Converting to Transparent Data Encryption with Oracle Data Guard using Fast Offline Conversion
Oracle Database 12.1 and Oracle Database 11.2

ORACLE WHITE PAPER | AUGUST 2017
Table of Contents

Introduction 1
  Prerequisites 1
  High Level Steps 1
Transparent Data Encryption Overview 2
  TDE Offline Data File Encryption Restrictions 2
Conversion Overview 2
  Prerequisites 2
  Conversion Example 3
  Enabling Transparent Data Encryption for Oracle 11.2 3
  Enabling Transparent Data Encryption for Oracle 12.1 5
  Convert Data Files 7
Hardware Keystore 10
Logical Standby Considerations 10
Decrypt 10
Conclusion 10
Appendix A – Alternative Methods to Convert to TDE 11
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 12.1.0.2 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:

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) 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 all nodes in the configuration (Oracle RAC primary nodes and all standby nodes).

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.


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 11g Release 1 (11.1) and higher.

Refer to the Oracle Database Advanced Security Administrator’s Guide for full details regarding TDE encryption.

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, UNDO, and TEMP tablespaces cannot be encrypted with TDE.

Conversion Overview

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.

1 https://docs.oracle.com/database/121/ASOAG/asotrans.htm#ASOAG10117
» COMPATIBLE is set to 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 switchover. 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
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 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 Oracle RAC nodes of a primary and standby database.

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 to pick up the wallet configuration which is being deferred at the initial primary until the switchover.*

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. Set the encryption wallet location.

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

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

3 | CONVERTING TO TRANSPARENT DATA ENCRYPTION USING DATA GUARD PHYSICAL STANDBY
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 by the new session.

4. Set the Master Encryption Key.

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

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

5. Open the wallet.

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

   System altered.

6. 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

7. 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>/*
8. Restart the standby database to mounted mode.

```sql
SQL> shutdown immediate
SQL> startup mount
```

OR

Ensure ORACLE_SID is set properly and:

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

9. Ensure the wallet is open on all nodes.

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

<table>
<thead>
<tr>
<th>INST_ID</th>
<th>WRL_TYPE</th>
<th>WRL_PARAMETER</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------</td>
<td>--------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>file</td>
<td>/u01/app/oracle/admin/TDE/$ORACLE_SID</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

Enabling Transparent Data Encryption for Oracle 12.1

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 all other nodes of a primary and standby database.

1. Create encryption keystore.

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

```sql
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 keystore. 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.

```bash
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 by the new session.
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 is contained in double quotation marks (" ").

5. Open the keystore.

```
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 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
'/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. Restart the standby database to mounted mode.

```
SQL> shutdown immediate
SQL> startup mount
```

OR

Ensure ORACLE_SID is set properly and:

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

10. Ensure the keystore is open on all nodes.

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

<table>
<thead>
<tr>
<th>INST_ID</th>
<th>WRL_TYPE</th>
<th>WRL_PARAMETER</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>----------</td>
<td>---------------</td>
<td>--------</td>
</tr>
</tbody>
</table>
Convert Data Files
Only user tablespace datafiles can be converted in Oracle Database releases 12.1 and 11.2. SYSTEM, SYSAUX, TEMP and UNDO tablespaces cannot be encrypted with TDE.

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
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
SQL> select * from gv$encryption_wallet;

INST_ID WRL_TYPE WRL_PARAMETER STATUS
------- -------- -------------------------------------- ------
1 file   /u01/app/oracle/admin/TDE/tdestby1 OPEN
```

2. Stop recovery.

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

Verify with the following statement:

```sql
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.

```sql
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.name not in
```
('SYSTEM','SYSAUX') and ts.name not in (select value from gv$parameter where name='undo_tablespace'));

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

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;

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 = +DATA1/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)

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;

Hardware Keystore

Hardware keystores are supported for Oracle Database 12c. 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 with the new command. Only datafiles encrypted with this offline encryption conversion command described in the paper can be decrypted with the following:

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 12.1.0.2 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 Oracle 11g Database Rolling Upgrades Made Easy or Oracle 12c 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 trade off 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 datafile. 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 datafile.

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.