EMC²
where information lives®

EMC CLARiiON SnapView and MirrorView for Oracle Database 10g Automatic Storage Management

Best Practices Planning

ORACLE®

EMC²
where information lives®

Abstract

The purpose of this document is to provide a comprehensive set of best practices and procedures when deploying Oracle Database 10g and the Automatic Storage Management (ASM) feature with EMC® CLARiiON® storage-based replication technologies.

June 2006

Table of Contents

Executive summary

The task of managing storage provisioning and data layouts to accommodate the ever changing and increasingly demanding requirements of a growing Oracle database has always been a key IT challenge. The Oracle database supporting the enterprise business and operation is key to its success. Protecting the database, and ensuring its continual integrity and availability, is now mandatory for such enterprises. The information content frequently needs to be replicated and repurposed to create competitive advantage for the business.

Oracle Corporation's innovation in their Oracle Database 10g Automatic Storage Management (ASM) feature aims at simplifying storage provisioning and data layouts challenges. EMC's proven storage replication technologies, working in combination with ASM, allows the Oracle data to be easily and readily replicated for both protection, as well as satisfying different repurposing needs.

This paper was jointly developed by EMC and Oracle to offer best practices when using EMC replication technologies with Oracle Database 10g and ASM.

Introduction

The purpose of this white paper is to provide a comprehensive set of best practices and procedures when deploying Oracle Database 10g and Automatic Storage Management (ASM) with EMC® CLARiiON® storage-based replication and consistency technologies. This includes EMC SnapView™ and MirrorView™ (Asynchronous and Synchronous), which have been validated in accordance with Oracle's Storage Compatibility Program (OSCP) and are being extended to include validation when using Oracle Database 10g Automatic Storage Management.

This paper will document the procedures and best practices for the following use cases:

- Using Oracle Database 10g hot backup with SnapView clones for database backup
- Using SnapView clones for database cloning
- Using MirrorView for DR and remote database cloning
- Using Oracle Database 10g hot backup with MirrorView for DR and remote database backup
- Restoring a database on a production host

Audience

This white paper is intended for those who have a basic understanding of Oracle Database 10g Automatic Storage Management and EMC SnapView and MirrorView technologies.

Oracle Automatic Storage Management

Automatic Storage Management (ASM) is a database file system that provides cluster file system and volume management capabilities integrated into the Oracle Database 10g at no additional cost. ASM lowers your total cost of ownership and increases storage utilization without compromising performance or availability. With ASM, a fraction of the time is needed to manage your database files.
ASM eliminates overprovisioning and maximizes storage resource utilization facilitating database consolidation. The ASM self-tuning feature evenly distributes the data files across all available storage. It delivers high performance similar to raw devices, sustained over time, with the ease of use of a file system. ASM's intelligent mirroring technology enables up to triple data protection, even on non-RAID storage arrays, empowering low-cost storage deployment reliably. ASM benefits are the following:

- Simplify and automate storage management
- Increase storage utilization and agility
- Predictably deliver on performance and availability service level agreements

ASM simplifies storage management tasks, such as creating/laying out databases and disk space management. Since ASM allows disk management to be done using familiar create/alter/drop SQL statements, database administrators (DBA) do not need to learn a new skill set or make crucial decisions on provisioning. In addition, ASM operations can be completely managed with 10g Enterprise Manager. ASM is a management tool specifically built to simplify the job of the DBA. It provides a simple storage management interface across all server and storage platforms. ASM provides the DBA flexibility to manage a dynamic database environment with increased efficiency. This feature is a key aspect of grid computing.

Oracle Recovery Manager

Recovery Manager (RMAN) is Oracle's utility to manage the backup, and more importantly the recovery, of the database. It eliminates operational complexity while providing superior performance and availability of the database. RMAN debuted with Oracle8 to provide DBAs with an integrated backup and recovery solution. RMAN determines the most efficient method of executing the requested backup, restore, or recovery operation, and then executes these operations in concert with the Oracle database server. RMAN and the server automatically identify modifications to the structure of the database and dynamically adjust the required operation to adapt to the changes.

Oracle Flash Recovery Area

The Flash Recovery Area is a unified storage location for all recovery-related files and activities in an Oracle database. By defining one init.ora parameter, all RMAN backups, archive logs, control file autobackups, and datafile copies are automatically written to a specified file system or ASM disk group. In addition, RMAN automatically manages the files in the Flash Recovery Area by deleting obsolete backups and archive logs that are no longer required for recovery. Allocating sufficient space to the Flash Recovery Area will ensure faster, simpler, and automatic recovery of the Oracle database.

EMC consistency technology

Beginning with the EMC CLARiiON release in Q3 of 2005, you can use the Flare Consistency Assist (FCA) feature to perform consistent splits on multiple, application content related business continuous volume (BCV) pairs. Consistent split for BCVs of multiple LUNs avoids inconsistencies and restart problems that can occur if you split a database-related BCV without first quiescing the database. When a consistent split is performed inside the CLARiiON, database writes are held at the storage level for a very short time while the foreground split occurs, maintaining dependent-write order consistency on the target devices.

Consistency technology covering SnapView snapshots or clones, and MirrorView mirroring groups, provides the capability to create an image of one or more databases that are DBMS restartable copies. It does this by momentarily holding all write I/O to the related LUNs while performing a split operation.

The resultant databases on the target volumes are in a data state that is equivalent to the state they would be in after a power failure. In the context of Oracle technology, a more appropriate analogy would be that they look the same as if all database instances were aborted simultaneously.

Since restarting an aborted instance does not in any way require the database to be in Oracle's hot backup mode, we are able to provide customers with a way to create restartable database clones without requiring the user to place the databases tablespaces in hot backup mode.

EMC SnapView overview

The CLARiiON SnapView software runs inside the CLARiiON storage processors. It provides the ability to efficiently create bit image replicas of a CLARiiON storage object called a logical unit number (LUN) inside the storage system, without requiring any host processing resources.

The replicated LUN may be a logical replica, called a snapshot, or a physical bit-for-bit replica, called a clone and also frequently referred to as a business continuous volume (BCV).

A maximum of eight snapshots or clones can be created against a single source LUN. Snapshots can also be created from clones of a source LUN.

Each snapshot represents a point-in-time logical image of the source LUN from which it is derived. Snapshots rely on a Copy-On-First-Write technology to track source LUN changes from the time each snapshot is created. Each snapshot can be used by server host or hosts as if they are independent CLARiiON LUNs, distinct from their source LUNs. However, additional physical storage is used only to hold different copies of data that have been changed in the source LUNs. A 1 TB source LUN and a point-in-time snapshot appear to server hosts as two storage LUNs, each 1 TB. However, the actual physical storage used to support both LUNs is typically much less than the actual 2 TB total.

SnapView clones, in contrast, are full bit-for-bit replicas of their respective source LUNs. When we establish a clone to the same 1 TB source LUN, a total of 2 TB of space is allocated in the CLARiiON storage system.

With snapshots, we logically create a usable copy of the source LUN content at a particular point in time by *starting a new snapshot* on the source LUN. With clones, we create a usable copy of the source LUN data content by first internally bitwise synchronizing the source LUN content to the clone LUN, then *splitting the clone* LUN off (and allow it to be independently accessed and manipulated) at a particular point in time as needed. Once the independent usage is completed, the clone can be bitwise resynchronized with its source, to prepare it to be split off again for independent use at a later time.

SnapView consistent sets (snapshots and clones)

SnapView supports the concept of a consistent LUN set. A set of source LUNs that contain application-level interrelated contents can be dynamically selected, to be used as the source dataset to create a multi-LUN point-in-time content coherent snapshot set. Alternatively, if each source LUN has a synchronized clone at a particular point in time, all the clone LUNs can be split off from their respective source LUNs via a single SnapView "atomic" clone split action.

The action of creating an entire set of snapshots via a single snapshot action, or causing a set of clones to be simultaneously split from their respective sources, guarantees a point-in-time content consistent replica set. When the SnapView consistent set action is invoked, all incoming modifications to each of the source LUNs selected for this set replication action will be held briefly by the storage system, long enough for the replication function for all members selected to be completed.

Hence, no host side special application-level provisions need to be made to ensure that there are no data changes made to the respective source data LUNs while the storage replication action is in progress. The result storage replicated LUN set is guaranteed to be point-in-time bitwise storage content consistent with all source LUNs in the set. For Oracle, the replicated SnapView dataset, snapshots, or clones would represent a point-in-time crash restartable image of the database if all relevant database LUNs are included.

MirrorView overview

MirrorView supports the storage managed propagation of changes made to data stored in a LUN on one CLARiiON system to a corresponding LUN on another CLARiiON storage system.

The propagation of changes from a LUN on one storage system, called the primary LUN, to a corresponding LUN, called the secondary LUN, on another system, can be done either in synchronous or asynchronous mode. In synchronous mode, a write that changes the content of the primary LUN is not acknowledged to the server host as successfully completed until the change is successfully propagated to the secondary storage system. In asynchronous mode, changes to the primary LUNs are not immediately

propagated. Instead, they are tracked and periodically, or per user's explicit direction. All the tracked changes are propagated from the primary LUN (s) to the secondary LUN (s) as a batch. With asynchronous mirroring, if a particular disk segment inside a LUN is modified multiple times within the change resynchronization interval, only the final segment content resulting from the aggregation of all the changes will be forwarded. So, locality of repeated data changes is a key consideration when sizing a solution involving a MirrorView/Asynchronous implementation. Figure 1 shows the MirrorView configuration possibilities.
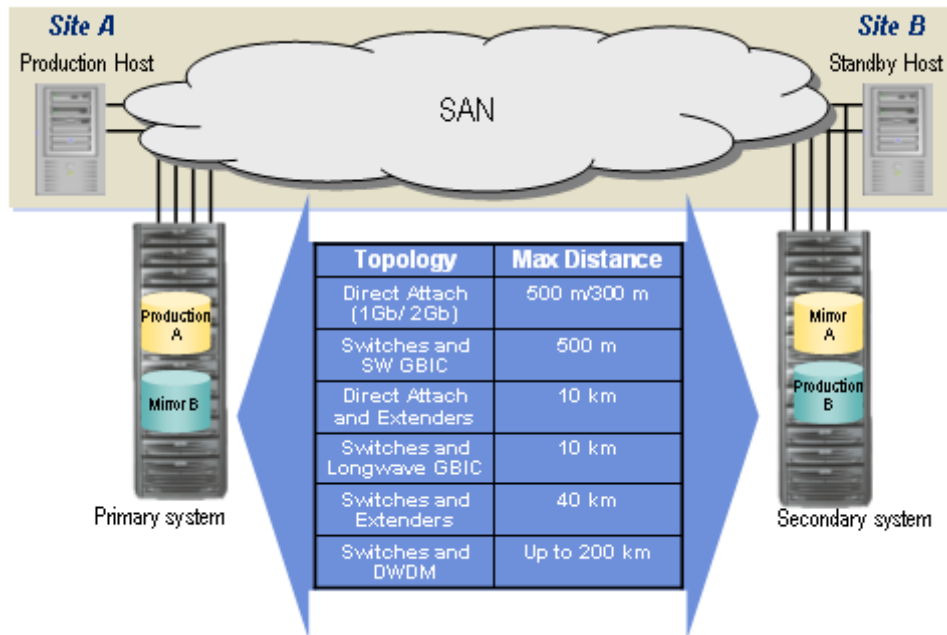

Figure 1. MirrorView configuration possibilities

MirrorView/Synchronous mode

In synchronous mode, all I/O from the local host is first written to the local storage processor memory and is then sent over the mirroring links to the remote CLARiiON unit. Once the remote CLARiiON unit reports that the data has reached its storage processor memory successfully, the I/O is acknowledged to the local host. Synchronous mode guarantees that the remote image is a complete duplication of the source image.

MirrorView/Asynchronous mode

Synchronous mirroring mode ensures that the data on the secondary CLARiiON is bitwise tracking all the production data. When changing data has to be kept in synchronization over significant distance, application performance will be compromised. The main premise of MirrorView/Asynchronous mode is to provide a consistent, point-in-time image on secondary data copy, which is not too far behind the primary production data, and that results in minimal data loss in the event of a disaster at the primary site.

Disaster restart and disaster recovery

Classical disaster recovery techniques have evolved over several decades. In most cases, a disaster recovery activity implies usage of data tapes that have been stored offsite in a secure location. Full backup tapes of disk data are usually taken periodically—often during a low transaction period, such as a Saturday or Sunday night. During the rest of the week, incremental backup tapes (or tapes capturing changed disk data since the last full or previous incremental backup) are gathered and sent offsite. By being stored offsite, these tapes use geographic separation to guard against any local disaster, such as a fire or a flood. In a

disaster situation, the user must gather all the tapes and apply them in sequence. Considering the time associated with running the full and incremental tape backups from disk, packaging the tapes for offsite transport, gathering them in the event of a disaster, and recovering them back to disk storage at the backup site, 48 hours may be considered a realistic expectation for the duration of a disaster recovery activity. These activities are also susceptible to human error, such as the tapes incorrectly applied in the disaster recovery sequence, lost tapes, damaged tapes, incompatible tapes, and so on. Once the remote site is running, a similarly lengthy outage (usually involving tapes) occurs returning the data to home once repairs have been made at the original data center.

Disaster restart, on the other hand, does not use computer tapes. Rather, data is transported by communication links to remote data storage. The remote replica of data serves as the restart point, and the user may restart the application using disk images at the remote site.

A major issue is whether the data in the remote location is logically consistent. MirrorView ensures the dependent write order consistency of the replication by synchronizing each and every dependent I/O (synchronous mode) or by synchronizing cycled chunks of data (asynchronous mode). In a true physical disaster at the source location, database restart operations can be completed at the remote site without the delays associated with finding and applying tapes in the correct sequence.

In addition to disaster restart benefits, MirrorView significantly enhances disaster recovery operations by using fast and reliable replication technology to offload the Oracle backup operations to a remote site and later return the restored data to the local site.

When a disaster recovery solution is required, the only way to create valid Oracle backups with any split mirror or snapshot technology is to ensure that the database is in hot backup mode during the split or by using Oracle Recovery Manager (RMAN). Refer to Oracle documentation for further details regarding RMAN.

Automatic Storage Management instance

In addition to the standard database instance, Oracle introduced a new type of instance called an ASM instance. The ASM instance is started with the INSTANCE_TYPE=ASM init.ora parameter. This parameter signals the Oracle initialization routine to start an ASM instance and not a standard database instance. Unlike the standard database instance, the ASM instance contains no physical files such as log files, control files or datafiles.

The OTN ASM home page has more information on ASM:

http://www.oracle.com/technology/products/database/asm/index.html

Rebalancing and consistency technology

ASM provides a seamless and non-intrusive mechanism to expand and shrink the diskgroup storage. When disk storage is added or removed, ASM will perform a redistribution (rebalancing) of the striped data[1]. This entire rebalance operation is done while the database is online, thus providing higher availability to the database. The main objective of the rebalance operation is to always provide an even distribution of file extents and space usage across all disks in the diskgroup.

It is considered a best practice to use ASM external redundancy for data protection when using EMC arrays. CLARiiON will provide protection against loss of media, as well as transparent failover in the event of a specific disk or component failure.

---

[1] A disk failure will also trigger a rebalance; however, this is specific to ASM Failure groups.

The split operation of storage-based replicas is sensitive to the rebalancing process that may cause ASM diskgroup inconsistencies if the diskgroup device members are split at slightly different times. These inconsistencies are a result of ASM metadata changes occurring while a split operation is in process. Upon startup, if ASM detects an inconsistency, metadata logs will be used to perform ASM instance recovery. In addition, Oracle provides tools and procedural steps to avoid inconsistencies when splitting storage-based replicas, however, these procedures can be simplified and streamlined with the use of EMC consistency technology.

Since EMC consistent split technology suspends database I/O to preserve write ordering, it also has the side effect of preventing any ASM metadata changes during the split. Performing a consistent split will prevent ASM metadata inconsistencies during the replication process, eliminating the otherwise extra steps or possible unusable replica if ASM rebalance was active while performing a non-consistent split.

Test cases and best practices

Test results show that Oracle Database 10g and Automatic Storage Management can be deployed non-disruptively with the EMC SnapView clones and MirrorView family of products when using consistency technology. If a rebalance operation is triggered while a consistent split is being performed, any ASM metadata changes are held until the source and target are in a synchronous state.

General tests configuration

It is assumed that:
- Oracle is installed on the target host with similar options to the production host and is configured for ASM use.
- Copies of the production init.ora files for the ASM instance and the database instance were copied to the target host and modified if required to fit the target host environment.
- The appropriate remote copies at the secondary site (whichever is appropriate for the test) are accessible by the target host.
- An RMAN recovery catalog is configured and operational.
- The backup server (target host) has Oracle Net connectivity to the recovery catalog database.
- You are using a Flash Recovery Area.
- The production host has connectivity to a LAN-based tape backup system

Test conditions were:
- OLTP load was running during the split.
- Transaction integrity test (defined by OSCP test kit) was running during the split.
- ASM rebalance was active during the split.

Test success was measured by:
- ASM and database instance were opened successfully on target host without any errors reported.
- Transaction integrity test passed.
- Rebalance operation continued automatically and completed successfully on target host.
- Database Verification utility tested all the datafiles, and no errors were found.

Navicli commands that trigger SnapView and MirrorView functions can be issued against the CLARiiON storage system from any servers across the IP network. In our test scenarios, the commands are shown as invoked from the production host. The term *production host* is used to specify the primary host if the source devices are used, and *target* or *backup host* is used to specify the host if the BCVs and snapshots are used.

Hardware

Tables 1 and 2 provide server and storage system configurations used in testing.

Table 1. Server system configurations used in testing

	Model	OS	Oracle version
Local "Production" Host	SUN	Solaris 2.8	10g release 2 (10.2.0.1)
Remote "Backup or Remote" Host	SUN	Solaris 2.8	10g release 2 (10.2.0.1)

Table 2. CLARiiON storage system configurations used in testing

	Name/serial number	Type	FLARE™ version
Local CLARiiON	APM000470488	CX700	Release 19 (02.9.700.5.06)
Remote CLARiiON	APM000470489	CX700	Release 19 (02.9.700.5.016)

ASM diskgroups / mount points

In all cases the databases were built using three distinct ASM diskgroups:

- The Datafile diskgroup contains all datafiles.

- The Flash Recovery diskgroup contains files such as multiplexed control files, backup sets, and flashback logs.

- The Online Redo diskgroup contains online redo logs for the database. Ordinarily, Oracle's best practice recommendation is for the redo logs files to be placed in the same diskgroup as all the database files (the Datafile diskgroup in this example). However, it is necessary to separate the online redo logs from the data diskgroup when planning to do recovery from split mirror snap copies since the current redo log files can not be used to recover the cloned database.

Oracle recommends that archive logs be placed in the Flash Recovery Area.

Diskgroup purpose	Diskgroup name / mount point	Path (on production host)	Production LUN in array (AUN)	Production array clones (AUN)	Secondary array LUN (AUN)
Flash Recovery Area	RECOVERY_AREA	/dev/rdsk/c3t2d0s6	003	023	003
		/dev/rdsk/c3t2d1s6	004	024	004
Online Redo Logs	REDO_AREA	/dev/rdsk/c3t2d2s6	005	025	005
		/dev/rdsk/c3t2d3s6	006	026	006
Datafiles	DATA_AREA	/dev/rdsk/c3t2d4s6	007	027	007
		/dev/rdsk/c3t2d5s6	008	028	008
		/dev/rdsk/c3t2d6s6	009	029	009
		/dev/rdsk/c3t2d7s6	010	030	010
		/dev/rdsk/c3t2d8s6	011	031	011
		/dev/rdsk/c3t2d9s6	012	032	012
		/dev/rdsk/c3t2d10s6	013	033	013
		/dev/rdsk/c3t2d11s6	014	034	014

Each device size was approximately 8.6 GB. All LUNs are 4+1 R5. For local SnapView clone testing, local Array Unit Numbers (AUN) that were 20+ in number were correspondingly clone synchronized with their production AUN. For MirrorView testing, the local AUNs were mirrored to their corresponding remote AUNs on the second array, *after* the primary AUNs were taken out of the clone pair relationship with their local clones.

This is an example of how to query the contents of the DATA_AREA diskgroup.

```
SQL> select path, dg.name   from v$asm_disk dk, v$asm_diskgroup dg where
dk.group_number=dg.group_number and dg.name='DATA_AREA' order by
failgroup;
```

```
PATH                 NAME
-------------------- --------------
/dev/rdsk/c3t2d4s6   DATA_AREA
/dev/rdsk/c3t2d5s6   DATA_AREA
/dev/rdsk/c3t2d6s6   DATA_AREA
/dev/rdsk/c3t2d7s6   DATA_AREA
/dev/rdsk/c3t2d8s6   DATA_AREA
/dev/rdsk/c3t2d9s6   DATA_AREA
/dev/rdsk/c3t2d10s6  DATA_AREA
/dev/rdsk/c3t2d11s6  DATA_AREA
```

ASM instance parameter file

```
INSTANCE_TYPE=ASM
ASM_DISKSTRING='/dev/rdsk/*s6'
ASM_DISKGROUPS='DATA_AREA','REDO_AREA','RECOVERY_AREA'
SHARED_POOL_SIZE=30m
```

Database instance parameter file

```
job_queue_processes=2
sga_target=900M
db_name = hrd10g
control_files = +DATA_AREA/control_001
parallel_max_servers = 10
recovery_parallelism = 40
db_files = 116
db_cache_size   = 341M
db_16k_cache_size  = 341M
dml_locks = 200
log_buffer = 1048576
processes = 70
sessions = 70
transactions = 70
shared_pool_size = 100M
cursor_space_for_time = TRUE
db_block_size = 8192
undo_management = auto
undo_retention = 2
_in_memory_undo=false
_undo_autotune=false
plsql_optimize_level=2
UNDO_TABLESPACE = undo_1
db_2k_cache_size = 20M
db_4k_cache_size = 20M
DB_RECOVERY_FILE_DEST = +RECOVERY_AREA
LOG_ARCHIVE_DEST_1 = 'LOCATION=USE_DB_RECOVERY_FILE_DEST'
```

```
log_archive_format= %t_%s_%r.arc
```

Using CLARiiON Navisphere Secure Command Line Interface

In the following examples, the CLARiiON Navisphere® Secure Command Line Interface (naviseccli) commands will be shown as part of the host side process to activate the appropriate CX storage replication functions in support of the operational scenarios.

Naviseccli provides a secure management facility for activating the required CLARiiON array replication support functionality in support of the usage scenarios below. In the naviseccli command examples, the standard format of invocation would include:

naviseccli –h arrray1 –User {admin_user} –Password {admin_password} …

where
-h   specifies the array system management port IP address or name alias to direct the command to
-User   allows the authorized array management user name {admin_user} to be supplied
-Password supplies the required authentication password for that user

As CLARiiON management is done over the IP network as an out-of-band management model, the authentication user identification and passwords are encrypted before transmit over the IP network. However, if it is desired that all CLI scripting be done without explicitly exposing the storage system management user identification, naviseccli supports the use of separate server-by-server-based user authentication. Under the server authentication setup, the explicit inclusion of management user identification can be omitted from the command line itself, with the server authentication file used instead against the OS user invoking the CLI command to validate permission for execution of these calls.

The *EMC Navisphere Management Command Line Interface Reference* manual on EMC Powerlink™ (http://Powerlink.EMC.com) provides further details of the naviseccli syntax and parameter definitions.

Case 1: Using Oracle Database 10g hot backup with SnapView/clones for database backup

While the Oracle database is in hot backup mode on the local host a SnapView/clone consistent split is performed. This creates an image of the active database that can be used to perform a backup to tape by offloading this process to the remote or backup server.

Preparing to split SnapView clones

To begin the process, the set of relevant clones should be verified to be in content synchronization state. Assuming the proper Solaris server host side Navisphere management software is properly installed, the status of each of the clone pair can be checked with the following:

```
# naviseccli –h array1 –User {admin_user}
     –Password  {admin_password} snapview –listclone
     –Name {clone_pair_group_name} -CloneState
```

If the state of the clone pair is either CONSISTENT or SYNCHRONIZED, the clone of that particular production LUN is ready to be split off.

Online backup on production host

Begin backup mode
```
# export ORACLE_SID=hrd10g
# sqlplus "/ as sysdba"
SQL> alter database begin backup;
```

Perform a consistent set clone split for database files
```
# naviseccli –h array1
      -user {admin_user} -password {admin_password} snapview
      -consistentfractureclones -CloneGroupNameCloneId
      name1 id1 ... nameN idN
```

Where name1, name 2, and nameN are all the production LUN clone pairs that are part of the ASM disk sets that should be atomically split together as a set (LUNs 007 through 014). The IDs for each pair are usually 010000000000000 if the clone pair has only one active clone (this provides support for multiple PIT clone set if more than one clone is associated to each of the production LUN).

End backup mode

```
SQL> alter database end backup;
```

Switch logs and create control files
Create two copies of the control file. One copy (control_start) will be used to start up the database in mount mode on the backup server. The second copy (control_bakup) will be used as a component of the backup set used by RMAN.

```
SQL> alter system archive log current;
```

Perform a consistent split snapshot of the Recovery Area to capture the archive log

```
# naviseccli –h array1
      -user {admin_user} -password {admin_password} snapview
      -consistentfractureclones -CloneGroupNameCloneId
      name1 id1 ... nameN id
```
Here, the clone pairs should include only the LUNs (003 and 004) used for the Flash_Recovery_Area.

```
RMAN> run {
      Allocate channel ctl_file type disk;
      Copy current controlfile to
      '+RECOVERY_AREA/control_file/control_start';
      Copy current controlfile to
      '+RECOVERY_AREA/control_file/control_bakup';
      Release Channel ctl_file;
      }
```

Resynchronize the RMAN catalog
This adds the most recent archive log to the recovery catalog.

```
RMAN> resync catalog;
```

Backup procedure
On the Backup host the snapshot can be used as a disk backup or a source for a tape backup. Some backup applications require the database to be mounted to perform backups.

Once the clones are split, check that:

- The clones as remounted as system devices on the backup server have Oracle permissions.
- ASM init.ora file parameter ASM_DISKSTRING includes the path to the cloned devices as remounted to the backup server.
- ASM init.ora file parameter ASM_DISKGROUPS contains the names of the diskgroups.
- If the database used any mount points (in this case the archive logs directory), mount the file system from the clones.

Start ASM instance
When the ASM instance is started, since the BCVs are included in the ASM_DISKSTRING parameter, it will identify them as the diskgroups from the production database. Also since the ASM_DISKGROUPS parameter contains the diskgroup names, they will be mounted automatically.

```
# export ORACLE_SID=+ASM
# sqlplus "/ as sysdba"
SQL> startup
```

Mount database instance
A database backup that was taken with hot backup mode is valid for backup only as long as it wasn't open with resetlogs options. For that reason it should be either mounted (prerequisite for media recovery and many backup applications) or open read-only (after at least enough recovery was done to allow the database to open).

Before the database is mounted, change the Backup database instance init.ora CONTROL_FILE parameter to point to the copied controlfile. For example:

```
control_files  = +RECOVERY_AREA/control_file/control_start
#control_files = +DATA_AREA/control_001

# export ORACLE_SID=hrd10g
# sqlplus "/ as sysdba"
SQL> startup mount
```

Backing up the database instance
Perform a RMAN backup on the Backup host. The previously backed up control file must be part of the backup set because once the database is mounted, the SCN will be updated and will no longer reflect the initial state of the control file.

```
RMAN> run { ALLOCATE CHANNEL t1 TYPE SBT_TAPE
        BACKUP FORMAT 'ctl %d/%s/%p/%t'
        CONTROLFILECOPY '+RECOVERY_AREA/control_file/control_bak';
        BACKUP
                FULL
                FORMAT 'ctl %d/%s/%p/%t'
                (database);
        BACKUP
                FORMAT 'al %d/%s/%p/%t'
                (archive all);
        RELEASE CHANNEL t1
        }
```

The format specifier %d is replaced with date, %t is replaced with a 4 byte timestamp, %s with the backup set number, and %p with the backup piece number.

Case 2: Using SnapView/clones for database cloning

While Oracle is open for read/write on the local host, a SnapView/clones consistent set split is performed on the established clones to create a restartable image of the active database that can serve as a repurposed database.

Create point-in-time consistent restartable clone set split

```
# naviseccli –h array1
      -user {admin_user} -password {admin_password} snapview
      -consistentfractureclones -CloneGroupNameCloneId
      name1 id1 ... nameN id
```

It is important to remember that in order to support restart, all the ASM group LUNs (003-014) have to be included in the list for this command invocation.

On target host
On the target host, once the clones are split, check that:
- The clones remounted as system devices on the target host have Oracle permissions.

- ASM init.ora file parameter ASM_DISKSTRING includes the paths to the clone devices.

- ASM init.ora file parameter ASM_DISKGROUPS contains the names of the diskgroups.

Start ASM instance
When the ASM instance is started, since the BCVs are included in the ASM_DISKSTRING parameter, it will identify them as the diskgroups from the production database. Also since the ASM_DISKGROUPS parameter contains the diskgroup names they will be mounted automatically.

```
# export ORACLE_SID=+ASM
# sqlplus "/ as sysdba"
SQL> startup
```

Start database instance
When the database instance is started, since the BCVs were split with the consistency option, the database server will automatically perform crash recovery and open the database. At the end of this step the database is opened and available for user connections.

```
# export ORACLE_SID=hrd10g
```

Connect to RMAN

```
RMAN > startup mount
RMAN > recover database;
RMAN > exit

nid target=sys/manager1            optionally, the db_name can be
                                   changed as well.  Please refer to
                                   the Oracle Recovery Guide for
                                   further information on the nid
                                   utility
```

```
SQL> startup mount
SQL> alter database open resetlogs
```

At the end of this step, the database is opened and available for user connections.

Case 3: Using MirrorView for DR and remote database cloning

When using MirrorView, the secondary LUNs are typically not directly remountable and usable on the server or servers connected to the secondary storage array. This is because the storage bit content of the MirrorView secondary LUNs would be continually changing based on activities on the production server side, and so direct use of the content of the secondary LUNs would not be meaningful. However, it is possible to leverage SnapView snaphot capability against the secondary LUNs to create useful point-in-time storage replicas based on the secondary LUNs. If the point-in-time consistent SnapView snapshot set is captured off the MirrorView secondary LUN set while the production primary LUNs are placed into an Oracle hot backup state, the snapshot set would also then represent the database in a hot backup state. In that case, the capture and usage scenario would be very comparable to Case 1.

On the other hand, if all the LUNs (003-104) are set up to mirror as a point-in-time consistent *group* for MirrorView, the snapshots of the secondary LUNs can then be taken, and the snapshot set, by virtue of the fact that each LUN represents the point-in-time content state of all the content consistent MirrorView secondary LUNs, would result in a restartable database disk image set.

Create MirrorView consistency group

It is again assumed here that readers are already familiar with how to create MirrorView primary-to-secondary mirroring relationships between corresponding LUNs on the two CLARiiON arrays. To ensure that all the relevant LUNs are being mirrored together as an "atomic" set, the different mirroring pair has to be placed into a MirrorView consistency group as follows:

```
# naviseccli –h array1
     -user {admin_user} -password {admin_password} mirror
     -sync -creategroup -name ALLDB_DG –o


# naviseccli –user {admin_user} –password {admin_password}

     -h array1 mirror -sync -addtogroup -name ALLDB_DG

     -mirrorname {LUN 03 mirror pair}
```
...... (one addtogroup per mirroring pair that needs to be put into the ALLDB_DG group)

```
# naviseccli –user {admin_user} –password {admin_password}

     -h array1 mirror -sync -addtogroup -name ALLDB_DG

     -mirrorname {LUN 14 mirror pair}
```

This sequence establishes a MirrorView/Synchronous consistency group ALLDB_DG with all mirroring pairs from LUN 03 to LUN 14 inclusive. From this point on, mirroring action will be done by MirrorView on the entire group. Either every member will be mirrored, or mirroring on all members will stop. The individual mirroring LUN pair can no longer be manipulated independently unless they are removed from the MirrorView consistency group ALLDB_DG completely.

On target host

Creating a point-in-time consistent snapshot set from the MirrorView secondary LUNs

First verify that the mirroring group state is consistent (no mirroring action still in progress) from the production host with:

```
# naviseccli –user {admin_user}
      -password {admin_password} -h array1 mirror -sync
      -listgroups ALLDB_DG –state
```

When the state is SYNCHRONIZED or CONSISTENT, we are ready to create the point-in-time SnapView snapshot LUN set for ALLDB_DG from the remote protection servers using:

```
# naviseccli –h array2  -user {admin_user} –password {admin_password}
      snapview startsession ALLDB_DG –lun 003 004 .. 014
      -consistent
```

This creates a SnapView snapshot session called ALLDB_DG for each of the MirrorView secondary LUNs 003, 004 to 014, and so on. The snapshot sets can then be presented as new system disk devices to the target host.

Setting up on the target host
- On the target host, make sure that the entire set of snapshots are discovered as new disk devices.
- ASM init.ora file parameter ASM_DISKSTRING includes the path to these new devices.
- ASM init.ora file parameter ASM_DISKGROUPS contains the names of the disk groups.

Start ASM instance
When the ASM instance is started, since the BCVs are included in the ASM_DISKSTRING parameter, it will identify them as the diskgroups from production database. Also since the ASM_DISKGROUPS parameter contains the diskgroup names they will be mounted automatically.

```
# export ORACLE_SID=+ASM
# sqlplus "/ as sysdba"
SQL> startup
```

Start database instance
When the database instance is started, since the BCVs were split with consistency option, the database server will automatically perform crash recovery and open the database. At the end of this step the database is opened and available for user connections.

```
# export ORACLE_SID=hrd10g
```

Connect to RMAN

```
RMAN > startup mount
RMAN > recover database;
RMAN > exit

nid target=sys/manager1            optionally, the db_name can be
                                   changed as well.  Please refer to
                                   the Oracle Recovery Guide for
                                   further information on the nid
                                   utility
```

```
SQL> startup mount
SQL> alter database open resetlogs
```

At the end of this step, the database is opened and available for user connections.

Case 4: Using Oracle Database 10g hot backup with MirrorView for DR and remote database backup

In Case 3, consistent snapshot set capability was leveraged to create a restartable storage image for both restarting the ASM instance as well as the database instance. If the production database LUNs were first put into hot backup state prior to taking the consistent snapshot set at the protection site, that snapshot set will be a valid database backup set.

Create MirrorView consistency groups

Because hot (online) backup requires the archive logs to be split at a different time than the rest of the database, two MirrorView consistency groups were created: one for the database files (ALLDB_DG) and one for archive logs (RECOV_DG). In general it is best practice for online backups to include only the Oracle datafiles. However, with ASM it is possible that control, redo, temp, and data files may be all mixed together in small number of ASM diskgroups.

The following command can be used to create the ALLDB_DG mirroring consistency group in the CLARiiON system:

```
# naviseccli -h array1
       -user {admin_user} -password {admin_password} mirror
       -sync -creategroup -name ALLDB_DG -o
```

```
# naviseccli –user {admin_user} –password {admin_password}
        –h array1 mirror -sync -addtogroup -name ALLDB_DG
       -mirrorname {LUN 03 mirror pair}
```
…… (one addtogroup per mirroring pair that needs to be put into the ALLDB_DG group)

```
# naviseccli –user {admin_user} –password {admin_password}
       –h array1 mirror -sync -addtogroup -name ALLDB_DG
       -mirrorname {LUN 14 mirror pair}
```

This sequence establishes a MirrorView/Synchronous consistency group ALLDB_DG mirroring all ASM members including the redo logs, and DB files.

```
# naviseccli -h -array1
       -user {admin_user} -password {admin_password} mirror
       -sync -creategroup -name RECOV_DG -o
```

```
# naviseccli –user {admin_user} –password {admin_password}
       –h array1 mirror -sync -addtogroup -name RECOV_DG
```

–mirrorname {LUN 03 mirror pair}

naviseccli –user {admin_user} –password {admin_password}

        –h array1 mirror –sync –addtogroup –name RECOV_DG

        –mirrorname {LUN 04 mirror pair}}

This sequence establishes a MirrorView/Synchronous consistency group RECOV_DG with all the mirroring pairs from LUN 03 and LUN 04 forming the ASM group holding the flash_recovery_area.

Online backup on production host

Begin backup mode

```
# export ORACLE_SID=hrd10g
# sqlplus "/ as sysdba"
SQL> alter database begin backup;
```

Perform a consistent split snapshot for database files

First verify that the mirroring group state is consistent from the production host (no mirroring action still in progress) with:

```
# naviseccli –user {admin_user}
        -password {admin_password} –h array1 mirror –sync
        -listgroups ALLDB_DG –state
```

When the state is SYNCHRONIZED or CONSISTENT, we are ready to create the point-in-time SnapView snapshot LUN set for ALLDB_DG from the remote protection servers using:

```
# naviseccli –h array2 snapview startsession ALLDB_DG
        -lun 005 006 .. 014-consistent
```

This creates a SnapView snapshot session called ALLDB_DG for each of the MirrorView secondary LUNs 005, 006 to 014, and so on. The snapshot sets can then be presented as new system disk devices to the target host.

End backup mode

```
SQL> alter database end backup;
```

Switch logs and create control files
Create two copies of the control file. One copy (control_start) will be used to start up the database in mount mode on the backup server. The second copy (control_backup) will be used as a component of the backup set used by RMAN.

```
SQL> alter system archive log current;
```

Perform a consistent split snapshot of the Recovery Area to capture archive logs

First verify that the mirroring group state is consistent from the production host (no mirroring action still in progress) with:

```
# naviseccli  -user {admin_user}
     -password {admin_password} -h array1 mirror -sync
     -listgroups RECOV_DG -state
```

When the state is SYNCHRONIZED or CONSISTENT, we are ready to create the point-in-time
SnapView/snapshot LUN set for RECOV_DG from the remote protection servers using:

```
# naviseccli -h array2 -user {admin_user} -password {admin_password}
     snapview startsession ALLDB_DG -lun 003 004 -consistent
```

This creates a SnapView snapshot session called RECOV_DG for each of the MirrorView secondary LUNs
003 and 004. The snapshot sets can then be presented as new system disk devices to the target host so that
they can be used to remount the ASM group RECOVERY_AREA.

```
RMAN> run {
     Allocate channel ctl_file type disk;
     Copy current controlfile to
     '+RECOVERY_AREA/control_file/control_start';
     Copy current controlfile to
     '+RECOVERY_AREA/control_file/control_bakup';
     }
```

Resynchronize the RMAN catalog
This adds the most recent archive log to the recovery catalog.

```
RMAN> resync catalog;
```

On target host

On the target host, once the snapshot devices are visible, verify the following:
- The snapshot devices have Oracle permissions.
- ASM init.ora file parameter ASM_DISKSTRING includes the path to these devices.
- ASM init.ora file parameter ASM_DISKGROUPS contains the names of the diskgroups.

Start ASM instance
When the ASM instance is started, since the BCVs are included in the ASM_DISKSTRING parameter, it
will identify them as the diskgroups from the production database. Also since the ASM_DISKGROUPS
parameter contains the diskgroup names, they will be mounted automatically.

```
# export ORACLE_SID=+ASM
# sqlplus "/ as sysdba"
SQL> startup
```

Mount database instance
A database backup that was taken with hot backup mode is valid for backup only as long as it wasn't open
with resetlogs options. For that reason it should be either mounted (prerequisite for media recovery and
many backup applications) or open read-only (after at least enough recovery was done to allow the
database to open).

Before the database is mounted change the target database instance init.ora CONTROL_FILE parameter to
point to the copied controlfile. For example:

```
control_files  = +RECOVERY_AREA/control_file/control_start
#control_files = +DATA_AREA/control_001


# export ORACLE_SID=hrd10g
# sqlplus "/ as sysdba"
SQL> startup mount
```

Backup target database instance
Perform an RMAN backup on the target host. The previously backed up control file must be part of the backupset because once the database is mounted, the SCN will be updated and will no longer reflects the initial state of the control file.

```
RMAN> run { ALLOCATE CHANNEL t1 TYPE SBT_TAPE
      BACKUP FORMAT 'ctl %d/%s/%p/%t'
      CONTROLFILECOPY '+RECOVERY_AREA/control_file/control_bak';
      BACKUP
            FULL
            FORMAT 'ctl %d/%s/%p/%t'
            (database);
      BACKUP
            FORMAT 'al %d/%s/%p/%t'
            (archive all);
      RELEASE CHANNEL t1
      }
```

The format specifier %d is replaced with date, %t is replaced with a 4 byte timestamp, %s with the backup set number, and %p with the backup piece number.

Case 5: Restoring a database on the production host

Assume that a local online backup as depicted in Case 1 has been done. The SnapView clones set can then be leveraged to support a fast restore/recovery of the production database leveraging the SnapView clone reverse synchronization mechanism. It is recommended that the SnapView clones should be initially established with PROTECTED RESTORE enabled.

The backup clone set as described in Case 1 represents the replica of the production database in hot backup state. From the time the clone set is split off from its respective production LUNs, data segments that have been changed in the production LUNs are tracked. When a SnapView reverse synchronization action is performed, only the changed tracks in the production LUNs would be "restored" using the corresponding tracks from the clones. This allows the production LUN content to be fully "restored" to the backup clones without the need to actually fully restore every data bit stored in the production LUNs.

The reverse synchronization action is performed in a PROTECTED mode. This means that while the "restore" process is in progress, if there are any further changes to any of the production LUNs, those changes would not get propagated to the clones. The clones remain as a "gold copy" backup set. Should the restore/recovery process fail for some unexpected reason, the same restore process can be restarted from the beginning.

Also of note is that as soon as the reverse synchronization process is initiated for each of the database LUNs in protected mode, the production database can be brought back online. The storage system keeps track of which LUN data segments have been restored, and which are still to be copied back. So, should the host side application ask for production LUN data that has yet to be restored, the storage system would automatically pull the appropriate data up from the clone. So, even though the storage data restore may

still take some time, the production database can be restarted even before the true data bit restore inside the storage system is completed.

Shut down the database instance on the production host:

```
# export ORACLE_SID=hrd10g
# sqlplus "/ as sysdba"
SQL> shutdown  [-immediate]
```

Unmount the ASM Data diskgroup on the production server.

```
SQL> alter diskgroup DATA_AREA dismount;
```

Note that in this case, we only restore datafile images from the production host. We do not want to overwrite the online redo logs (if they are still available) as they contain the last committed transactions. Also, we do not want to overwrite the Flash Recovery Area, which contains more recent archive logs. If the Flash Recovery Area was also damaged, then it would be necessary to unmount the damaged group from the production host and leverage SnapView clone reverse synchronization as the means to also restore that group to a usable state (at the loss of archived logs). If the redo log group has been damaged, the redo log group should just be dropped and re-created. The logs will be re-created when the database is re-opened with resetlogs.

Perform a restore of the database files only

The Recovery Area must not be restored to the production server in general (see previous section). The redo logs and DB file ASM disks can be storage restored by

```
# navisecccli –h array1
     -user {admin_user} -password {admin_password} snapview
     -reversesyncclone –Name name1 –Cloneid 010000000000000
     -UseProtectedRestore 1 –o
```

where name1 is the clone group name for one of the SnapView clone pair including LUN 005 through 014 (the redo logs and DB file ASM disks). This command is repeated 10 times, once for each of those 10 LUNs. Note that LUN 003 and 004, making up the FLASH_RECOVERY_AREA ASM diskgroup, do not have to be storage restored, as those would be coming from the separate RMAN backups done to the SBT channels.

Because the production ASM disk sets are currently unmounted, the storage members can be individually and independently storage restored (through the reverse synchronization action). Special provision for point-in-time consistent action for restoring the entire LUN set is not required. (The clone set used for initiating the individual protection LUN restore was captured as a point-in-time consistent set using the group consistent split action.)

Mount the ASM diskgroups

```
# export ORACLE_SID=+ASM
# sqlplus "/ as sysdba"

SQL> alter diskgroup DATA_AREA mount;
```

Start up the local database in MOUNT

```
# export ORACLE_SID=hrd10g
# sqlplus "/ as sysdba"
SQL> startup mount
```

Perform complete or point-in-time recovery with RMAN.

If you are performing incomplete recovery, set the recovery marker for "until time" or "until SCN".

```
RMAN> run {
      SET UNTIL TIME '06-dec-05 12:00';
      RECOVER DATABASE;
      }
```

Open the database

After you are sure that all files are correctly restored and recovered, you can open the database using the resetlogs option. Opening with the option resetlogs will create a new incarnation of the database, which must be also registered in the RMAN catalog.

```
RMAN> alter database open resetlogs;
```

Conclusion

The use cases discussed cover most of the typical deployment considerations for leveraging CLARiiON replication software features to augment ASM based deployments to facilitate Oracle database protections and different repurposing needs. The features in both Oracle and EMC array software effectively augment each other. Customers who already have invested in EMC CLARiiON hardware and software can confidently adopt an Oracle Database 10g ASM based deployment without sacrificing any of their current storage investment value.

References

- *Using Oracle 10g's Automatic Storage Management with EMC Storage Technology*
- *EMC TimeFinder and SRDF Best Practices for Oracle Database 10g Automatic Storage Management*
- *Understanding EMC Consistent Split with Oracle Databases*
- *Oracle Database 10g Automatic Storage Management Best Practices*
- *EMC CLARiiON Navisphere Command Line Interface (NAVICLI) including consistency support*