Oracle Maximum
Availability Architecture

Converting to Transparent Data Encryption with Oracle Data Guard using Fast Offline Conversion

Oracle Database 12c and Oracle Database 11.2

ORACLE WHITE PAPER | AUGUST 2018

Table of Contents

Introduction

With an increasing number of applications requiring 24/7 availability, both unplanned outages and planned downtime become a significant concern. The Oracle Cloud also requires that production databases be encrypted using Oracle Advanced Security Transparent Data Encryption (TDE). To minimize downtime required to encrypt an unencrypted database either for migration to the cloud or to provide additional security for on-premises databases, Oracle Database 12c and Oracle Database 11.2.0.4 now enable offline in-place conversion of data files to TDE. Combining this capability with Data Guard physical standby enables production workloads to run unaffected on an unencrypted primary database while its standby is converted to TDE. Regardless of the size of the database, total downtime is limited to a brief brownout while application connections are switched to the new encrypted copy of production (a Data Guard switchover) after the process is complete. This new functionality is enabled by a patch for 12.1.0.2 and 11.2.0.4 that must be requested from Oracle Support. See My Oracle Support Note 2148746.1 for instructions on how to access the patch.

Once installed, the patch enables offline, in-place TDE conversion of data files at a Data Guard standby with a DDL command instead of having to reload data which can be time consuming, tedious, and in some cases complex. The patch must be applied to all Oracle Homes in the Data Guard configuration. This process is the recommended Oracle Maximum Availability Architecture best practice for converting to TDE with minimal downtime and least complexity. This supersedes previous methods for converting to TDE described in Appendix A- Alternative Methods.

Whether you 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:

NOTE: As of Oracle Database 12.2, Online encryption can be performed against a primary database which in turn encrypts the standby. See Encryption Conversions for Existing Online Tablespaces in the 12.2 documentation for more details.

Prerequisites

- See MOS 2148746.1 for the most recent information on the patch to the database home for the primary and standby databases required by this process.
- Take or ensure there is a viable backup available.
- Understand TDE implications and restrictions and develop a process for maintaining wallets and keys. Refer to the *Oracle Database Advanced Security Administrator's Guide* (11.2|12.1|12.2) and MOS 1228046.1 for further details.

High Level Steps

1. Verify that the Data Guard configuration is healthy and contains no gaps.
2. Create the encryption wallet, and set the master key.
3. Copy the wallet files to the standby database environment.
4. Place the standby in a mounted state with recovery stopped.
5. On the standby: Encrypt data files in-place and in parallel.
6. On the standby: Restart redo apply and catch up.
7. Execute a Data Guard switchover making the encrypted standby the new primary and the unencrypted primary the new standby.
8. On the NEW standby: Place the new standby database in a mounted state with recovery stopped.

9. On the NEW standby: Encrypt data files in-place and in parallel.
10. On the NEW standby: Restart redo apply and catch up.
11. Optionally execute a Data Guard switchover to reestablish the original configuration.

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.

As always, be sure to execute and verify this process in a test environment before executing in production.

Transparent Data Encryption Overview

Transparent Data Encryption (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 11*g* Release 1 (11.1) and higher.

Refer to the *Oracle Database Advanced Security Administrator's Guide* and MOS 2359020.1 for full details regarding TDE encryption[1].

TDE Offline Data File 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, you must use Oracle Data Pump.
- The offline encryption process described in this paper will use the AES128 encryption algorithm with the key identifier listed in V$DATABASE_KEY_INFO.
- This process is only for application tablespace data files. SYSTEM, SYSAUX, and TEMP tablespaces cannot be encrypted with TDE prior to Oracle Database version 12.2.0.1. While SYSTEM and SYSAUX can be encrypted for 12.2, it is not yet recommended.
- Oracle does not recommend encrypting offline the UNDO tablespace in these releases. Doing so prevents the keystore from being closed, and this prevents the database from functioning. In addition, encrypting the UNDO tablespace while the database is offline is not necessary because all undo records that are associated with any encrypted tablespaces are automatically encrypted in the UNDO tablespace.

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

» The UNDO and TEMP metadata that is generated from sensitive data in an encrypted tablespace is already automatically encrypted. Therefore, encrypting UNDO and TEMP is optional.

Conversion Overview

There are many configuration options for TDE tablespace encryption including but not limited to the type of keystore and storage used. This document describes the most common options. Refer to the Database Advance Security Guide for your specific version for full details. 11.2 | 12.1 | 12.2

Prerequisites

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

» There is an existing physical standby database.

» A current backup has been taken of the database prior to converting data files.

» The patch described in MOS 2148746.1 has been applied to both primary and standby RDBMS installation.

» COMPATIBLE is set to 12.2.0.1, 12.1.0.2 or 11.2.0.4 respectively.

» 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 role transition. To ensure there are no unrecoverable blocks the following query at the primary database should return no rows. See MOS 290161.1 for additional details on nologging operations and handling.

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

Note: If UNRECOVERABLE_CHANGE# is >0 for any datafile, compare the value on the primary to the value on the standby. If they are the same the datafile was copied after the unrecoverable change and no action is necessary.

» A log archive destination (LAD) must be set for each database to transport redo when it is a primary database. If the broker is configured and utilized as per MAA best practices, the LAD will be set automatically after the role transition.

Conversion Example

TDE utilizes wallets and keystores to store the master encryption key. While the default database wallet can be used, Oracle recommends using a separate 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 the primary database and must be manually copied to the standby environment. In RAC configurations the MAA best practice is to create the wallet in a shared location such as an ASM disk group. If no shared storage is available, the wallet must be copied to all nodes of the primary and standby clusters.

If the database is already using an encryption wallet or keystore, the same may be reused but Oracle recommends a new master key be set.

Enabling Transparent Data Encryption for Oracle 11.2

NOTE: The new wallet should not be used for encryption on the primary database until the end of this process. A database bounce is required for all database processes to pick up the wallet location from the sqlnet.ora which is being deferred at the initial primary until the switchover.

1. Set the encryption wallet location.

NOTE: ASM storage of the encryption wallet is not supported until RDBMS version 12.1.0.1

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_UNQNAME)
      )
    )
```

NOTE: Using an environment variable like `$ORACLE_UNQNAME` in the directory path ensures that all databases on a shared system use their own wallet. Using this environment variable requires that the variable be set in CRS(where applicable) and the OS user's environment.

2. Create the corresponding directory on all nodes or shared storage with the proper `ORACLE_UNQNAME`.

```
$ mkdir -p /u01/app/oracle/admin/TDE/<ORACLE_UNQNAME>
```

3. Set the environment variable in CRS(if applicable) and for the OS session.

```
$ srvctl setenv database -d <ORACLE_UNQNAME> -T ORACLE_UNQNAME=<ORACLE_UNQNAME>
```

```
$ export ORACLE_UNQNAME=<ORACLE_UNQNAME>
```

NOTE: Set the ORACLE_UNQNAME variable in login scripts or environment scripts as well. Any session that may start the database or instance must have the variable set in order to find the wallet

4. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora and the environment variable to be picked up by the new session.

5. Set the Master Encryption Key.

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

This step may cause redo apply to stop on the standby. It can be restarted once the wallet is copied to the standby in the step 8.

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

6. Open the wallet.

```
SQL> ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "AbCdEfGh!";


System altered.
```

7. 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_UNQNAME -auto_login
```

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_UNQNAME/*
oracle@<host>:/u01/app/oracle/admin/TDE/<ORACLE_UNQNAME>/
```

9. Restart the standby database to mounted mode.

```
Ensure ORACLE_UNQNAME is set properly and:
SQL> shutdown immediate
SQL> startup mount

OR
```

```
$ srvctl stop database -d <standby DB>
$ srvctl start database -d <standby DB> -o mount
```

10. 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_UNQNAME  OPEN
```

Enabling Transparent Data Encryption for Oracle 12.1 and 12.2

TDE utilizes keystores to store the master encryption key. Oracle recommends using a specific wallet for TDE by using the ENCRYPTION_WALLET_LOCATION parameter in sqlnet.ora. Additionally, using an auto login keystore relieves the administrator from opening the keystore manually each time the database is started.

The keystore will be created on one primary instance and must be manually copied to the standby database environment.

1. Create encryption keystore.

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

```
ENCRYPTION_WALLET_LOCATION =
   (SOURCE = (METHOD = FILE)
     (METHOD_DATA =
      (DIRECTORY = +DATA/$ORACLE_UNQNAME/TDE)
     )
   )
```

NOTE: Using an environment variable like `$ORACLE_UNQNAME` in the directory path ensures that all databases on a shared system use their own wallet. Using this environment variable requires that the variable be set in CRS(where applicable) and the OS user's environment.

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

```
ASMCMD> mkdir +DATAC1/<ORACLE_UNQNAME>/TDE
```

3. Set the environment variable in CRS(if applicable) and for the OS session.

```
$ srvctl setenv database -d <ORACLE_UNQNAME> -T ORACLE_UNQNAME=<ORACLE_UNQNAME>
```

```
$ export ORACLE_UNQNAME=<ORACLE_UNQNAME>
```

NOTE: Set the ORACLE_UNQNAME variable in login scripts or environment scripts as well. Any session that may start the database or instance must have the variable set in order to find the wallet

4. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora and environment variable to be picked up by the new session.

5. Create the password-based keystore.

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '+DATA/<ORACLE_UNQNAME>/TDE' IDENTIFIED BY
"AbCdEfGh!";
```

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

6. Open the keystore.

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

7. Set the Encryption Key.

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "AbCdEfGh!"  WITH BACKUP USING 'TDE'
[CONTAINER=ALL|CURRENT];
```

NOTE: If the database is a non-CDB, omit the CONTAINER clause. Each PDB including the ROOT container (but not PDB$SEED) will have a different master encryption key, and setting CONTAINER=ALL generates a master key for each PDB including ROOT using one command. If only a subset of PDBs requires TDE, first set the ROOT master key followed by each PDB individually.

8. Create auto login keystore.

An auto login keystore removes the requirement of manually opening the keystore when the database is started.

```
ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'+DATA/<ORACLE_UNQNAME>/TDE' IDENTIFIED BY "AbCdEfGh!";
```

8. Copy the files generated in the keystore directory to each standby environment.

Copy files to each node:

```
ASMCMD> cp +DATA/<PRIMARY ORACLE_UNQNAME>/TDE/cwallet.sso /tmp
ASMCMD> cp +DATA/<PRIMARY ORACLE_UNQNAME>/TDE/ewallet.p12 /tmp

<primary host>$ scp /tmp/cwallet.sso ewallet.p12 oracle@<standby host>:/tmp

<standby host> ASMCMD> cp /tmp/cwallet.sso +DATA/<STANDBY ORACLE_UNQNAME>/TDE/
<standby host> ASMCMD> cp /tmp/ewallet.p12 +DATA/<STANDBY ORACLE_UNQNAME>/TDE/

Alternatively, the files can be copied directly from ASM to ASM.

ASMCMD>cp cwallet.sso sys/<password>@stbyhost.+ASM1:+DATA/<STANDBY ORACLE
UNQNAME>/TDE/
ASMCMD>cp ewallet.p12 sys/<password>@stbyhost.+ASM1:+DATA/<STANDBY ORACLE
UNQNAME>/TDE/
```

9. Restart the standby database to mounted mode.

```
Ensure ORACLE_UNQNAME is set properly and:
SQL> shutdown immediate
SQL> startup mount
```

OR

```
$ srvctl stop database -d <standby DB>
$ srvctl start database -d <standby DB> -o mount
```

10. Ensure the keystore is open on all nodes.

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

INST_ID WRL_TYPE WRL_PARAMETER                          STATUS

------- -------- ------------------------------------- ------

      1 file     +DATA/<ORACLE_UNQNAME>/TDE             OPEN
```

Convert Data Files

Only user tablespace data files can be converted in Oracle Database releases 12.1 and 11.2. With release 12.2 SYSTEM, SYSAUX and UNDO tablespaces can also be encrypted.

In version 12.2, the TEMP tablespaces can be encrypted but they cannot be converted. To have an encrypted TEMP tablespace, create an encrypted TEMP tablespace in Oracle Database 12c release 2 (12.2), make it the default temporary tablespace, and then drop the original TEMP tablespace. Remember not to use the wallet to encrypt on the primary until either after a role transition or the primary database has been restarted.

The UNDO and TEMP metadata that is generated from sensitive data in an encrypted tablespace is already automatically encrypted. Therefore, encrypting UNDO and TEMP tablespaces is optional.

This is a datafile by datafile conversion; all datafiles of a given tablespace must be encrypted before the tablespace can be brought online or the database can be opened.

1. Ensure the standby database is mounted and the keystore is open.

```
SQL> select inst_id,database_role,open_mode from gv$database;

   INST_ID DATABASE_ROLE    OPEN_MODE

---------- ---------------- --------------------

         1 PHYSICAL STANDBY MOUNTED

         2 PHYSICAL STANDBY MOUNTED

SQL> select * from gv$encryption_wallet;

INST_ID WRL_TYPE WRL_PARAMETER                            STATUS

------- -------- ---------------------------------------- ------

      1 file     +DATA/<ORACLE_UNQNAME>/TDE               OPEN
```

2. Stop recovery.

```
DGMGRL> edit database tdestby set state=APPLY-OFF;
```

Verify with the following statement:

```
SQL> select * from gv$managed_standby where process='MRP0';

no rows selected
```

While Redo Apply is stopped, redo is still being transported from the primary to the standby database while it is mounted, minimizing the Recovery Point Objective (RPO) in the event of a primary database failure or disaster.

3. Generate the commands for the data files to be encrypted.

Only application tablespace data files can be converted in Oracle Database releases 12.1 and 11.2. SYSTEM, SYSAUX, UNDO, and TEMP cannot be encrypted with TDE.
For 12.2 encrypting UNDO and SYSTEM tablespaces is not recommended.

```
SQL> select 'alter database datafile '||chr(39)||df.name||chr(39)||' encrypt;'
COMMAND from v$tablespace ts, v$datafile df where ts.ts#=df.ts# and
ts.con_id=df.con_id and (ts.name not in ('SYSTEM','SYSAUX') and ts.name not in
(select value from gv$parameter where name='undo_tablespace'));
```

Remove 'ts.con_id=df.con_id and' from the query above for RDBMS version 11.2

This query assumes that all user tablespaces will be encrypted. If a subset is desired, change the WHERE clause accordingly.

For container databases use the following query instead (You may need to open the database in read only mode to get results from this query. Remember to stop and start the database in mounted mode before continuing.) :

```
set lines 120
set pages 9999
spool encrypt.sql
select 'alter session set container='||pdb.name||';'||chr(10)||'alter database datafile '||chr(39)||df.name||chr(39)||'
encrypt;' COMMAND
from v$tablespace ts, v$datafile df, v$pdbs pdb where ts.ts#=df.ts# and ts.con_id=df.con_id and (ts.name not in
('SYSTEM','SYSAUX')
and df.file# not in (select distinct file_id from cdb_undo_extents where con_id > 1)
and ts.name not in (select value from gv$parameter where name='undo_tablespace')) and df.con_id=pdb.con_id;
spool off
```

```
COMMAND
--------------------------------------------------------------------------------
alter database datafile '+DATA/TDESTBY/DATAFILE/users.1163.910254789' encrypt;
alter database datafile '+DATA/TDESTBY/DATAFILE/users.1133.909661023' encrypt;
alter database datafile '+DATA/TDESTBY/DATAFILE/users.1161.910254777' encrypt;
alter database datafile '+DATA/TDESTBY/DATAFILE/soets.1132.909661007' encrypt;
```

4. Convert datafiles in parallel.

Each datafile can be encrypted simultaneously in its own session. In this example, the four data files were converted at the same time in four different windows.

Note: This process uses CPU and I/O resources. In a shared environment, some care should be taken to monitor resources so that multiple encryption processes do not infringe on other databases' resource requirements.

```
SQL> alter database datafile '+DATA/TDESTBY/DATAFILE/users.1163.910254789' encrypt;
Database altered.
SQL> alter database datafile '+DATA/TDESTBY/DATAFILE/users.1133.909661023' encrypt;
Database altered.
SQL> alter database datafile '+DATA/TDESTBY/DATAFILE/users.1161.910254777' encrypt;
```

```
Database altered.

SQL> alter database datafile '+DATA/TDESTBY/DATAFILE/soets.1132.909661007' encrypt;

Database altered.
```

Note: The offline encryption operation is idempotent; if the conversion of a data file is interrupted for any reason simply re-run the alter database command.

5. Optional: Use DBVERIFY to confirm used blocks are encrypted.

```
$ dbv file=+DATAC1/TDESTBY/DATAFILE/users.1161.910254777 USERID=<user>/<password>


DBVERIFY - Verification starting : FILE = +DATAC1/TDESTBY/DATAFILE/users.1161.910254777


DBVERIFY - Verification complete


Total Pages Examined         : 1310720
Total Pages Processed (Data) : 0
Total Pages Failing   (Data) : 0
Total Pages Processed (Index): 0
Total Pages Failing   (Index): 0
Total Pages Processed (Other): 0
Total Pages Processed (Seg)  : 0
Total Pages Failing   (Seg)  : 0
Total Pages Empty            : 559645
Total Pages Marked Corrupt   : 0
Total Pages Influx           : 0
Total Pages Encrypted        : 751075 <-- Total Pages Examined - Total Pages Empty
Highest block SCN            : 0 (0.0)
```

In the DBVerify output, any pages marked as 'Data' or 'Index' are not encrypted. Pages labeled as 'Other' are not Data or Index pages and do not need to be encrypted, any value for 'Other' can be ignored. All encrypted pages whether Data or Index will be grouped together as Encrypted.

6. Restart recovery.

After all datafiles are encrypted restart recovery. Any new redo applied to the encrypted data files will be encrypted. Once the standby is current with no gaps, proceed to step 7.

```
DGMGRL> edit database tdestby set state=APPLY-ON;
```

7. Switchover.

Execute a switchover making the standby the new primary. All new data applied to the standby will be encrypted.

```
DGMGRL> switchover to tdestby;
```

This step is potentially the only outage required for this process (an optional switchback would also incur an outage). Refer to the 'Role Transition Best Practices' whitepaper to optimize switchover and reduce application down time. MAA testing has produced switchovers in less than 1 minute.

Note: While this switchover does not have to occur immediately after datafile encryption on the standby, it is not recommended that the state of a non-encrypted primary with an encrypted standby be left indefinitely.

8. Restart the new standby (the original primary) to pick up the sqlnet.ora changes, and repeat steps 1-5 for the new standby database and all other standby databases in the configuration.

Note: The datafile names are potentially different on the primary than those on the standby.

9. As before, when the conversion steps are complete and the standby has caught up with the primary, optionally switch back to the original configuration.

```
DGMGRL> switchover to tde;
```

Asymmetrical Configurations

When one database in the Data Guard configuration is encrypted and the other is not, the state of encryption of a recovered block changes depending on the RDBMS version.

For Oracle Database 12.2 and later, recovery honors the encryption properties of the database. If a data file is encrypted at the standby, a recovered block to that data file will be encrypted regardless of whether the redo (source primary) is encrypted.

For Oracle Database 12.1 and 11.2, the encryption state of the redo (source primary) is honored, so regardless of the encryption state of the data file at the standby, redo from an unencrypted data file on the primary is stored unencrypted at the standby.

Therefore in Oracle Database 11.2 and 12.1, to ensure all blocks at the standby are encrypted, an additional execution of the encryption commands is potentially required if any redo from an unencrypted primary has been applied. This second execution should be much shorter in duration because there should be fewer blocks to encrypt.

For example:

1. Encrypt standby data files using the described process.

2. Recover the standby to current.
3. Switchover when ready.
4. Encrypt original primary data files.
5. Recover the standby (original primary) to current.
6. Switch back.
7. Run DBVerify against the standby data files, and if unencrypted data blocks are found, encrypt standby data files a second time replacing the 'encrypt' clause with the 'force encrypt' clause.

For Example:

```
SQL> alter database datafile '+DATA/TDESTBY/DATAFILE/users.1163.910254789'
force encrypt;
```

Hardware Keystore

Hardware keystores are supported for Oracle Database 12*c*. For additional details with regard to hardware keystores please review the documentation.

Logical Standby Considerations

Physical standby (Data Guard Redo Apply) is the recommended method for converting to TDE as described in this paper due to reduced complexity. Customers with existing logical standby databases (Data Guard SQL Apply), however, may also use this same process. A logical standby should be treated the same as a physical standby in that the encryption key must be present at the logical standby database. Likewise datafiles containing data from the primary should be encrypted. Repeat steps 1-6 of the 'Convert Datafiles' section for the appropriate datafiles.

For additional details on logical standby databases and TDE please see the relevant documentation (11.2|12.1).

Decrypt

If datafiles need to be decrypted for any reason, there is also a DECRYPT clause. Only datafiles encrypted with this offline encryption conversion method described in the paper can be decrypted with the following method:

```
ALTER DATABASE DATAFILE '<file name>' DECRYPT;
```

Note: Requirements of a mounted database or an offline data file apply to the DECRYPT command as well.

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 Data Guard physical standby database to facilitate the encryption process minimizes the impact to the availability of the application while achieving the goal of encrypted data at rest. The new offline encrypt feature is available with a patch in Oracle Database releases 12c and 11.2.0.4 to greatly simplify TDE conversion to a DDL

avoiding the reloading of data. Coupling this feature with the standby-first process described in this paper helps achieve the highest availability while converting to a secure data-at-rest solution.

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.

- Transient logical process: This process, normally used for upgrades with minimal down time, can also be used for maintenance such as this. The process converts a physical standby database to a logical standby, at which point a new encrypted tablespace can be created and loaded with the desired data through Data Pump export/import. Refer to Database Rolling Upgrades Using Data Guard or Oracle 12*c* DBMS_Rolling for more information.
- 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 tradeoff is that operational investment can vary depending on targeted objects.
- Use ALTER TABLE MOVE: Create a like sized encrypted tablespace and move each segment to the encrypted tablespace.
- Take an outage: This process was written to minimize the down time to the application and performance impact during migration; however, if availability is not a concern there are two options.
    - While the database is open, take each application/user tablespace offline and execute the ALTER DATABASE DATAFILE command for each underlying data file. This can be done in a rolling fashion or all at once.
    - Take a complete outage by bringing the primary database to a mounted state and running the ALTER DATABASE DATAFILE commands for each application/user tablespace data file.

    For either case above, if there are standby databases, they can be converted in the same manner at the same time provided recovery has been stopped.

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