



Transparent Data Encryption

リリース 15.2.0

EnterpriseDB Corporation

2023 年 02 月 19 日

目次

第 1 章	TDE で暗号化されるのは何ですか?	3
1.1	TDE はパフォーマンスにどのように影響しますか?	3
第 2 章	TDE はどのように機能しますか?	5
2.1	TDE はどのようにデータを暗号化しますか?	5
2.2	TDE でデータはディスクにどのように保存されますか?	5
2.2.1	Securing the data encryption key	6
2.2.2	Enabling TDE	10
2.2.3	Limitations	12
2.2.4	Commands affected by TDE	12
2.2.5	Troubleshooting with encrypted WAL files	13
2.2.6	Working with encrypted backup files	14
2.2.7	Upgrading a TDE system	14
2.2.8	Single-user mode	15
2.2.9	Testing a TDE configuration	15

透過的データ暗号化（TDE）は、高可用性を備えた EDB Postgres Advanced Server および EDB Postgres Extended Server のバージョン 15 でサポートされているオプション機能です。

データベースシステムに保存されているユーザーデータを暗号化します。この暗号化はユーザーに対して透過的です。ユーザーデータには、オブジェクトの名前などのシステムカタログデータだけでなく、テーブルやその他のオブジェクトに格納されている実際のデータが含まれます。

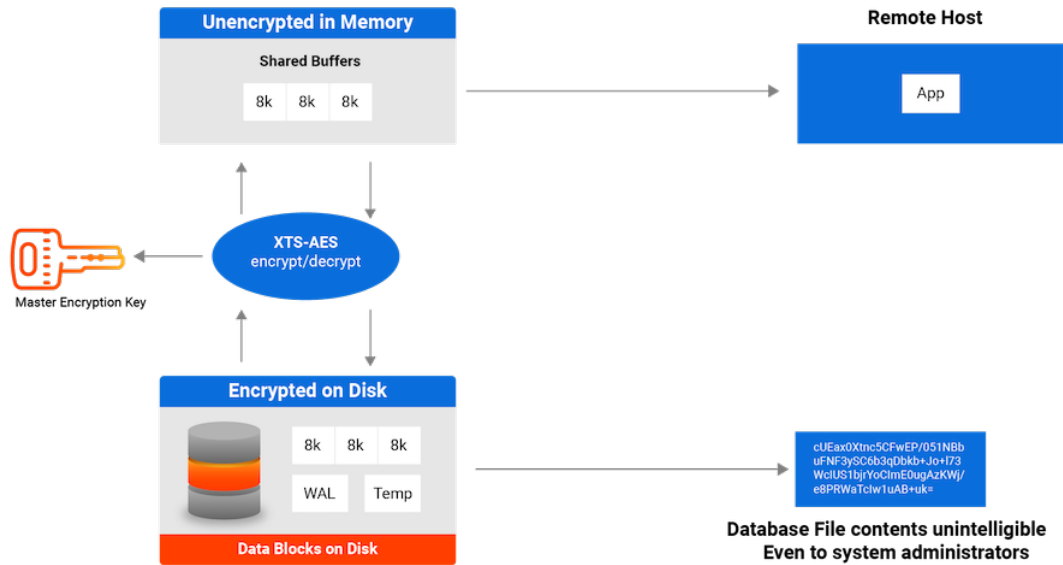


図 1: Architecture diagram

第1章 TDEで暗号化されるのは何ですか？

TDE は以下を暗号化します。

- TOAST テーブルとシステムカタログを含み、すべてのフォークを含むテーブル、シーケンス、インデックスの基礎となるファイル。これらのファイルは_データファイル_と呼ばれます。
- ログ先行書き込み（WAL）。
- クエリ処理やデータベースシステムの運用時に使用される各種一時ファイル。

以下は、TDE によって暗号化または偽装されません。

- トランザクションステータス（たとえば、pg_subtrans および pg_xact）などのユーザデータを含まないデータベースシステムの操作内部のメタデータ。
- データディレクトリ内のファイル名とファイルシステム構造。これは、データベースシステムの全体的なサイズ、データベースの数、テーブルの数、それらの相対的なサイズ、および最終アクセス時間などのファイルシステムのメタデータがすべて復号化されないことを意味します。
- 外部テーブルのデータ。
- サーバー診断ログ。
- 構成ファイル。

1.1 TDE はパフォーマンスにどのように影響しますか？

パフォーマンスは、AES の一般的なオーバーヘッドと一致しています。

第2章 TDE はどのように機能しますか？

TDE は、データベースサーバーおよびバックアップストレージ上のオペレーティングシステムファイルのデータの不正表示を防ぎます。データは、盗まれたり紛失したりすると、権限のないユーザーには理解できなくなります。

データの暗号化と復号化はデータベースによって管理されるため、アプリケーションの変更やクライアントドライバの更新は必要ありません。

EDB Postgres Advanced Server と EDB Postgres Extended Server は、データベース外部のキー管理へのフックを提供します。これらのフックにより、単純なパズフレーズの暗号化/復号化、またはエンタープライズキー管理ソリューションとの統合が可能になります。詳細については、[Securing the data encryption key](#) を参照してください。

2.1 TDE はどのようにデータを暗号化しますか？

TDE は、AES-128-XTS を使用してデータファイルを暗号化します。XTS「微調整」はデータベース OID、relfilenode、およびブロック番号を使用します。

WAL は AES-128-CTR を使用して暗号化されます。「カウンター」には WAL LSN が含まれます。

ブロックによってアクセスされる一時ファイルも AES-128-XTS を使用して暗号化されます。その他の一時ファイルは AES-128-CBC を使用して暗号化されます。

2.2 TDE でデータはディスクにどのように保存されますか？

この例では、tbfoo テーブルのデータは暗号化されています。pg_relation_filepath ファンクションは、tbfoo テーブルに対応するデータファイルを見つけます。

```
insert into tbfoo values (abc,123);
INSERT 0 1

select pg_relation_filepath(tbfoo);

pg_relation_filepath
-----
base/5/16416
```

文字を探してデータを grepp しても何も返されません。最後の 5 行を表示すると、暗号化されたデータが返されます。

```
$ hexdump -C 16416 | grep abc
$

$ hexdump -C 16416 | tail -5
00001fc0 c8 0f 1d c8 9a 63 3d dc 7d 4e 68 98 b8 f2 5e 0a |.....c=.}Nh...^.|
00001fd0 9a eb 20 1d 59 ad be 94 6e fd d5 6e ed 0a 72 8c |.. .Y...n...n..r.|
00001fe0 7b 14 7f de 5b 63 e3 84 ba 6c e7 b0 a3 86 aa b9 |{...[c...l.....|
00001ff0 fe 4f 07 50 06 b7 ef 6a cd f9 84 96 b2 4b 25 12 |.O.P...j.....K%.|
00002000
```

2.2.1 Securing the data encryption key

透過的なデータ暗号化のキー（データキー）は、通常 `initdb` によって生成され、`data` ディレクトリの下のファイル `pg_encryption/key.bin` に保存されます。このファイルには、実際には実行時にさまざまな目的で使用されるいくつかのキーが含まれています。ただし、データキーに関しては、単一のランダムバイトシーケンスが含まれています。

さらにアクションを行わないと、このファイルには安全ではないプレーンテキストのキーが含まれています。暗号化されたデータディレクトリにアクセスできる人は誰でも平文のキーにアクセスできますが、これは暗号化の目的に反します。したがって、このセットアップはテスト目的のみに適しています。

データキーを適切に保護するには、別のキーで暗号化して「ラップ」します。大まかに、これを調整するには2つのアプローチを使用できます。

- データキーをパスフレーズで保護します。ラッピングキーはパスフレーズから取得され、データキーの暗号化に使用されます。
- ラッピング鍵は、キーストアとも呼ばれるキー管理システムなど、他の場所に保存されています。この2番目のキーは、**キーラッピングキー**または**マスターキー**とも呼ばれます。

テストなどでキーのラッピングが必要ない場合は、`wrap` コマンドと `unwrap` コマンドを空の文字列に設定する必要があります。この設定は、さらに処理せずにファイルのキーを使用することを指定します。このアプローチは、`wrap` または `unwrap` コマンドをまったく設定しないこととは異なります。透過的なデータ暗号化が使用されているときに `wrap` または `unwrap` コマンドを設定しないと、影響を受けるユーティリティプログラムの実行時に致命的なエラーが発生します。

Postgres はこの構成をユーザーに任せるため、ローカルの要件に合わせてセットアップを調整し、既存のキー管理ソフトウェアなどと統合できます。データキー保護を構成するには、ラッピング（暗号化）とアンラッピング（復号化）を行う一対の外部コマンドを指定する必要があります。

パスフレーズの使用

openssl コマンドラインユーティリティを使用して、パスフレーズでデータキーを保護できます。以下は、この保護を設定する例です。

```
initdb -D datadir -y --key-wrap-command=openssl enc -e -aes128-wrap -pbkdf2 -out "%p"
↪--key-unwrap-command=openssl enc -d -aes128-wrap -pbkdf2 -in "%p"
```

この例では、ランダムに生成されたデータキー (initdb によって内部的に実行) を、AES-128-WRAP (AESKW) アルゴリズムを使用して暗号化することによりラップします。暗号化には、PBKDF2 キー導出関数とランダムに生成されたソルトを使用してパスフレーズから導出されたキーが使用されます。ターミナルはパスフレーズの入力を求めます。(これらのオプションの詳細については、openssl-enc のマニュアルページを参照してください。使用可能なオプションはバージョンによって異なります。) プレースホルダー%p は、ラップされたキーを保存するファイルの名前に置き換えられます。

unwrap コマンドは逆の操作を実行します。initdb にはアンラップ操作は必要ありません。ただし、起動時に使用する初期化されたクラスターの postgresql.conf ファイルに保存します。

注釈: サーバーが pg_ctl または systemd などのサービスマネージャーを介して起動された場合、端末でパスワードの入力を求めるキーのラップ解除コマンドが機能しません。これらの環境では、サーバーは端末から切断されます。サーバーの起動時に対話型のパスワードプロンプトが必要な場合は、systemd-ask-password を使用するなど、より複雑な構成が必要です。

key wrap コマンドは標準入力でプレーンテキストキーを受け取り、ラップされたキーを%p プレースホルダーで指定されたファイルシステムの場所に配置する必要があります。key unwrap コマンドは、%p プレースホルダーで指定されたファイルシステムの場所からラップされたキーを読み取り、アンラップされたキーを標準出力に書き込む必要があります。

pg_rewind や pg_upgrade などのユーティリティプログラムは、データディレクトリまたはバックアップなどのコピーを直接操作します。これらのプログラムは、状況に応じて、key unwrap コマンドについても通知する必要があります。これらにはそれぞれ、この目的のためのコマンドラインオプションがあります。

操作を簡素化するために、環境変数にキーのラップコマンドとラップ解除コマンドを設定することもできます。これらは、対応するコマンドラインオプションを指定しない場合、影響を受けるすべてのアプリケーションで受け入れられます。例：

```
PGDATAKEYWRAPCMD=openssl enc -e -aes128-wrap -pbkdf2 -out "%p"
PGDATAKEYUNWRAPCMD=openssl enc -d -aes128-wrap -pbkdf2 -in "%p"
export PGDATAKEYWRAPCMD PGDATAKEYUNWRAPCMD
```

キーストアの使用

外部のキー管理システムのキーストアを使用して、データ暗号化キーを管理できます。テスト済みでサポートされているキーストアは次のとおりです。

- Amazon AWS Key Management Service (KMS)
- Google Cloud - Cloud Key Management Service
- Microsoft Azure Key Vault
- タレス暗号トラストマネージャー

AWS Key Management Service の例

AWS Key Management Service でキーを作成します。

```
aws kms create-key
aws kms create-alias --alias-name alias/pg-tde-master-1 --target-key-id "..."
```

alias/pg-tde-master-1 キーを指定して `aws kms` コマンドを使用して、データ暗号化キーをラップおよびアンラップします。

```
PGDATAKEYWRAPCMD=aws kms encrypt --key-id alias/pg-tde-master-1 --plaintext fileb:///
→dev/stdin --output text --query CiphertextBlob | base64 -d > "%p"
PGDATAKEYUNWRAPCMD=aws kms decrypt --key-id alias/pg-tde-master-1 --ciphertext-blob
→fileb://"%p" --output text --query Plaintext | base64 -d
```

注釈: この例のように、パイプを使用したシェルコマンドには問題があります。パイプの終了ステータスは最後のコマンドの終了ステータスであるためです。最初のより興味深いコマンドの失敗は、適切に報告されません。Postgres は、`wrap` または `unwrap` コマンドが何も書いていないかどうかを認識することにより、これをいくらか処理します。ただし、これをより堅牢にすることをお勧めします。たとえば、一部のシェルで使用可能な ``pipefail`` オプション、または一部のオペレーティングシステムで使用可能な ``mispipe`` コマンドを使用します。より複雑なコマンドをインラインで定義するのではなく、外部シェルスクリプトまたは他のプログラムに配置します。

または、`crypt utility` を使用して、データ暗号化キーをラップおよびラップ解除できます。

```
PGDATAKEYWRAPCMD=crypt encrypt aws --out %p --region us-east-1 --kms alias/pg-tde-
→master-1
PGDATAKEYUNWRAPCMD=crypt decrypt aws --in %p --region us-east-1
```

Azure Key Vault の例

Azure Key Vault でキーを作成します。

```
az keyvault key create --vault-name pg-tde --name pg-tde-master-1
```

pg-tde-master-1 キーを指定して az keyvault コマンドを使用して、データ暗号化キーをラップおよびアンラップします。

```
PGDATAKEYWRAPCMD=crypt encrypt azure --vaultURL https://pg-tde.vault.azure.net --name pg-tde-master-1 --version fa2bf368449e432085318c5bf666754c --out %p
PGDATAKEYUNWRAPCMD=crypt decrypt azure --vaultURL https://pg-tde.vault.azure.net --name pg-tde-master-1 --version fa2bf368449e432085318c5bf666754c --in %p
```

この例では、*Securing the data encryption key* を使用します。一部の機能がないため、この目的で Azure CLI を直接使用することはできません。

Google Cloud KMS の例

Google Cloud KMS でキーを作成します。

```
gcloud kms keys create pg-tde-master-1 --location=global --keyring=pg-tde --purpose=encryption
```

pg-tde-master-1 キーを指定して az keyvault コマンドを使用して、データ暗号化キーをラップおよびアンラップします。

```
PGDATAKEYWRAPCMD=gcloud kms encrypt --plaintext-file=- --ciphertext-file=%p --location=global --keyring=pg-tde --key=pg-tde-master-1
PGDATAKEYUNWRAPCMD=gcloud kms decrypt --plaintext-file=- --ciphertext-file=%p --location=global --keyring=pg-tde --key=pg-tde-master-1
```

または、*crypt utility* を使用して、データ暗号化キーをラップおよびラップ解除できます。

```
PGDATAKEYWRAPCMD=crypt encrypt gcp --out=%p --location=global --keyring=pg-tde --key=pg-tde-master-1 --project your-project-123456
PGDATAKEYUNWRAPCMD=crypt decrypt gcp --in=%p --location=global --keyring=pg-tde --key=pg-tde-master-1 --project your-project-123456
```

2.2.2 Enabling TDE

initdb TDE オプションの使用

を使用してデータベースクラスターを初期化するとき、透過的なデータ暗号化を有効にします。

initdb TDE オプションの使用

暗号化を有効にするには、`initdb` コマンドまたはそのフォールバック環境変数で次のオプションを使用します。

`-y, --data-encryption`

透過的なデータ暗号化を使用して新しいデータベースクラスターを初期化します。詳細については、*Transparent Data Encryption* を参照してください。

`--copy-key-from=<file>`

指定された場所からデータ暗号化キーをコピーします。`pg_upgrade` のターゲットとして新しいクラスターを準備するときに、このオプションを使用して既存のクラスターからキーをコピーできます。`--key-wrap-command=<command>` 生成したデータ暗号鍵をラップ（暗号化）するコマンドを指定します。コマンドには、ラップされたキーを書き込むファイルを指定するプレースホルダー`%p`を含める必要があります。ラップされていないキーは、標準入力コマンドに提供されます。このオプションを指定しない場合、環境変数 `PGDATAKEYWRAPCMD` が使用されます。キーラッピングコマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、*Securing the data encryption key* を参照してください。

`--key-unwrap-command=<command>`

データ暗号化キーをアンラップ（復号化）するコマンドを指定します。コマンドには、ラップされたキーを読み取るファイルを指定するプレースホルダー`%p`を含める必要があります。このコマンドは、ラップされていないキーを標準出力に書き込む必要があります。このオプションを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、*[Securing the data encryption key]* を参照してください。 `--no-key-wrap`

キーラッピングを無効にします。代わりに、データ暗号化キーはデータディレクトリに平文で保存されます。（このオプションは、`wrap` コマンドと `unwrap` コマンドの両方を空の文字列に設定するためのショートカットです。）.. **Note::**

このオプションの使用は安全ではありません。テスト目的でのみ使用してください。データ暗号化を選択し、このオプションを指定しない場合、キーのラップおよびラップ解除コマンドを提供する必要があります。それ以外の場合、`initdb` はエラー終了します。

環境変数の使用

操作を簡素化するために、環境変数にキーのラップコマンドとラップ解除コマンドを設定できます。

例：

```
PGDATAKEYWRAPCMD=openssl enc -e -aes128-wrap -pbkdf2 -out "%p"
PGDATAKEYUNWRAPCMD=openssl enc -d -aes128-wrap -pbkdf2 -in "%p"
export PGDATAKEYWRAPCMD PGDATAKEYUNWRAPCMD
```

postgresql.conf にキーパラメーターを設定する

クラスターの TDE を有効にすると、`initdb` コマンドは `postgresql.conf` 構成ファイルの `data_encryption_key_unwrap_command` パラメーターを初期化します。`data_encryption_key_unwrap_command` で指定された文字列は、データ暗号化キーをアンラップ（復号化）します。

コマンドには、プレースホルダー`%p`を含める必要があります。これは、ラップを解除するキーを含むファイルの名前に置き換えられます。このコマンドは、ラップされていない（復号化された）キーを標準出力に出力する必要があります。

このパラメーターを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。

キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。

データ暗号化を使用している場合は、このパラメーターまたは環境変数 `fallback` を指定する必要があります。詳細については、[Securing the data encryption key](#) を参照してください。

このパラメーターは、サーバーの起動時にのみ設定できます。

このパラメーターは通常、`initdb` によって初期化されます。キーラップ方法を変更する場合にのみ変更してください。

構成ファイルの詳細については、[PostgreSQL File Locations documentation](#) を参照してください。

例

この例では、Linux プラットフォームで実行されている EDB Postgres Advanced Server 15 を使用します。`openssl` を使用して、生成されたデータ暗号化キーをラップおよびラップ解除するパスキーを定義します。

1. データ暗号化キー（ラップ）と復号化（アンラップ）環境変数を設定します。

```
export PGDATAKEYWRAPCMD=openssl enc -e -aes-128-cbc -pass pass:ok -out %p
export PGDATAKEYUNWRAPCMD=openssl enc -d -aes-128-cbc -pass pass:ok -in %p
` `` !!!note Windows を使用している場合、変数値を一重引用符で囲む必要はありません。
```

1. 暗号化を有効にした `initdb` を使用してクラスターを初期化します。このコマンドは、`postgresql.conf` ファイルに `data_encryption_key_unwrap_command` パラメーターを設定します。

(次のページに続く)

(前のページからの続き)

```
``shell
/usr/edb/as15/bin/initdb --data-encryption -D /var/lib/edb/as15/data
```

1. クラスターを起動します。

```
/usr/edb/as15/bin/pg_ctl -D /var/lib/edb/as15/data start
```

1. postgresql.conf で grep を実行して、data_encryption_key_unwrap_command の設定を確認します。

```
grep data_encryption_key_unwrap_command /var/lib/edb/as15/data/postgresql.conf
__OUTPUT__
data_encryption_key_unwrap_command = openssl enc -d -aes-128-cbc -pass pass:ok -in %p
```

2.2.3 Limitations

FILE_COPY

透過的なデータ暗号化が有効になっている場合、CREATE DATABASE の strategy パラメーターで FILE_COPY 戦略を使用することはできません。

詳細については、[PostgreSQL CREATE DATABASE documentation](#) を参照してください。

2.2.4 Commands affected by TDE

TDE が有効になっている場合、次のコマンドに TDE 固有のオプションがあるか、環境変数または構成ファイルの TDE 設定を読み取ります。

- pg_waldump
- pg_resetwal
- pg_verifybackup
- pg_rewind
- pg_upgrade
- *postgresql.conf* にキーパラメーターを設定する

2.2.5 Troubleshooting with encrypted WAL files

WAL ファイルを暗号化できます。暗号化された WAL フォールでトラブルシューティングを行う場合、WAL コマンドオプションを使用できます。

TDE で暗号化された WAL ファイルのダンプ

暗号化された WAL ファイルを操作するには、

`pg_waldump` はラップ解除キーを認識する必要があります。次のオプションを使用して

`unwrap` コマンドのキーを `pg_waldump` コマンドに渡すか、フォールバック環境変数に依存します。

`--data-encryption`

暗号化する WAL ファイルを検討し、処理する前に復号化します。WAL ファイルが透過的なデータ暗号化によって暗号化された場合、このオプションを指定する必要があります。`pg_waldump` は、WAL ファイルが暗号化されているかどうかを自動的に検出できません。

`--key-file-name=<file>`

指定された場所からデータ暗号化キーをロードします。

`--key-unwrap-command=<command>`

データ暗号化キーをアンラップ（復号化）するコマンドを指定します。コマンドには、ラップされたキーを読み取るファイルを指定するプレースホルダー`%p`を含める必要があります。このコマンドは、ラップされていないキーを標準出力に書き込む必要があります。このオプションを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、[Securing the data encryption key](#) を参照してください。

破損した TDE で暗号化された WAL ファイルのリセット

破損した暗号化された WAL ファイルをリセットするには、

`pg_resetwal` コマンドはアンラップキーを認識する必要があります。次のオプションを使用して

`unwrap` コマンドのキーを `pg_resetwal` コマンドに渡すか、フォールバック環境変数に依存します。

`--key-unwrap-command=<command>`

データ暗号化キーをアンラップ（復号化）するコマンドを指定します。コマンドには、ラップされたキーを読み取るファイルを指定するプレースホルダー`%p`を含める必要があります。このコマンドは、ラップされていないキーを標準出力に書き込む必要があります。このオプションを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、[Securing the data encryption key](#) を参照してください。

2.2.6 Working with encrypted backup files

TDE システムのバックアップを検証する

暗号化されたバックアップファイルを検証するには、

`pg_verifybackup` コマンドはラップ解除キーを認識する必要があります。次のオプションを使用して `unwrap` コマンドのキーを `pg_verifybackup`

コマンドに渡すか、フォールバック環境変数に依存します。

`--key-unwrap-command=<command>`

データ暗号化キーをアンラップ（復号化）するコマンドを指定します。コマンドには、ラップされたキーを読み取るファイルを指定するプレースホルダー `%p` を含める必要があります。このコマンドは、ラップされていないキーを標準出力に書き込む必要があります。このオプションを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、[Securing the data encryption key](#) を参照してください。

TDE システムでタイムラインを再同期する

暗号化されたクラスターをバックアップと再同期するには、

`pg_rewind` コマンドはラップ解除キーを認識する必要があります。次のオプションを使用して

`unwrap` コマンドのキーを `pg_rewind` コマンドに渡すか、フォールバック環境変数に依存します。

`--key-unwrap-command=<command>`

データ暗号化キーをアンラップ（復号化）するコマンドを指定します。コマンドには、ラップされたキーを読み取るファイルを指定するプレースホルダー `%p` を含める必要があります。このコマンドは、ラップされていないキーを標準出力に書き込む必要があります。このオプションを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。

キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、[Securing the data encryption key](#) を参照してください。

2.2.7 Upgrading a TDE system

`pg_upgrade` のこれらのオプションは、暗号化されたクラスターのアップグレードに役立ちます。

`--copy-by-block`

ファイル全体を一度にコピーするデフォルトではなく、ブロックごとにファイルを新しいクラスターにコピーします。このオプションはデフォルトモードと同じですが、やや遅くなります。ただし、暗号化設定が異なるクラスター間のアップグレードはサポートされています。暗号化設定が異なるクラスター間でアップグレードする場合、つまり、非暗号化から暗号化、暗号化から非暗号化、またはその両方を異なるキーで暗号化する場合は、このオプションを使用する必要があります。ファイルを新しいクラスターにコピーするときに、ファイルを復号化し、新しいクラスターのキーと設定で再暗号化します。より確実にするため

に、古いクラスターが暗号化されており、新しいクラスターが非暗号化として初期化された場合、このオプションは古いクラスターからデータを復号化し、暗号化せずに新しいクラスターにコピーします。古いクラスターが暗号化されておらず、新しいクラスターが暗号化されたものとして初期化された場合、このオプションは古いクラスターのデータを暗号化し、暗号化された新しいクラスターに配置します。`pg_upgrade` のターゲットとして新しいクラスターを準備するときに既存のクラスターからキーをコピーする方法については、

`initdb --copy-key-from=<file>` option の説明を参照してください。

`--key-unwrap-command=<command>`

データ暗号化キーをアンラップ（復号化）するコマンドを指定します。コマンドには、ラップされたキーを読み取るファイルを指定するプレースホルダー`%p`を含める必要があります。このコマンドは、ラップされていないキーを標準出力に書き込む必要があります。このオプションを指定しない場合、環境変数 `PGDATAKEYUNWRAPCMD` が使用されます。キーのラップ解除コマンドを適用しない場合は、空の文字列を使用します。データ暗号化を使用している場合は、このオプションまたは環境変数 `fallback` を指定する必要があります。詳細については、[Securing the data encryption key](#) を参照してください。

2.2.8 Single-user mode

TDE を有効にしてシングルユーザーモードで `postgresql.conf` にキーパラメーターを設定する を呼び出すと、`postgres` コマンドは次のいずれかを読み取ります。

- `PGDATAKEYUNWRAPCMD` 環境変数（設定されている場合）
- `postgresql.conf` ファイルの `data_encryption_key_unwrap_command` 値

2.2.9 Testing a TDE configuration

透過的なデータ暗号化を有効にして *Single-user mode* でテストを実行するには、環境変数 `PG_TEST_USE_DATA_ENCRYPTION` を設定します。例：

```
make check PG_TEST_USE_DATA_ENCRYPTION=1
```

こちらをご覧ください

- [Enabling TDE](#)
- [PostgreSQL postgres command documentation](#)
- [PostgreSQL Running the Tests documentation](#)