to be dropped on TSM. The latter is a little too dramatic for most cases, especially where there are more than one database instance using the node.

The following technique has been used successfully to clean up a 9.2 RMAN catalog. Although there is nothing that is 9.2 specific it has not been tested by the author on any other database version.

The use of Tivoli TSM requires that if the catalog entries for databases that reside upon other nodes (existing or former) that there are appropriate tdpo.opt and TDPO.node_oracle password files exists for the databases concerned. This is all described in section 5 above.

Method:

1. On the database holding the RMAN repository, generate a listing of the currently registered databases

   select * from rc_database order by name;

   The listing is used below.

2. Check for the existence of the appropriate TDP files for the nodes where the databases reside exist upon the current node. If they do not exist generate or copy the files from the appropriate server.

3. Create a dummy test Oracle instance. This instance will be started in NOMOUNT mode, so no database files will be created.

4. Open a RMAN session so that the command prompt shows:

   RMAN>

5. Issue a set dbid command to access the required database information.

   RMAN:> set dbid=xxxxxxx;

   Where xxxxxxx is the DBID of the database as shown on the list generated is step 1.

6. Generate a list of the current backups for this DBID.

   RMAN:> list backup summary;

7. If the list shows that backups exist for the node then there is a need to allocate a maintenance channel before they can be removed. Typically this would be performed using a command as follows:

   RMAN:> allocate channel for maintenance type 'SBT_TAPE' FORMAT '%d_%U_%s' parms 'ENV=(TDPO_OPTFILE=/usr/tivoli/tsm/client/oracle/bin64/tdpo_*node*.opt)';

   Where the node name is the name of the machine upon which the database resides or resided.

8. It may be possible to just run a 'delete obsolete' command, perhaps preceded with a 'crosscheck backup' to ensure that the backup sets are set to expired. There are likely to be a few backup sets remaining even if the 'delete obsolete' succeeds. In these situations and where the delete obsolete fails to work, it will be necessary to delete the backup sets individually by issuing the following command:

   RMAN:> delete backupset 1111111,2222222,3333333,4444444;

   Where 1111111, 2222222 etc are the backupset numbers generated from the summary backup list command is step 6.

9. If there are still some backups that it is desired to keep then the steps complete at this stage. However if all the backups are deleted and there is no desire to retain the RMAN catalog entry for this database at all, then go to step 10.

10. To remove the database from the RMAN catalog it is necessary to open a SQL session as the 'rman' catalog owner and issue the following command:

    SQL:> exec dbms_rcvcat.unregisterdatabase(db_key, dbid);

    Were the db_key and the dbid are shown on the list generated in step 1 above.

    i.e. exec dbms_rcvcat.unregisterdatabase(1072489,693855127);

10 Top Backup and Recovery best practices.

This section assumes that one is already performing the Backup and Recovery basics

- Running in Archivelog mode
- Multiplexing the controlfile
- Taking regular backups
- Periodically doing a complete restore to test your procedures.

1. Turn on block checking.

REASON: The aim is to detect, very early the presence of corrupt blocks in the database. This has a slight performance overhead, but Checksums allow Oracle to detect early corruption caused by underlying disk, storage system, or I/O system problems.

SQL> alter system set db_block_checking = true scope=both;

2. Turn on block tracking when using RMAN backups (if running 10g)

REASON: This will allow RMAN to backup only those blocks that have changed since the last full backup, which will reduce the time taken to back up, as less blocks will be backed up.

SQL> alter database enable block change tracking using file
'/u01/oradata/ora1/change_tracking.f';

3. Duplex log groups and members and have more than one archive log dest.

REASON: If an archivelog is corrupted or lost, by having multiple copies in multiple locations, the other logs will still be available and could be used.

If an online log is deleted or becomes corrupt, you will have another member that can be used to recover if required.

SQL> alter system set log_archive_dest_2='location=/new/location/archive2' scope=both;

SQL> alter database add logfile member '/new/location/redo21.log' to group 1;

4. When backing up the database use the 'check logical' parameter

REASON: This will cause RMAN to check for logical corruption within a block as well as the normal head/tail checksumming. This is the best way to ensure that you will get a good backup.

RMAN> backup check logical database plus archivelog delete input;

5. Test your backup.

REASON: This will do everything except actually restore the database. This is the best method to determine if your backup is good and usable before being in a situation where it is critical and issues exist.

RMAN> restore validate database;

6. Have each datafile in a single backup piece

REASON: When doing a partial restore RMAN must read through the entire piece to get the datafile/archivelog requested. The smaller the backup piece the quicker the restore can complete. This is especially relevent with tape backups of large databases or where the restore is only on individual / few files.

RMAN> backup database filesperset 1 plus archivelog delete input;

7. Maintain your RMAN catalog/controlfile

REASON: Choose your retention policy carefully. Make sure that it compliments your tape subsystem retention policy, requirements for backup recovery strategy. If not using a catalog, ensure that your controlfile record keep time instance parameter matches your retention policy.

SQL> alter system set control_file_record_keep_time=21 scope=both;

This will keep 21 days of backup records.

Run regular catalog maintenance.

REASON: Delete obsolete will remove backups that are outside your retention policy. f obsolete backups are not deleted, the catalog will continue to grow until performance becomes an issue.

RMAN> delete obsolete;

REASON: crosschecking will check that the catalog/controlfile matches the physical backups. If a backup is missing, it will set the piece to 'EXPIRED' so when a restore is started, that it will not be eligible, and an earlier backup will be used. To remove the expired backups from the catalog/controlfile use the delete expired command.

RMAN> crosscheck backup;

RMAN> delete expired backup;

8. Prepare for loss of controlfiles.

set autobackup on

REASON: This will ensure that you always have an up to date controlfile available that has been taken at the end of the current backup not during.

RMAN> configure controlfile autobackup on;

keep your backup logs

REASON: The backup log contains parameters for your tape access, locations on controlfile backups that can be utilised if complete loss occurs.

9. Test your recovery

REASON: During a recovery situation this will let you know how the recovery will go without actually doing it, and can avoid having to restore source datafiles again.

SQL> recover database test;

10. Do not specify 'delete all input' when backing up archivelogs
REASON: Delete all input' will backup from one destination then delete both copies of the archivelog where as 'delete input' will backup from one location and then delete what has been backed up. The next backup will back up those from location 2 as well as new logs from location 1, then delete all that are backed up. This means that you will have the archivelogs since the last backup available on disk in location 2 (as well as backed up once) and two copies backup up prior to the previous backup.