NEC

Transparent Data Encryption for PostgreSQL

行単位暗号化 透過的暗号化機能 利用の手引

ＯＳＳＤＢＴＤＥ０８－０１

ご注意

1. 本書の内容の一部または全部を無断転載することは、禁止されています。
2. 本書の内容に関しては将来予告なしに変更することがあります。
3. 本書の内容について万全を期して作成いたしましたが、万一ご不審な点や誤り、記載漏れなど、お気付きのことがありましたらご連絡ください。

輸出する際の注意事項

本製品（ソフトウェア）は、外国為替管理令に定める提供を規制される技術に該当致しますので、日本国外へ持ち出す際には日本国政府の役務取引許可申請等必要な手続きをお取りください。

許可手続き等にあたり特別な資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

はしがき

本書は、Transparent Data Encryption for PostgreSQL の機能の一つである透過的暗号化機能に関する環境設定および運用方法について説明したものです。

本書の構成は、次の通りです。

章	タイトル	内容
1	透過的暗号化機能の概要	透過的暗号化機能の概要
2	透過的暗号化機能の設計	透過的暗号化機能を利用する際の設計事項や動作環境
3	透過的暗号化機能の導入	透過的暗号化機能の導入方法
4	透過的暗号化機能の運用	透過的暗号化機能の運用方法
5	コマンドリファレンス	透過的暗号化機能で提供するコマンド
6	ファンクションリファレンス	透過的暗号化機能で提供するファンクション
7	パラメータリファレンス	透過的暗号化機能で提供するパラメータ
8	開発サンプル	透過的暗号化機能を利用したサンプルコード
A	透過的暗号化機能で出力されるメッセージ	透過的暗号化機能で出力されるメッセージ
B	改訂履歴	本マニュアルの改訂履歴
C	ライセンス	本ソフトウェアで利用しているソフトウェアのライセンス

備考

1. 本書に説明しているすべての機能はプログラムプロダクトであり、次のプロダクト型番に対応しています。

プロダクト型番	プロダクト名	対応モデル
UL4027-H201-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 (1CPU)(1 年間)	64 ビット
UL4027-H231-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 1CPU 追加(1 年間)	64 ビット
UL4027-H203-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 Cluster Option(1 年間)	64 ビット
UL4027-H211-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 (1CPU)(3 年間)	64 ビット
UL4027-H212-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 1CPU 追加(3 年間)	64 ビット
UL4027-H213-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 Cluster Option(3 年間)	64 ビット
UL4027-J201-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 (1CPU)(1 年間)(時間延長保守)	64 ビット
UL4027-J231-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 1CPU 追加(1 年間)(時間延長保守)	64 ビット
UL4027-J203-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 Cluster Option(1 年間)(時間延長保守)	64 ビット
UL4027-J211-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 (1CPU)(3 年間)(時間延長保守)	64 ビット
UL4027-J212-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 1CPU 追加(3 年間)(時間延長保守)	64 ビット
UL4027-J213-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.1 Linux 版 Cluster Option(3 年間)(時間延長保守)	64 ビット

2. Linux は、Linus Torvalds 氏の米国およびその他の国における登録商標または商標です。
3. Red Hat、Red Hat Enterprise Linux は、米国 Red Hat, Inc.の登録商標です。
4. その他、記載されている会社名および製品名は、一般的にそれぞれ各社の商標または登録商標です。

本書の表記規則

本書では、注意すべき事項、重要な事項および関連情報を以下のように表記します。

注

この表記は、重要であるがデータ損失やシステムおよび機器の損傷には関連しない情報を表します。

重要

この表記は、データ損失やシステムおよび機器の損傷を回避するために必要な情報を表します。

ヒント

この表記は、お客様に役立つ可能性のある情報を表します。

実行例およびファイルの設定例は以下のように表記します

```
コマンドラインの実行例を示します
```

```
ファイルの設定例を示します
```

また、本書では以下の表記法を使用します。

表記	使用方法	例
コマンドライン中の [ ] 角かっこ	かっこ内の値の指定が省略可能であることを示します	cipher_setup.sh [-s {1|2} [path] | -h］
コマンドライン中の { } 波かっこ	かっこ内の値のいずれかを指定する必要があることを示します	cipher_setup.sh [-s {1|2} [path] | -h] 上記例の場合角かっこ内に波かっこがあるため、"-s" オプションを指定した場合、 "1" または "2" を指定する必要があります
#	OS の管理者ユーザで発行するコマンドを示すプロンプトです	# `./cipher_setup.sh`
$	OS の一般ユーザ（postgres など）で発行するコマンドを示すプロンプトです	$ `psql`
=#	PostgreSQL のスーパーユーザーで SQL を発行する場合は、「=#」のように表記しますが、明示的に接続しているデータベース名を示す場合は、「postgres=#」や「testdb=#」のように先頭にデータベース名を含みます	=# SELECT count(*) FROM public.cipher_key_table;
=>	PostgreSQL の一般ユーザーで SQL を発行する場合は、「=>」のように表記しますが、明示的に接続しているデータベース名を示す場合は、「postgres=>」や「testdb=>」のように先頭にデータベース名を含みます	=> SELECT c1 FROM t1;
CMD>	Windows のコマンドプロンプトで発行するコマンドを示します	CMD>`ipconfig`
モノスペースフォント斜体	ユーザーが有効な値に置き換えて入力する項目	tdeforpg2_pg<*PostgreSQL メジャーバージョン*>-<*Transparent Data Encryption for PostgreSQL バージョン*>-.<*Red Hat Enterprise Linux バージョン*>.x86_64.rpm

最新情報の入手先

最新の製品情報については、以下の Web サイトを参照してください。

https://jpn.nec.com/tdeforpg/

目次

第 1 章 透過的暗号化機能の概要 1

1.1 透過的暗号化機能とは 1

1.2 システム構成 1

1.3 機能概要 1

1.4 利用可能な機能と提供されるサービス 2

第 2 章 透過的暗号化機能の設計 3

2.1 透過的暗号化機能の利用の流れ 3

2.2 透過的暗号化機能をセットアップするために必要な情報 3

2.3 利用するモードの検討 4

2.4 利用する暗号化アルゴリズムの検討 5

2.5 暗号鍵のパスフレーズの検討 5

2.6 通信経路の暗号化 6

2.7 動作環境 6

2.7.1 データベースサーバー 7

2.7.1.1 ハードウェア要件 7

2.7.1.2 ソフトウェア要件 7

2.7.2 クライアント 7

2.7.2.1 ハードウェア要件 8

2.7.2.2 ソフトウェア要件 8

第 3 章 透過的暗号化機能の導入 10

第 4 章 透過的暗号化機能の運用 11

4.1 透過的暗号化機能の管理 11

4.1.1 暗号鍵の登録・更新 11

4.1.2 利用するモードの変更 12

4.1.3 最新の暗号鍵によるデータ再暗号化 12

4.1.4 透過的暗号化機能の利用状況の表示 13

4.2 透過的暗号化機能の利用 13

4.2.1 暗号化対象テーブルの作成 13

4.2.2 暗号化/復号処理 13

4.3 バックアップとリストア 16

4.3.1 バックアップ対象について 16

4.3.2 バックアップ手法 17

4.3.3 物理バックアップについて 17
4.3.4 論理バックアップについて 18
4.3.5 COPY コマンドについて 19

第 5 章 コマンドリファレンス 21

5.1 透過的暗号化機能コマンド（pgtde） 21
5.1.1 暗号鍵の登録・更新（-m regist） 25
5.1.2 透過的暗号化機能の利用モードの変更（-m switch） 27
5.1.3 最新の暗号鍵によるデータ再暗号化（-m cipher） 29
5.1.4 透過的暗号化機能の利用状況の表示（-m show） 30

第 6 章 ファンクションリファレンス 32

6.1 ログ出力無効化 （cipher_key_disable_log） 33
6.2 セッション開始（pgtde_begin_session） 34
6.3 ログ出力有効化（cipher_key_enable_log） 36
6.4 セッション終了（pgtde_end_session） 37
6.5 暗号鍵バックアップ（cipher_key_backup） 38

第 7 章 パラメータリファレンス 40

7.1 バックアップファイル出力先設定（encrypt.backup） 41
7.2 暗号鍵パラメータ（encrypt.cipherkey） 42

第 8 章 開発サンプル 43

8.1 環境構築 43
8.2 データ検索 44

付録 A. 透過的暗号化機能で出力されるメッセージ 46

A.1 コマンドエラーメッセージ 46
A.2 透過的暗号化機能により PostgreSQL が出力するメッセージ一覧 47

付録 B. 改訂履歴 50

付録 C. ライセンス 51

C.1 PostgreSQL ライセンス(PostgreSQL, psql, libpq) 51
C.2 The PostgreSQL JDBC Driver 52
C.3 Apache License Version 2.0(Apache Commons Logging, Apache commons Lang, Apache Commons Codec, Apache HttpComponents, Jackson 1.x, Jackson 2.x, Joda-Time) 53

第 1 章
透過的暗号化機能の概要

本章では、透過的暗号化機能の紹介と Edition ごとの提供機能やサービスについて説明します。

1.1　透過的暗号化機能とは

透過的暗号化機能とは、Transparent Data Encryption for PostgreSQL の機能で、Linux 上で稼動している PostgreSQL のデータを暗号化し、そのデータの暗号化/復号をアプリケーションから透過的に行う機能と、暗号鍵の管理を簡単に行う機能を提供をします。

一般的にファイルシステム上に格納されているデータベースファイルは、アクセス権限以外の方法でデータの秘匿性は確保されていません。そのため、悪意を持ったユーザがファイルのアクセス権限を保有している場合にはそのデータベースファイルから情報が漏洩してしまう可能性があります。透過的暗号化機能を使用することで、データベースファイル内のデータを暗号化することができるため、暗号鍵情報を持たない人への情報漏洩を防ぐことができます。

1.2　システム構成

透過的暗号化機能による基本的なシステム構成を下記に示します。

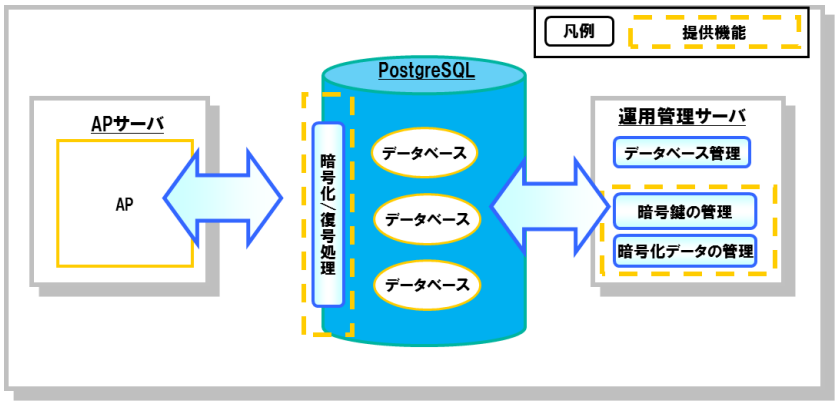

図 1-1　システム構成図

透過的暗号化機能は暗号化/復号処理機能、暗号鍵の管理機能、暗号化データの管理機能で構成されています。

1.3　機能概要

透過的暗号化機能で提供する機能は以下のとおりです。

表 1-1　透過的暗号化機能一覧

機能名	機能内容
暗号化/復号処理	ユーザが暗号化対象テーブルに対してデータの挿入/更新/削除/参照した際に、データを透過的に暗号化/復号する機能のことです。
暗号鍵の管理	透過的暗号化機能で使用する暗号鍵の登録・更新、モードの変更、暗号鍵の利用状況を確認する機能のことです。
暗号化データの管理	古い鍵で暗号化されたデータを最新の暗号鍵を使って一括で再暗号化する機能のことです。

1.4　利用可能な機能と提供されるサービス

Transparent Data Encryption for PostgreSQL には、商用版の Enterprise Edition があります。利用可能な機能と提供されるサービスを示します。

表 1-2　機能/サービス

機能/サービス	Enterprise Edition for Linux
Transparent Data Encryption 機能	
行単位の暗号化機能	○
鍵の更新、バージョン管理機能	○
簡易 TDE モード	○
サポートサービス	
Transparent Data Encryption for PostgreSQL の PP サポートサービス	○
PostgreSQL 本体の保守サポートサービス	○

第 2 章
透過的暗号化機能の設計

本章は、透過的暗号化機能を使用するために事前に設計しておくべき点や必要な動作環境について説明します。

2.1　透過的暗号化機能の利用の流れ

ヒント

下記作業の前に Transparent Data Encryption for PostgreSQL のインストールおよび透過的暗号化機能を有効化する必要があります。詳細は『行単位暗号化 セットアップカード』をご確認ください。

1. 「2.3　利用するモードの検討（4 ページ）」
2. 「2.4　利用する暗号化アルゴリズムの検討（5 ページ）」
3. 「2.5　暗号鍵のパスフレーズの検討（5 ページ）」
4. 「2.6　通信経路の暗号化（6 ページ）」
5. 「4.1.1　暗号鍵の登録・更新（11 ページ）」
6. 「4.2.1　暗号化対象テーブルの作成（13 ページ）」

2.2　透過的暗号化機能をセットアップするために必要な情報

透過的暗号化機能をセットアップする際に事前に確認、決定しておくべきことについて次の表に一覧を記載しています。各項目の詳細は次節以降に説明します。

表 2-1　透過的暗号化機能をセットアップするために必要な情報

確認項目		概要
PostgreSQL の接続情報	ポート番号	透過的暗号化機能をセットアップするデータベースが定義された PostgreSQL のサービス待ち受けポート番号です。
	データベース名	透過的暗号化機能をセットアップするデータベースの名前です。
	スーパユーザ名	透過的暗号化機能をセットアップするデータベースに接続するためのスーパユーザです。
	スーパユーザのパスワード	透過的暗号化機能をセットアップするデータベースに接続するためのスーパユーザのパスワードです。
	セキュリティ管理ユーザ名	透過的暗号化機能の暗号鍵を管理するための専用のユーザです。
	セキュリティ管理ユーザのパスワード	透過的暗号化機能の暗号鍵を管理するための専用のユーザのパスワードです
	アプリケーション管理ユーザ名	透過的暗号化機能を利用しているユーザデータに対する暗号化・復号権限を持つユーザです。
	アプリケーション管理ユーザのパスワード	透過的暗号化機能を利用しているユーザデータに対する暗号化・復号権限を持つユーザのパスワードです。

確認項目	概要
利用するモード	透過的暗号化機能が提供する 2 つのモード（簡易 TDE モード、標準 TDE モード）から利用するモードを選択します。
利用する暗号化アルゴリズム	透過的暗号化機能が提供する 2 つの暗号化アルゴリズム（aes、bf）から利用する暗号化アルゴリズムを選択します。
暗号鍵のパスフレーズ	透過的暗号化機能の標準 TDE モードおよび簡易 TDE モードを利用する際の暗号鍵のパスフレーズを決定します。

2.3　利用するモードの検討

透過的暗号化機能で利用する暗号鍵が不正に利用されないよう、管理者は責任を持って暗号鍵を管理する必要があります。暗号鍵の秘匿性を確保するためには、暗号鍵はデータベースサーバから独立して管理されることが望ましいとされています。一方セキュリティレベルを向上することで、利用のための準備作業や検討項目が増加します。そのため、透過的暗号化機能ではセキュリティ要件やセキュリティレベルに合わせてこれを実現するために簡易 TDE モード、標準 TDE モードの 2 つのモードを提供しています。以下に各モードの機能要件差異を示します。

表 2-2　各モードの機能要件比較

比較対象要件	簡易 TDE モード	標準 TDE モード
セッションごとのファンクション実行の要否	不要	必要
鍵管理のユーザ独自設計要否	必要	必要
鍵情報漏洩の可能性	鍵管理方式に依存	鍵管理方式に依存

- セッションごとのファンクション実行の要否

  セッションごとにファンクションの実行の要否を示します。標準 TDE モードでは、暗号化データにアクセスするセッションごとに、セッション開始と終了時にファンクションを実行する必要があります。簡易 TDE モードでは、その処理を内部的に実行するため、ユーザ側でファンクションを実行する必要がありません。ただし、データベースに接続可能なユーザであれば暗号化したデータを参照できるため、相対的にセキュリティレベルが低下します。

- 鍵管理のユーザ独自設計要否

  暗号鍵を管理者が責任を持って管理する必要があるかを示します。簡易 TDE モードと標準 TDE モードでは暗号鍵を登録する際に設定したパスフレーズを不正に利用されないよう管理する必要があります。標準 TDE モードではセッション開始ファンクションでパスフレーズを設定する必要があるため、セキュリティ管理者に加えてアプリケーション管理者（アプリケーション開発者）も管理する必要があります。簡易 TDE モードでは、セキュリティ管理者のみでパスフレーズを管理することが可能です。

- 鍵情報漏洩の可能性

  暗号鍵が漏洩する可能性を示します。暗号鍵が漏洩した場合、暗号化データを別環境で復号される恐れがあります。簡易 TDE モードは暗号鍵を Transparent Data Encryption for PostgreSQL の一定のルールで秘匿し、データベースサーバに格納しています。一方

で標準 TDE モードではアプリケーションから渡された暗号鍵を利用して鍵管理情報を操作します。そのため、データベース管理者とアプリケーションの管理を分けることができればデータベース管理者に対する暗号鍵の漏えいのリスクを低減できます。

なお、どのモードを利用した場合でも、透過的暗号化機能のセキュリティを向上させるため、『行単位暗号化 セットアップカード』の「よりセキュアな運用のための設定」を行って頂くことを推奨いたします。

2.4　利用する暗号化アルゴリズムの検討

透過的暗号化機能で使用できる暗号化アルゴリズム、およびそれぞれのアルゴリズムによるデータサイズへの影響は以下のとおりです。

表 2-3　暗号化アルゴリズム

暗号化アルゴリズム名	概要	暗号化後データサイズ
aes(Rijndael-128)	Rijndael-128 方式で暗号化を行います。鍵として指定した文字列の 33byte 目以降は無視され、ブロック長は 128bit(16byte)固定です。CPU の命令セット AES-NI を利用することで高速な暗号化/復号が可能です。	行ヘッダを除いた((<*元レコードサイズ*>/16)+1)*16+4
bf(Blowfish)	Blowfish 方式で暗号化を行います。鍵として指定した文字列の 57byte 目以降は無視され、ブロック長は 64bit(8byte)固定です。	行ヘッダを除いた((<*元レコードサイズ*>/8)+1)*8+4

2.5　暗号鍵のパスフレーズの検討

暗号化/復号に使用される暗号鍵の長さは一般的に長くなるほどセキュリティレベルは向上しますが、性能は遅くなります。セキュリティ要件と性能要件の両方を考慮した上で適切な暗号鍵の長さを選択してください。暗号化アルゴリズム別に使用可能な暗号鍵の長さの詳細は以下を参照してください。

- **aes(Rijndael-128)**
  - **16byte(128bit)** : (暗号鍵長<=16byte)のとき使われます。暗号鍵の長さが 16byte より短いときは 16byte までパディングされます。
  - **24byte(198bit)** : (16byte<暗号鍵長<=24byte)の時使われます。暗号鍵の長さが 24byte より短いときは 24byte までパディングされます。
  - **32byte(256bit)** : (24byte<暗号鍵長)の時使われます。暗号鍵の長さが 32byte より短いときは 32byte までパディングされます。暗号鍵の長さが 32byte を超える場合は 32byte を超過した部分の暗号鍵は無視されます。
- **bf(Blowfish)**
  - **最大 56byte(448bit)** : 56byte までの入力した暗号鍵がそのまま使われます。暗号鍵の長さが 56byte を超える場合は 56byte を超過した部分の暗号鍵は無視されます。

2.6　通信経路の暗号化

透過的暗号化機能が提供するコマンドやユーザ定義関数は暗号鍵を入力として受け付けます。暗号化/復号処理はデータベース内で実行されるため、暗号鍵の情報や暗号化対象のデータについては平文で通信されることになります。PostgreSQL データベースへの接続には、ローカル接続もしくは SSL 接続の利用を推奨します。

2.7　動作環境

重要

透過的暗号化機能では PostgreSQL が提供する以下の機能や手続き言語などはサポート対象外です。

- サポート対象外の PostgreSQL の機能
  - バックグラウンドワーカプロセス
  - サーバプログラミングインタフェース
  - パラレルクエリ
    * PostgreSQL サーバの設定ファイル（`postgresql.conf`）を以下のように設定してパラレルクエリを無効にしてください。

      [postgresql.conf 設定例]

      ```
      max_parallel_maintenance_workers=0
      max_parallel_workers_per_gather=0
      ```
  - JIT コンパイル
  - 論理レプリケーション（ロジカルデコーディング）
  - TOAST
    * 格納可能な行データの最大長は 8123 バイトとし、TOAST を使用することができません。行データの最大長が 8123 バイトを超える場合は、Transparent Data Encryption for PostgreSQL 列単位暗号化をご利用ください。
- サポート対象外の PostgreSQL の contrib パッケージ
  - citext
  - fuzzystrmatch
- サポート対象外の PostgreSQL の手続き言語
  - PL/Tcl
  - PL/Perl
  - PL/Python

2.7.1　データベースサーバー

Transparent Data Encryption for PostgreSQL をインストールする PostgreSQL がインストールされているサーバーのハードウェアとソフトウェア要件について説明します。

2.7.1.1　ハードウェア要件

Transparent Data Encryption for PostgreSQL のインストールには下記のハードウェア要件を満たす必要があります。

プロセッサ	x86_64 プロセッサ
メモリ容量	約 200M バイト以上を推奨
ディスク容量	任意のディスクに約 100M バイト以上の空き領域

ヒント

AES-NI の利用

AES による暗号化および復号の高速化を目的とした CPU の命令セット AES-NI を利用するためには、以下の条件を満たす必要があります。

- PostgreSQL 12 以上に対して透過的暗号化機能が有効となっていること
- Linux では Transparent Data Encryption for PostgreSQL V2.1.0 以降が利用されていること
- OpenSSL がインストールされていること
  - Red Hat Enterprise Linux では通常 OpenSSL 1.0.2 系（RHEL7）または OpenSSL 1.1.1 系（RHEL8）がインストールされています。

2.7.1.2　ソフトウェア要件

Transparent Data Encryption for PostgreSQL のインストールには下記のソフトウェア要件を満たす必要があります。

PostgreSQL バージョン	オペレーティングシステム（Linux）	
	Red Hat Enterprise Linux 7.1 以上	Red Hat Enterprise Linux 8.1 以上
12	○	○
必要パッケージ（Linux）	zlib.x86_64 glibc.x86_64	

注

SELinux（Security-Enhanced Linux）機能はサポートしていません。

2.7.2　クライアント

Transparent Data Encryption for PostgreSQL が構成された PostgreSQL にアクセスするクライアント（アプリケーションサーバーなど）のハードウェアとソフトウェア要件について説明します。

2.7.2.1　ハードウェア要件

クライアント側に Transparent Data Encryption for PostgreSQL をインストールするには下記のハードウェア要件を満たす必要があります。

プロセッサ	x86_64 プロセッサ
メモリ容量	約 200M バイト以上を推奨
ディスク容量	任意のディスクに約 100M バイト以上の空き領域 Transparent Data Encryption for PostgreSQL 対応 JDBC ドライバや Transparent Data Encryption for PostgreSQL 対応 ODBC ドライバを利用する場合は 20M バイト以上の空き領域

2.7.2.2　ソフトウェア要件

Transparent Data Encryption for PostgreSQL のインストールには下記のソフトウェア要件を満たす必要があります。

PostgreSQL バージョン	オペレーティングシステム	
	Red Hat Enterprise Linux 7.1 以上	Red Hat Enterprise Linux 8.1 以上
12	○	○
対応アプリケーションプログラムインターフェース	• libpq • PostgreSQL JDBC Driver • psqlODBC	

PostgreSQL JDBC Driver

注

行単位暗号化方式では、PostgreSQL コミュニティから提供されている PostgreSQL JDBC Driver を利用して暗号化/復号処理を実施することが可能です。ただし、PostgreSQL JDBC Driver が出力するログに暗号鍵情報が含まれることがありますので取り扱いにご注意ください。

PostgreSQL JDBC Driver は Transparent Data Encryption for PostgreSQL のインストール媒体に同梱されている Transparent Data Encryption for PostgreSQL 対応 JDBC ドライバをご利用ください。このドライバでは列単位暗号化方式の暗号化データ型属性に対応する後述の O/R マッパをサポートする修正が追加されています。列単位暗号化方式と行単位暗号化方式の両方で利用可能な PostgreSQL 標準データ型に対する処理に関しても同一バージョンの PostgreSQL JDBC Driver と機能的互換性を保持しています。 Transparent Data Encryption for PostgreSQL 対応 JDBC ドライバはインストール媒体の `drivers/jdbc` 配下に格納されています。

`drivers/jdbc` 配下には以下の JDBC ドライバファイルが格納されています。これら JDBC ドライバファイルの使用方法は、PostgreSQL コミュニティから提供されている PostgreSQL JDBC Driver と同様です。

- `postgresql-tdeforpg-42.2.19.1.jar` [JDBC42（JDK1.8）ドライバ]

表 2-5　透過的暗号化機能対応 JDBC の動作検証済 O/R マッパ一覧

O/R マッパ名	バージョン
MyBatis	3.3

O/R マッパ名	バージョン
Hibernate ORM	5.0

psqlODBC

注

行単位暗号化方式では、PostgreSQL コミュニティから提供されている psqlODBC を利用して暗号化/復号処理を実施することが可能です。但し、psqlODBC が出力するログに暗号鍵情報が含まれることがありますので取り扱いにご注意ください。

Transparent Data Encryption for PostgreSQL 対応 ODBC ドライバは Windows(Windows Server 2012 R2 検証済み)のみ対応します。ログに暗号鍵情報が流出することを防ぐ目的で psqlODBC から出力されるログメッセージに対して暗号鍵情報をマスクする仕様となっています。ただし、ODBC トレースの実行により出力されるログは対象外であるため、取り扱いにご注意ください。

psqlODBC は Transparent Data Encryption for PostgreSQL のインストール媒体に同梱されている Transparent Data Encryption for PostgreSQL 対応 ODBC ドライバをご利用ください。このドライバでは列単位暗号化方式の暗号化データ型属性に対応する修正が追加されています。列単位暗号化方式と行単位暗号化方式の両方で利用可能な PostgreSQL 標準データ型に対する処理は同一バージョンの PostgreSQL ODBC Driver と機能的互換性を保持しています。Transparent Data Encryption for PostgreSQL 対応 ODBC ドライバはインストール媒体の `drivers/psqlodbc` 配下に格納されています。

`drivers/psqlodbc` 配下には以下の ODBC ドライバファイルが格納されています。これら ODBC ドライバファイルの使用方法は、PostgreSQL コミュニティから提供されている psqlODBC と同様です。

- `tdeforpg_psqlodbc_12.02.0001_x64.msi` [Windows 対応 psqlODBC 64 ビット]
- `tdeforpg_psqlodbc_12.02.0001_x86.msi` [Windows 対応 psqlODBC 32 ビット]

第 3 章
透過的暗号化機能の導入

Transparent Data Encryption for PostgreSQL のインストールおよび透過的暗号化機能を有効化する必要があります。詳細は『行単位暗号化 セットアップカード』をご確認ください。

第 4 章
透過的暗号化機能の運用

本章は、透過的暗号化機能に関連する運用作業について説明します。

4.1　透過的暗号化機能の管理

4.1.1　暗号鍵の登録・更新

透過的暗号化機能を使用するためには、暗号鍵の登録を行う必要があります。暗号鍵は運用中の適切なタイミングで繰り返し更新することが可能です。暗号鍵の登録・更新をするためには、透過的暗号化機能で提供する pgtde コマンドの-m regist オプションを使用します。

また、透過的暗号化機能では、暗号鍵のバージョンを管理しています。暗号鍵のバージョンを管理することによって暗号鍵を更新する際にすべての暗号化データを再暗号化する必要がなくなります。なお、暗号鍵更新後に確立させたセッションは、暗号化対象テーブルに暗号化/復号処理を最新の暗号鍵で行います。

pgtde コマンドおよび-m regist の詳細は「5.1.1　暗号鍵の登録・更新（-m regist）（25 ページ）」をご確認ください

重要

- 標準 TDE モードを利用している場合、暗号鍵を更新した際にアプリケーションに組み込まれている暗号鍵のパスフレーズを併せて変更する必要があります。
- バージョン管理している暗号鍵情報の一部をユーザの操作ミス等によって削除してしまったり、破損してしまうと、暗号化データを復号して参照することができなくなってしまいます。この様な自体を避ける為、暗号鍵情報をバックアップする機能を提供しています。詳細は「6.5　暗号鍵バックアップ（cipher_key_backup）（38 ページ）」をご確認ください。

注

- 暗号鍵を更新する場合、今までの鍵を使用しているコネクションは、そのまま使用可能ですが、データの暗号化に更新後の鍵は使用されません。また、更新後の鍵によって暗号化されたデータを参照することもできません。その場合は一旦セッションを終了し、セッション開始ファンクションを再度実行してください。
- 空文字の暗号鍵を登録することはできません。

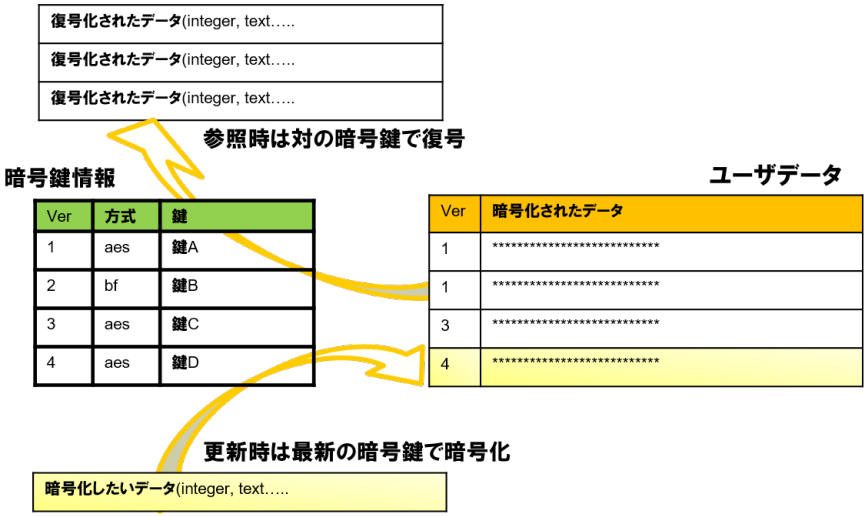

図 4-1　暗号鍵のバージョン管理

4.1.2　利用するモードの変更

透過的暗号化機能を利用するデータベースのセキュリティ要件やセキュリティレベルが変更された場合などに、「2.3　利用するモードの検討（4 ページ）」を参考に利用するモードを決定し、透過的暗号化機能で提供する pgtde コマンドの-m switch オプションを使用します。

pgtde コマンドおよび-m switch の詳細は「5.1.2　透過的暗号化機能の利用モードの変更（-m switch）（27 ページ）」をご確認ください。

重要

利用するモードを変更する場合、アプリケーションに透過的暗号化機能が提供するファンクションを組み込んだり、除外するといった変更が必要です。

4.1.3　最新の暗号鍵によるデータ再暗号化

暗号鍵の登録・更新を繰り返し行うことで、透過的暗号化機能を利用するデータベースは、暗号化されたデータに対応する暗号鍵バージョンが複数混在する状態になります。また古い暗号鍵で暗号化されたデータに対してはインデックスを使った検索が不可能になります。複数の暗号鍵バージョンで暗号化されたユーザデータを最新の暗号鍵バージョンに統一するために、データベースにスーパユーザで接続し、データベース全体に対して VACUUM FULL を実施してください。詳細は「5.1.3　最新の暗号鍵によるデータ再暗号化（-m cipher）（29 ページ）」をご確認ください。

注

- 再暗号化処理は対象のテーブル全てに対して排他ロックを取得するため、再暗号化処理中は対象のテーブルへのアクセスは行わないで下さい。
- 最新の暗号鍵によるデータ再暗号化は負荷状況に注意する必要があります。

- 暗号化対象テーブルに対してマテリアライズドビューを利用している場合、本コマンドを実行してもマテリアライズドビューは再暗号化されません。再暗号化にはマテリアライズドビューのリフレッシュが必要となります。

4.1.4　透過的暗号化機能の利用状況の表示

透過的暗号化機能が有効化されたデータベースの利用しているモードや暗号化アルゴリズム、暗号鍵のバージョンの確認をするために透過的暗号化機能で提供する pgtde コマンドの-m show オプションを使用します。

pgtde コマンドおよび-m show オプションの詳細は「5.1.4　透過的暗号化機能の利用状況の表示（-m show）（30 ページ）」をご確認ください

4.2　透過的暗号化機能の利用

4.2.1　暗号化対象テーブルの作成

透過的暗号化機能を利用するためには、CREATE TABLE 文でテーブルを作成する際に USING tdeforpg2 を指定します。Table Access Method を利用することで、データを行単位で暗号化してデータベースに格納することができます。

次の例では、暗号化対象テーブルを作成する例を記載します。

```
=# CREATE TABLE Employee(
  EmployeeID Integer PRIMARY KEY,
  Name TEXT,
  Address TEXT,
  TelephoneNumber TEXT
)USING tdeforpg2;
```

注

暗号化対象テーブルを以下のオブジェクト、およびルールで使用することはサポートしていません。

- ルール(RULE)
- トリガ(TRIGGER)

4.2.2　暗号化/復号処理

暗号化対象テーブルを透過的に暗号化/復号するために、透過的暗号化機能で提供するセッション開始ファンクションおよびセッション終了ファンクションを利用します。セッション開始ファンクションを利用するには、暗号化されたデータベースで使用している最新の暗号鍵情報が必要になります。セッションが開始され暗号化対象テーブルに対するアプリケーション処理が完了後、セッション終了ファンクションを利用します。

セッション開始ファンクションを利用する際に PostgreSQL のログ設定によっては、暗号鍵の内容が PostgreSQL のログに出力されてしまう可能性があるため、暗号鍵情報が PostgreSQL のログに出力されないようログ出力無効化ファンクションをセッション開始ファンクションの前に利用します。セッション開始ファンクション実行後、ログ出力有効化ファンクションを利用します。

ヒント

それぞれのファンクションの詳細は以下をご確認ください。

- 「6.1　ログ出力無効化（cipher_key_disable_log）（33 ページ）」
- 「6.2　セッション開始（pgtde_begin_session）（34 ページ）」
- 「6.3　ログ出力有効化（cipher_key_enable_log）（36 ページ）」
- 「6.4　セッション終了（pgtde_end_session）（37 ページ）」

透過的暗号化機能では 2 つのモードを提供しており、そのうちの 1 つである簡易 TDE モードでは前述のファンクションは内部的に実行されるため、ユーザがアプリケーションに組み込む必要はありません。その他の標準 TDE モードでは前述のファンクションを実行する必要があります。各モードの詳細は「2.3　利用するモードの検討（4 ページ）」をご確認ください。

次にセッション開始ファンクションとログ出力設定ファンクションの使用方法について説明します。

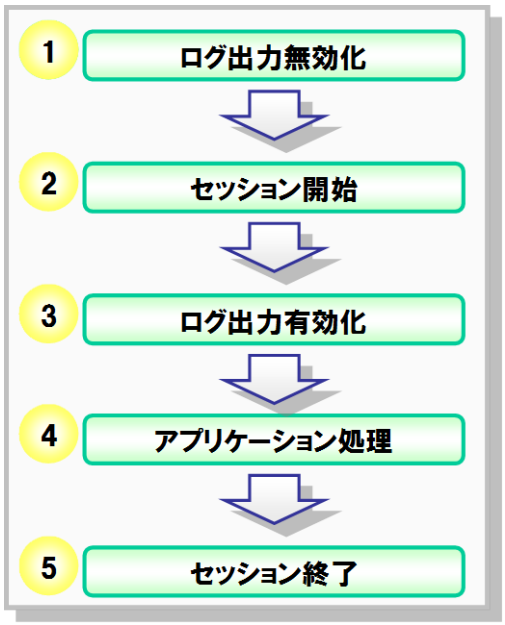

図 4-2　標準 TDE モードにおける透過的暗号化セッション利用方法

上記の図は、透過的暗号化機能を使用してアプリケーション処理を行うための手順になります。また、透過的暗号化機能の標準 TDE モードを使用する場合には、必ずこの手順を踏んでください。

1. ログ出力無効化

ログ出力無効化ファンクションを実行して、セッション開始ファンクションに指定する暗号鍵情報が PostgreSQL のログへ出力されないようにします。

2. セッション開始

   セッション開始ファンクションを実行して、セッションを開始します。

3. ログ出力有効化

   ログ出力有効化ファンクションを実行することにより、ログ出力設定を元に戻します。

4. アプリケーション処理

   アプリケーションとデータベース間の処理を実行します。

5. セッション終了

   セッション終了ファンクションを実行することにより、メモリ上の暗号鍵情報を削除してセッションを終了します。

セッション開始ファンクション実行後、暗号化対象テーブルに挿入/更新/削除/参照することでデータは透過的に暗号化/復号されます。

次の例では暗号化/復号処理を行う場合の動作について説明します。下記例ではデータベースが使用するテーブルを行単位で暗号化してあります。

ヒント

パラメータおよびバックアップの手順の詳細は以下をご確認ください。

- 「7.2　暗号鍵パラメータ（encrypt.cipherkey）（42 ページ）」
- 「4.3　バックアップとリストア（16 ページ）」

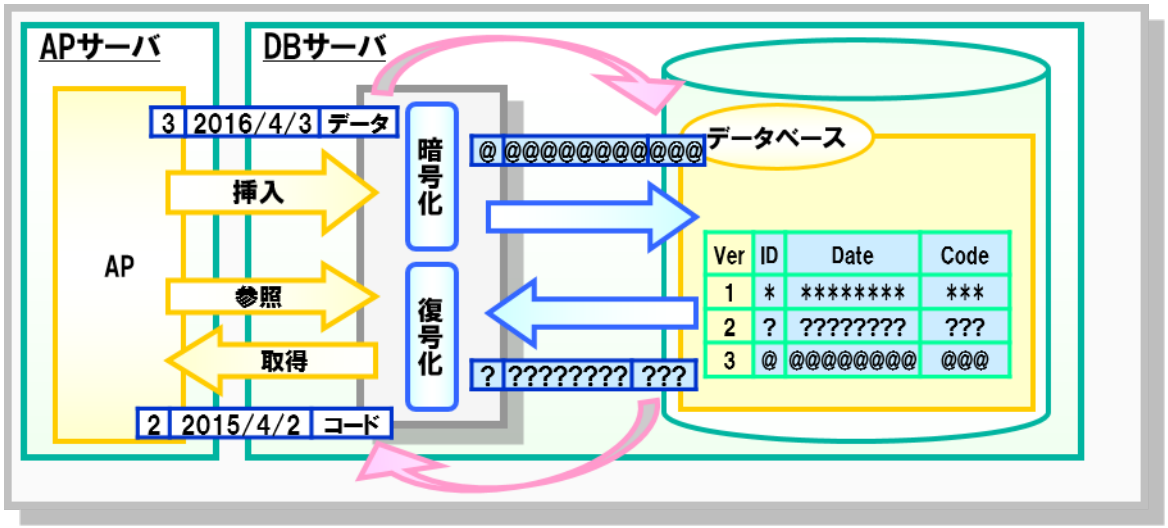

図 4-3　暗号化/復号処理を行う場合

1. データ挿入

アプリケーションから「3,2016/04/3,データ」のデータをテーブルへ挿入するクエリを実行すると、透過的暗号化機能が自動的に行単位で暗号化してから対象のテーブルへデータを挿入します。

2. データ参照

   アプリケーションから「ID」列が「2」の行を参照するクエリを実行すると、透過的暗号化機能が自動的に行単位で復号してアプリケーションへ復号されたデータを返却します。

4.3　バックアップとリストア

Transparent Data Encryption for PostgreSQL を使用したデータベースへのバックアップ・リストアの手法について説明します。

4.3.1　バックアップ対象について

Transparent Data Encryption for PostgreSQL を利用した場合のバックアップ対象は以下になります。

- PostgreSQL のデータベースインスタンス（データベースクラスタ）

  PostgreSQL のデータ領域です。データベースクラスタのバックアップについては、「4.3.2　バックアップ手法（17 ページ）」をご確認ください。

- PostgreSQL サーバの設定ファイル（`postgresql.conf`）やログファイル

  PostgreSQL サーバの設定ファイル（`postgresql.conf`）やログファイルは pg_dump や pg_dumpall ではバックアップされません。そのため、個別にファイルをバックアップします。

- Transparent Data Encryption for PostgreSQL の各種ファイル

  Transparent Data Encryption for PostgreSQL の以下のファイルをバックアップします。

  - 設定ファイル：*<インストールディレクトリ>*`/conf/*`
  - ログファイル：*<インストールディレクトリ>*`/log/*`

  注

  ログファイルの出力先や設定ファイルの配置場所を変更している場合は、変更した配置場所をバックアップしてください。

  設定ファイルやログファイルのバックアップについては、どのように確保するかあわせて検討してください。

4.3.2　バックアップ手法

透過的暗号化機能が有効となっているデータベースのバックアップ手法について説明します。PostgreSQL のバックアップ手法としては、ファイルシステムレベルでのバックアップ（本書では物理バックアップと記載）と SQL によるダンプ（本書では論理バックアップと記載）の大きく 2 つが存在します。また、COPY 文を利用したテーブル単位でのバックアップが存在します。

透過的暗号化機能が有効となっているデータベースもこれらのバックアップ手法を利用できます。以下の表にそれぞれのバックアップ手法のメリットとデメリットを記載しています。目的にあわせて利用するバックアップ手法をご検討ください。

表 4-1　バックアップ手法のメリット・デメリット

項目	物理バックアップ	論理バックアップ	COPY コマンド
概要	データベースクラスタ全体をファイルシステムレベルで取得する	データベースから検索した結果をスクリプト形式またはアーカイブファイル形式で取得する	テーブルのデータをファイルにエクスポートして取得する
使用ツールまたは SQL 文	OS のコマンド（cp、tar、rsync 等)やバックアップツールなどを利用する	pg_dump や pg_dumpall を利用する。	COPY TO (エクスポート）、COPY FROM (インポート）文を実行する
オンラインバックアップ	可能	可能	可能
オフラインバックアップ	可能	不可能	不可能
バックアップ・リストア時間	他の手法より短い	物理バックアップより長い	物理バックアップより長い（暗号化・復号をする場合は、もっとも時間が長い）
バックアップ単位	データベースクラスタ単位	テーブル単位 データベース単位 データベースクラスタ単位	テーブル単位
データベースの論理的なデータ検査の実施有無	未実施	実施	実施
データのバックアップ	暗号化されたデータ	復号したデータ	復号したデータ

4.3.3　物理バックアップについて

物理バックアップでは、オンライン・オフラインバックアップの両方でバックアップを取ることができます。バックアップやリストアの方法については、透過的暗号化機能を利用していないデータベースと変わりません。 透過的暗号化機能を利用している場合、暗号化されたデータは暗号化されたままバックアップされますのでデータ漏えいのリスクを低下させることができます。 なお、復号したデータをバックアップすることはできません。

具体的な手法や手順については、PostgreSQL のマニュアルをご覧ください。

関連リンク

ファイルシステムレベルのバックアップ (最新バージョンの PostgreSQL マニュアル URL https://www.postgresql.jp/document/current/html/backup-file.html)

継続的アーカイブとポイントインタイムリカバリ（PITR） (最新バージョンの PostgreSQL マニュアル URL https://www.postgresql.jp/document/current/html/continuous-archiving.html ）

4.3.4　論理バックアップについて

論理バックアップでは、バックアップ対象のデータを検索(SELECT)した結果をスクリプト形式 またはアーカイブファイル形式で出力するユーティリティツールである pg_dump、pg_dumpall を利用します 。リストアする場合は取得したバックアップファイルをリストアするユーティリティツールである psql または pg_restore を利用します（以降 pg_dump、pg_dumpall、psql または pg_restore をまとめて記載する場合は、バックアップ・リストアユーティリティツールと記載します）。

pg_dump は 1 つのデータベース内のデータを、pg_dumpall はデータベースクラスタ全体（すべてのデータベース）をバックアップする場合に用います。

論理バックアップでは、オンラインバックアップが可能です。バックアップ対象に透過的暗号化機能が有効なデータベースが含まれる場合、データは復号した状態でバックアップされます。バックアップしたデータは復号した状態のため、漏えいのリスクがありますのでご注意ください。

復号したデータをバックアップするためには、バックアップ・リストアユーティリティツールを実行する前に PGOPTIONS 環境変数で encrypt.cipherkey パラメータにデータ鍵の情報を設定します（簡易 TDE モードの場合データ鍵の情報の設定は不要です）。

pg_dump のバックアップの手順例

次の例では、透過的暗号化機能が有効な対象のデータベースに pg_dump を使用してバックアップする例を記載します（簡易 TDE モードの場合 encrypt.cipherkey パラメータは不要です）。出力先ダンプファイル名には「/pgsql/backup/tdedb.bak」を、バックアップ対象データベース名には「tdedb」を指定しています。

- 標準 TDE モードの場合

```
$ PGOPTIONS="-c encrypt.cipherkey=key1234567890" pg_dump -f /pgsql/backup/tdedb
.bak -Fc tdedb
```

- 簡易 TDE モードの場合

```
$ pg_dump -f /pgsql/backup/tdedb.bak -Fc tdedb
```

セットアップ機能 cipher_setup.sh の再実施

出力先のデータベースを空にした状態からリストアする場合、セットアップ機能 cipher_setup.sh の再実施を行います。セットアップ機能 cipher_setup.sh については『行単位暗号化 セットアップカード（Linux 版）』の「透過的暗号化機能の有効化」を参照してください。

暗号鍵の再登録

出力先のデータベースを空にした状態からリストアする場合、暗号鍵の再登録を行います。暗号鍵の登録については、「5.1.1　暗号鍵の登録・更新（-m regist）（25 ページ）」を参照してください。

pg_restore によるリストアの手順例

次の例では、対象のデータベースにバックアップファイルを pg_restore を使用してリストアする例を記載します（簡易 TDE モードの場合 encrypt.cipherkey パラメータは不要です）。 -a や--data-only オプションなどを利用してデータのみリストアする場合は、リストア先データベースに透過的暗号化機能が有効になっている必要があります。リストアするダンプファイル名には「/pgsql/backup/tdedb.bak」を、リストア対象データベース名には「tdedb」を指定しています。

- 標準 TDE モードの場合

```
$ PGOPTIONS="-c encrypt.cipherkey=key1234567890" pg_restore -d tdedb -e /pgsql/backup/tdedb.bak
```

- 簡易 TDE モードの場合

```
$ pg_restore -d tdedb -e /pgsql/backup/tdedb.bak
```

関連リンク

pg_dump について (最新バージョンの PostgreSQL マニュアル https://www.postgresql.jp/document/current/html/app-pgdump.html)

pg_dumpall について (最新バージョンの PostgreSQL マニュアル https://www.postgresql.jp/document/current/html/app-pg-dumpall.html)

pg_restore について (最新バージョンの PostgreSQL マニュアル https://www.postgresql.jp/document/current/html/app-pgrestore.html)

4.3.5　COPY コマンドについて

COPY コマンドは PostgreSQL のテーブルをファイルにエクスポートしたり、ファイルから PostgreSQL にインポートするために用意されているコマンドです。このコマンドを使用することで任意のテーブルを CSV ファイルなどに出力することが可能です。

また、COPY コマンドを用いることでデータベースの暗号化対象テーブルを復号して CSV に出力することができます。

暗号化対象テーブルから復号したデータをエクスポート/インポートする

次の例では、暗号化対象テーブルから復号したデータを CSV ファイルとしてエクスポートする例を記載します。なお、本実行例では復号したデータをエクスポートするため、データベース管理者以外のユーザが COPY コマンドを実行することを想定して、psql が提供する\copy コマンドを使用する方法を示しています。暗号鍵として「key1234567890」を、復号したデータを出力する CSV ファイル名に「/export/user_table.csv」を指定します。

```
=>select public.cipher_key_disable_log();
=>select public.pgtde_begin_session('key1234567890');
=>select public.cipher_key_enable_log();
=>\copy apuser.user_table TO '/export/user_table.csv' WITH CSV;
=>select public.pgtde_end_session();
```

エクスポートした CSV ファイルを暗号化対象テーブルにインポートする例を以下に記載します。

```
=>select public.cipher_key_disable_log();
=>select public.pgtde_begin_session('key1234567890');
=>select public.cipher_key_enable_log();
=>\copy apuser.user_table FROM '/export/user_table.csv' WITH CSV;
=>select public.pgtde_end_session();
```

重要

COPY コマンドを利用する場合、 public スキーマに定義されている透過的暗号化機能が利用するテーブルである cipher_key_table と key_management_table もインポートしてください。cipher_key_table と key_management_table のデータがエクスポートされたファイルの出力先やファイル名は「7.1　バックアップファイル出力先設定（encrypt.backup）（41 ページ）」や「6.5　暗号鍵バックアップ（cipher_key_backup）（38 ページ）」をご確認ください。

関連リンク

COPY について (最新バージョンの PostgreSQL マニュアル https://www.postgresql.jp/document/current/html/sql-copy.html)

第 5 章
コマンドリファレンス

本章では、透過的暗号化機能で提供しているコマンドについて説明します。それぞれのコマンドについて、次の内容を説明します。

構文

コマンドの入力方法を記載します。また、コマンドの基本的な使用方法を簡単に説明します。

説明

コマンドの目的や使用方法を記載します。

パラメータ

構文の中に含まれるそれぞれのパラメータの内容ついて記載します。

使用方法

コマンドの使用方法およびコマンドの動作に関する追加情報を記載します。

注意制限事項

コマンドの注意制限事項を記載します。

実行例

コマンドの実行例を記載します。

5.1　透過的暗号化機能コマンド（pgtde）

構文

```
pgtde -m { regist | cipher [CIPHER_OPTIONS] | switch {SWITCH_OPTIONS} | show} [CONNECTION_OPTI
ONS]
```

説明

透過的暗号化機能コマンド（pgtde）は、透過的暗号化機能で利用する暗号鍵の管理機能と暗号化データの管理機能を提供します。提供する機能は以下の通りです。

- 「5.1.1　暗号鍵の登録・更新（-m regist）（25 ページ）」

  透過的暗号化機能に使用する暗号鍵の登録・更新をすることができます。

- 「5.1.3　最新の暗号鍵によるデータ再暗号化（-m cipher）（29 ページ）」

  暗号化済みのユーザデータを一括で再暗号化することができます。

  重要

  - 本オプションは制限事項の為、現在使用することができません。
  - 再暗号化には対象のテーブルに対して VACUUM FULL コマンドを実行してください。

- 「5.1.2　透過的暗号化機能の利用モードの変更（-m switch）（27 ページ）」

  透過的暗号化機能のモードを切り替える機能を提供します。

- 「5.1.4　透過的暗号化機能の利用状況の表示（-m show）（30 ページ）」

  透過的暗号化機能の利用状況を表示することができます。

ヒント

それぞれの機能は『行単位暗号化 セットアップカード』の「よりセキュアな運用のための設定」を行っている場合、以下の表のとおり、透過的暗号化機能コマンド（pgtde）の-m オプション毎に実行ユーザが制限されます。

表 5-1　-m オプション毎の実行ユーザ制限

実行可能ユーザ	-m オプション
セキュリティ管理者	regist（暗号鍵の登録・更新）
	switch（透過的暗号化機能の利用モードの変更）
	show（透過的暗号化機能利用状況の表示）

パラメータ

表 5-2　-m オプションパラメーター覧

パラメータ		説明	分類
-m	regist	暗号鍵登録・更新します。	選択
	switch	透過的暗号化機能の利用モードの変更します。	選択
	show	透過的暗号化機能の利用状況の表示します。	選択

表 5-3　データベース接続パラメーター覧

パラメータ	説明	分類
-d *dbname*	接続先のデータベース名を指定します。	任意

パラメータ	説明	分類
-h *host*	接続先のホスト名または IP アドレスを指定します。	任意
-p *port*	接続先のポート番号を指定します。	任意
-U *user*	接続先データベースユーザー名を指定します。	任意
-pw *password*	接続先データベースユーザーのパスワードを指定します。	任意
-conf *dbconfigfile*	データベース接続情報を記載した設定ファイルを指定します。このオプションを指定した場合は-d, -h, -p, -U, -pw オプションは使用できません。	任意

透過的暗号化機能で提供する透過的暗号化機能コマンドのデータベース接続情報は、設定ファイルで指定することも可能です。設定ファイルのテンプレート「pgtde.properties.template」は、透過的暗号化機能のインストールディレクトリ（Linux 版 Transparent Data Encryption for PostgreSQL のデフォルトは/opt/nec/tdeforpg2_pgXX（XX は PostgreSQL のメジャーバージョン。12 の場合は 12））の「template」ディレクトリ配下の格納されています。設定ファイルを別の場所から読み込みたい場合は-conf オプションで対象ファイルパスを**絶対パス**で指定してください。

[pgtde.properties 設定例]

```
pgtdedatabase=pgtdedb
pgtdehost=hostname
pgtdeport=5432
pgtdeuser=username
pgtdepassword=userpassword
```

ヒント

本コマンドのデータベース接続情報は、接続パラメータ以外に環境変数（環境変数を記載したファイル）からも設定することができます。環境変数を使用してデータベース接続情報を設定した場合、コマンド上での接続パラメータの指定を省略することが可能です。

表 5-4　環境変数および設定ファイルの設定項目一覧

環境変数名	設定ファイル	内容
PGDATABASE	pgtdedatabase	接続するデータベース名
PGHOST	pgtdehost	接続するサーバアドレス(ホスト名)
PGPORT	pgtdeport	接続するポート番号
PGUSER	pgtdeuser	接続するユーザー名
PGPASSWORD	pgtdepassword	接続するユーザのパスワード

[環境変数設定例（Linux）]

```
$ export PGDATABASE=pgtdedb
$ export PGHOST=hostname
$ export PGPORT=5432
$ export PGUSER=username
$ export PGPASSWORD=userpassword
```

上記の接続パラメータまたは設定ファイル、環境変数が同時に設定されている場合には、下図のとおり接続パラメータまたは設定ファイル、環境変数の順番で優先順位が決められています。

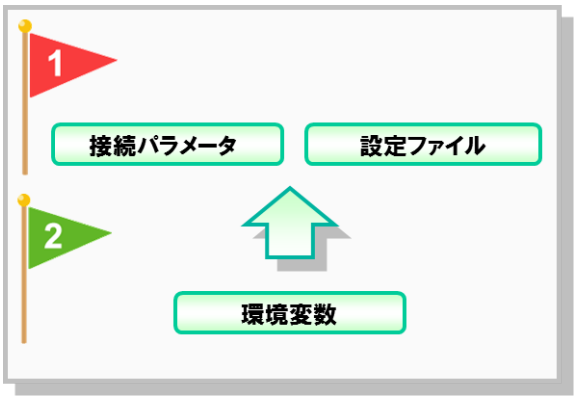

図 5-1　接続パラメータ優先順位図

使用方法

透過的暗号化機能コマンド（pgtde）の戻り値は以下の通りです。

表 5-5　透過的暗号化機能コマンド戻り値

パターン	戻り値
正常終了	0
異常終了	1

注意制限事項

透過的暗号化機能コマンド（pgtde）の-m オプション共通の注意事項を記載します。

- 透過的暗号化機能コマンド（pgtde）を同一データベースに対して並列で実行することはサポートしていません。
- 透過的暗号化機能コマンド（pgtde）は対話型による利用を前提とした仕様であるため、バックグラウンドジョブとして実行できません。
- 接続パラメータである、データベース名、ホスト名、ユーザー名、パスワードにマルチバイト文字を指定することはできません。
- 接続パラメータと設定ファイルをコマンドから同時に参照させることはできません。
- データベース接続情報にシェルのメタ文字を含む場合は、それぞれのメタ文字の直前にエスケープ文字を入力する必要があります。データベース接続情報のパスワードを「\"$@&」としている場合は以下のように入力します。

```
-pw \\\"\$\@\&
```

実行例

以下のコマンドを入力することにより、透過的暗号化機能コマンドの Usage を参照することができます。

- [Linux]

```
Usage:
 pgtde CONNECTION_OPTIONS MODE_OPTIONS

MODE_OPTIONS:
 -m regist                      register new data key to specified database.
 -m cipher [CIPHER_OPTIONS]     cipher data with latest key
 -m switch {SWITCH_OPTIONS}     switch key management mode
 -m show                         show status of using Transparent Data Encryption in specif
ied database

CIPHER_OPTIONS:                 [--reset]
 --reset                        including reset cipher key's version

SWITCH_OPTIONS:                 --simple-tde | --standard-tde
 --simple-tde                   switch key management mode to simple TDE mode
 --standard-tde                 switch key management mode to standard TDE mode

CONNECTION_OPTIONS:             [-h] [-p] [-U] [-pw] [-d] | -conf
 -h           HOSTNAME          database server host or socket directory
 -p           PORT              database server port number
 -U           USERNAME          user name for authentication
 -pw          PASSWORD          password for authentication
 -d           DBNAME            database name
 -conf        DbConfigfile      database connection information config file

HELP:
 --help                         show this help, then exit
```

5.1.1　暗号鍵の登録・更新（-m regist）

構文

- [Linux]

```
pgtde -m regist [CONNECTION_OPTIONS]
```

説明

暗号鍵の登録・更新（`-m regist`）は、接続対象データベースで利用する暗号鍵の登録・更新をすることができます。本`-m` オプションは対話的にコマンドを実行します。

なお、ストリーミングレプリケーションを利用している場合は、プライマリサーバ上で実行してください。実行結果はスタンバイサーバにも反映されます。

使用方法

暗号鍵登録・更新（`-m regist`）オプションを指定した際の処理の流れについて説明します。

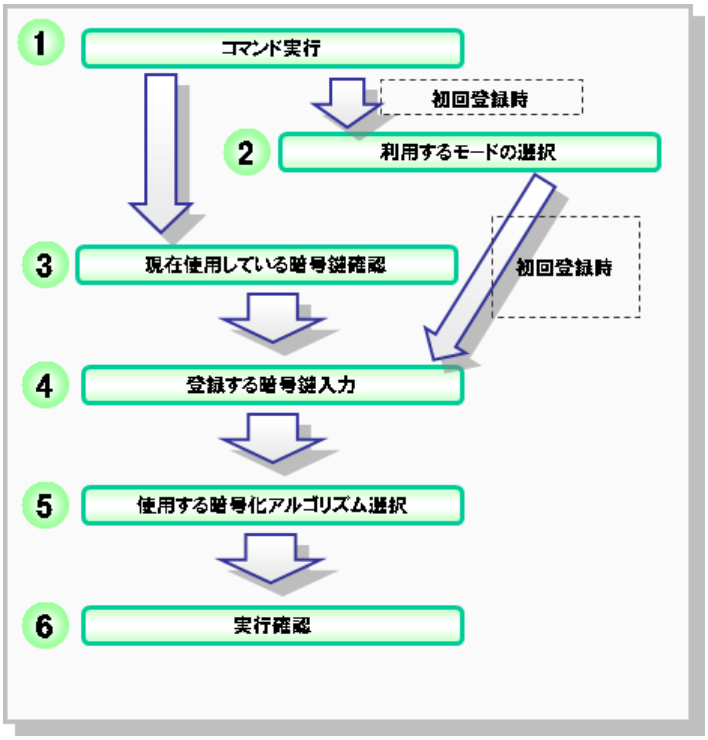

図 5-2　暗号鍵登録・更新

1. コマンド実行

   透過的暗号化機能コマンド（pgtde）を実行します。

2. 利用するモードの選択(初回登録時)

   利用するモードを選択します。

   - [Linux]
     - 1. Simple TDE mode.（簡易 TDE モードを利用する場合）
     - 2. Standard TDE mode.（標準 TDE モードを利用する場合）

   上記 「1」または「2」 を指定します。

3. 現在使用している暗号鍵を入力

   初回登録時を除き、現在使用している暗号鍵を入力します。

4. 登録する暗号鍵を入力

   新たに登録する暗号鍵を入力します。

5. 使用する暗号化アルゴリズムを選択

   透過的暗号化で使用する暗号化アルゴリズムを選択します。選択できる暗号化アルゴリズムは以下のとおりです。

   - 1. aes
   - 2. bf

   上記から使用するアルゴリズムを選択して番号を入力します。

6. 実行決定

   上記までに行った設定で暗号鍵登録・更新を実行するためには、「Y」か「y」を入力します。なお、実行を中止する場合には、「Y」および「y」以外を入力します。

注意制限事項

- 標準 TDE モードを利用している場合、暗号鍵を更新した際にアプリケーションに組み込まれている暗号鍵のパスフレーズを併せて変更する必要があります。
- バージョン管理している暗号鍵情報の一部をユーザの操作ミス等によって削除してしまったり、破損してしまうと、暗号化データを復号して参照することができなくなってしまいます。この様な自体を避ける為、暗号鍵情報をバックアップする機能を提供しています。詳細は「6.5　暗号鍵バックアップ（cipher_key_backup）（38 ページ）」をご確認ください。
- 暗号鍵を更新する場合、今までの鍵を使用しているコネクションは、そのまま使用可能ですが、データの暗号化に更新後の鍵は使用されません。また、更新後の鍵によって暗号化されたデータを参照することもできません。その場合は一旦セッションを終了し、セッション開始ファンクションを再度実行してください。
- 空文字の暗号鍵を登録することはできません。

実行例

- [Linux]
  - 次の例では、簡易 TDE モードで暗号鍵を登録します。データベースの接続には設定ファイル「/tmp/pgtde_secuser.properties」を利用します。

```
$ pgtde -m regist -conf /tmp/pgtde_secuser.properties
Key management mode is not yet set.
Please select key management mode:
1. Simple TDE mode.
2. Standard TDE mode.
1
Enter new data key:
Retype new data key:
Select algorithm:
1. aes
2. bf
1
Are you sure you want to Regist new key to "tdedb"(DATABASE) with "aes" algorithm? (Press Y(y)
 key to execute): Y
New key version 1 is registered to tdedb
```

5.1.2　透過的暗号化機能の利用モードの変更（-m switch）

コマンド構文

- [Linux]

```
pgtde -m switch {--simple-tde | --standard-tde} [CONNECTION_OPTIONS]
```

説明

透過的暗号化機能の利用モードの変更（-m switch）では、透過的暗号化機能の利用モードををスムーズに切り替える機能を提供します。

なお、ストリーミングレプリケーションを利用している場合は、プライマリサーバ上で実行してください。実行結果はスタンバイサーバにも反映されます。

パラメータ

表 5-6　透過的暗号化機能の利用モードの変更 パラメーター覧

パラメータ	説明	分類
--simple-tde	簡易 TDE モードに変更する場合に指定します。	選択
--standard-tde	標準 TDE モードに変更する場合に指定します。	選択

使用方法

透過的暗号化機能の利用モードの変更（-m switch）オプション処理の流れについて説明します。

簡易 TDE モードと標準 TDE モードを切り替える場合

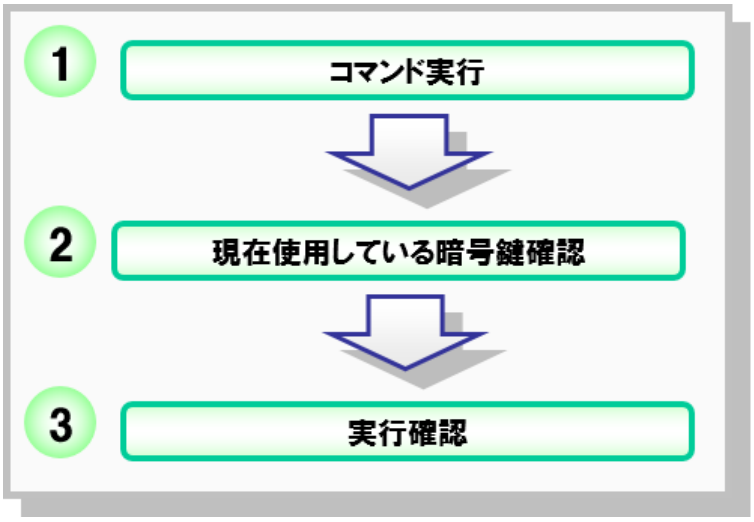

図 5-3　透過的暗号化機能の利用モードの変更（簡易 TDE モードと標準 TDE モードを切り替える）

1. コマンド実行
   - 簡易 TDE モードに変更する場合：「--simple-tde」パラメータを指定して pgtde コマンドを実行します。
   - 標準 TDE モードに変更する場合：「--standard-tde」パラメータを指定して pgtde コマンドを実行します。

2. 現在使用している暗号鍵入力

   現在使用中の暗号鍵を入力します。

3. 実行確認

   上記までに行った設定で過的暗号化機能の利用モードの変更を実行するためには、「Y」か「y」を入力します。なお、実行を中止する場合には、(「Y」および「y」)以外を入力します。

注意制限事項

- 利用するモードを変更する場合、アプリケーションに透過的暗号化機能が提供するファンクションを組み込んだり、除外するといった変更が必要です
- 簡易 TDE モードと標準 TDE モードを切り替えた場合は新しい暗号鍵は登録されません。
- 透過的暗号化機能によって出力された暗号鍵ファイルを修正したファイルは、透過的暗号化機能は正しく読み取ることができなくなるため、出力された暗号鍵ファイルの修正は禁止しています。

実行例

- [Linux]
  - 次の例では、簡易 TDE モードから標準 TDE モードに変更します。データベースの接続には設定ファイル「/tmp/pgtde_secuser.properties」を利用します。

```
$ pgtde -m switch --standard-tde -conf /tmp/pgtde_secuser.properties
Enter current data key:
Are you sure you want to switch key management mode of "tdedb"(DATABASE) to "Standard TDE mode
" ? (Press Y(y) key to execute): y
New key version 1 is registered to tdedb

Switch key management mode to Standard TDE mode mode is successfully.
```

5.1.3　最新の暗号鍵によるデータ再暗号化（-m cipher）

重要

本オプションは制限事項の為、現在使用することができません。

- 最新の暗号鍵によるデータ再暗号化を実施する場合、データベースにスーパユーザで接続し、データベース全体に対して VACUUM FULL を実施してください。以下はセッション開始時に暗号鍵として 「key1234567890」を指定した例となります。

```
#= select cipher_key_disable_log();
 cipher_key_disable_log
------------------------
 t
(1 行)

#= select pgtde_begin_session('key1234567890');
```

```
 pgtde_begin_session
---------------------
 t
(1 行)

#= select cipher_key_enable_log ();
 cipher_key_enable_log
-----------------------
 t
(1 行)

#= VACUUM FULL VERBOSE;
・・・
VACUUM
#= select pgtde_end_session();
 pgtde_end_session
---------------------
 t
(1 行)
```

注

暗号鍵のリセットを実施したい場合は、PP サポートサービスへお問い合わせください。

5.1.4　透過的暗号化機能の利用状況の表示（-m show）

コマンド構文

```
pgtde -m show [CONNECTION_OPTIONS]
```

説明

透過的暗号化機能の利用状況の表示（-m show）では透過的暗号化機能の利用状況を表示します。利用している暗号化アルゴリズムや現在の暗号鍵バージョンを表示します。

パラメータ

なし

透過的暗号化機能の利用状況の表示オプションの流れ

本オプションの処理の流れについて説明します。

1. コマンド実行

   本コマンドを実行します。

2. 暗号鍵の情報を表示

   以下の情報を表示します

   - 利用している暗号化アルゴリズム
   - 暗号鍵バージョン

実行例

- [Linux]
  - 次の例では、簡易 TDE モードを利用している際に透過的暗号化機能の利用状況を確認しています。データベースの接続には設定ファイル「/tmp/pgtde_secuser.properties」を利用します。

```
$ pgtde -m show -conf /tmp/pgtde_secuser.properties

Database "tdedb" is operating in "Simple TDE mode" key management mode.

### Cipher key information #####
* Algorithm: aes
* Version: 5
################################
```

第 6 章
ファンクションリファレンス

本章では、透過的暗号化機能では PostgreSQL のユーザ定義関数を利用して機能を提供しています。提供するユーザ定義関数を「ファンクション」として記載します。

ヒント

簡易 TDE モードでは「ログ出力無効化」、「セッション開始」、「ログ設定有効化」、「セッション終了」ファンクションは内部で実行されるため、ユーザ側で実行せずとも透過的暗号化機能を利用できます。

表 6-1　ファンクション一覧

機能名	機能内容	詳細
ログ出力無効化	ログへの暗号鍵情報出力を抑制します。	「ログ出力無効化（33 ページ）」
セッション開始	透過的暗号化に使用する暗号鍵情報をセッションに設定します。	「セッション開始（34 ページ）」
ログ設定有効化	「ログ出力無効化」により変更されたログ出力設定を元に戻します。	「ログ出力有効化（36 ページ）」
セッション終了	透過的暗号化に使用する暗号鍵情報をセッションからクリアします。	「セッション終了（37 ページ）」
暗号鍵バックアップ	鍵管理機能で管理している暗号鍵情報をバックアップします。	「暗号鍵バックアップ（38 ページ）」

本章では、透過的暗号化機能で提供しているコマンドについて説明します。それぞれのコマンドについて、次の内容を説明します。

構文

ファンクションの入力方法を記載します。

説明

ファンクションの目的や使用方法を記載します。

引数

構文の中に含まれるそれぞれの引数の内容ついて記載します。

戻り値

ファンクションの戻り値について記載します。

注意制限事項

ファンクションの注意制限事項を記載します。

実行権限

ファンクションを実行するために必要な権限をを記載します。

影響範囲

ファンクションの影響範囲を記載します。影響範囲としてファンクション実行時のみとセッションの 2 種類があります。

実行例

ファンクションの実行例を記載します。

6.1　ログ出力無効化 （cipher_key_disable_log）

構文

```
cipher_key_disable_log()
```

説明

透過的暗号化機能を利用するには「セッション開始」ファンクション（pgtde_begin_session）を使用する必要があります。「セッション開始」ファンクションの引数には暗号鍵情報を入力します。この時、PostgreSQL の設定によっては、暗号鍵情報が PostgreSQL のログやアクティビティ情報に出力されてしまう可能性があります。本ファンクションを実行することで引数として入力した暗号鍵情報が PostgreSQL のログやアクティビティ情報に出力されないようになります。なお、「セッション開始」ファンクションを実行する前には、本ファンクションを実行する必要があります。

引数

なし

戻り値

「cipher_key_disable_log」ファンクションの戻り値は以下のとおりです。

表 6-2　cipher_key_disable_log ファンクション戻り値

戻り値	パターン
true	正常終了

注意制限事項

- 「セッション開始」ファンクションを実行する際は予め本ファンクションを実行する必要があります。
- ログ出力無効化中は、実行される SQL の最初の"("から最後の")"までの全ての文字列がログに出力されないようになります。
- psql -c で複数クエリを実行すると、PostgreSQL は 1 つのクエリと認識するため、ログマスク処理が動作しないことがあります。

実行権限

PostgreSQL の一般ユーザー権限以上

影響範囲

セッション

実行例

```
=>SELECT cipher_key_disable_log();
 cipher_key_disable_log
------------------------
 t
```

6.2　セッション開始（pgtde_begin_session）

構文

```
pgtde_begin_session('key')
```

説明

透過的暗号化機能を利用するには「セッション開始」ファンクション(pgtde_begin_session)を使用する必要があります。本ファンクションにより暗号化に使用する暗号鍵の情報をセッションに設定します。本ファンクションに接続対象の最新の暗号鍵を指定することにより、セッションを開始することができます。

引数

表 6-3　pgtde_begin_session 引数一覧

引数	説明	種別
key	透過的暗号化機能を適用したデータベースに登録されている最新の暗号鍵を指定します。	必須

戻り値

「pgtde_begin_session」ファンクションの戻り値は以下のとおりです。

表 6-4　pgtde_begin_session ファンクション戻り値

戻り値	パターン
true	正常終了
false	暗号鍵が未登録の場合

注意制限事項

- 本ファンクション実行前には「ログ出力無効化」ファンクションを実行している必要があります。ログ出力無効化ファンクションを実行せずに本ファンクションを実行した場合、エラーとなり、ログに引数として指定した暗号鍵情報が出力される可能性があります。
- 本ファンクション実行後は「ログ出力有効化」ファンクションを実行しないと、その後のクエリで性能劣化が発生する可能性があります。
- `psql` 等から「セッション開始」ファンクションを実行する場合、`psql` のコマンド履歴ファイル（`.psql_history`）に指定した暗号鍵文字列が平文で記載されてしまう場合があります。`psql` のコマンド履歴ファイルに履歴を残さないようにするには以下のオプションをつけて psql を実行する必要があります。

```
$ psql --no-readline
```

実行権限

PostgreSQL の一般ユーザー権限以上

影響範囲

セッション

実行例

```
=>SELECT pgtde_begin_session('key');
 pgtde_begin_session
---------------------
 t
```

注

簡易 TDE モードで実行した場合、以下の NOTICE メッセージが表示されます。

```
=> SELECT pgtde_begin_session('key');
NOTICE:  TDE-N0001 No need to execute pgtde_begin_session in simple TDE mode[01].
 pgtde_begin_session
---------------------
 t
```

6.3　ログ出力有効化（cipher_key_enable_log）

構文

```
cipher_key_enable_log()
```

説明

本ファンクションでは、「ログ出力無効化」ファンクション（cipher_key_disable_log）で設定された暗号鍵情報出力抑制状態を解除することができます。

引数

なし

戻り値

「cipher_key_enable_log」ファンクションの戻り値は以下のとおりです。

表 6-5　cipher_key_enable_log ファンクション戻り値

戻り値	パターン
true	正常終了

注意制限事項

なし

実行権限

PostgreSQL の一般ユーザー権限以上

影響範囲

セッション

実行例

「cipher_key_enable_log」ファンクションを実行します。

```
=>SELECT cipher_key_enable_log();
 cipher_key_enable_log
-----------------------
 t
```

6.4　セッション終了（pgtde_end_session）

ファンクション構文

```
pgtde_end_session()
```

説明

透過的暗号化機能で提供している「セッション開始」ファンクションを使用して開始したセッションを終了することができます。また、本ファンクションは、セッションを終了する際に暗号鍵情報をクリアします。なお、コネクションプール機能を使用している場合には、本ファンクションを使用してセッションを終了しない場合、暗号鍵の情報がメモリー上に残留してしまうので注意が必要です。

引数

なし

戻り値

「pgtde_end_session」ファンクションの戻り値は以下のとおりです。

表 6-6　pgtde_end_session ファンクション戻り値

戻り値	パターン
true	正常終了

戻り値	パターン
false	セッションを開始していない場合

注意制限事項

なし

実行権限

PostgreSQL の一般ユーザー権限以上

影響範囲

セッション

実行例

「pgtde_end_session」ファンクションを実行します。

```
=>SELECT pgtde_end_session();
 pgtde_end_session
-------------------
 t
```

6.5　暗号鍵バックアップ（cipher_key_backup）

構文

```
cipher_key_backup()
```

説明

透過的暗号化機能は暗号鍵情報が何らかの原因で壊れ暗号鍵情報を取得できない状態になると暗号化対象テーブルを復号してデータを参照することができなくなってしまいます。そのため、暗号鍵情報を格納する cipher_key_table テーブルと key_management_table テーブルをバックアップする機能をファンクションで提供します。なお、透過的暗号化機能で提供している暗号鍵登録・更新コマンド（`pgtde -m regist`）を使用した際には、自動的にこのバックアップ機能が実行されます。本ファンクションで取得したバックアップファイルの保存先は、「バックアップファイル出力先設定」オプションで任意の場所に変更することができます。詳細は「バックアップファイル出力先設定（41 ページ）」をご覧ください。

バックアップファイル名は cipher_key_table が「ck_backup_*データベース名*」、key_management_table が「mk_backup_*データベース名*」です。また、cipher_key_table は一世代前のバックアップファイル名が「ck_backup_*データベース名*.sv」として残ります。key_management_table は一世代前のバックアップファイルを残しません)。なお、バックアップファイル名のデータベース名には、暗号鍵のバックアップを実施したデータベース名が入ります。

引数

なし

戻り値

「cipher_key_backup」ファンクションの戻り値は以下のとおりです。

表 6-7　cipher_key_backup ファンクション戻り値

戻り値	パターン
true	正常終了
false	異常終了

注意制限事項

なし

実行権限

PostgreSQL のスーパユーザーと cipher_setup.sh で作成したセキュリティユーザ

実行例

「cipher_key_backup」ファンクションを実行します。

```
=# SELECT cipher_key_backup();
 cipher_key_backup
-------------------
 t
```

第 7 章
パラメータリファレンス

透過的暗号化機能では動作オプションとして PostgreSQL のカスタムパラメータを提供しています。提供しているカスタムパラメータは以下のとおりです。

表 7-1　動作オプション一覧

機能名	機能内容	詳細
バックアップファイル出力先設定	本ファンクションで提供している「暗号鍵バックアップ（38 ページ）」のバックアップファイルの出力先を変更することができます。	「バックアップファイル出力先設定（41 ページ）」
暗号鍵パラメータ	標準 TDE モードにおいて、pg_dump などのコマンド実行時のように、「セッション開始」ファンクション(pgtde_begin_session)が実行できないケースでの暗号鍵の指定に使用します。	「暗号鍵パラメータ（42 ページ）」

本動作オプションの設定をデータベース全体に設定するためには、PostgreSQL の設定ファイル（postgresql.conf）に記述する必要があります。設定を反映するには、PostgreSQL の再起動か構成のリロードが必要です。以下に、「バックアップファイル出力先設定」の設定例を記載します。

[postgresql.conf 設定例]

```
encrypt.backup = '/backup'
```

それぞれのパラメータについて、次の内容を説明します。

構文

パラメータの設定方法を記載します。

説明

パラメータの目的や使用方法を記載します。

設定値

パラメータが取りうる値について記載します。

注意制限事項

パラメータの注意制限事項を記載します。

設定項目の反映タイミング

パラメータが反映されるタイミングを記載します。

実行例

パラメータの実行例を記載します。

7.1　バックアップファイル出力先設定（encrypt.backup）

構文

```
encrypt.backup = { '' | 'backup_directory' }
```

概要

本パラメータを設定することで、「暗号鍵バックアップ」ファンクションで出力するバックアップファイルの保存先を変更することができます。なお、既定の設定では「SHOW data_directory」クエリで出力されるディレクトリにバックアップファイルが出力されます。

設定値

表 7-2　バックアップファイル出力先設定

設定値	内容	種別
''（空文字）	本パラメータに既定で設定されている値です。「SHOW data_directory」クエリで出力されるディレクトリにバックアップファイルが出力されます。	既定値
'*backup_directory*'	「暗号鍵バックアップ」ファンクションで作成されるバックアップファイルの保存先ディレクトリパスを指定します。**絶対パス**で指定してください。指定するディレクトリは PostgreSQL の起動ユーザがファイルの書き込み権限を持っている必要があります。	

注意制限事項

なし

設定項目の反映タイミング

個々のセッション毎に PostgreSQL のスーパユーザのみが設定可能です。設定値は即時セッションに反映されます。

実行例

次の例では「/backup」にバックアップファイル出力先を変更します。

```
=# SET encrypt.backup = '/backup';
SET
```

7.2　暗号鍵パラメータ（encrypt.cipherkey）

構文

```
encrypt.cipherkey = key
```

概要

標準 TDE モードにおいて、pg_dump などのコマンド実行時のように、「セッション開始」ファンクション(pgtde_begin_session)が実行できないケースでの暗号鍵の指定に使用します。

設定値

表 7-3　暗号鍵パラメータ

設定値	内容	種別
key	透過的暗号化機能を適用したデータベースに登録されている最新の暗号鍵を指定します。	

注意制限事項

なし

設定項目の反映タイミング

個々のセッション毎に設定可能です。設定値は即時セッションに反映されます。

実行例

次の例では本パラメータで設定した暗号鍵を使用して pg_dump を実行します。

```
$ PGOPTIONS="-c encrypt.cipherkey=key" pg_dump -f /pgsql/backup/tdedb.bak -Fc tdedb
```

第 8 章
開発サンプル

本章では、透過的暗号化機能を利用したテーブルを検索するサンプルコードについて説明します。なお、既存のデータベースに透過的暗号化機能を正常にセットアップして、データベースに対して暗号鍵が登録されているものとします。

8.1　環境構築

サンプルテーブル

サンプルテーブルを作成します。以下の「SampleTable.sql」で示すテーブル定義ファイルを作成し、psql コマンドの「-f」パラメータに作成したファイルを指定して実行してください。

- SampleTable.sql

```
--ログ出力無効化
select cipher_key_disable_log();
--セッション開始
select pgtde_begin_session('cipherkey');
--ログ出力有効化
select cipher_key_enable_log ();
--Employee テーブル作成
CREATE TABLE Employee(
  EmployeeID Integer PRIMARY KEY,
  Name TEXT,
  Address TEXT,
  TelephoneNumber TEXT
)USING tdeforpg2;
--セッション終了
select pgtde_end_session();
```

サンプルデータ

サンプルテーブルにサンプルデータを挿入します。以下の「DataModel.sql」で示すデータ挿入スクリプトファイルを作成し、psql コマンドの「-f」パラメータに作成したファイルを指定して実行してください。なお、下線で示している「cipherkey」には、データベースに設定した最新の暗号鍵を指定してください。

- DataModelA.sql

```
--ログ出力無効化
select cipher_key_disable_log();
--セッション開始
select pgtde_begin_session('cipherkey');
```

```
--ログ出力有効化
select cipher_key_enable_log ();
--サンプルデータ挿入
insert into Employee values(1,'従業員 1','滋賀','003-0001-0001');
insert into Employee values(2,'従業員 2','京都','003-0001-0002');
insert into Employee values(3,'従業員 3','大阪','003-0001-0003');
insert into Employee values(4,'従業員 4','兵庫','003-0001-0004');
insert into Employee values(5,'従業員 5','奈良','003-0001-0005');
insert into Employee values(6,'従業員 6','和歌山','003-0001-0006');
insert into Employee values(7,'従業員 7','鳥取','003-0002-0001');
insert into Employee values(8,'従業員 8','島根','003-0002-0002');
insert into Employee values(9,'従業員 9','岡山','003-0002-0003');
insert into Employee values(10,'従業員 10','広島','003-0002-0004');
insert into Employee values(11,'従業員 11','山口','003-0002-0005');
insert into Employee values(12,'従業員 12','徳島','003-0002-0006');
--セッション終了
select pgtde_end_session();
```

8.2 データ検索

サンプルコード(データ検索)

```
import java.sql.*;
public class sampleCipher {
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try{
            conn = DriverManager.getConnection(
                    "jdbc:postgresql://host:port/databaseName",
                    "userName", "password");
            stmt = conn.createStatement();
            //ログ出力無効化
            stmt.executeQuery("select cipher_key_disable_log()");
            //セッション開始
            stmt.executeQuery("select pgtde_begin_session('cipherkey')");
            //ログ出力有効化
            stmt.executeQuery("select cipher_key_enable_log()");
            //データの検索をする
            rs = stmt.executeQuery("select * from Employee");
            Integer count = rs.getMetaData().getColumnCount();
            while( rs.next() ){
                for( int i = 1 ; i <= count ; i++ ) {
                    System.out.print( rs.getObject(i) );
                    if( i != count )
                        System.out.print(" , ");
                }
                System.out.println();
            }
            rs.close();
            //アクセスを終了する。
            stmt.executeQuery("select pgtde_end_session()");
```

```
            stmt.close();
            conn.close();
        }catch(Exception e){
            // 例外処理
            System.err.println(e.getMessage());
        }
    }
}
```

注

下線で示している「host」、「port」、「databaseName」、「userName」、「password」については、それぞれの環境の値を指定してください。また、「cipherkey」にはデータベースで使用している最新の暗号鍵を指定してください。

サンプルコード(データ検索)の実行した結果を以下に示します。実行結果から、データが透過的に復号されていることを確認することができます。

- データ検索プログラム実行例

```
1 , 従業員 1 , 滋賀 , 003-0001-0001
2 , 従業員 2 , 京都 , 003-0001-0002
3 , 従業員 3 , 大阪 , 003-0001-0003
4 , 従業員 4 , 兵庫 , 003-0001-0004
5 , 従業員 5 , 奈良 , 003-0001-0005
6 , 従業員 6 , 和歌山 , 003-0001-0006
7 , 従業員 7 , 鳥取 , 003-0002-0001
8 , 従業員 8 , 島根 , 003-0002-0002
9 , 従業員 9 , 岡山 , 003-0002-0003
10 , 従業員 10 , 広島 , 003-0002-0004
11 , 従業員 11 , 山口 , 003-0002-0005
12 , 従業員 12 , 徳島 , 003-0002-0006
```

付録 A. 透過的暗号化機能で出力されるメッセージ

透過的暗号化機能実行時に出力されるメッセージには、コマンドを実行時に異常を検知すると出力されるメッセージと PostgreSQL が出力するメッセージがあります。

A.1 コマンドエラーメッセージ

コマンド実行時に出力されるメッセージには「FATAL」、「ERROR」、「WARN」の 3 つのレベルがあります。エラーメッセージ一覧の対処方法を参照し、原因を取り除いてください。

表 A-1 コマンドエラーメッセージ一覧

レベル	エラーコード	エラーメッセージ	対処方法
FATAL	F000	*その他のエラー*	メッセージを確認して、不明点があれば PP サポートサービスにご連絡ください。
FATAL	F001	UNDEFINED ERROR	システム内部に原因があるため、PP サポートサービスにご連絡ください。
FATAL	F002	ILLEGAL ERROR CODE :"*エラーコード*"	システム内部に原因があるため、PP サポートサービスにご連絡ください。
FATAL	F009	No console	システム内部に原因があるため、PP サポートサービスにご連絡ください。
FATAL	F016	cipher_key_table or key_management_table table does not exist.	システム内部に原因があるため、PP サポートサービスにご連絡ください。
FATAL	F999	INTERNAL ERROR. Please confirm pgtde.log and PostgreSQL server log for details.	内部処理でエラーが発生しています。pgtde.log および PostgreSQL サーバログに出力されているメッセージを含め、PP サポートサービスにご連絡ください。
ERROR	E002	Failed to connect to PostgreSQL	共通パラメータを確認してください。また、PostgreSQL 側が接続を受け付ける設定になっているか確認してください。
ERROR	E003	Given command's argument is incorrect. Try "--help" for more information	正しい引数を入力してください。--help パラメータで機能のヘルプ情報を参照することができます。
ERROR	E122	Invalid number format for "数値"	正しい数値を指定してください。
ERROR	E123	User name is not set	ユーザー名を入力してください。
ERROR	E125	Database name is not set	データベース名を入力してください。
ERROR	E126	Port number is not set	ポート番号を入力してください。
ERROR	E127	Host name is not set	ホスト名を入力してください。
ERROR	E136	Input mode is invalid	実行コマンドに対応したモードを入力してください。
ERROR	E137	Data key is not yet registed	暗号鍵の登録をしてください。
ERROR	E138	Mode is not input	モードを指定してください。
ERROR	E144	New data key does not match	もう一度新しい暗号鍵を入力してください。
ERROR	E145	Selected algorithm is invalid	メニューに表示されている項目を選択してください。
ERROR	E147	Data key's length must not be zero	暗号鍵は空文字以外を入力してください。
ERROR	E321	This feature is not implemented due to technical reason.	「-m cipher」は制限事項です。
ERROR	E709	The specified cipher key is not valid.	正しい暗号鍵を入力してください。

レベル	エラーコード	エラーメッセージ	対処方法
ERROR	E713	Invalid option. Can not use -conf and "オプション" options simultaneously.	-conf オプションと{-h|-p|-d|-pw|-U}は同時に指定しないでください。
ERROR	E714	Could not open "ファイル名" file specified in -conf option.	-conf で指定したファイルが存在し適切な権限が設定されているかどうか確認してください。
ERROR	E718	Selected key management mode is invalid.	選択可能な鍵管理方式を選択してください。
ERROR	E719	Could not open "ファイル名" file.	アクセス可能なファイルを指定してください。
ERROR	E720	Failed to register new data key.	発生条件と、出力されたメッセージを PP サポートサービスにご連絡ください。
ERROR	E722	Failed to reencrypt data.	発生条件と、出力されたメッセージを PP サポートサービスにご連絡ください。
ERROR	E723	Length of input must not be zero.	空文字以外を入力してください。
ERROR	E724	Could not write to file: "ファイル名"	フォルダの権限を確認してください。
ERROR	E727	Failed to switch key management mode.	発生条件と、出力されたメッセージを PP サポートサービスにご連絡ください。
ERROR	E728	Failed to show status of TDE.	発生条件と、出力されたメッセージを PP サポートサービスにご連絡ください。
ERROR	E729	Key management mode is already in "鍵管理方式名"	現在既に指定された鍵管理方式で運用されています。現在の設定を確認してください。

A.2 透過的暗号化機能により PostgreSQL が出力するメッセージ一覧

透過的暗号化機能利用時に PostgreSQL により出力されるメッセージは「ERROR」「WARNING」「INFO」「LOG」ログがあります。エラーメッセージ一覧の対処方法を参照し、原因を取り除いてください。

表 A-2 PostgreSQL が出力するエラーメッセージ一覧

レベル	エラーコード	エラーメッセージ	対処方法
ERROR	E0001	log_statement must not be 'all'	「log_statement」の設定を「all」以外に変更してください。
ERROR	E0002	new cipher key is invalid	暗号鍵に空文字は指定できません。登録可能な暗号鍵文字列を入力してください。
ERROR	E0003	invalid cipher algorithm "アルゴリズム名"	暗号化アルゴリズムは"aes"か"bf"のみサポートしています。いずれかの暗号化アルゴリズムを入力してください。
ERROR	E0007	encrypt key version over 2147483647 for database "*データベース名*"	暗号鍵を登録しようとしたデータベースの暗号鍵のバージョンが最大値に達しているため、PP サポートサービスへお問い合わせください。
ERROR	E0008	current cipher key is not correct	指定した暗号鍵が現在使用されている暗号鍵であることを確認してください。
ERROR	E0012	cipher key is not correct	引数に指定した暗号鍵が正しいことを確認してください。
ERROR	E0014	could not get data directory path	cipher_key_table テーブルのバックアップに失敗しました。適切な権限で関数を実行しているか確認してください。encrypt.backup パラメータにバックアップ先ファイルパスを指定して実行してください。
ERROR	E0015	could not rename old backup file of cipher key	暗号鍵テーブルのバックアップ時に、前回のバックアップファイルのリネームに失敗しました。暗号鍵バックアップファイルの出力先の権限設定を確認してください。

レベル	エラーコード	エラーメッセージ	対処方法
ERROR	E0016	could not encrypt data, because key was not set	暗号化テーブルに透過的にアクセスするため、事前にセッション開始関数を実行してください。
ERROR	E0017	could not decrypt data, because key was not set	暗号化テーブルに透過的にアクセスするため、事前にセッション開始関数を実行してください。
ERROR	E0037	Error while executing query in SPI mode.	SPI モードでクエリの実行に失敗しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0038	Data key is not yet registered.	データ鍵を登録してください。
ERROR	E0040	Could not create hash table for cipher key table.	SPI モードで鍵情報用ハッシュテーブルの作成に失敗しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0042	Error while setting key in simple TDE mode. Error code: %d	SPI モードで鍵情報用ハッシュテーブルの作成に失敗しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0043	Error while creating management key. Error code: %d	SPI モードで鍵情報用ハッシュテーブルの作成に失敗しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0044	Error while registering new key. Error code: %d	SPI モードで鍵情報用ハッシュテーブルの作成に失敗しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0045	Error while reencrypting cipher_key_table. Error code: %d	SPI モードで鍵情報用ハッシュテーブルの作成に失敗しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0046	could not find management key	簡易 TDE モードにおいて、鍵管理テーブル（key_management_table）にデータが無い状態で、暗号化対象テーブルにデータの挿入が行われました。発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0047	could not reset key info	暗号鍵のリセット処理の前に正常に既存の鍵情報が取得できません。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0048	could not encrypt data len, because key was not set	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0049	could not get cipher provider.	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0050	length of decrypted data is less than 0.	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0051	invalid encrypt key version.	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0052	invalid data.	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0053	could not encrypt data.	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0054	could not decrypt data.	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。

レベル	エラーコード	エラーメッセージ	対処方法
ERROR	E0055	Error while setting key in internal TDE mode. Error code: %d	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0056	encrypt error: %s	内部エラーが発生しました。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
ERROR	E0057	The input value of encrypt.cipherkey is incorrect.	encrypt.cipherkey パラメータの入力値が間違っています。設定した暗号鍵が正しいか確認してください。
ERROR	E0058	Failed to execute pgtde_begin_session with parameter encrypt.cipherkey. Error code: %s	encrypt.cipherkey パラメータを使用して pgtde_begin_session を実行できませんでした。再度実行しても失敗する場合は、発生条件とエラーメッセージを PP サポートサービスにご連絡ください。
WARNING	W0001	encrypt key version over 30000 for database "*データベース名*"	暗号鍵を登録しようとしたデータベースの暗号鍵のバージョンの最大値が 2100000000 を超えています。
WARNING	W0002	number of encrypt key version over 10 for database "*データベース名*"	pgtde_begin_session で取得した暗号鍵の数が 10 を超過しています。
WARNING	W0003	The current mode is simple TDE mode. encrypt_cipherkey can be used in standard TDE mode	encrypt.cipherkey パラメータは簡易 TDE モードの場合は設定不要です。

付録 B. 改訂履歴

本マニュアルの改訂履歴は以下のとおりです。

表 B-1 改訂履歴一覧

版数	発行日	改訂履歴
第一版	2021 年 4 月	• 初版作成

付録 C. ライセンス

本ソフトウェアで利用しているソフトウェアのライセンスについて記載します。

Transparent Data Encryption for PostgreSQL がバンドルしているソフトウェア、および使用しているオープンソースソフトウェアは以下となります。

表 C-1 ソフトウェア一覧

項番	ソフトウェア名称	ライセンス
1	PostgreSQL	PostgreSQL License
2	psql	PostgreSQL License
4	libpq	PostgreSQL License
5	The PostgreSQL JDBC Driver	BSD License
6	Apache log4j	Apache Software License Ver2.0
7	Apache Commons Logging™	Apache Software License Ver2.0
8	Apache Commons Lang™	Apache Software License Ver2.0
9	Apache Commons Codec™	Apache Software License Ver2.0
10	Apache HttpComponents™	Apache Software License Ver2.0
11	Jackson 1.x	Apache Software License Ver2.0
12	Jackson 2.x	Apache Software License Ver2.0
13	Joda-Time	Apache Software License Ver2.0

C.1 PostgreSQL ライセンス(PostgreSQL, psql, libpq)

```
    PostgreSQL Database Management System
    (formerly known as Postgres, then as Postgres95)

    Portions Copyright (c) 1996-2020, PostgreSQL Global Development Group

       Portions Copyright (c) 1994, The Regents of the University of
California

    Permission to use, copy, modify, and distribute this software and its
      documentation for any purpose, without fee, and without a written
agreement
    is hereby granted, provided that the above copyright notice and this
    paragraph and the following two paragraphs appear in all copies.

     IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY
FOR
       DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES,
INCLUDING
    LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS
     DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED
OF THE
```

C.2 The PostgreSQL JDBC Driver

The PostgreSQL JDBC Driver is distributed under the BSD-2-Clause License. The simplest explanation of the licensing terms is that you can do whatever you want with the product and source code as long as you don't claim you wrote it or sue us. You should give it a read though, it's only half a page.

C.3 Apache License Version 2.0(Apache Commons Logging, Apache commons Lang, Apache Commons Codec, Apache HttpComponents, Jackson 1.x, Jackson 2.x, Joda-Time)

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,

excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the

Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

＜ア行＞

■ 暗号化列

透過的暗号化機能が提供する属性を用いたユーザテーブルの列のことです。PostgreSQL のテーブル定義に暗号化列を利用することで透過的にデータを暗号化することが出来ます。

■ 暗号鍵

透過的暗号化機能でデータの暗号化に使用するための鍵で 透過的暗号化機能を開始するセッションを接続する際にこの鍵を利用します。

＜カ行＞

■ 簡易 TDE モード

TDE 機能を利用するにあたって接続時にデータ鍵を渡すことを不要とするモードのことです。本モードではセッションごとの関数実行が不要となり、クライアント側は暗号化機能を完全に意識せずに使用可能となります。

■ 行単位暗号化

V2.1 で実装される Table Access Method を利用した暗号化方式のことです。行単位暗号化方式では Table Access Method を利用した暗号化方式を採用しており、暗号化対象テーブルに対して暗号化をすることができます。

＜サ行＞

■ 削除タプル

PostgreSQL は追記型のアーキテクチャを採用しており、データを削除、または更新した場合は 元のデータに削除したフラグを立てて管理しています。 この削除タプル（レコード）は VACUUM によって実際に削除されるまではデータが残っています。

■ サルベージ

メンテナンス機能が提供するデータベースの救出機能のことです。データベースを停止した状態で データベースファイルから直接データをできる限り救出し、 そのデータを別の PostgreSQL データベースに復旧させることができます。

■ システムカタログ

システムカタログとは、リレーショナルデータベース管理システムが テーブルや列の情報などのスキーマメタデータと内部的な情報を格納する場所です。 PostgreSQL のシステムカタログは通常のテーブルに格納されています。 詳細については PostgreSQL のマニュアルをご参照ください。

＜タ行＞

■ データベース

本マニュアルでは PostgreSQL の CREATE DATABASE 構文で作成するデータオブジェクトのことを指します。

■ 透過的暗号化機能

Transparent Data Encryption for PostgreSQL の透過的暗号化機能全体を表す名称です。

＜ハ行＞

■ ブロック

PostgreSQL のデータベースをファイルに格納する最小の I/O 単位です。 デフォルトでは 8kB となっています。

■ 標準 TDE モード

透過的暗号化機能で暗号鍵を独自に管理し、セッションごとの関数実行が必要となるモードのことです。

＜ラ行＞

■ 列単位暗号化

列単位で暗号化する方式のことです。CREATE TABLE 文によるテーブル定義時に暗号化データ型を指定するだけでデータの暗号化をすることができます。

■ ローカル鍵管理

透過的暗号化機能で暗号鍵を独自に管理する方式のことです。

＜A-Z＞

■ heapam

Table Access Method を用いて既存の heap と同じ動作を行い、PostgreSQL12 で実装されているアクセスメソッドです。V2.1 は heapam を流用して実装しています。

■ Table Access Method

PostgreSQL12 の新機能であり、テーブルから行データを読み書きする動作を独自の実装に変更することができます。V2.1 の行単位暗号化ではこの機能をベースとしており、PostgreSQL に対する制約を低減します。

■ TOAST

過大属性格納技法：The Oversized-Attribute Storage Technique の略で、 PostgreSQL がページをまたがるサイズのタプル（レコード）を格納するための技法のことです。 一定のサイズを超えた データを圧縮または複数のレコードに分割して別のテーブル（ファイル）に管理します。 詳細については PostgreSQL のマニュアルをご参照ください。

マニュアルコメント用紙

読者各位

説明書に関するご意見、ご要望、内容不明確な部分について具体的にご記入のうえ、販売店または、当社担当営業、担当ＳＥにお渡しください。		お客様ご提出日	年　月　日
		〒 ご住所	〒
マニュアルコード	ＯＳＳＤＢＴＤＥ０８－０１	貴社名 所 属	
マニュアル名	Transparent Data Encryption for PostgreSQL 行単位暗号化 透過的暗号化機能 利用の手引	お名前	

項番	ﾍﾟｰｼﾞ	行・図番	指摘区分	指 摘 内 容	添付資料
1					

備考　指摘区分　１：誤り　２：誤字・脱字　３：難解　９：ご要望

ご協力ありがとうございます。

（注意）販売店員または、当社営業部員、ＳＥは、すみやかに所定の手続きに従ってマニュアル担当までお送りください。（メール：２２－Ａ０７０３）

なお、ＮＥＣメールがない場合は、下記まで郵送してください。

〒２１１－８６６６ 神奈川県川崎市中原区下沼部１７５３

日本電気(株) AI プラットフォーム事業部 SDP グループ宛

販売店員 営業部員 ＳＥ記入	販売店名 または 所属名		担当		ﾒｰﾙ TEL	

ＮＥＣ

Transparent Data Encryption for PostgreSQL
行単位暗号化 透過的暗号化機能 利用の手引

ＯＳＳＤＢＴＤＥ０８－０１

2021 年 04 月 第一版 発行

日本電気株式会社