Oracle Maximum
Availability Architecture

Converting to Transparent Data Encryption Using Data Guard Transient Logical Standby

Oracle Database 11g Release 2

ORACLE WHITE PAPER | MAY 2015

Table of Contents

Introduction 1

TDE Overview 1

- TDE Tablespace Encryption Restrictions 2

Conversion Overview 2

- Transient Logical Standby Restrictions 2

Prerequisites 3

Conversion Example 4

- Enabling Transparent Data Encryption 4
- Transient Logical Script - First Execution 5
- Move Data to TDE Tablespace 7
- Transient Logical Script-Second Execution (Planned Maintenance Window starts) 11
- Transient Logical Script-Final Execution 13

Conclusion 14

Appendix A - First execution of the physru.sh script 15

Appendix B - Second execution of the physru.sh script 18

Appendix C- Third execution of the physru.sh script 20

Appendix D- physru.sh statistics 23

Appendix E- Alternative Methods to Convert to TDE 24

.

Introduction

Encrypting data with Oracle Advanced Security Transparent Data Encryption (TDE) requires that the data go through the process of encryption. Accomplishing this task with minimal application down time is a significant concern with the growing desire for 24/7 availability of many applications.

Oracle MAA best practices recommend using Data Guard transient logical standby for this purpose to avoid affecting production database performance during the conversion process and to minimize downtime. Downtime is minimal regardless of the size of the database since the TDE conversion occurs in a separate database (the standby database), while production runs unaffected. Alternative methods are described in Appendix E- Alternative Methods to Convert to TDE.

Whether you already have an existing physical standby database or are using a new physical standby database deployed just for facilitating conversion to TDE, the process of conversion includes the following high level steps:

1. Presence of a Data Guard physical standby database with no archive log gaps.
2. Conversion of the physical standby to a logical standby.
3. Pausing the standby apply process.
4. Rebuilding tablespaces with TDE and setup of the TDE configuration at the logical standby.
5. Starting the logical apply process to resynchronize the standby (now encrypted) with the primary database.
6. Data Guard switchover. The estimated application downtime using best practices is less than 5 minutes.
7. Conversion of the old primary (momentarily a logical standby) to a new physical standby database.
8. Starting the Data Guard physical apply process on the new standby database (the original primary).
9. Optionally – switching production back to the original primary. Estimated downtime using best practices is less than 5 minutes.

This Oracle Maximum Availability Architecture (Oracle MAA) best practices white paper is intended for database administrators who wish to convert a non-encrypted Oracle Database to TDE with minimal downtime. This paper assumes the reader has a technical understanding of Data Guard and TDE.

TDE Overview

TDE provides encryption of data at rest in an Oracle database. "At rest" implies that the data is encrypted at the operating system and storage level where data is stored. TDE decrypts data transparently when it hits the buffer cache where it is subject to normal database authentication and authorization rules.

There are two forms of TDE encryption. TDE column encryption encrypts specific columns of data while TDE tablespace encryption encrypts all data within a TDE encrypted tablespace. Tablespace encryption takes advantage of bulk encryption to enhance performance while relieving the administrator of the task of analyzing each column to determine which should be encrypted. Additionally, there are fewer restrictions with tablespace encryption compared to column encryption. This paper describes how to convert to TDE tablespace encryption. TDE Tablespace encryption is available in Oracle Database 11g Release 1 (11.1) and higher.

Refer to the Oracle Database Advanced Security Administrator's Guide for full details regarding TDE encryption[1].

1 http://docs.oracle.com/cd/E11882_01/network.112/e40393/asotrans.htm#ASOAG600

TDE Tablespace Encryption Restrictions

There are few restrictions with TDE tablespace encryption because encrypt/decrypt takes place during read/write as opposed to the SQL layer with column encryption. TDE tablespace encryption restrictions are:

- External Large Objects (BFILEs) cannot be encrypted using TDE tablespace encryption because these files reside outside the database.
- To perform import and export operations on TDE encrypted tablespaces, use Oracle Data Pump.

Conversion Overview

TDE tablespace encryption can only be enabled during the creation of a tablespace. Existing tablespaces cannot be altered to enable TDE. A Data Guard Transient Logical Standby Database can be used to limit the impact to application performance and availability during conversion to TDE tablespace encryption. Data is exported from the transient logical standby using Oracle Data Pump, the existing tablespace is dropped and a new TDE enabled tablespace is created followed by an import.

For a detailed description of the rolling upgrade process please refer to Oracle MAA best practices for Data Guard Transient Logical standby databases[2]. The process will be similar except the conversion will take place in lieu of a database upgrade.

Transient Logical Standby Restrictions

Since this process utilizes a logical standby database, the logical standby restrictions apply. A list of the most commonly encountered restrictions follows. Please refer to Data Guard documentation for a complete list of logical standby prerequisites and restrictions[3].

- Data Guard Broker must be disabled.
- Data Guard protection mode must be set to MAXIMUM PERFORMANCE or MAXIMUM AVAILABILITY.
- LOG_ARCHIVE_DEST_n for the standby database must be OPTIONAL.
- Logical standby databases do not support Oracle Label Security.
- Logical standby databases do not fully support an Oracle E-Business Suite implementation because there are tables that contain unsupported data types.
- Transportable tablespaces cannot transport encrypted tablespaces.
- Transportable tablespaces cannot transport tablespaces containing tables with encrypted columns.
- Data type restrictions (11.2):
  - BFILE
  - Collections (including VARRAYS and nested tables)
  - Multimedia data types (including Spatial, Image, and Oracle Text)
  - ROWID, UROWID
  - User-defined types

Note: Extended Datatype Support can be utilized to mitigate data type restrictions. See My Oracle Support Note 949516.1 (SQL Apply Extended Datatype Support - 11.2)[4].

2 http://www.oracle.com/technetwork/database/features/availability/maa-wp-11g-upgrades-made-easy-131972.pdf

3 http://docs.oracle.com/cd/E11882_01/server.112/e41134/data_support.htm#SBYDB00305

4 https://support.oracle.com/epmos/faces/DocumentDisplay?id=949516.1

Prerequisites

This process requires the following prerequisites to ensure a successful execution.

- There is an existing physical standby database.
- COMPATIBLE is set to a minimum of 11.1.0 though to enable enhanced features a setting of 11.2 is required.
- Oracle MAA Best practices require the primary database to have forced logging enabled. This is required for replication and will protect against unrecoverable objects during switchover. To ensure there are no unrecoverable blocks the following query should return no rows:

```
SQL> select NAME from V$DATAFILE where UNRECOVERABLE_CHANGE#>0;
no rows selected
```

- Flashback database must be enabled on both primary and standby. The following query should return 'YES' on both the primary and the standby.

```
SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
YES
```

- Any existing restore points will be dropped by this process. Make sure this is acceptable for your application.
- The described method is not compatible with Data Guard Broker, therefore Broker must be disabled and parameter DG_BROKER_START must be set to FALSE on both primary and standby databases.
- During this process a datapump export will be taken for all tablespaces designated for TDE encryption. This excludes the SYSTEM and SYSAUX tablespaces. There must be ample space to take these exports. The estimate_only=YES option on expdp should be used to get a rough estimate of space used by the export.

NOTE: expdp estimates do NOT take into account compression if you intend to compress the datapump export, compression saves space but takes longer to export and import.

- A log archive destination must be set for each database to transport redo when it is a primary database. If broker is regularly configured, the destination for the standby to primary may not be set and should be done so manually after disabling the configuration. The standby to primary destination should use `valid_for(online_logfiles,primary_role)` in order to prevent errors from redo from being shipped from the logical standby to the primary.
- fal_server must be set properly for each database.
- Parameter STANDBY_FILE_MANAGEMENT should be set to AUTO on primary and standby databases to facilitate the creation of new datafiles during redo apply.
- DB_FILE_NAME_CONVERT should be set on both primary and standby databases. **This is especially important for local standby databases so that files are not overwritten**.
- Static services are required to one instance of each database so that the script described later in this paper can start the databases throughout the process of database rolling maintenance. My Oracle Support Note 1387859.1, Oracle Data Guard Broker and Static Service Registration[5], can be used to assist in setting up static services if not already defined.

5 https://support.oracle.com/epmos/faces/DocumentDisplay?id=1387859.1

» NOTE: if a static service has already been configured for Data Guard Broker they may be used.

» Net service names which use a connect descriptor for the static service are required in the tnsnames.ora where the script will be run.

Conversion Example

Enabling Transparent Data Encryption

TDE utilizes wallets to store the master encryption key. While the default database wallet can be used, Oracle strongly recommends using a specific wallet for TDE by using the ENCRYPTION_WALLET_LOCATION parameter in sqlnet.ora. Additionally, using an auto-login wallet relieves the administrator from opening the wallet manually each time the database is started.

Logical standby databases restrict the creation of wallets. Therefore, the wallets must be created on the primary and copied to the standby prior to the execution of this process.

NOTE: The new wallet should not be used for encryption on a primary RAC database until the end of this process.

The wallet will be created on one primary instance and must be manually copied to all other nodes of a primary and standby database.

1. Create encryption wallet

Set the wallet location in the sqlnet.ora on all nodes of primary and standby.

```
ENCRYPTION_WALLET_LOCATION =
   (SOURCE = (METHOD = FILE)
     (METHOD_DATA =
      (DIRECTORY = /u01/app/oracle/admin/TDE/$ORACLE_SID)
     )
   )
```

NOTE: Using ORACLE_SID in the directory path ensures that all databases do not share the wallet. If there is just one database on the system the ORACLE_SID is not necessary.

2. Create the corresponding directory on all nodes with the proper ORACLE_SID.

```
mkdir -p /u01/app/oracle/admin/TDE/$ORACLE_SID
```

3. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora to be picked up.

4. Set the Master Encryption Key

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "AbCdEfGh!";
```

NOTE: Ensure the password string in double quotation marks (" ").

5. Create Auto-login wallet

An Auto-login wallet removes the requirement of manually opening the wallet when the database is started.

```
orapki wallet create -wallet /u01/app/oracle/admin/TDE/$ORACLE_SID -auto_login
```

6. Copy the files generated in the keystore directory to all nodes of the primary and standby.

Copy files to each node:

```
scp  /u01/app/oracle/admin/TDE/$ORACLE_SID/* oracle@<host>:/u01/app/oracle/admin/TDE/<SID_NAME>/
```

7. Ensure the wallet is open on all nodes

```
SQL> select * from gv$encryption_wallet;

   INST_ID WRL_TYPE
---------- --------------------
WRL_PARAMETER
--------------------------------------------------------------------------------
STATUS
------------------
         1 file
/u01/app/oracle/admin/TDE/$ORACLE_SID
OPEN.
```

Transient Logical Script - First Execution

The process used for database rolling upgrades using a transient logical standby database can be applied to many types of migration that would otherwise require an extended outage. The transient logical process gets its name because it begins and ends with a physical standby database. The logical apply process is only used while the configuration is replicating between different versions of the database - a primary database that has not yet been upgraded or migrated and a standby database where the upgrade or migration is complete. While the transient logical process may be performed manually as described in Data Guard documentation, the best practices described in this paper utilize a script provided by Oracle that automates many of the manual steps along with additional levels of validation that occur automatically throughout the process.

For a detailed description of the transient logical rolling maintenance process utilizing the Oracle provided script please refer to Oracle MAA best practices for Data Guard Transient Logical standby databases[6]. The physru.sh script described in this paper can be downloaded from My Oracle Support Note 949322.1[7].

Convert your physical standby database to a logical standby.

1. Verify all the assumptions are correct in the environment.

2. Verify that the standby is caught up and only lagging by seconds.

3. If standby is RAC, stop all instances but one, set cluster_database=false scope=spfile and bring the one instance to mounted state. The static listener should reside on the node where the one standby instance is running. This is required because a standby must be mounted in order to create restore points which are used by the script.

```
SQL> alter system set cluster_database=false scope=spfile;

System altered.
```

6 http://www.oracle.com/technetwork/database/features/availability/maa-wp-11g-upgrades-made-easy-131972.pdf

7 https://support.oracle.com/epmos/faces/DocumentDisplay?id=949322.1

```
$ srvctl stop database -d standby
$ srvctl start instance -d standby -i standby1 -o mount

SQL> recover managed standby database using current logfile disconnect;
Media recovery complete.
```

4. Run the "physru" Upgrade Script to completion. See Appendix A for sample output.

```
Example: physru.sh <username> <static_primary_tns> <static_standby_tns>
<primary_name> <standby_name> <upgrade_version>

Arguments:

  <username>        = dba username
  <primary_tns>     = static tns service name to primary
  <standby_tns>     = static tns service name to physical standby
  <primary_name>    = db_unique_name of primary
  <standby_name>    = db_unique_name of standby
  <upgrade_version> = Since there is no upgrade, use current version
```

You may receive the following warning if there are unsupported data types in the database:

```
WARN: Objects have been identified on the primary database which will not be
replicated on the transient logical standby. The complete list of objects and
their associated unsupported datatypes can be found in the
dba_logstdby_unsupported view. For convenience, this script has written the
contents of this view to a file - physru_unsupported.log.

Various options exist to deal with these objects such as:

 - disabling applications that modify these objects
 - manually resolving these objects after the upgrade
 - extending support to these objects (see My Oracle Support Note: 559353.1)

If you need time to review these options, you should enter 'n' to exit
the script.  Otherwise, you should enter 'y' to continue with the rolling
upgrade.

Are you ready to proceed with the rolling upgrade? (y/n):
```

In this case review the physru_unsupported.log file from the execution directory for details and resolve accordingly.

Note, if you receive the error below, it indicates that the standby is open read only. If this is encountered simply mount a single instance of the standby with cluster_database=false and re-execute the script.

```
ERROR at line 1:
ORA-16000: database open for read-only access
ORA-06512: at line 6

### The offending sql code in its entirety:
set pagesize 0 feedback off verify off heading off echo off tab off
whenever sqlerror exit sql.sqlcode
```

```
declare
  cursor curs is
    select name from v$restore_point where name like 'PRU_%';
begin
  for r_curs in curs loop
    execute immediate 'drop restore point ' || r_curs.name;
  end loop;
end;
/
exit;
```

```
Apr 03 20:23:51 2015 [0-1] ERROR: failed to purge script state from database standby
```

Move Data to TDE Tablespace

This part of the migration uses data pump export[8] and import[9] to unload and reload the data. There are many ways to approach this. There should be some thought given to how to break up the reload of data whether by tablespace or by schema or the full database. For best performance the PARALLEL parameter should be used on export and import and EXCLUDE=statistics should be used on the import followed by gathering statistics once the logical becomes a primary. COMPRESSSION can be used to reduce the disk storage requirement of the export files but that may extend the export and import time.

Note: SYSTEM and SYSAUX tablespaces are not compatible with TDE and should not be converted.

1. Stop logical apply

```
SQL> alter database stop logical standby apply;
```

2. Run a Datatpump export for all tablespaces which are to be converted to TDE tablespace encryption. Using the parallel option can improve overall performance, as can setting DB_BLOCK_CHECKING=FALSE and DB_BLOCK_CHECKSUM=FALSE

```
$ expdp "'"sys as sysdba"'" compression=all parallel=4 dumpfile=TDE_%U.dmp
logfile=TDE_exp.log tablespaces=TS1[,TS2,…]
```

This example will use the default directory DATA_PUMP_DIR as the export directory. Ensure there is sufficient space in the directory as suggested in the assumptions. A different directory may also be configured and used.

3. Disable guard status

This step ensures indexes can be rebuilt or else the import will fail on indexes.

```
SQL> select guard_status from v$database;
```

8 http://docs.oracle.com/cd/E11882_01/server.112/e22490/dp_export.htm#SUTIL200

9 http://docs.oracle.com/cd/E11882_01/server.112/e22490/dp_import.htm#SUTIL300

```
GUARD_S
-------
ALL

SQL> alter database guard none;
SQL> select guard_status from v$database;

GUARD_S
-------
NONE
```

4. Drop all guaranteed restore points created by the script and any existing prior to the script.

Tablespaces cannot be modified nor dropped when guaranteed restore points exist so they must be dropped.

First, gather the scn and name for each existing restore point.

```
SQL> col name format a50
SQL> script STANDBY_restore_point_history.log

SQL> select name,scn from v$restore_point order by TIME;
NAME                                                        SCN
---------------------------------------------------- ----------
PRU_0000_0001                                       2019288602
PRU_0101                                            2019288602
PRU_0201                                            2019290873
PRU_0202                                            2019293221
PRU_0203                                            2019293425
PRU_0204                                            2019308174

SQL> script STANDBY_restore_point_history.log
```

As a protective measure, also gather this information from the primary.

```
SQL> col name format a50
SQL> script PRIMARY_restore_point_history.log

SQL> select name,scn from v$restore_point order by TIME;
NAME                                SCN
---------------------------------------------------- ----------
PRU_0000_0001                        2019288658

SQL> script PRIMARY_restore_point_history.log
```

This block can be used to drop all restore points only on the standby:

```
set serveroutput on
declare
  cursor curs is
    select name from v$restore_point ;
```

```
begin
  for r_curs in curs loop
    execute immediate 'drop restore point ' || r_curs.name;
  end loop;
end;
```

5. Drop exported tablespaces

The DBMS_METADATA.GET_DDL procedure can be used to retrieve the tablespace DDL.

```
SQL> set long 99999
SQL> select dbms_metadata.get_ddl('TABLESPACE','TS1') from dual;


DBMS_METADATA.GET_DDL('TABLESPACE','TS1')
--------------------------------------------------------------------------------

  CREATE BIGFILE TABLESPACE "TS1" DATAFILE
  SIZE 3221225472
  AUTOEXTEND ON NEXT 1073741824 MAXSIZE 33554431M
  LOGGING ONLINE PERMANENT BLOCKSIZE 8192
  EXTENT MANAGEMENT LOCAL AUTOALLOCATE DEFAULT
 NOCOMPRESS  SEGMENT SPACE MANAGEMENT AUTO

SQL> drop tablespace TS1 including contents and datafiles;
```

6. Recreate exported tablespaces with encryption clause as below.

NOTE: The following encryption algorithms are available with TDE:

- 3DES168
- AES128
- AES192
- AES256

```
SQL> CREATE BIGFILE TABLESPACE "TS1" DATAFILE

  SIZE 3221225472

  AUTOEXTEND ON NEXT 1073741824 MAXSIZE 33554431M

  LOGGING ONLINE PERMANENT BLOCKSIZE 8192

  EXTENT MANAGEMENT LOCAL AUTOALLOCATE ENCRYPTION using 'AES256' DEFAULT
STORAGE(ENCRYPT) SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.
```

7. Import database to TDE encrypted tablespace.

```
$ impdp "'"sys as sysdba"'" PARALLEL=4 EXCLUDE=statistics DUMPFILE=TDE_%U.dmp
LOGFILE=TDE_imp.log
```

Using PARALLEL in a RAC database requires that files are on shared storage accessible by all instances. If other instances are left down in the step, the requirement is removed.

8. Re-enable guard status and start logical apply

```
SQL> alter database guard all;

Database altered.

SQL> select guard_status from v$database;

GUARD_S
-------
ALL

SQL> ALTER DATABASE START LOGICAL STANDBY APPLY immediate;

Database altered.
```

9. **THIS STEP IS CRITICAL** to ensure the transient logical script continues at the proper place:

Recreate the first and last restore points from the data gathered in step 4 in order. These restore points are simply a method to track where the script left off. The SCN is not relevant, only the name and order of creation.

```
SQL> create restore point PRU_0000_0001;

Restore point created.

SQL> create restore point PRU_0204;

Restore point created.
```

10. RAC can now be re-enabled at the standby in preparation of it becoming the primary database.

```
SQL> alter system set cluster_database=true scope=spfile;

System altered.

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

$ srvctl start database -d standby -o open
```

11. Start logical apply on the standby instance configured with the static listener used by the script.

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY immediate;

Database altered.
```

12. Ensure the the remote destination is set for the current primary database so that redo shipping can continue after switchover.

```
SQL> select value from v$parameter where name='log_archive_dest_2';

VALUE
--------------------------------------------------------------------------------
service="primary" ASYNC db_unique_name="primary"
valid_for=(all_logfiles,primary_role)
```

13. Compare the current scn of the primary to the applied scn of the logical standby.

STANDBY

SQL> select APPLIED_SCN from V$LOGSTDBY_PROGRESS;

APPLIED_SCN

2019754453

PRIMARY

SQL> select current_scn from v$database;

CURRENT_SCN

2019754466

Proceed when the two SCNs are within a couple hundred values of each other. See Oracle documentation for additional guidance managing and tuning a logical standby database[10].

Transient Logical Script-Second Execution (Planned Maintenance Window starts)

Running the physru.sh script for the second time with identical inputs executes the switch over to the transient logical where TDE was enabled, flashes back the new standby (the original primary) and converts it to a physical standby of the new primary database.

1. Execute physru.sh with the same arguments as the previous execution. See Appendix B for sample output. This should be done during a planned maintenance window when there is low activity.

```
$ ./physru.sh sys primary1_static standby1_static primary standby 11.2.0.4.0
```

10 http://docs.oracle.com/cd/E11882_01/server.112/e41134/manage_ls.htm#SBYDB00800

2. Step 5 will pause while the old primary is brought to a single mounted instance(cluster_database does not need to be changed here as before):

```
WARN: primary is a RAC database.  Before this script can continue, you
must manually reduce the RAC to a single instance.  This can be accomplished
with the following step:

 1) Shutdown all instances other than instance primary1.
    eg: srvctl stop instance -d primary -i primary2 -o abort

Once these steps have been performed, enter 'y' to continue the script. If
desired, you may enter 'n' to exit the script to perform the required steps,
and recall the script to resume from this point.

$ srvctl stop instance -d primary -i primary2 -o abort
```

3. Enter 'y' in the script prompt to continue

4. The script will stop after the new standby is shut down. This is required when this script is used for a rolling upgrade to change the binaries, however, it is only necessary at this point to start the new standby in a mounted state before continuing. All instances may be started in a RAC configuration.

```
NOTE: Database primary has been shutdown, and is now ready to be started
      using the newer version Oracle binary.  This script requires the
      database to be mounted (on all active instances, if RAC) before calling
      this script to resume the rolling upgrade.

NOTE: Database primary is no longer limited to single instance operation since
      the database has been successfully converted into a physical standby.
      For increased availability, Oracle recommends starting all instances in
      the RAC on the newer binary by performing the following step:

        1) Startup and mount all instances for database primary
        eg: srvctl start database -d primary -o mount

$ srvctl start database -d primary -o mount
```

5. Drop the restore points on the new standby**. Failure to do so will cause recovery to fail once it hits the dropping of the tablespace(s)**. This same block can be used to drop the restore points on the new standby only.

```
set serveroutput on
declare
  cursor curs is
    select name from v$restore_point ;
begin
  for r_curs in curs loop
    execute immediate 'drop restore point ' || r_curs.name;
```

```
  end loop;
end;
```

Transient Logical Script-Final Execution

The final execution will synchronize the new standby (original primary) with all redo generated by the new primary. This includes all application activity that occurred at the original primary database while the transient logical database was being migrated to TDE, plus all redo directly associated with the conversion to TDE, and all transactions that have occurred at the new primary after the initial switchover. At the end of the script you have to option to switch back to the previous primary/standby configuration

1. Execute physru.sh with the same arguments as the previous executions one last time. See Appendix C for sample output.

./physru.sh sys primary1_static standby_static primary standby 11.2.0.4.0

2. Note that you may see the following message in the alert log if you are monitoring it.

```
ORA-19906: recovery target incarnation changed during recovery
Managed Standby Recovery not using Real Time Apply
Recovery Slave PR00 previously exited with exception 19906
```

This is due to the remote log archive destination on the new primary timing out. You can ignore this and reconnection will eventually occur or you can set the log_archive_dest_state_n=enable on the new primary to force a reconnection.

3. Depending on the size of the online redo logs(which need to be cleared before MRP starts), the amount of redo needed to apply and the speed of the system it is possible for the script to time out after 10 minutes.

```
Apr 02 14:07:17 2015 [6-1] ERROR: timed out after 10 minutes of inactivity
```

If this occurs simply restart the script with the same arguments.

4. At the end of the script there will be a prompt to choose whether to switch back to the original configuration. If so desired this can be performed immediately by entering ‘y’ otherwise enter ‘n’ and perform the switch back when so desired.

```
NOTE: At this point, you have the option to perform a switchover
     which will restore primary back to a primary database and
     standby back to a physical standby database.  If you answer 'n'
     to the question below, primary will remain a physical standby
     database and standby will remain a primary database.

Do you want to perform a switchover? (y/n):
```

5. If the switchover is chosen, the new primary database must be brought to single instance for which the script will prompt.

```
WARN: standby is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
      accomplished with the following step:
```

```
        1) Shutdown all instances other than instance standby1.
           eg: srvctl stop instance -d standby -i standby2

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n):

$ srvctl stop instance -d standby -i standby2
```

6. Enter ‘y’ to continue the script

7. Only the first instance of the primary is now open, open any mounted instances as directed by the script.

8. The standby has only one mounted instance. Restart the standby to the desired state (Active Data Guard or mounted), and start managed recovery.

9. The script then produces statistics about the process, an example of which can be seen in appendix D.

10. Gather statistics on the objects which have been reloaded. This can also be done while the new old primary is applying redo from the conversion or if the reversed roles will be in place for a period of time.

Conclusion

Converting to Oracle Advanced Security Transparent Data Encryption provides protection for data at rest in an Oracle Database. The process of encrypting the data is a challenge with 24/7 requirements for many applications. Using a transient logical standby database can minimize the impact to availability of application while achieving the goal of encrypted data at rest.

Appendix A - First execution of the physru.sh script

```
oracle@slcc32adm05 andy]$ ./physru.sh sys primary1_static standby1_static primary
standby 11.2.0.4.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Apr 04 08:22:58 2015 [0-1] Identifying rdbms software version
Apr 04 08:22:58 2015 [0-1] database primary is at version 11.2.0.4.0
Apr 04 08:22:58 2015 [0-1] database standby is at version 11.2.0.4.0
Apr 04 08:22:58 2015 [0-1] verifying flashback database is enabled at primary and
standby
Apr 04 08:22:59 2015 [0-1] verifying available flashback restore points
Apr 04 08:22:59 2015 [0-1] verifying DG Broker is disabled
Apr 04 08:22:59 2015 [0-1] looking up prior execution history
Apr 04 08:22:59 2015 [0-1] purging script execution state from database primary
Apr 04 08:22:59 2015 [0-1] purging script execution state from database standby
Apr 04 08:22:59 2015 [0-1] starting new execution of script

### Stage 1: Backup user environment in case rolling upgrade is aborted
Apr 04 08:22:59 2015 [1-1] creating restore point PRU_0000_0001 on database standby
Apr 04 08:23:00 2015 [1-1] backing up current control file on standby
Apr 04 08:23:01 2015 [1-1] created backup control file
/u01/app/oracle/product/11.2.0.4/dbhome_1/dbs/PRU_0001_standby_f.f
Apr 04 08:23:01 2015 [1-1] creating restore point PRU_0000_0001 on database primary
Apr 04 08:23:03 2015 [1-1] backing up current control file on primary
Apr 04 08:23:06 2015 [1-1] created backup control file
/u01/app/oracle/product/11.2.0.4/dbhome_1/dbs/PRU_0001_primary_f.f

NOTE: Restore point PRU_0000_0001 and backup control file PRU_0001_standby_f.f
      can be used to restore standby back to its original state as a
      physical standby, in case the rolling upgrade operation needs to be aborted
      prior to the first switchover done in Stage 4.

### Stage 2: Create transient logical standby from existing physical standby
Apr 04 08:23:06 2015 [2-1] verifying RAC is disabled at standby
Apr 04 08:23:06 2015 [2-1] verifying database roles
Apr 04 08:23:06 2015 [2-1] verifying physical standby is mounted
Apr 04 08:23:07 2015 [2-1] verifying database protection mode
```

```
Apr 04 08:23:07 2015 [2-1] verifying transient logical standby datatype support

WARN: Objects have been identified on the primary database which will not be
      replicated on the transient logical standby.  The complete list of
      objects and their associated unsupported datatypes can be found in the
      dba_logstdby_unsupported view.  For convenience, this script has written
      the contents of this view to a file - physru_unsupported.log.

      Various options exist to deal with these objects such as:
        - disabling applications that modify these objects
        - manually resolving these objects after the upgrade
        - extending support to these objects (see metalink note: 559353.1)

      If you need time to review these options, you should enter 'n' to exit
      the script.  Otherwise, you should enter 'y' to continue with the
      rolling upgrade.

Are you ready to proceed with the rolling upgrade? (y/n): y

Apr 04 08:23:11 2015 [2-1] continuing
Apr 04 08:23:11 2015 [2-2] starting media recovery on standby
Apr 04 08:24:30 2015 [2-2] confirming media recovery is running
Apr 04 08:24:30 2015 [2-2] waiting for apply lag to fall under 30 seconds
Apr 04 08:24:37 2015 [2-2] apply lag measured at 7 seconds
Apr 04 08:24:37 2015 [2-2] stopping media recovery on standby
Apr 04 08:24:38 2015 [2-2] executing dbms_logstdby.build on database primary
Apr 04 08:25:02 2015 [2-2] converting physical standby into transient logical standby
Apr 04 08:26:23 2015 [2-3] opening database standby
Apr 04 08:26:25 2015 [2-4] configuring transient logical standby parameters for
rolling upgrade
Apr 04 08:26:25 2015 [2-4] starting logical standby on database standby
Apr 04 08:26:33 2015 [2-4] waiting until logminer dictionary has fully loaded
Apr 04 08:27:34 2015 [2-4] dictionary load 75% complete
Apr 04 08:27:44 2015 [2-4] dictionary load is complete
Apr 04 08:27:45 2015 [2-4] waiting for apply lag to fall under 30 seconds
Apr 04 08:27:51 2015 [2-4] apply lag measured at 6 seconds

NOTE: Database standby is now ready to be upgraded.  This script has left the
      database open in case you want to perform any further tasks before
```

```
      upgrading the database.  Once the upgrade is complete, the database must
      opened in READ WRITE mode before this script can be called to resume the
      rolling upgrade.

NOTE: If standby was previously a RAC database that was disabled, it may be
      reverted back to a RAC database upon completion of the rdbms upgrade.
      This can be accomplished by performing the following steps:

          1) On instance standby1, set the cluster_database parameter to TRUE.
          eg: SQL> alter system set cluster_database=true scope=spfile;

          2) Shutdown instance standby1.
          eg: SQL> shutdown abort;

          3) Startup and open all instances for database standby.
          eg: srvctl start database -d standby
```

Appendix B - Second execution of the physru.sh script

```
[oracle@slcc32adm05 andy]$ ./physru.sh sys primary1_static standby1_static primary
standby 11.2.0.4.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Apr 04 08:53:45 2015 [0-1] Identifying rdbms software version
Apr 04 08:53:45 2015 [0-1] database primary is at version 11.2.0.4.0
Apr 04 08:53:45 2015 [0-1] database standby is at version 11.2.0.4.0
Apr 04 08:53:46 2015 [0-1] verifying flashback database is enabled at primary and
standby
Apr 04 08:53:46 2015 [0-1] verifying available flashback restore points
Apr 04 08:53:46 2015 [0-1] verifying DG Broker is disabled
Apr 04 08:53:46 2015 [0-1] looking up prior execution history
Apr 04 08:53:46 2015 [0-1] last completed stage [2-4] using script version 0001
Apr 04 08:53:46 2015 [0-1] resuming execution of script

### Stage 3: Validate upgraded transient logical standby
Apr 04 08:53:47 2015 [3-1] database standby is no longer in OPEN MIGRATE mode
Apr 04 08:53:47 2015 [3-1] database standby is at version 11.2.0.4.0

### Stage 4: Switch the transient logical standby to be the new primary
Apr 04 08:53:49 2015 [4-1] waiting for standby to catch up (this could take a while)
Apr 04 08:53:49 2015 [4-1] waiting for apply lag to fall under 30 seconds
Apr 04 08:53:53 2015 [4-1] apply lag measured at 4 seconds
Apr 04 08:53:53 2015 [4-2] switching primary to become a logical standby
Apr 04 08:54:23 2015 [4-2] primary is now a logical standby
Apr 04 08:54:23 2015 [4-3] waiting for standby standby to process end-of-redo from
primary
Apr 04 08:54:23 2015 [4-4] switching standby to become the new primary
Apr 04 08:54:36 2015 [4-4] standby is now the new primary

### Stage 5: Flashback former primary to pre-upgrade restore point and convert to
physical
Apr 04 08:54:37 2015 [5-1] verifying instance primary1 is the only active instance

WARN: primary is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
```

```
      accomplished with the following step:

        1) Shutdown all instances other than instance primary1.
           eg: srvctl stop instance -d primary -i primary2 -o abort

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n): y

Apr 04 08:58:45 2015 [5-1] continuing
Apr 04 08:58:45 2015 [5-1] verifying instance primary1 is the only active instance
Apr 04 08:58:45 2015 [5-1] shutting down database primary
Apr 04 08:59:07 2015 [5-1] mounting database primary
Apr 04 08:59:20 2015 [5-2] flashing back database primary to restore point
PRU_0000_0001
Apr 04 08:59:21 2015 [5-3] converting primary into physical standby
Apr 04 08:59:22 2015 [5-4] shutting down database primary

NOTE: Database primary has been shutdown, and is now ready to be started
      using the newer version Oracle binary.  This script requires the
      database to be mounted (on all active instances, if RAC) before calling
      this script to resume the rolling upgrade.

NOTE: Database primary is no longer limited to single instance operation since
      the database has been successfully converted into a physical standby.
      For increased availability, Oracle recommends starting all instances in
      the RAC on the newer binary by performing the following step:

        1) Startup and mount all instances for database primary
        eg: srvctl start database -d primary -o mount
```

Appendix C- Third execution of the physru.sh script

```
[oracle@slcc32adm05 andy]$ ./physru.sh sys primary1_static standby1_static primary
standby 11.2.0.4.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Apr 04 09:08:12 2015 [0-1] Identifying rdbms software version
Apr 04 09:08:12 2015 [0-1] database primary is at version 11.2.0.4.0
Apr 04 09:08:12 2015 [0-1] database standby is at version 11.2.0.4.0
Apr 04 09:08:12 2015 [0-1] verifying flashback database is enabled at primary and
standby
Apr 04 09:08:13 2015 [0-1] verifying available flashback restore points
Apr 04 09:08:13 2015 [0-1] verifying DG Broker is disabled
Apr 04 09:08:13 2015 [0-1] looking up prior execution history
Apr 04 09:08:13 2015 [0-1] last completed stage [5-4] using script version 0001
Apr 04 09:08:13 2015 [0-1] resuming execution of script

### Stage 6: Run media recovery through upgrade redo
Apr 04 09:08:14 2015 [6-1] upgrade redo region identified as scn range [1893786448,
1893791142]
Apr 04 09:08:14 2015 [6-1] starting media recovery on primary
Apr 04 09:08:21 2015 [6-1] confirming media recovery is running
Apr 04 09:08:21 2015 [6-1] waiting for media recovery to initialize
v$recovery_progress
Apr 04 09:09:34 2015 [6-1] monitoring media recovery's progress
Apr 04 09:13:21 2015 [6-2] last applied scn 1893763258 is approaching upgrade redo
start scn 1893786448
Apr 04 09:13:37 2015 [6-3] recovery of upgrade redo at 59% - estimated complete at
Apr 04 09:13:55
Apr 04 09:13:52 2015 [6-4] media recovery has finished recovering through upgrade

### Stage 7: Switch back to the original roles prior to the rolling upgrade

NOTE: At this point, you have the option to perform a switchover
     which will restore primary back to a primary database and
     standby back to a physical standby database.  If you answer 'n'
     to the question below, primary will remain a physical standby
     database and standby will remain a primary database.
```

```
Do you want to perform a switchover? (y/n): y

Apr 04 09:14:26 2015 [7-1] continuing
Apr 04 09:14:27 2015 [7-2] verifying instance standby1 is the only active instance

WARN: standby is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
      accomplished with the following step:

        1) Shutdown all instances other than instance standby1.
           eg: srvctl stop instance -d standby -i standby2

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n): y

Apr 04 09:18:59 2015 [7-2] continuing
Apr 04 09:18:59 2015 [7-2] verifying instance standby1 is the only active instance
Apr 04 09:19:00 2015 [7-2] waiting for apply lag to fall under 30 seconds
Apr 04 09:19:01 2015 [7-2] apply lag measured at 1 seconds
Apr 04 09:19:02 2015 [7-3] switching standby to become a physical standby
Apr 04 09:19:06 2015 [7-3] standby is now a physical standby
Apr 04 09:19:06 2015 [7-3] shutting down database standby
Apr 04 09:19:06 2015 [7-3] mounting database standby
Apr 04 09:19:17 2015 [7-4] waiting for standby primary to process end-of-redo from
primary
Apr 04 09:19:18 2015 [7-5] switching primary to become the new primary
Apr 04 09:19:23 2015 [7-5] primary is now the new primary
Apr 04 09:19:23 2015 [7-5] opening database primary
Apr 04 09:19:26 2015 [7-6] starting media recovery on standby
Apr 04 09:20:44 2015 [7-6] confirming media recovery is running

NOTE: Database primary has completed the switchover to the primary role, but
      instance primary1 is the only open instance.  For increased availability,
      Oracle recommends opening the remaining active instances which are
      currently in mounted mode by performing the following steps:
```

1) Shutdown all instances other than instance primary1.
eg: srvctl stop instance -d primary -i primary2

2) Startup and open all inactive instances for database primary.
eg: srvctl start database -d primary

NOTE: Database standby is no longer limited to single instance operation since it has completed the switchover to the physical standby role. For increased availability, Oracle recommends starting the inactive instances in the RAC by performing the following step:

1) Startup and mount inactive instances for database standby
eg: srvctl start database -d standby -o mount

Appendix D- physru.sh statistics

```
### Stage 8: Statistics
script start time:                                  04-Apr-15 08:42:28
script finish time:                                 04-Apr-15 09:21:27
total script execution time:                              +00 00:38:59
wait time for user upgrade:                               +00 00:11:14
active script execution time:                             +00 00:27:45
transient logical creation start time:
transient logical creation finish time:
primary to logical switchover start time:           04-Apr-15 08:53:53
logical to primary switchover finish time:          04-Apr-15 08:54:37
primary services offline for:                             +00 00:00:44
total time former primary in physical role:               +00 00:14:55
time to reach upgrade redo:                               +00 00:04:03
time to recover upgrade redo:                             +00 00:00:15
primary to physical switchover start time:          04-Apr-15 09:14:26
physical to primary switchover finish time:         04-Apr-15 09:19:26
primary services offline for:                             +00 00:05:00

SUCCESS: The physical rolling upgrade is complete.
```

Appendix E- Alternative Methods to Convert to TDE

There are alternatives to the method used in this paper when converting to TDE encryption. Depending on the size of the database, storage available and tolerance of downtime the following approaches may also be considered but will require in-house development of steps.

- Use DBMS_REDEFINITION in the active primary database. This is an option for databases without a standby database or for administrators more comfortable with the DBMS_REDEFINITION process. The main benefit is zero to very low downtime. The trade off is that operational investment can vary depending on targeted objects.
- Use ALTER TABLE MOVE in lieu of data pump in this process. Create a like sized encrypted tablespace and move each segment to the encrypted tablespace. This method may be preferable in situations where storage is not a limiting factor as the database will double in size.
- Take an outage- This process was written to minimize the down time to the application and performance impact during migration, however, if these are not concerns then the changes can be made in the primary database during a maintenance window. The most likely approach to perform this conversion is with data pump.

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

AUTHOR: ANDREW STEINORTH

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together