Oracle Maximum
Availability Architecture

# Oracle Data Guardで高速オフライン変換を使用した透過的データ暗号化への変換

## Oracle Database 12cおよびOracle Database 11.2

Oracle ホワイト・ペーパー | 2018 年 8 月

ORACLE

# 目次


概要 ..... 2

前提条件 ..... 2

手順の概要 ..... 3

透過的データ暗号化の概要 ..... 3

TDE によるオフラインでのデータファイル暗号化の制限事項 ..... 4

変換の概要 ..... 4

前提条件 ..... 4

変換例 ..... 5

Oracle 11.2 での透過的データ暗号化の有効化 ..... 5

Oracle 12.1 と 12.2 での透過的データ暗号化の有効化 ..... 7

データファイルの変換 ..... 10

非対称構成 ..... 14

ハードウェア・キーストア ..... 15

ロジカル・スタンバイに関する考慮事項 ..... 15

復号化 ..... 15

結論 ..... 16

付録 A – TDE への別の変換方法 ..... 17

# 概要

24 時間 365 日の可用性が必要となるアプリケーションが増える中で、計画外停止と計画停止時間の両方が大きな懸案事項になっています。また、Oracle Cloud では、Oracle Advanced Security の透過的データ暗号化（TDE）を使用して本番データベースを暗号化する必要があります。クラウドに移行する場合でもオンプレミス・データベースのセキュリティを強化する場合でも、暗号化されていないデータベースの暗号化に必要な停止時間を最小化するために、Oracle Database 12c と Oracle Database 11.2.0.4 ではデータファイルを TDE にオフラインでインプレース変換できるようになりました。この機能を Data Guard のフィジカル・スタンバイと組み合わせると、本番環境のワークロードを暗号化されていないプライマリ・データベース上で影響を受けずに実行しながら、スタンバイ・データベースを TDE に変換できます。データベースのサイズに関係なく、総停止時間を短時間の一時停止に抑えながら、プロセスが完了すると、アプリケーション接続が新たに暗号化された本番環境のコピーに切り替わります（Data Guard スイッチオーバー）。この新機能は、12.1.0.2 と 11.2.0.4 用のパッチで提供されており、Oracle Support からこのパッチを入手する必要があります。このパッチの入手方法の説明は、My Oracle Support Note 2148746.1 を参照してください。

このパッチをインストールすると、DDL コマンドを使用して Data Guard のスタンバイ・データベースでデータファイルを TDE にオフラインでインプレース変換でき、長い時間や手間がかかる複雑なデータ再ロードを行う必要がありません。このパッチは Data Guard 構成のすべての Oracle ホームに適用する必要があります。このプロセスは、TDE への変換を最小限の停止時間と複雑さで実現するために推奨される Oracle Maximum Availability Architecture のベスト・プラクティスです。これは、「付録 A - TDE への別の変換方法」で説明している、従来の TDE への変換方法に取って代わる方法です。

既存のフィジカル・スタンバイ・データベースを使用する場合も、TDE への変換を容易にするためだけに新しいフィジカル・スタンバイ・データベースを展開して使用する場合も、変換プロセスは以下のとおりです。

*注：Oracle Database 12.2 以降、オンライン暗号化をプライマリ・データベースに対して実行できるようになっています。実行後、プライマリ・データはスタンバイを暗号化します。詳しくは、12.2 のドキュメントの「Encryption Conversions for Existing Online Tablespaces」を参照してください。*

## 前提条件

» このプロセスで必要なプライマリ・データベースとスタンバイ・データベースのデータベース・ホームへのパッチに関する最新情報は、MOS 2148746.1 を参照してください。

» バックアップを作成するか、有効なバックアップがあることを確認します。

» TDE の影響と制限を把握し、ウォレットと鍵を保持するプロセスを作成します。詳しくは、『Oracle Database Advanced Security 管理者ガイド』（11.2|12.1|12.2）と MOS 1228046.1 を参照してください。

### 手順の概要

1. Data Guard の構成が正しく、ギャップがないことを確認します。
2. 暗号化ウォレットを作成し、マスター鍵を設定します。
3. ウォレット・ファイルをスタンバイ・データベース環境にコピーします。
4. スタンバイ・データベースをマウント状態にし、リカバリを停止します。
5. スタンバイ側：データファイルをインプレースでパラレルに暗号化します。
6. スタンバイ側：REDO Apply を再起動して更新します。
7. Data Guard スイッチオーバーを実行して、暗号化されたスタンバイを新しいプライマリに、暗号化されていないプライマリを新しいスタンバイにします。
8. 新しいスタンバイ側：新しいスタンバイ・データベースをマウント状態にし、リカバリを停止します。
9. 新しいスタンバイ側：データファイルをインプレースでパラレルに暗号化します。
10. 新しいスタンバイ側：REDO Apply を再起動して更新します。
11. 必要に応じて、Data Guard スイッチオーバーを実行して元の構成を復元します。

この Oracle Maximum Availability Architecture（Oracle MAA）ベスト・プラクティスのホワイト・ペーパーは、暗号化されていない Oracle Database を最小限の停止時間で TDE に変換したいと考えているデータベース管理者を対象としています。このホワイト・ペーパーでは、Data Guard と TDE を技術的に理解していることを前提としています。

*このプロセスをテスト環境で実行、検証してから、本番環境で実行してください。*

## 透過的データ暗号化の概要

透過的データ暗号化（TDE）によって、Oracle データベース内に格納されているデータを暗号化できます。"格納されている"とは、データが保存されているオペレーティング・システム・レベルとストレージ・レベルでデータが暗号化されることを示します。TDE では、通常のデータベース認証ルールと認可ルールに従って、バッファ・キャッシュにヒットしたデータが透過的に復号化されます。

TDE 暗号化には次の 2 つの形式があります。TDE 列暗号化では特定の列データが暗号化され、TDE 表領域暗号化では TDE で暗号化された表領域内の全データが暗号化されます。表領域暗号化では、バルク暗号化の利用でパフォーマンスが向上します。また、管理者はどの列を暗号化すべきか判断するために各列を分析する必要がありません。さらに、表領域暗号化では、制限が列暗号化より少なくなります。このホワイト・ペーパーでは、TDE 表領域暗号化への変換方法を説明します。TDE 表領域暗号化は、Oracle Database 11g Release 1（11.1）以降で利用可能です。

TDE 暗号化の詳細については、『*Oracle Database Advanced Security 管理者ガイド*』と MOS 2359020.1 を参照してください [1]。

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

### TDEによるオフラインでのデータファイル暗号化の制限事項

TDE 表領域暗号化の制限が少ないのは、暗号化と復号化が読取り/書込み操作中に実行されるためで、SQL レイヤーで実行される列暗号化とはこの点が異なります。TDE 表領域暗号化の制限は次のとおりです。

» TDE 表領域暗号化を使用して外部ラージ・オブジェクト（BFILE）を暗号化することはできません。これらのファイルはデータベース外部に常駐しているためです。

» TDE で暗号化された表領域でインポート操作とエクスポート操作を実行するには、Oracle Data Pump を使用する必要があります。

» このホワイト・ペーパーで説明しているオフライン暗号化プロセスでは、V$DATABASE_KEY_INFO に記載された鍵識別子と AES128 暗号化アルゴリズムが使用されます。

» このプロセスは、アプリケーション表領域のデータファイルにのみ適用されます。Oracle Database バージョン 12.2.0.1 より前のバージョンでは、SYSTEM、SYSAUX、TEMP の各表領域は TDE では暗号化できません。12.2 では SYSTEM と SYSAUX を暗号化できますが、推奨されません。

» これらのリリースでは、UNDO 表領域をオフラインのまま暗号化しないことを推奨します。暗号化すると、キーストアを閉じることができなくなり、データベースが機能しなくなります。また、データベースがオフラインのときに UNDO 表領域を暗号化する必要はありません。暗号化された表領域に関連付けられた UNDO レコードはすべて、UNDO 表領域で自動的に暗号化されるためです。

» 暗号化された表領域の機密データから生成される UNDO メタデータと TEMP メタデータは、すでに自動的に暗号化されています。したがって、UNDO と TEMP の暗号化は任意の操作になります。

## 変換の概要

使用されるキーストアとストレージのタイプなど、TDE 表領域の暗号化には多くの構成オプションがあります。このドキュメントでは、もっとも一般的なオプションについて説明します。詳しくは、使用しているバージョンの『Oracle Database Advance Security ガイド』を参照してください。11.2 | 12.1 | 12.2

### 前提条件

このプロセスを正しく実行するには、以下の前提条件を満たしている必要があります。

» フィジカル・スタンバイ・データベースが存在する。

» データベースの現在のバックアップをデータファイルの変換前に実行している。

» MOS 2148746.1 に記載されているパッチがプライマリ RDBMS インストールとスタンバイ RDBMS インストールの両方に適用されている。

» COMPATIBLE がそれぞれ 12.2.0.1、12.1.0.2、または 11.2.0.4 に設定されている。

» Oracle MAA ベスト・プラクティスでは、プライマリ・データベースの強制ロギングを有効にすることを求めています。これはレプリケーションのために必要で、ロール移行中のリカバリ不能なオブジェクトを防ぎます。リカバリ不能なブロックがないことを確認するには、プライマリ・データベースで次の問合せを実行します。行は返されないはずです。NOLOGGING 処理と操作に関する詳細は、MOS 290161.1 を参照してください。

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

*注：UNRECOVERABLE_CHANGE#の値が 0 より大きいデータファイルがある場合は、プライマリの値とスタンバイの値を比較してください。値が同じ場合、データファイルはリカバリ不能な変更後にコピーされるので、対応は不要です。*

» プライマリ・データベースの場合は、ログ・アーカイブ先（LAD）を設定して各データベースが REDO を転送するようにします。ブローカが MAA ベスト・プラクティスどおりに構成されて使用されている場合は、LAD はロール移行後、自動的に設定されます。

### 変換例

TDE では、ウォレットとキーストアを利用してマスター暗号鍵が保存されます。デフォルトのデータベース・ウォレットを使用できますが、sqlnet.ora に ENCRYPTION_WALLET_LOCATION パラメータを指定して TDE の個別のウォレットを使用することをオラクルでは推奨しています。また、自動ログイン・ウォレットを使用すると、管理者はデータベースを起動するたびに手動でウォレットを開く必要がありません。

ウォレットはプライマリ・データベース上で作成され、スタンバイ環境に手動でコピーする必要があります。Oracle RAC 構成における MAA ベスト・プラクティスは、ASM ディスク・グループなど、共有された場所でウォレットを作成することです。共有ストレージがない場合、ウォレットをプライマリ・クラスタとスタンバイ・クラスタの全ノードにコピーする必要があります。

データベースですでに暗号化ウォレットかキーストアを使用している場合は、同じものを再利用しても構いませんが、オラクルでは新しいマスター鍵を設定することを推奨しています。

## Oracle 11.2での透過的データ暗号化の有効化

*注：このプロセスが終了するまで、新しいウォレットをプライマリ・データベースでの暗号化に使用しないでください。スイッチオーバーまで初期プライマリに保留されているウォレットの場所を sqlnet.ora から選択するには、全データベース・プロセスでデータベースを再起動する必要があります。*

1. 暗号化ウォレットの場所を設定します。

*注：暗号化ウォレットの ASM ストレージは、RDBMS バージョン 12.1.0.1 以前はサポートされていません。*

プライマリ・データベースとスタンバイ・データベースのすべてのノードで、sqlnet.ora にウォレットの場所を設定します。

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

*注：ディレクトリ・パス内で$ORACLE_UNQNAME のような環境変数を使うと、共有システムのすべてのデータベースが確実に自身のウォレットを使用します。この環境変数を使用する場合、変数を CRS（該当する場合）と OS ユーザーの環境で設定する必要があります。*

2. 対応するディレクトリを適切な ORACLE_UNQNAME ですべてのノードまたは共有ストレージ上で作成します。

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

3. 環境変数を CRS（該当する場合）および OS セッションで設定します。

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

$ export ORACLE_UNQNAME=<ORACLE_UNQNAME>
```

*注：ORACLE_UNQNAME 変数をログイン・スクリプトまたは環境スクリプト内に設定します。データベースまたはインスタンスを起動できるセッションでウォレットを見つけるには、これらのセッションでこの変数を設定する必要があります。*

4. 新しい SQL*Plus セッションを起動します。これにより、sqlnet.ora が変更され、環境変数が新しいセッションに反映されます。

5. マスター暗号化鍵を設定します。

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

この手順によって、REDO APPLY がスタンバイで停止されることがあります。手順 8 でウォレットがスタンバイにコピーされた後、再起動できます。

*注：パスワードを二重引用符（" "）で囲んでください。*

6. ウォレットを開きます。

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

System altered.
```

7. 自動ログイン・ウォレットを作成します。

自動ログイン・ウォレットを作成すると、データベースを起動するときにウォレットを手動で開く必要がなくなります。

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

8. キーストア・ディレクトリで生成されたファイルをプライマリとスタンバイのすべてのノードにコピーします。

ファイルを各ノードにコピーします。

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

9. スタンバイ・データベースをマウント・モードで再起動します。

ORACLE_UNQNAME が正しく設定されていることを確認して、次を実行します。

```
SQL> shutdown immediate
SQL> startup mount OR
$ srvctl stop database -d <standby DB>
$ srvctl start database -d <standby DB> -o mount
```

10. ウォレットがすべてのノードで開かれていることを確認します。

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

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

## Oracle 12.1と12.2での透過的データ暗号化の有効化

TDE では、キーストアを利用してマスター暗号鍵が保存されます。sqlnet.ora に ENCRYPTION_WALLET_LOCATION パラメータを指定して TDE 固有のウォレットを使用することをオラクルでは推奨しています。また、自動ログイン・キーストアを使用すると、管理者はデータベースを起動するたびに手動でキーストアを開く必要がありません。

キーストアは、1 つのプライマリ・インスタンスで作成され、スタンバイ・データベース環境に手動でコピーする必要があります。

1. 暗号化キーストアを作成します。

プライマリ・データベースとスタンバイ・データベースのすべてのノードで、sqlnet.ora にキーストアの場所を設定します。

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

*注：ディレクトリ・パス内で$ORACLE_UNQNAME のような環境変数を使うと、共有システムのすべてのデータベースが確実に自身のウォレットを使用します。この環境変数を使用する場合、変数を CRS（該当する場合）と OS ユーザーの環境で設定する必要があります。*

2. 適切な ORACLE_UNQNAME を使用し、対応するディレクトリをすべてのノード上に作成します。

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

3. 環境変数を CRS（該当する場合）および OS セッションで設定します。

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

*注：ORACLE_UNQNAME 変数をログイン・スクリプトまたは環境スクリプト内に設定します。データベースまたはインスタンスを起動できるセッションでウォレットを見つけるには、これらのセッションでこの変数を設定する必要があります。*

4. 新しい SQL*Plus セッションを起動します。これにより、sqlnet.ora が変更され、環境変数が新しいセッションに反映されます。

5. パスワードベースのキーストアを作成します。

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

*注：パスワードを二重引用符（" "）で囲んでください。*

6. キーストアを開きます。

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

7. 暗号化鍵を設定します。

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

*注：データベースが非 CDB の場合は、CONTAINER 句を省略します。ROOT コンテナ（ただし、PDB$SEED は除く）を含む各 PDB には異なる暗号鍵があり、CONTAINER=ALL と設定すると、1 つのコマンドを使って、ROOT を含む各 PDB のマスター鍵が生成されます。PDB のサブセットのみで TDE が必要な場合は、最初に ROOT マスター鍵を設定してから、各 PDB を個別に設定します。*

8. 自動ログイン・キーストアを作成します。

自動ログイン・キーストアを作成すると、データベースを起動するときにキーストアを手動で開く必要がなくなります。

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

8. キーストア・ディレクトリで生成されたファイルを各スタンバイ環境にコピーします。

ファイルを各ノードにコピーします。

```
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. スタンバイ・データベースをマウント・モードで再起動します。

ORACLE_UNQNAME が正しく設定されていることを確認して、次を実行します。

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

または

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

10. キーストアがすべてのノードで開かれていることを確認します。

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

INST_ID WRL_TYPE WRL_PARAMETER                              STATUS
------- -------- ------------------------------------------ ------
      1 file      +DATA/<ORACLE_UNQNAME>/TDE                OPEN
```

## データファイルの変換

Oracle Database Release 12.1 と 11.2 で変換できるのは、ユーザー表領域のデータファイルのみです。リリース 12.2 の SYSTEM、SYSAUX、UNDO 表領域も暗号化することができます。

*バージョン 12.2 では、TEMP 表領域の暗号化は可能ですが、変換はできません。TEMP 表領域を暗号化するには、暗号化した TEMP 表領域を Oracle Database 12c Release 2（12.2）で作成し、デフォルトの一時表領域にしてから、元の TEMP 表領域を破棄します。ロールが移行されるまで、またはプライマリ・データベースが再起動されるまで、ウォレットを使ってプライマリで暗号化を実行しないでください。*

*暗号化された表領域の機密データから生成される UNDO メタデータと TEMP メタデータは、すでに自動的に暗号化されています。したがって、UNDO 表領域と TEMP 表領域の暗号化は任意の操作になります。*

変換はデータファイルごとに実行されます。所定の表領域をオンラインにしたりデータベースをオープンしたりするには、その表領域のデータファイルをすべて暗号化する必要があります。

1. スタンバイ・データベースがマウントされており、キーストアが開いていることを確認します。

```
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. リカバリを停止します。

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

次の文で検証します。

```
SQL> select * from gv$managed_standby where process='MRP0';
no rows selected
```

*REDO Apply が停止しても、マウント中はプライマリ・データベースからスタンバイ・データベースに REDO が引き続き転送されるため、プライマリ・データベースに障害や災害が発生した場合にリカバリ・ポイント目標（RPO）が最小限に抑えられます。*

3. データファイルを暗号化するコマンドを生成します。

*Oracle Database Release 12.1 と 11.2 で変換できるのは、アプリケーション表領域のデータファイルのみです。*
*SYSTEM、SYSAUX、UNDO、TEMP の各表領域は TDE では暗号化できません。*
*12.2 の場合、UNDO 表領域と SYSTEM 表領域の暗号化は推奨されません。*

```
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*
*この問合せではすべてのユーザー表領域が暗号化されることを想定しています。サブセットが必要な場合は、WHERE 句を適時変更します。*

*コンテナ・データベースの場合、代わりに以下の問合せを使用します（この問合せから結果を取得するには、データベースを読取り専用モードで開く必要があるかもしれません。続行する前に、データベースをマウント・モードで停止および起動してください）。*

```
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. データファイルをパラレルで変換します。

各データファイルを個別のセッションで同時に暗号化できます。この例では、4 つのデータファイルを 4 つの異なるウィンドウで同時に変換しています。

*注：このプロセスは CPU リソースと I/O リソースを消費します。共有環境では、リソースを監視して複数の暗号化プロセスが他のデータベースのリソース要求と衝突しないように注意する必要があります。*

```
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.
```

*注：オフライン暗号化操作は冪等性（べきとうせい）があります。データファイルの変換が何らかの理由で中断されても、alter database コマンドを実行するだけです。*

5. オプション：DBVERIFY を使用して、使用済みブロックが暗号化されていることを確認します。

```
$ 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)
```

*DBVerify 出力の中で、'Data'または'Index'とマークされているページは暗号化されていません。'Other'というラベルが付いているページは Data ページまたは Index ページではなく、暗号化する必要はありません。'Other'の値はすべて無視してもかまいません。Data または Index であるかに関わらず暗号化されたページはすべて、Encrypted としてまとめてグループ化されます。*

6. リカバリを再開します。

データファイルをすべて暗号化したら、リカバリを再開します。暗号化されたデータファイルに適用された新しい REDO がすべて暗号化されます。スタンバイがギャップなく最新になったら、手順 7 に進みます。

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

7. スイッチオーバーを実行します。

スイッチオーバーを実行し、スタンバイを新しいプライマリにします。スタンバイに適用された新しいデータがすべて暗号化されます。

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

このプロセスでこれが唯一停止の可能性がある手順です（オプションのスイッチバックでも停止が発生することがあります）。スイッチオーバーの最適化とアプリケーション停止時間の短縮については、ホワイト・ペーパー『ロール移行のベスト・プラクティス』を参照してください。Oracle MAA のテストでは、スイッチオーバーは 1 分未満で完了しています。

*注：このスイッチオーバーはスタンバイのデータファイルの暗号化後直ちに実行する必要はありませんが、スタンバイが暗号化されているがプライマリが暗号化されていないという状態を無期限に放置することは推奨されません。*

8. sqlnet.ora への変更を反映するために新しいスタンバイ（元のプライマリ）を起動し、新しいスタンバイ・データベースおよび構成内の他のすべてのスタンバイ・データベースで手順 1～5 を繰り返します。

*注：データファイル名はプライマリとスタンバイで異なる可能性があります。*

9. 前回同様、変換手順が完了してスタンバイがプライマリと同じ状態になったら、必要に応じて元の構成に切り替えます。

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

## 非対称構成

Oracle Data Guard の 1 つのデータベースが暗号化され、もう一方が暗号化されていない場合、リカバリされたブロックの暗号化の状態は、RDBMS バージョンに応じて異なります。

Oracle Database 12.2 以上では、データベースの暗号化プロパティはリカバリ後も維持されます。スタンバイでデータファイルが暗号化されている場合、そのデータファイルに対するリカバリされたブロックは、REDO（ソース・プライマリ）が暗号化されているかどうかに関係なく暗号化されます。

Oracle Database 12.1 と 11.2 の場合、REDO（ソース・プライマリ）の暗号化の状態が維持されるので、スタンバイでのデータファイルの暗号化の状態に関係なく、プライマリ上の暗号化されていないデータファイルの REDO はスタンバイ上で暗号化されないまま保存されます。

したがって Oracle Database 11.2 と 12.1 の場合、スタンバイでの全ブロックを確実に暗号化するには、暗号化されていないプライマリの REDO が適用されるのであれば、追加の暗号化コマンドを実行しなければならない可能性があります。暗号化するブロックの数は少ないはずなので、この 2 回目の実行にかかる時間は通常はるかに短くなります。

次に例を示します。

1. 説明したプロセスを使って、スタンバイのデータファイルを暗号化します。
2. スタンバイを現在の状態にリカバリします。
3. 準備ができたらスイッチオーバーを実行します。
4. 元のプライマリ・データファイルを暗号化します。
5. スタンバイ（元のプライマリ）を現在の状態にリカバリします。
6. スイッチバックを実行します。
7. スタンバイ・データファイルに対して DBVerify を実行し、暗号化されていないデータ・ブロックが見つかったら、'encrypt'句を'force encrypt'句に置き換えて 2 回目の暗号化をスタンバイ・データファイルに対して実行します。

次に例を示します。

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

## ハードウェア・キーストア

ハードウェア・キーストアは、Oracle Database 12c でサポートされています。ハードウェア・キーストアに関する詳細は、こちらのドキュメントを参照してください。

## ロジカル・スタンバイに関する考慮事項

このホワイト・ペーパーで説明したとおり、フィジカル・スタンバイ（Data Guard REDO Apply）は複雑さが軽減されるため、TDE への変換方法として推奨されています。ただし、すでにロジカル・スタンバイ・データベース（Data Guard SQL Apply）を使用している場合も、この同じプロセスを使用できます。ロジカル・スタンバイはフィジカル・スタンバイと同じように扱い、暗号化鍵をロジカル・スタンバイ・データベースに格納する必要があります。さらに、プライマリのデータを含むデータファイルは暗号化する必要があります。該当するデータファイルに対して、「データファイルの変換」セクションの手順 1～6 を繰り返します。

ロジカル・スタンバイ・データベースと TDE の詳細は、関連ドキュメント（11.2|12.1）を参照してください。

## 復号化

データファイルを何らかの理由で復号化する必要のある場合には、DECRYPT 句を使用できます。復号化できるのは、このホワイト・ペーパーに記載されている、オフラインの暗号化変換メソッドで暗号化されたデータファイルだけで、メソッドは次のとおりです。

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

*注：マウントされたデータベースまたはオフラインのデータファイルの要件は、DECRYPT コマンドにも適用されます。*

## 結論

Oracle Advanced Security の透過的データ暗号化への変換によって、Oracle Database に格納されているデータを保護できます。データを暗号化するプロセスでは、多くのアプリケーションで 24 時間 365 日の要件が課題となります。暗号化プロセスを容易にするために Data Guard のフィジカル・スタンバイ・データベースを使用すると、アプリケーションの可用性に及ぼす影響を最小限に抑えながら、格納データを暗号化するという目的を達成できます。この新しいオフライン暗号化機能は Oracle Database Release 12c と 11.2.0.4 のパッチで利用でき、DDL への TDE 変換を大幅に簡素化して、データの再ロードを避けることができます。 このホワイト・ペーパーに記載されている Standby-First プロセスとこの機能を組み合わせると、最高レベルの可用性を達成しながら、セキュアな格納データ・ソリューションに変換できます。

# 付録A – TDEへの別の変換方法

TDE 暗号化への変換では、このホワイト・ペーパーで使用した方法以外に複数の方法を使用できます。データベースのサイズ、利用可能なストレージ、許容できる停止時間に応じて以下のアプローチを検討できますが、手順の社内開発が必要です。

» 一時ロジカル・プロセス：このプロセスは、通常はアップグレードの停止時間を最小限にするために使用されますが、このようなメンテナンスにも使用できます。このプロセスでフィジカル・スタンバイ・データベースをロジカル・スタンバイに変換し、その時点で、暗号化された表領域を新たに作成し、Data Pump のエクスポート/インポートを使って必要なデータをロードできます。詳しくは、「Data Guard を使用したデータベース・ローリング・アップグレード」または Oracle 12c の DBMS_Rolling を参照してください。

» アクティブなプライマリ・データベースでの DBMS_REDEFINITION の使用：これは、スタンバイ・データベースを持たないデータベースの場合か、管理者が DBMS_REDEFINITION プロセスに慣れている場合の選択肢です。おもなメリットは、停止時間がゼロか非常に短いことです。デメリットは、運用上の投資が対象オブジェクトによって大きく変わる可能性があることです。

» ALTER TABLE MOVE の使用：同じサイズの暗号化表領域を作成し、各セグメントを暗号化表領域に移動します。

» 停止措置：このプロセスは、移行中にアプリケーションの停止時間とパフォーマンスへの影響を最小限にするために作成されています。ただし、可用性が重要でない場合は、選択肢が 2 つあります。

  » データベースがオープンされた状態で、各アプリケーション/ユーザー表領域をオフラインにし、基盤となる各データファイルに対して ALTER DATABASE DATAFILE コマンドを実行します。これはローリング方式で実行することも一度にすべて実行することも可能です。

  » プライマリ・データベースをマウント状態にして、各アプリケーション/ユーザー表領域のデータファイルに対して ALTER DATABASE DATAFILE コマンドを実行し、完全に停止します。

  上記いずれの場合も、スタンバイ・データベースがある場合は、同じ方法でスタンバイ・データベースを同時に変換できます。ただし、リカバリが停止していることが条件です。

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

**海外からのお問い合わせ窓口**
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

CONNECT WITH US

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

**Hardware and Software, Engineered to Work Together**