Oracle Maximum
Availability Architecture

Converting to Transparent Data Encryption Using Active Data Guard (DBMS_ROLLING)

Oracle Database 12c

ORACLE WHITE PAPER | MAY 2015

Table of Contents

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 an Active Data Guard standby database and the DBMS_ROLLING PL/SQL package[1] to avoid affecting production database performance and availability during the process of converting to TDE. 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 solely for facilitating conversion to TDE, the process of conversion includes the following high level steps:

1. Presence of an Active Data Guard physical standby database with no archive log gaps.
2. Conversion of the physical standby to a logical standby using the DBMS_ROLLING PL/SQL package.
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 Active 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 Active 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[2].

1 http://st-doc.us.oracle.com/database/121/SBYDB/dbms_rolling_upgrades.htm#SBYDB5432

2 https://docs.oracle.com/database/121/ASOAG/asotrans.htm#ASOAG10117

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

Existing tablespaces cannot be altered to enable TDE. Tablespace encryption can only be enabled during the creation of a tablespace. Oracle MAA best practice recommends using an Active Data Guard standby database to eliminate any impact to primary database performance or availability while tablespaces are being converted to TDE. The migration to TDE begins by using the DBMS_ROLLING PL/SQL package to temporarily convert a physical standby database to a transient logical standby. The administrator then exports the data using Oracle Data Pump, drops the existing tablespace and then uses import to create the new TDE enabled tablespace. Once complete, Active Data Guard automatically resynchronizes the standby with all transactions that had occurred at the primary while data was being encrypted. This all occurs without any impact to production running at the primary database. Application downtime is limited to the time required to switch production users to the new encrypted copy of the production database.

The Active Data Guard DBMS_ROLLING PL/SQL package is used to automate:

- Conversion of the physical standby to a logical standby.
- Resynchronization after the logical standby has been converted to TDE.
- Switchover of production to the TDE encrypted logical standby to make it the new primary database.
- Conversion of the original primary into a new physical standby and its conversion to TDE.
- Resynchronization of the new standby with the new primary database.
- Switchback of production to the original primary database.

Note: DBMS_ROLLING requires a license for Active Data Guard. DBMS_ROLLING greatly simplifies use of a transient logical standby database to perform database maintenance and upgrades in rolling fashion.

DBMS_ROLLING / Logical Standby Restrictions

Since DBMS_ROLLING utilizes a logical standby database, any 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. You can work around this limitation by replicating just those tables post Data Guard role transition.

3 http://st-doc.us.oracle.com/database/121/SBYDB/data_support.htm#SBYDB00305

- Transportable tablespaces cannot transport encrypted tablespaces.
- Transportable tablespaces cannot transport tablespaces containing tables with encrypted columns.
- Data type restrictions (12.1):
  - BFILE
  - ROWID, UROWID
  - Collections (including VARRAYs and nested tables)
  - Objects with nested tables and REFs
  - The following Spatial types are not supported:
    - MDSYS.SDO_GEORASTER
    - MDSYS.SDO_TOPO_GEOMETRY
  - Identity columns

Note: Extended Datatype Support can be utilized to mitigate data type restrictions. See the Oracle documentation for more information about Extended Datatype Support with Oracle Database 12c[4].

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. The Broker must be disabled on both the primary and the 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.

4 http://st-doc.us.oracle.com/database/121/SBYDB/manage_ls.htm#SBYDB5149

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 the broker is regularly configured, the transport 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 redo shipping errors as a result of redo being shipped logical standby to the primary.
- fal_server must be set properly for each database.
- The 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**.

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.

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. Create the password-based keystore

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
'/u01/app/oracle/admin/TDE/<ORACLE_SID>' IDENTIFIED BY "AbCdEfGh!";
```

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

5. Open the wallet

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "AbCdEfGh!";
```

6. Set the Encryption Key

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "AbCdEfGh!"  WITH BACKUP USING 'TDE';
```

7. Create Auto-login wallet

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

```
ADMINISTER KEY MANAGEMENT CREATE  AUTO_LOGIN KEYSTORE FROM KEYSTORE
'/u01/app/oracle/admin/TDE/$ORACLE_SID' IDENTIFIED BY “AbCdEfGh!”;
```

8. 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>/
```

9. 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/primary1
OPEN.
```

Convert Physical Standby to Logical Standby

The DBMS_ROLLING package will be used for the transient logical part of the process. The DBMS_ROLLING operations are restart-able so if any errors are encountered, simply correct the issue and rerun the step.

1. Initialize plan with DBMS_ROLLING.

```
SQL> exec DBMS_ROLLING.init_plan(‘standby’);

PL/SQL procedure successfully completed.
```

2. Bring the standby to a single mounted instance and restart recovery as required by DBMS_ROLLING.

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

```
SQL> recover managed standby database disconnect;
```

3. Build DBMS_ROLLING plan

```
SQL> exec dbms_rolling.build_plan;
```

```
PL/SQL procedure successfully completed.
```

The plan can be viewed with the following query:

SQL> col instid format 999
SQL> col target format a10
SQL> col phase format a10
SQL> col description format a65
SQL> set lines 99
SQL> set pages 999
SQL> SELECT instid, target, phase, description FROM DBA_ROLLING_PLAN;

4. Start DBMS_ROLLING plan (this converts the physical standby to logical)

```
SQL> exec dbms_rolling.start_plan;
```

5. Verify the standby is a logical and applying redo from the primary by running query below to see that APPLIED_SCN is incrementing:

```
SQL> select database_role,open_mode from v$database;
```

```
DATABASE_ROLE    OPEN_MODE
---------------- --------------------
LOGICAL STANDBY  READ WRITE
```

```
SQL> SELECT APPLIED_SCN FROM V$LOGSTDBY_PROGRESS;
```

Note: if redo is not applying and there are errors in the logical standby's alert log like the one below you may be hitting bug 20889894.

```
ORA-20000: Unable to gather statistics concurrently: Resource Manager is not enabled.
```

As a workaround execute the following and restart logical apply :

```
SQL> exec dbms_stats.set_global_prefs('CONCURRENT', 'FALSE');
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY immediate;
```

Move Data to TDE Tablespace

1. Stop logical apply

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

2. Run a Datapump export for all tablespaces which are to be converted to TDE tablespace encryption.

```
$ expdp "'"sys as sysdba"'" compression=all dumpfile=TDE.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;
```

```
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 DBMS_ROLLING package.

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
---------------------------------------------------- ----------
DBMSRU_INITIAL                                       197267580
```

```
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
--------------------------------------------------- ----------
DBMSRU_INITIAL                       197267789
```

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"'" DUMPFILE=TDE.dmp LOGFILE=TDE_imp.log
```

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. Ensure the remote destination is set for the current primary database so that redo shipping can continue after switchover and set to point to the primary if necessary.

```
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)
```

10. 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.

Note: if redo is not applying and there are errors in the logical standby's alert log like the one below you may be hitting bug 20889894.

```
ORA-20000: Unable to gather statistics concurrently: Resource Manager is not enabled.
```

As a workaround execute the following and restart logical apply :

```
SQL> exec dbms_stats.set_global_prefs('CONCURRENT', 'FALSE');

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

Switchover - Logical Standby Becomes Primary

1. From the current primary, run the DBMS_ROLLING.switchover procedure to convert the logical standby to a primary database

```
SQL> exec dbms_rolling.switchover;
```

2. Verify the old logical is now a primary database and open read/write

```
SQL> select database_role,open_mode from v$database;

DATABASE_ROLE    OPEN_MODE
---------------- --------------------
PRIMARY          READ WRITE
```

3. At this point only one new primary instance is started. Start any additional instances.

4. Restart the new standby (original primary) as single mounted instance for conversion to physical standby.

```
$ srvctl stop database -d primary
```

```
$ srvctl start instance -d primary -i primary1 -o mount
```

5. On the new primary execute the final steps of the DBMS_ROLLING plan which will flash back the old primary, convert it to a physical standby and apply the redo for the new primary. This command will be expected to fail when recovery detects redo changes for dropped tablespace commands. It is helpful to monitor the alert log of the old primary while this step is executed.

FINISH_PLAN will eventually fail when recovery hits the dropping of the old unencrypted tablespace due to the existence of a guaranteed restore point which is created by DBMS_ROLLING.

```
SQL> exec dbms_rolling.finish_plan;
BEGIN dbms_rolling.finish_plan; END;
```

```
*
```

```
ERROR at line 1:
ORA-45415: instruction execution failure
ORA-06512: at "SYS.DBMS_ROLLING", line 36
ORA-06512: at line 1
```

The old primary alert log will show the following message:

```
Thu Apr 09 11:34:15 2015
Errors in file
/u01/app/oracle/diag/rdbms/primary/primary1/trace/primary1_pr00_98452.trc:
ORA-38882: Cannot drop tablespace TS1 on standby database due to guaranteed restore
points.
Thu Apr 09 11:34:15 2015
Managed Standby Recovery not using Real Time Apply
Thu Apr 09 11:34:15 2015
Recovery interrupted!
```

Confirmation of the failure's cause can be confirmed on the new primary with the following queries:

DBA_ROLLING_EVENTS tracks completion of steps and error messages for the plan.

```
SQL> select event_time,MESSAGE from dba_rolling_events where TYPE='ERROR';
```

```
EVENT_TIME
-------------------------------------------------------------------------------
MESSAGE
-------------------------------------------------------------------------------

09-APR-15 11.34.37.120484 AM
failed with status 45426 in instruction 62

09-APR-15 11.34.37.122758 AM
failure while executing one or more instructions from batch 44

09-APR-15 11.34.37.129782 AM
DBMS_ROLLING.FINISH_PLAN halted due to error
```

Take the failed instruction number and determine that it was waiting for recovery to catch up:

DBA_ROLLING_PLAN lists the steps for the plan

```
SQL> SELECT instid, target, phase, description FROM DBA_ROLLING_PLAN where instid=62;

    INSTID TARGET     PHASE      DESCRIPTION
---------- ---------- ---------- --------------------------------------------------
---
        62 primary    FINISH     Wait until upgrade redo has been fully recovered
```

6. Now that the restore point created by DBMS_ROLLING has been used to flash back the old primary, it can be dropped and recovery restarted on the old primary.

```
SQL> drop restore point DBMSRU_INITIAL;

Restore point dropped.

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

7. Restart DBMS_ROLLING.finish_plan which will start where it left off.

```
SQL> exec dbms_rolling.finish_plan;
```

8. Once complete all instances of the old primary can be started.

9. A switch back to the original configuration is possible at this point is desired.

```
SQL> alter database switchover to primary;
```

Restart original primary:

```
$ srvctl stop database -d primary
```

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

10. Restart original standby and start recovery (or replace the broker configuration)

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

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 standby database to facilitate the encryption process minimizes the impact to availability of application while achieving the goal of encrypted data at rest. Active Data Guard 12c provides new automation with the DBMS_ROLLING PL/SQL package to greatly simplify using a physical standby database and the transient logical rolling maintenance process for this purpose.

Appendix A – 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