NEC

Transparent Data Encryption for PostgreSQL

列単位暗号化 透過的暗号化機能 利用の手引

ＯＳＳＤＢＴＤＥ０２－１１

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。

注

Transparent Data Encryption for PostgreSQL V1.1.4 までのローカル鍵管理方式と AWS KMS[*1] 管理方式およびこれらの総称である鍵管理方式は、Transparent Data Encryption for PostgreSQL V1.2.0 よりそれぞれ標準 TDE モード、AWS KMS モード、およびこれらの総称をモードと名称が変更されています。Transparent Data Encryption for PostgreSQL V1.1.4 以前をご利用の際には適宜読み替えていただきますようお願いいたします。

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	64 ビット

*1　AWS Key Management Service の略称です。

プロダクト型番	プロダクト名	対応モデル
	Enterprise Edition V2.1 Linux 版 Cluster Option(3 年間)	
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
UL1298-H201-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU(1 年間)	64 ビット
UL1298-H202-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU 追加(1 年間)	64 ビット
UL1298-H203-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 待機用 1CPU(1 年間)	64 ビット
UL1298-H211-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU(3 年間)	64 ビット
UL1298-H212-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU 追加(3 年間)	64 ビット
UL1298-H213-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 待機用 1CPU(3 年間)	64 ビット
UL1298-J201-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU(1 年間)(時間延長保守)	64 ビット
UL1298-J202-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU 追加(1 年間)(時間延長保守)	64 ビット
UL1298-J203-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 待機用 1CPU(1 年間)(時間延長保守)	64 ビット
UL1298-J211-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU(3 年間)(時間延長保守)	64 ビット
UL1298-J212-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 1CPU 追加(3 年間)(時間延長保守)	64 ビット
UL1298-J213-I	Transparent Data Encryption for PostgreSQL Enterprise Edition V2.2 Windows 版 待機用 1CPU(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. Oracle、Oracle Linux は、Oracle Corporation およびその子会社，関連会社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。
5. Microsoft、Windows、Windows Server、Windows PowerShell は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
6. Amazon Web Services およびすべての AWS 関連の商標、ならびにその他の AWS のグラフィック、ロゴ、ページヘッダーボタンアイコン、スクリプト、サービス名は、米国

および／またはその他の国における、AWS の商標、登録商標またはトレードドレスです。

7. 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 の管理者ユーザーで発行するコマンドを示すプロンプトです	# `./cipher_setup.sh`
$	OS の一般ユーザー（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	tde_for_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 Edition ごとの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 6

2.5 暗号鍵のパスフレーズの検討 6

2.6 通信経路の暗号化 7

2.7 動作環境 7

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 8

2.7.1.2 ソフトウェア要件 8

2.7.2 クライアント 9

2.7.2.1 ハードウェア要件 9

2.7.2.2 ソフトウェア要件 9

第 3 章 透過的暗号化機能の導入 11

第 4 章 透過的暗号化機能の運用 12

4.1 透過的暗号化機能の管理 12

4.1.1 暗号鍵の登録・更新 12

4.1.2 利用するモードの変更 13

4.1.3 最新の暗号鍵によるデータ再暗号化 14

4.1.4 透過的暗号化機能の利用状況の表示 14

4.2 透過的暗号化機能の利用 15

4.2.1 暗号化データ型を含むテーブルの作成 15

4.2.2 暗号化/復号処理 16

4.3 暗号化データ型の仕様 19

4.3.1 暗号化データ型の注意制限事項 19

4.3.2 比較規則 20

4.3.3 キャスト 21
4.3.4 インデックス定義 22
4.4 AWS Key Management Service (AWS KMS)の利用 23
4.4.1 AWS KMS について 23
4.4.2 AWS KMS 利用時の注意事項 24
4.4.3 AWS KMS 利用のための準備(AWS KMS 設定) 24
4.4.4 AWS KMS 利用のための準備(透過的暗号化機能設定) 26
4.5 バックアップとリストア 28
4.5.1 バックアップ対象について 28
4.5.2 バックアップ手法 29
4.5.3 物理バックアップについて 30
4.5.4 論理バックアップについて 30
4.5.5 COPY コマンドについて 31
第 5 章 コマンドリファレンス 34
5.1 透過的暗号化機能コマンド（pgtde） 34
5.1.1 暗号鍵の登録・更新（-m regist） 40
5.1.2 透過的暗号化機能の利用モードの変更（-m switch） 46
5.1.3 最新の暗号鍵によるデータ再暗号化（-m cipher） 52
5.1.4 透過的暗号化機能の利用状況の表示（-m show） 55
第 6 章 ファンクションリファレンス 58
6.1 ログ出力無効化 （cipher_key_disable_log） 59
6.2 セッション開始（pgtde_begin_session） 60
6.3 ログ出力有効化（cipher_key_enable_log） 62
6.4 セッション終了（pgtde_end_session） 63
6.5 暗号鍵バックアップ（cipher_key_backup） 65
第 7 章 パラメータリファレンス 67
7.1 暗号化/復号制御（encrypt.enable） 68
7.2 バックアップファイル出力先設定（encrypt.backup） 69
7.3 進捗ログ出力設定（encrypt.logoutpernum） 70
7.4 暗号化データ型比較制御（encrypt.diff_version） 71
7.5 AWS KMS 復号処理タイムアウト（encrypt.kms_timeout） 72
7.6 透過的暗号化機能ライブラリパス（encrypt.tde_rootpath） 73
第 8 章 開発サンプル 75

8.1 環境構築 75
8.2 データ検索 76
付録 A. 透過的暗号化機能で出力されるメッセージ 78
A.1 コマンドエラーメッセージ 78
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 80
付録 B. 改訂履歴 83
付録 C. ライセンス 85
C.1 PostgreSQL ライセンス(PostgreSQL, psql, pgcrypto, libpq) 85
C.2 The PostgreSQL JDBC Driver 86
C.3 Apache License Version 2.0(AWS SDK for Java, Apache Commons Logging, Apache commons Lang, Apache Commons Codec, Apache HttpComponents, Jackson 1.x, Jackson 2.x, Joda-Time) 87

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、Windows および 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ー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。

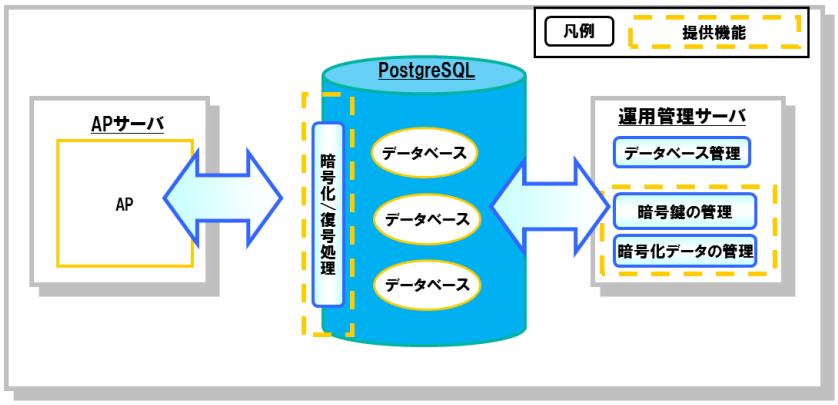

図 1-1　システム構成図

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	ユーザーが暗号化データ型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　Edition ごとの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 と OSS として公開している Free Edition があります。各 Edition で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　Edition による機能/サービスの違い

機能/サービス		Enterprise Edition for Linux	Enterprise Edition for Windows	Free Edition
Transparent Data Encryption 機能				
列単位の暗号化機能	テキスト	○	○	○
	バイト列（画像など）	○	○	○
	NUMERIC	○	○	×
	整数型（smallint,integer,bigint）	○	○	×
	日付・時刻	○	○	×
鍵の更新、バージョン管理機能		○	○	△*1
AWS Key Management Service を利用した鍵管理		○	×	×
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		○	○	×

*1　暗号鍵のバージョン管理機能なし。一括更新のみ可

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『列単位暗号化 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（6 ページ）」
3. 「2.5　暗号鍵のパスフレーズの検討（6 ページ）」
4. 「2.6　通信経路の暗号化（7 ページ）」
5. 「4.1.1　暗号鍵の登録・更新（12 ページ）」
6. 「4.2.1　暗号化データ型を含むテーブルの作成（15 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。
	スーパーユーザー名	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ーユーザーです。
	スーパーユーザーのパスワード	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ーユーザーのパスワードです。
	セキュリティ管理ユーザー名	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ーです。
	セキュリティ管理ユーザーのパスワード	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ーのパスワードです
	アプリケーション管理ユーザー名	透過的暗号化機能を利用しているユーザーデータに対する暗号化・復号権限を持つユーザーです。
	アプリケーション管理ユーザーのパスワード	透過的暗号化機能を利用しているユーザーデータに対する暗号化・復号権限を持つユーザーのパスワードです。

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 3 つのモード（簡易 TDE モード、標準 TDE モード、AWS KMS 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ー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、AWS KMS モードの 3 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。

ヒント

Transparent Data Encryption for PostgreSQL V1.1.4 までは、標準 TDE モードをローカル鍵管理方式、AWS KMS モードを AWS KMS 管理方式と表記していました。またこれらの総称として鍵管理方式と表記していましたが、Transparent Data Encryption for PostgreSQL V1.2.0 より、総称をモードに変更しています。

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	AWS KMS 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	必要
鍵管理のユーザー独自設計要否	必要	必要	不要
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	低い
セッション開始にかかる時間	速い	速い	遅い
鍵管理コマンド実行時間	速い	速い	遅い
データベースサーバーから外部へのアクセス許可	不要	不要	必要

- 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 モードおよび AWS KMS 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、ユーザー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ー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。

- 鍵管理のユーザー独自設計要否

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。AWS KMS モードでは暗号鍵を AWS Key Management Service で管理するため、管理者は暗号鍵の管理から解放されます。

- 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。AWS KMS モードでは透過的暗号化機能利用者は、データを暗号化するための鍵を CMK で暗号化したものを暗号鍵として使用します。これによって透過的暗号化機能利用者は直接暗号鍵情報を参照できなくなるため、鍵情報漏洩の可能性は低くなります。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ーに格納しています。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。

- セッション開始にかかる時間

  AWS KMS モードでは、セッション開始の際、都度 AWS KMS にアクセスして暗号鍵を復号する必要が生じるため、セッション開始にかかる時間が AWS KMS モードを利用しない場合と比べて遅くなります。

- 鍵管理コマンド実行時間

  AWS KMS モードでは、鍵管理コマンドの実行の都度 AWS KMS にアクセスする必要が生じるため、AWS KMS モードを利用しない場合と比べて遅くなります。

- データベースサーバーから外部へのアクセス許可

  前述の通り、AWS KMS モードでは AWS KMS にアクセスする必要があるため、データベースサーバーからインターネットを介した AWS KMS へのアクセス許可が必要です。その他のモードでは不要です。

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、『列単位暗号化 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。

ヒント

AWS KMS モードを利用する場合は「4.4　AWS Key Management Service (AWS KMS)の利用（23 ページ）」をご確認ください。

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	
		ENCRYPT_TIMESTAMP および ENCRYPT_INTEGER 以外	ENCRYPT_TIMESTAMP および ENCRYPT_INTEGER
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。	((<*元データサイズ*>/16)+1)*16+2+<長さ情報※> ※元データサイズが 111 バイトまでは 1 バイト、112 バイト以上は 4 バイトとなります。	19 バイトもしくは 35 バイト
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。	((<*元データサイズ*>/8)+1)*8+2+<長さ情報※> ※元データサイズが 119 バイトまでは 1 バイト、120 バイト以上は 4 バイトとなります。	19 バイトもしくは 27 バイト

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ー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
  - ロジカルデコーディング

    Windows 版 TDEforPG ではサポート対象外です。Linux 版ではサポートしますが、PostgreSQL 10.3 まで PostgreSQL の不具合により'cache lookup failed for type' エラーが発生する可能性があります。既に修正パッチがコミュニティにコミットされており PostgreSQL 10.4 以降では本エラーは発生しません。
  - バックグラウンドワーカプロセス
  - サーバープログラミングインタフェース
  - バイナリー書式モード
  - パラレルクエリ（PostgreSQL 9.6 以降）
  - CREATE STATISTICS（PostgreSQL 10 以降）
  - 宣言的 Partitioning（PostgreSQL 10 以降）
  - JIT コンパイル（PostgreSQL 11 以降）
- 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
- Windows では Transparent Data Encryption for PostgreSQL V2.2.0 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 および Oracle Linux では通常 OpenSSL 1.0.2 系（RHEL7、OL7）または OpenSSL 1.1.1 系（RHEL8、OL8）がインストールされています。
  - コミュニティ推奨 Windows インストーラーによりインストールされた PostgreSQL を利用します。（同梱されている OpenSSL を利用するため）

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 Oracle Linux 7.1 以上	Red Hat Enterprise Linux 8.1 以上 Oracle Linux 8.1 以上
12	○	○
13	○	○
必要パッケージ（Linux）	zlib.x86_64 glibc.x86_64	

PostgreSQL バージョン	オペレーティングシステム（Windows）	
	Windows Server 2016	Windows Server 2019
12	○	○
13	○	○
必要パッケージ（Windows）	Microsoft Visual C++ 2015-2022 Redistributable(x64) OpenJDK Version 8(JRE 8)、または Java™ Platform, Standard Edition Runtime Environment Version 8	

重要

Windows プラットフォームではコミュニティが推奨している Windows インストーラーからインストールされた PostgreSQL のみをサポートします。https://www.postgresql.org/download/windows/

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 ドライバーや Transparent Data Encryption for PostgreSQL 対応 ODBC ドライバーを利用する場合は 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 Oracle Linux 7.1 以上	Red Hat Enterprise Linux 8.1 以上 Oracle Linux 8.1 以上	Windows Server 2016	Windows Server 2019
12	○	○	○	○
13	○	○	○	○
対応アプリケーションプログラムインターフェース	• libpq※1 • PostgreSQL JDBC Driver • psqlODBC			

注

※1

Transparent Data Encryption for PostgreSQL はバイナリー書式モードをサポートしないため、libpq が提供する以下の API を利用する際に、テキスト書式モードを paramFormats 引数にて設定してください。

- PQsendQuery
- PQsendQueryParams
- PQsendQueryPrepared
- PQexecParams
- PQexecPrepared

PostgreSQL JDBC Driver

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 ドライバーをご利用ください。このドライバーでは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。非暗号化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 ドライバー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 ドライバーファイルが格納されています。これら JDBC ドライバー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-42.4.1.1.jar` [JDBC42（JDK1.8）ドライバー]

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

O/R マッパ名	バージョン
MyBatis	3.3
Hibernate ORM	5.0

psqlODBC

注

Transparent Data Encryption for PostgreSQL 対応 ODBC ドライバーは 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 ドライバーをご利用ください。このドライバーでは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。非暗号化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 ドライバー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 ドライバーファイルが格納されています。これら ODBC ドライバー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_13.02.0001_x64.msi` [Windows 対応 psqlODBC 64 ビット]
- `tdeforpg_psqlodbc_13.02.0001_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『列単位暗号化 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、暗号化データ型に暗号化/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）（40 ページ）」をご確認ください

重要

- 標準 TDE モードや AWS KMS 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ー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）（65 ページ）」をご確認ください。

注

- 暗号鍵登録・更新に何らかの原因で失敗した場合、無効な暗号鍵ファイルが生成される可能性があります。この暗号鍵ファイルを残していてもセキュリティ上問題はありませんが、必要に応じて削除されることを推奨いたします。
- 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。

- 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。
- 空文字の暗号鍵を登録することはできません。

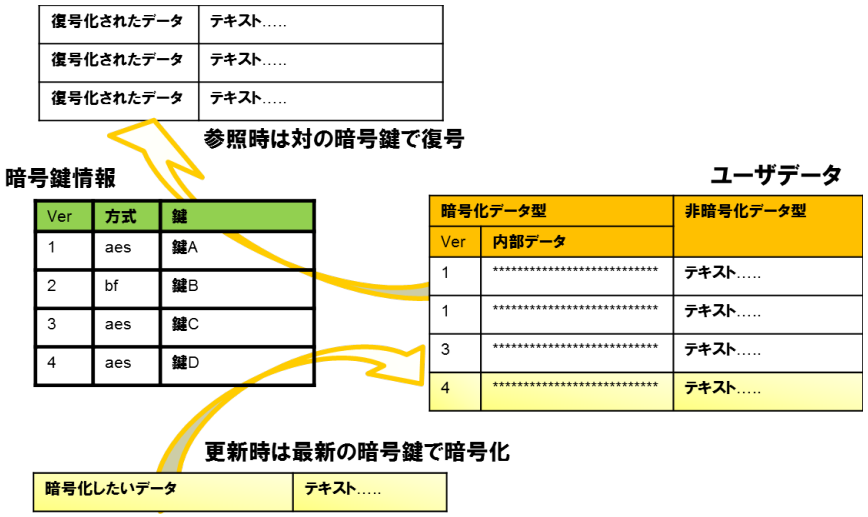

図 4-1　暗号鍵のバージョン管理

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）（46 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。

注

- 簡易 TDE モードまたは標準 TDE モードから AWS KMS モードに切り替える際に、何らかの原因で失敗した場合、無効な暗号鍵ファイルが生成される可能性があります。この暗号鍵ファイルを残していてもセキュリティ上問題はありませんが、必要に応じて削除されることを推奨いたします。
- 簡易 TDE モードまたは標準 TDE モードから AWS KMS モードに切り替えた場合、自動的に新しい暗号鍵が透過的暗号化機能データベースに登録されます。AWS KMS モードから簡易 TDE モードまたは標準 TDE モードに切り替える場合は新しい暗号鍵は登録されません。ま

た、簡易 TDE モードと標準 TDE モードを切り替えた場合も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。

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ーデータを最新の暗号鍵バージョンに統一するために、透過的暗号化機能で提供する `pgtde` コマンド`-m cipher` オプションを使用します。また、`--reset` パラメータを使用することで再暗号化により不要となった暗号鍵を削除することも可能です。詳細は「5.1.3　最新の暗号鍵によるデータ再暗号化（-m cipher）（52 ページ）」をご確認ください。

注

- 再暗号化を実行する際、暗号化列を定義したテーブルのルールまたはトリガは削除または無効化(ALTER TABLE … DISABLE RULE …)する必要があります。ルールまたはトリガが有効な状態では、再暗号化処理中にルールに定義した処理が実行されます。
- 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。
- 「`--reset`」パラメータを指定して再暗号化を実行している場合は、暗号鍵の登録・更新・削除等の操作は実行されず、再暗号化の完了を待ちます。
- 「`--reset`」パラメータを指定して再暗号化を実行する前から接続を開始している場合は、再暗号化処理完了後に再接続する必要があります。
- 「`--reset`」パラメータを指定して再暗号化を実行した後は、暗号化データ型を定義した全てのユーザーテーブルに ANALYZE を実施していただく必要があります。なお、ANALYZE の実施には対象のテーブルを所有する権限が必要となります。
- 暗号化列を持つ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や AWS アクセス情報(Access Key ID,Secret Access Key)を入

力することで AWS KMS の登録状況を確認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）（55 ページ）」をご確認ください

注

表示される CMK の作成日付はシステムのタイムゾーンに合わせて表示されます。

4.2　透過的暗号化機能の利用

4.2.1　暗号化データ型を含むテーブルの作成

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暗号化データ型を指定します。暗号化データ型とは、透過的暗号化機能が提供する暗号化/復号処理を行うための専用のデータ型のことです。暗号化したいデータに暗号化データ型を使用することで、データを暗号化した状態でデータベースに格納することができます。なお、暗号化データ型を利用する際には後述の「4.3　暗号化データ型の仕様（19 ページ）」をあわせてご確認ください。

透過的暗号化機能では、暗号化データ型として「ENCRYPT_TEXT」、「ENCRYPT_BYTEA」、「ENCRYPT_NUMERIC」、「ENCRYPT_INTEGER」、「ENCRYPT_TIMESTAMP」 を提供します。

表 4-1　暗号化列のデータ型

データ型名	概要
ENCRYPT_TEXT	TEXT 型データを暗号化／復号して格納するためのデータ型
ENCRYPT_BYTEA	BYTEA 型データを暗号化／復号して格納するためのデータ型
ENCRYPT_NUMERIC	NUMERIC 型データを暗号化／復号して格納するためのデータ型
ENCRYPT_INTEGER	整数型（smallint、integer、bigint）データを暗号化／復号して格納するためのデータ型
ENCRYPT_TIMESTAMP	TIMESTAMP WITHOUT TIMEZONE 型データを暗号化/復号して格納するためのデータ型

注

暗号化データ型を以下のオブジェクト、制約、およびルールで使用することはサポートしていません。

- 一時テーブル(TEMP TABLE)
- 外部データ(FOREIGN TABLE)
- 主キー制約(PRIMARY KEY)
- 一意性制約(UNIQUE)
- 検査制約(CHECK)

- デフォルト値(DEFAULT)
- 排他制約(EXCLUDE)
- 外部キー制約(REFERENCES)
- ルール(RULE)
- トリガ(TRIGGER)

4.2.2　暗号化/復号処理

暗号化データ型を透過的に暗号化/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データ型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）（59 ページ）」
- 「6.2　セッション開始（pgtde_begin_session）（60 ページ）」
- 「6.3　ログ出力有効化（cipher_key_enable_log）（62 ページ）」
- 「6.4　セッション終了（pgtde_end_session）（63 ページ）」

透過的暗号化機能では 3 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、ユーザー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 モードと AWS KMS 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。

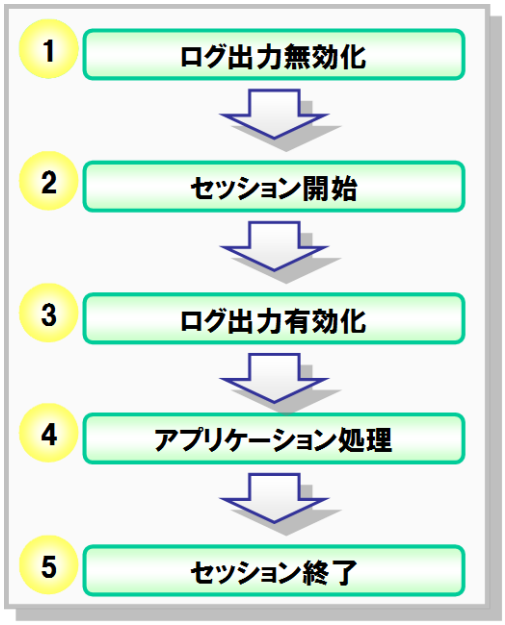

図 4-2　標準 TDE モードと AWSKMS モードにおける透過的暗号化セッション利用方法

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 モードと AWS KMS 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、暗号化データ型に挿入/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。なおバックアップやリストア目的で暗号化されたままのデータを取得したい場合には、透過的暗号化機能で提供する「暗号化/復号制御(encrypt.enable=OFF)」パラメータを利用します。

次の例では暗号化/復号処理を行う場合(ON)とバックアップやリストア目的で暗号化/復号処理を行わない場合(OFF)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

テーブルの一つに、透過的暗号化機能で提供する暗号化データ型の「Code」列が定義してあります。

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。

- 「7.1　暗号化/復号制御（encrypt.enable）（68 ページ）」
- 「4.5　バックアップとリストア（28 ページ）」

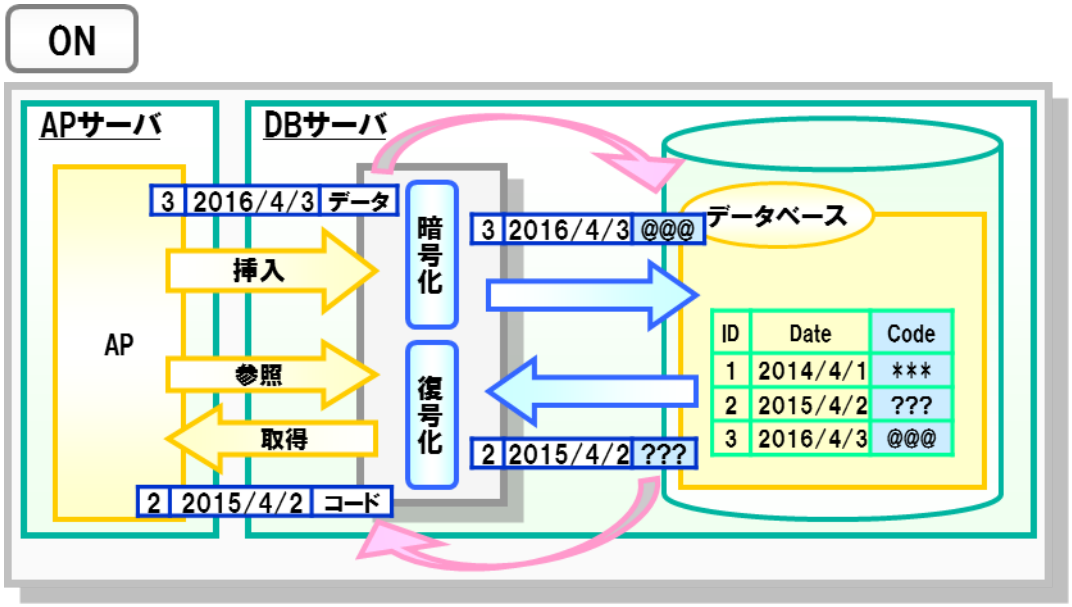

図 4-3　暗号化/復号処理を行う場合(ON)

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「Code」列を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「Code」列を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。

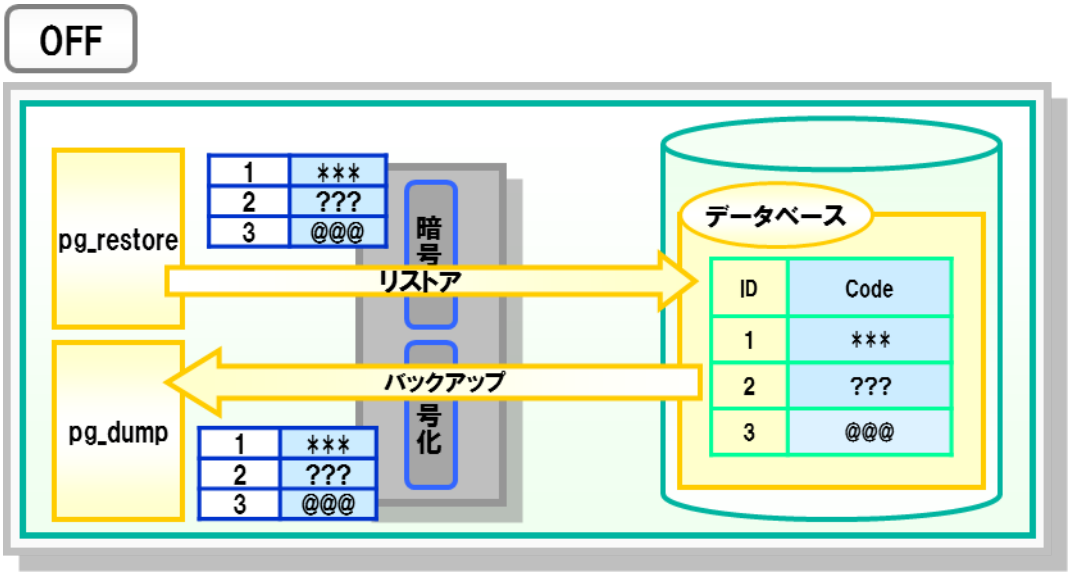

図 4-4　暗号化/復号処理を行わない場合(OFF)

4.3　暗号化データ型の仕様

本章では、透過的暗号化機能で提供する暗号化データ型の仕様動作について説明します。

4.3.1　暗号化データ型の注意制限事項

暗号化データ型の比較規則、キャスト、インデックス定義以外の注意制限事項を記載します。

注

- EXPLAIN ANALYZE に関する注意事項

  ANALYZE オプション付きの EXPLAIN は、実際に SQL を実行して実行時間等を取得しますが、カラムをそのまま出力する SQL の場合、データ型の出力処理がスキップされます。そのため、暗号化データ型を含むテーブルの場合、復号処理のオーバヘッドが含まれずに、実際の実行時間とは異なってしまう可能性があります。

  次の例では 、t1 テーブルの暗号化データ型の c1 列を SELECT を実行していますが、復号処理がスキップされます。

```
EXPLAIN ANALYZE SELECT c1 FROM t1;
```

  次の例では、t1 テーブルの暗号化データ型の c1 列に対して substring 関数を実行することで復号処理が実行されます

```
EXPLAIN ANALYZE SELECT substring(c1, 1, 3) FROM t1;
```

  → substring 関数への適用のため復号処理が実行されます。

- search_path に関する注意事項

透過的暗号化機能関連オブジェクトは public スキーマに定義されるため、search_path パラメータのデフォルト値を変更する場合は、public スキーマを含めるように設定してください。

- 行単位のセキュリティポリシーの定義に関する注意事項

  CREATE POLICY 構文の USING 句に指定する検査式の文字列は、pg_policies システムカタログに記録されます。暗号化データ型と比較するリテラル文字列は暗号化されずに保存されますのでご注意ください。

  (例)

```
CREATE POLICY a_user_select ON test_acl FOR SELECT TO a_user USING ('A' = AN
Y(acl));
=> select qual from pg_policies where tablename = 'test_acl' limit 1;
              qual
--------------------------------
 ('A'::encrypt_text = ANY (acl))
(1 row)
```

- psql に関するの注意事項

  psql で ENCRYPT_INTEGER、ENCRYPT_NUMERIC のデータを表示する際に、対応するデータ型と異なり left alignment（左寄せ）で表示されます。

4.3.2　比較規則

暗号化データ型の比較処理は復号せずに比較できる場合を除き、非暗号化データ型への暗黙的なキャストを利用することにより復号してから比較を行います。

次に暗号化データ型の比較処理仕様一覧を示します。

表 4-2　暗号化データ型比較処理仕様

暗号化データ型	比較対象データ型	比較処理方法
ENCRYPT_TEXT ENCRYPT_BYTEA ENCRYPT_NUMERIC ENCRYPT_TIMESTAMP ENCRYPT_INTEGER	リテラル値	等価比較の場合、リテラル値をそれぞれ対応する暗号化データ型にキャストした上で比較します。ただし、暗号鍵のバージョンが複数ある場合は注意事項をご確認ください。大小比較の場合、それぞれのデータ型を非暗号化データ型にキャストした上で比較します。
ENCRYPT_TEXT ENCRYPT_BYTEA ENCRYPT_NUMERIC ENCRYPT_TIMESTAMP ENCRYPT_INTEGER	ENCRYPT_TEXT ENCRYPT_BYTEA ENCRYPT_NUMERIC ENCRYPT_TIMESTAMP ENCRYPT_INTEGER	等価比較の場合、暗号化データ型のまま比較します。ただし、暗号鍵のバージョンが複数ある場合は注意事項をご確認ください。大小比較の場合、それぞれのデータ型を非暗号化データ型にキャストした上で比較します。
ENCRYPT_TEXT	CHARACTER VARCHAR TEXT NAME	それぞれのデータ型を TEXT データ型にキャストした上で比較します。
ENCRYPT_BYTEA	BYTEA	ENCRYPT_BYTEA データ型を BYTEA 型にキャストした上で比較します。
ENCRYPT_NUMERIC	NUMERIC	ENCRYPT_NUMERIC データ型を NUMERIC 型にキャストした上で比較します。
ENCRYPT_NUMERIC	FLOAT4 FLOAT8	ENCRYPT_NUMERIC データ型を FLOAT8 型にキャストした上で比較します。
ENCRYPT_TIMESTAMP	TIMESTAMP DATE	ENCRYPT_TIMESTAMP を TIMESTAMP データ型にキャストした上で比較します。

暗号化データ型	比較対象データ型	比較処理方法
ENCRYPT_INTEGER	INT2（SMALLINT） INT4（INTEGER） INT8（BIGINT）	ENCRYPT_INTEGER を INT8（BIGINT）型にキャストした上で比較します。

暗号鍵のバージョンが複数ある場合の注意事項

暗号化された状態でデータを比較するため、復号すればデータが一致していたとしても暗号化に別のバージョンの暗号鍵を使用していると異なるデータとして処理されてしまいます。この動作を変更するためには、透過的暗号化機能の動作を制御するカスタムパラメータ「diff_version」を利用します。詳細は「暗号化データ型比較制御（71 ページ）」をご確認ください。

4.3.3　キャスト

「4.3.2　比較規則（20 ページ）」で示しているとおり、透過的暗号化機能で提供する暗号化データ型と非暗号化データを比較するには、キャストする必要があります。透過的暗号化機能では非暗号化データ型から暗号化データ型へのキャストに加え、暗号化データ型から非暗号化データ型へのキャストもサポートしています。サポートするキャストには暗黙キャストと代入キャストがあります。代入キャストの場合は、非暗号化データ型と暗号化データ型の演算子を用いた評価の際、明示的なキャストが必要になる点ご注意ください。暗黙的なキャストを使用することにより、アプリケーションなどはキャストを意識せずに、非暗号化データ型と暗号化データ型との比較、評価を行うことが可能です。

注

- 以下の場合、ENCRYPT_NUMERIC と NUMERIC では動作が異なります。
  - ENCRYPT_NUMERIC は、sum()、min()、max()以外の集計関数は暗黙的に DOUBLE PRECISION にキャストして動作することがあり、NUMERIC と同様に動作させる場合には、明示的なキャストが必要です。
  - PostgreSQL の算術関数 ceil()、ceiling()、exp()、floor()、ln()、log()、round()、sign()、sqrt()、trunc()、avg()のように DOUBLE PRECISION、NUMERIC の両方を引数として取ることができる場合、ENCRYPT_NUMERIC は DOUBLE PRECISION にキャストして動作することがあり、NUMERIC と同様に動作させる場合には、明示的なキャストが必要です。
- 以下の場合、ENCRYPT_INTEGER と INT2（SMALLINT）、INT4（INTEGER）INT8（BIGINT)では動作が異なります。
  - ENCRYPT_INTEGER は、sum()、min()、max()以外の集計関数は暗黙的に DOUBLE PRECISION にキャストして動作することがあり、INT2（SMALLINT）、INT4（INTEGER）、INT8（BIGINT)と同様の動作にするには、明示的なキャストが必要です。
- 以下の場合、ENCRYPT_TIMESTAMP と TIMESTAMP では動作が異なります。

  - PostgreSQL の日付/時刻関数 age()のように TIMESTAMP、TIMESTAMPTZ の両方に解析される可能性がある関数の場合、日時定数と ENCRYPT_TIMESTAMP を同時に引数として使用すると TIMESTAMPTZ にキャストされて演算されます。

- TIMESTAMP の格納サイズがデフォルトの 8 バイト整数以外でコンパイルされている場合、ENCRYPT_TIMESTAMP は利用できません。
- GROUPING SETS、CUBE、ROLLUP を使用して複数のグループ化セットを作成し、暗号化データ型の列をその対象列に指定する際には、非暗号化データ型に明示的なキャストが必要です。これはグループ化セットの演算でソートを利用したグループ化が行われる場合があり、暗号化データ型がソート処理をサポートしていないため、当該処理を行えないことに起因しています。この条件に合致する場合、以下のエラーが出力されます。

```
ERROR:  operator 0 is not a valid ordering operator
```

  なお、PostgreSQL 10 の不具合により、GROUPING SETS 句に暗号化データ型の列(2 つ以上)のみを指定した場合、セグメンテーションフォルトが発生します。既に修正パッチはコミュニティにコミットされているため、PostgreSQL 10.4 以降は本問題を回避することができる見込みです。

- UNION、INTERSECT [ALL]、EXCEPT [ALL]、DISTINCT、GROUP BY 句の対象列に、record、money、varbit、tid、tsquery、tinterval、bit、tsvector データ型と 暗号化データ型を一緒に扱うことはできません。一緒に扱うためには、暗号化データ型を非暗号化データ型に明示的なキャストが必要です。これは暗号化データ型がハッシュ処理のみをサポートしており、上記データ型がハッシュ処理をサポートしていないため、上記キーワードによる処理が行えないことに起因しています。 例えば GROUP BY 句において上記条件に合致する場合、以下のエラーが出力されます。

```
ERROR: could not implement GROUP BY
DETAIL: Some of the datatypes only support hashing, while others only suppor
t sorting.
```

- abs()、max()、min()、mod()、sum()、to_char()以外の関数は、INT2（SMALLINT）、INT4（INTEGER）の引数に ENCRYPT_INTEGER データ型を指定する場合、明示的に INT2（SMALLINT）または INT4（INTEGER）にキャストしてください。
- 暗号化データ型の列を ORDER BY、PARTITION BY の対象列に指定する際には、非暗号化データ型への明示的なキャストが必要となります。

4.3.4　インデックス定義

暗号化データ型のインデックス定義には、ハッシュインデックスのみ使用できます。また、完全一致検索のみをサポートしています。

注

暗号化データ型には以下の注意事項があります。

- 暗号鍵のバージョンの異なるデータはハッシュ値が異なるため、暗号鍵のバージョンが統一されていない場合、インデックスを介したデータの抽出ができない可能性があります。
- 暗号化データ型の列を使用した式インデックスや部分インデックスはサポートしていません。

4.4　AWS Key Management Service (AWS KMS)の利用

本章では、AWS Key Management Service (AWS KMS)を使ったモードについて記載いたします。

4.4.1　AWS KMS について

AWS Key Management Service (AWS KMS) とは、Amazon.com,Inc.が提供する暗号鍵を作成および管理できるマネージドサービスです。AWS KMS を Transparent Data Encryption for PostgreSQL から利用することで、透過的暗号化機能で利用する暗号鍵管理のための開発・運用コストを低減し、透過的暗号化機能を利用したデータベースのよりセキュアな運用を容易に実現できます。

AWS Key Management Service については以下の Web ページをご覧ください。(2015 年 7 月時点)

- AWS Key Management Service とは

AWS KMS モードを利用した場合のシステム構成図は以下のとおりです。

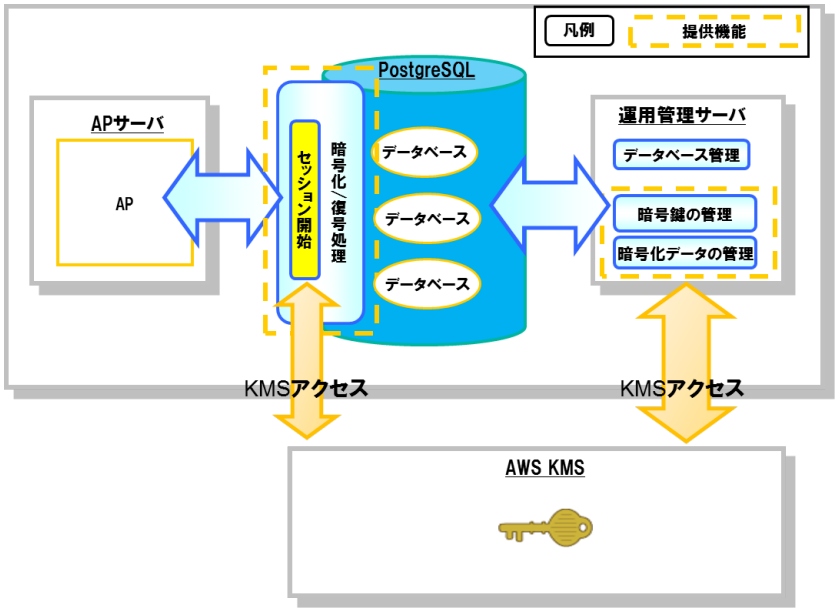

図 4-5　AWS KMS モード利用時の透過的暗号化機能システム構成図

4.4.2　AWS KMS 利用時の注意事項

AWS KMS 利用時には以下の注意事項があります。

注

- AWS KMS 利用の前提条件

  AWS KMS 管理方式を使って透過的暗号化機能をセキュアに運用するためには、『列単位暗号化 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」を参考に各種設定を行っている必要があります。また AWS KMS モードでは、『列単位暗号化 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」で作成した各 OS ユーザーそれぞれが AWS KMS モードで利用する各セキュリティ情報に対しての参照ルールを守る必要があります。

表 4-3　AWS KMS 利用時の各 OS ユーザーのセキュリティ情報参照ルール

セキュリティ情報	データベース管理者	セキュリティ管理者	アプリケーション管理者（アプリケーション開発者）
KeyId	○	○	×
透過的暗号化機能の暗号鍵	×	○	○
AWS KMS へのアクセス情報	○	○	×

- AWS KMS 処理中のキャンセル

  透過的暗号化機能が KMS と通信中にユーザーからのキャンセル要求を受け付けると、以下のエラーメッセージが出力される場合があります。

```
ERROR:Child process did not exit properly.
```

- 同時に使用可能な AWS リージョン数

  一つの透過的暗号化機能で同時に複数の AWS リージョンを使用することはできません。

- AWS KMS 利用にかかる料金

  AWS KMS を利用した場合、AWS KMS の各オペレーションごとに料金が発生します。具体的な料金については AWS KMS の料金表をご確認ください。

- AWS KMS 利用のために必要となる透過的暗号化機能設定ファイルに関する注意事項

  kms_info.properties ファイルは当該サーバーにログインできる任意のユーザーから参照可能となる仕様になっています。

- 使用中の CMK に関する注意事項

  使用している CMK が削除された場合、その CMK を使って暗号化されたデータは復号できなくなります。

4.4.3　AWS KMS 利用のための準備(AWS KMS 設定)

透過的暗号化機能で AWS KMS モードを利用する場合は、まずは AWS KMS 上で Customer Master Key（CMK）と呼ばれるマスターキーを作成していただく必要があります。

AWS KMS の利用方法については以下の Web ページをご覧ください。（2018 年 4 月時点)

ご利用開始にあたって

CMK の作成方法については以下の Web ページをご覧ください。(2018 年 4 月時点)

キーの作成

作成したキーは AWS Identity and Access Management （IAM）から以下のようなイメージで確認することができます。

図 4-6　IAM 上での CMK リスト一覧(イメージ)

透過的暗号化機能では、AWS KMS を鍵管理サーバーとして使用する際に作成した CMK の上図「キー ID」列に表示されている値を「KeyId」として使用します。

また、透過的暗号化機能では、セキュリティ管理をより強固とするため、透過的暗号化機能を利用するユーザー毎に以下の IAM ポリシーを割り当てる必要があります。なお、アプリケーション管理者は AWS KMS に直接アクセスする必要はなく、対応する IAM ユーザーを作成する必要もありません。各ユーザー種別については『列単位暗号化 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」で作成した OS ユーザーにそれぞれ対応しています。

表 4-4　透過的暗号化機能利用ユーザー毎の必要な IAM ポリシー

ユーザー種別	必要な IAM ポリシー
セキュリティ管理者	Decrypt,DescribeKey,GenerateDataKey,GetKeyRotationStatus,List*
データベース管理者	Decrypt
アプリケーション管理者	なし

IAM ユーザーへの IAM ポリシーの割り当てについては以下の Web ページをご覧ください (2015 年 7 月時点)

IAM ポリシーの概要

各ユーザー毎の IAM ポリシー設定例を以下に記載いたします。

セキュリティ管理者用 IAM ポリシー例

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1435558550000",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:GetKeyRotationStatus",
                "kms:List*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

DB 管理者用 IAM ポリシー例

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1435550586000",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

4.4.4　AWS KMS 利用のための準備(透過的暗号化機能設定)

AWS KMS を透過的暗号化機能で利用するためには、以下の設定ファイルに必要な情報を記載する必要があります。設定ファイルはインストールディレクトリの conf 配下に格納されています。

表 4-5　AWS KMS 利用のために必要となる透過的暗号化機能設定ファイル

設定ファイル名	概要
kms_info.properties	AWS KMS エンドポイント接続情報設定ファイル
aws_info.properties	データベース管理者ユーザーの IAM アクセス情報設定ファイル

注

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、プライマリサーバーとスタンバイサーバーの両方に設定を行う必要があります。

以下に各設定ファイル毎の説明と設定例を記載いたします。

表 4-6　kms_info.propeties

パラメータ	概要	デフォルト値
endpoint	AWS KMS エンドポイントの URL を指定します。プロコトルを含めた URL 形式で指定してください。	https://kms.ap-northeast-1.amazonaws.com
connection_timeout	AWS KMS 接続時のタイムアウトを指定します。単位はミリ秒です。0 を指定するとタイムアウトしません。	50000
socket_timeout	AWS KMS 通信時のタイムアウトを指定します。単位はミリ秒です。0 を指定するとタイムアウトしません。	50000
proxyhost	プロキシを使用する場合のプロキシサーバーのホスト名もしくは IP アドレスを指定してください。	なし
proxyport	プロキシを使用する場合のプロキシサーバーのポートを指定してください。	なし
protocol	プロキシを指定する場合のプロキシサーバーへの接続プロトコルを指定してください。"http"か"https"が指定可能です。	なし
proxyuser	プロキシを指定する場合のプロキシサーバーの認証ユーザー名を指定して下さい。	なし
proxypassword	プロキシを指定する場合のプロキシサーバーの認証ユーザーのパスワードを指定してください。	なし

[設定例]

次の設定例では、https://kms.ap-northeast-1.amazonaws.com に、connection_timeout および socket_timeout がそれぞれ 50 秒でプロキシサーバー「http://proxyserver1.ne.jp:8080」を利用して接続します。

```
endpoint = https://kms.ap-northeast-1.amazonaws.com
# connection_timeout = 50000
# socket_timeout = 50000
proxyhost=proxyserver1.ne.jp
proxyport=8080
protocol=http
# proxyuser=
# proxypassword=
```

表 4-7　aws_info.propeties

パラメータ	概要	デフォルト値
accesskeyid	DB 管理者ユーザーの AccessKeyId を指定してください。	なし
secretaccesskey	DB 管理者ユーザーの SecretAccessKey を指定してください。	なし

[設定例]

次の設定例では、AccessKeyId が「AACCCCEESSSSKKEEYY00」であり、SecretAccessKey が「AwS+SeCrEt+AcCeSsKeY+AwS」であるユーザーを暗号鍵の復号の接続情報として使用します。

```
accesskeyid = AACCCCEESSSSKKEEYY00
secretaccesskey = AwS+SeCrEt+AcCeSsKeY+AwS
```

4.5　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。

4.5.1　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、「4.5.2　バックアップ手法（29 ページ）」をご確認ください。

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

  PostgreSQL サーバーの設定ファイル（`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/*`
  - システムファイル：*<インストールディレクトリ>*`/sys/*`

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。

列単位暗号化を利用した環境でバックアップしたものに対して、行単位暗号化を利用した環境でリストアすることはサポートしていません。

また、行単位暗号化を使用した環境でバックアップしたものに対して、列単位暗号化を利用した環境でリストアすることはサポートしていません。

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。

4.5.2　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 4-8　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。本製品ではバイナリー形式での出力のみサポートする	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
復号データのバックアップ	不可能	不可能	可能

4.5.3　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)

4.5.4　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、暗号化された状態でデータをバックアップ・リストアする必要があります。バックアップしたデータは暗号化された状態のため、漏えいのリスクを低下させることができます。なお、pg_dump や pg_dumpall では復号したデータをバックアップすることはできません。

暗号化された状態で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.enable パラメータを off に設定します（簡易 TDE モードの場合も同様です）。

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 モードの場合も同様です）。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。

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

ヒント

pg_dump を利用して特定のテーブルのみをバックアップする場合は、 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 も同時にバックアップすることをお勧めします。

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 モードの場合も同様です）。 -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。

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

注

暗号鍵の更新前に取得した 論理バックアップから pg_restore を利用して特定のテーブルのみリストアする場合、最新の暗号鍵によるデータ再暗号化（pgtde -m cipher --reset）などで古い暗号鍵を削除してしまっているとリストアしたデータを読めません。そのため、暗号鍵の更新前に取得した 論理バックアップから特定のテーブルのみリストアしたい場合は、透過的暗号化機能が有効かつ暗号鍵が未登録の環境に cipher_key_table テーブルと key_management_table テーブルを一緒にリストアする必要があります。以上の理由から暗号鍵を更新するたびに新しいバックアップを取得することをお勧めします。なお、key_management_table テーブルについては、リストア前にテーブルを SQL コマンド（TRUNCATE や DELETE など）で空の状態にしてください。

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)

4.5.5　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属性をもつテーブルを復号して CSV に出力することや、暗号化したままの状態で任意のテーブルをファイルに出力することもできます。

暗号化属性を含む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、暗号化属性を含む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ーが 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 ファイルを暗号化属性を含む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();
```

暗号化属性を含むテーブルから暗号化した状態でデータをエクスポート/インポートする

COPY コマンドを使用して暗号化された状態のままファイルにエクスポートするには、COPY ... WITH [FORMAT] BINARY 構文を使用します。なお、本実行例では psql コマンドから実行する方法を記載します。CSV ファイル名に「/export_dba/user_table.bin」を指定します。

```
$ psql -c "COPY apuser.user_table TO '/export_dba/user_table.bin' WITH BINARY;"
```

エクスポートされた暗号化属性を含むバイナリーデータファイルをインポートする例を以下に記載します。

```
$ psql -c "COPY apuser.user_table FROM '/export_dba/user_table.bin' WITH BINARY;"
```

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「7.2　バックアップファイル出力先設定（encrypt.backup）（69 ページ）」や「6.5　暗号鍵バックアップ（cipher_key_backup）（65 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 [REGIST_OPTIONS] | cipher [CIPHER_OPTIONS] | switch {SWITCH_OPTIONS} | show}
 [CONNECTION_OPTIONS]
```

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。なお、Transparent Data Encryption for PostgreSQL V1.2.0 より、オプションの形式が変更されたため、次節以降のオプションの説明では Transparent Data Encryption for PostgreSQL V1.1.4 までの形式と V1.2.0 からの形式を記載します。

- 「5.1.1　暗号鍵の登録・更新（-m regist）（40 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）（52 ページ）」

  暗号化済みのユーザーデータを一括で再暗号化することができます。

- 「5.1.2　透過的暗号化機能の利用モードの変更（-m switch）（46 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）（55 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『列単位暗号化 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ーが制限されます。

表 5-1　-m オプション毎の実行ユーザー制限

実行可能ユーザー	-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）
アプリケーション管理者	cipher（最新の暗号鍵によるデータ再暗号化）

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
	cipher	最新の暗号鍵によりデータの再暗号化します。	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
-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/tdeforpgXX、Windows 版 Transparent Data Encryption for PostgreSQL のデフォルトは C:\Program Files\NEC\TDEforPGXX（XX は PostgreSQL のメジャーバージョン。13 の場合は 13））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	接続するサーバーアドレス(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	接続するユーザーのパスワード

[環境変数設定例（Linux）]

```
$ export PGDATABASE=pgtdedb
$ export PGHOST=hostname
```

```
$ export PGPORT=5432
$ export PGUSER=username
$ export PGPASSWORD=userpassword
```

[環境変数設定ファイル設定例（Windows）]

```
CMD>SET PGDATABASE=pgtdedb
CMD>SET PGHOST=hostname
CMD>SET PGPORT=5432
CMD>SET PGUSER=username
CMD>SET 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。

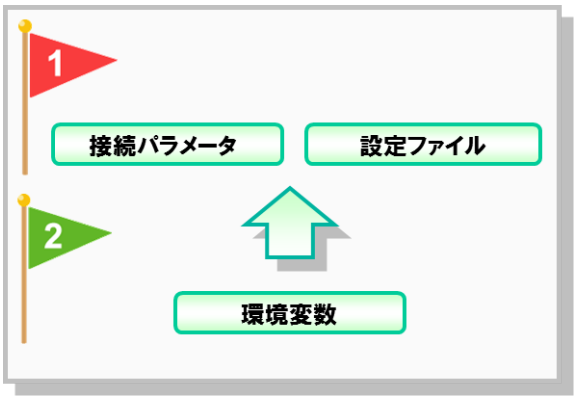

図 5-1　接続パラメータ優先順位図

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 \\\"\$\@\&
```

- AWS KMS 利用時、全てのコマンドの実行時間は AWS KMS に対するリクエストにかかる時間に影響を受けます。

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。

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [Linux]

```
Usage:
 pgtde CONNECTION_OPTIONS MODE_OPTIONS

MODE_OPTIONS:
 -m regist [REGIST_OPTIONS]    register new data key to specified database.
                               KeyId will be updated if --keyid is specified.
 -m cipher [CIPHER_OPTIONS]    cipher data with latest key
 -m switch {SWITCH_OPTIONS}    switch key management mode
 -m show                       show status of using KeyId from AWS-KMS in specified datab
ase

REGIST_OPTIONS:                [--keyid] [--cipherfile]
 --keyid                       KeyId will be updated if specify.
 --cipherfile                  if you use this, input not directly specified cipher key b
ut cipher key file path.

CIPHER_OPTIONS:                [-i INTERVAL] [--cipherfile] [--reset]
 -i  INTERVAL                  interval value
 --reset                       including reset cipher key's version
 --cipherfile                  if you use this, input not directly specified cipher key b
ut cipher key file path.

SWITCH_OPTIONS:                {--simple-tde | --standard-tde} [--cipherfile] | --aws-kms
 --simple-tde                  switch key management mode to simple TDE mode
 --standard-tde                switch key management mode to standard TDE mode
 --aws-kms                     switch key management mode to AWS-KMS mode
 --cipherfile                  if you use this, input not directly specified cipher key b
ut cipher key file path.

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
```

- [Windows]

```
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:                 [-i INTERVAL] [--reset]
 -i  INTERVAL                   interval value
 --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
```

【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

```
Usage:
 pgtde CONNECTION_OPTIONS MODE_OPTIONS

MODE_OPTIONS:
 -m cipher [CIPHER_OPTIONS]     cipher data with latest key
 -m regist [REGIST_OPTIONS]     register new data key to specified database.
                                KeyId will be updated
                                if --keyid is specified.
 -m switch {SWITCH_OPTIONS}     switch key management mode
 -m show                        show status of using KeyId from AWS-KMS in
                                specified database

CIPHER_OPTIONS:                 [-i INTERVAL] [--cipherfile] [--reset]
 -i  INTERVAL                   interval value
 --reset                        including reset cipher key's version
 --cipherfile                   if you use this,
                                input not directly specified
                                cipher key but cipher key file path.

REGIST_OPTIONS:                 [--keyid] [--cipherfile]
 --keyid                        KeyId will be updated if specify.
 --cipherfile                   if you use this,
                                input not directly specified
                                cipher key but cipher key file path.

SWITCH_OPTIONS:                 --aws-kms | --tde-only [--cipherfile]
 --aws-kms                      switch key management mode to AWS-KMS mode
 --tde-only                     switch key management mode to TDE local mode
 --cipherfile                   if you use this,
                                input not directly specified
                                cipher key but cipher key file path.

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

【Transparent Data Encryption for PostgreSQL 全バージョン共通】

- [Linux]

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

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [Windows]

```
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。

AWS KMS モードを利用する場合、透過的暗号化機能コマンド（pgtde）実行後に暗号鍵情報が記載されたファイルが出力されます。出力された暗号鍵ファイルを参照いただき、任意の方法で暗号鍵情報をアプリケーション管理者（アプリケーション開発者）に通知してください。通知完了後はセキュリティ確保のため、速やかに当該ファイルは削除していただくことを推奨いたします。

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、プライマリサーバー上で実行してください。実行結果はスタンバイサーバーにも反映されます。

パラメータ

表 5-6　暗号鍵登録・更新パラメーター覧

パラメータ	説明	分類
--keyid	AWS KMS モードの場合に、現在使用している CMK とは異なる CMK を使用して暗号鍵の登録・更新を行いたい場合に指定します。	任意
--cipherfile	AWS KMS モードの場合に、本パラメータを指定することで、暗号鍵の登録・更新に必要な暗号鍵を入力したファイルパスから読み込みます。	任意

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。

簡易 TDE モードおよび標準 TDE モードの場合

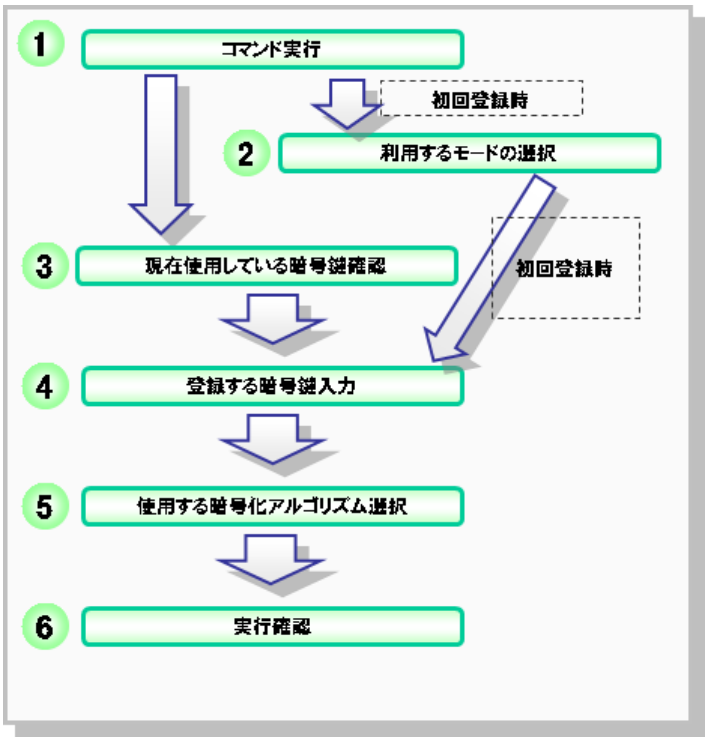

図 5-2　暗号鍵登録・更新（簡易 TDE モードおよび標準 TDE モードの場合)

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。

   【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

   - [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）
     - 3. AWS-KMS mode.（AWS KMS モードを利用する場合）
   - [Windows]
     - 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。

   【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

- 1. AWS-KMS mode.（暗号鍵の管理方式に AWS KMS 管理方式を採用する場合）
- 2. TDE local mode.(暗号鍵の管理方式にローカル鍵管理方式を採用する場合)

上記 「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。

AWS KMS モードの場合

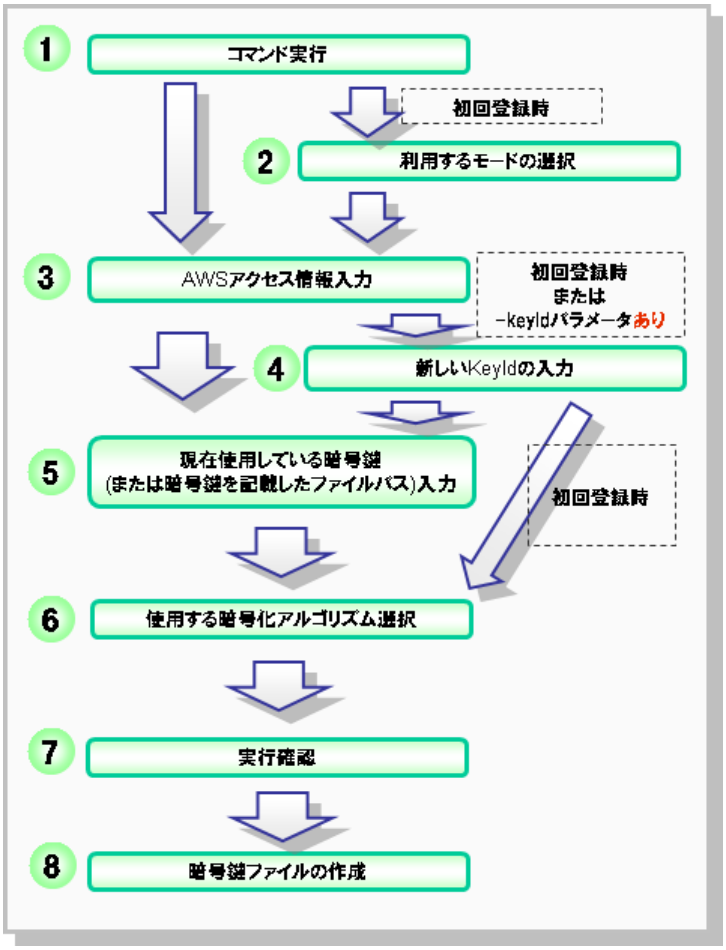

図 5-3　暗号鍵登録・更新（AWS KMS モードの場合)

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。

   【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

   注

   Windos 版 Transparent Data Encryption for PostgreSQL は AWS KMS モードをサポートしません。

   - [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）
     - 3. AWS-KMS mode.（AWS KMS モードを利用する場合）

   上記 「3」 を指定します。

   【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux)】

- 1. AWS-KMS mode.（暗号鍵の管理方式に AWS KMS 管理方式を採用する場合）
- 2. TDE local mode.(暗号鍵の管理方式にローカル鍵管理方式を採用する場合)

上記 「2」 を指定します。

3. AWS アクセス情報入力

   セキュリティ管理者用の AWS アクセス情報(Access Key ID,Secret Access Key)を入力します。

4. 新しい KeyId の入力

   初回登録時、または CMK を変更したい場合(--keyid オプションを付与した場合)、新しい KeyId を入力します。

5. 現在使用している暗号鍵(または暗号鍵ファイルパス)入力

   初回登録時を除き、現在使用中の暗号鍵を入力します。--cipherfile オプションを付与している場合は暗号鍵のみが記載されたファイルの絶対パスを入力します。

6. 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。

7. 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。

8. 暗号鍵ファイルの出力

   暗号鍵の登録・更新に成功すると、key_YYYYMMDDHHMMSS の書式で暗号鍵が記載されたファイルがコマンド実行時のカレントディレクトリに出力されます。

注意制限事項

- 標準 TDE モードや AWS KMS 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ー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）（65 ページ）」をご確認ください。

- 暗号鍵登録・更新に何らかの原因で失敗した場合、無効な暗号鍵ファイルが生成される可能性があります。この暗号鍵ファイルを残していてもセキュリティ上問題はありませんが、必要に応じて削除されることを推奨いたします。
- 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。
- 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。
- 空文字の暗号鍵を登録することはできません。

実行例

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [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.
3. AWS-KMS 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
```

  - 次の例では、AWS KMS モードで暗号鍵を更新します。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。また、AWS KMS モードで暗号鍵を更新する際には--cipherfile パラメータを利用します。

```
$ pgtde -m regist -conf /tmp/pgtde_secuser.properties --cipherfile
Enter AWS Access Key ID: AACCCCEESSSSKKEEYY11
Enter AWS Secret Access Key: aaa+SeCrEt+AcCeSsKeY+cccddd
Enter current cipher key file path: /home/secuser/key_20180325180308
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 4 is registered to tdedb
```

```
Cipher key file name: /home/secuser/key_20180325183944
INFO: <I001> For security purposes please delete this cipher key file after used.
```

- [Windows]
  - 次の例では、標準 TDE モードで暗号鍵を更新します。データベースの接続には設定ファイル「C:\Program Files\NEC\TDEforPG13\conf\pgtde_secuser.properties」を利用します。

```
CMD>pgtde.bat -m regist -conf "C:\Program Files\NEC\TDEforPG13\conf\pgtde_secuser.properties"
Enter current data key:
Enter new data key:
Retype new data key:
Select algorithm:
1. aes
2. bf
2
Are you sure you want to Regist new key to "tdedb"(DATABASE) with "bf" algorithm? (Press Y(y)
key to execute): y
New key version 2 is registered to tdedb
```

【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

- 次の例ではローカル鍵管理方式で暗号鍵を登録します。

```
$ pgtde -m regist -conf /tmp/pgtde_secuser.properties
Key management mode is not yet set.
Please select key management mode:
1. AWS-KMS mode.
2. TDE local mode.
2
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 "testdb"(DATABASE) with "aes" algorithm? (Press Y(y) key
 to execute): y
New key version 1 is registered to testdb
```

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

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [Linux]

```
pgtde -m switch {--simple-tde | --standard-tde | --aws-kms} [--cipherfile] [CONNECTION_OP
TIONS]
```

- [Windows]

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

【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

- `pgtde -m switch {--aws-kms | --tde-only} [--cipherfile] [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、プライマリサーバー上で実行してください。実行結果はスタンバイサーバーにも反映されます。

パラメータ

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

表 5-7　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
--aws-kms	AWS KMS モードに変更する場合に指定します。	選択
--cipherfile	AWS KMS モードに変更する場合に、本パラメータを指定することで、AWS KMS モードを変更するために必要な暗号鍵を入力したファイルパスから読み込みます。	任意

【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

表 5-8　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
--aws-kms	鍵の管理方式をローカル鍵管理方式から AWS KMS 管理方式に変更する場合に指定します。	選択
--tde-only	鍵の管理方式を AWS KMS 管理方式からローカル鍵管理方式に変更する場合に指定します。	選択
--cipherfile	AWS KMS 管理方式からローカル鍵管理方式に変更する場合に、当該パラメータを指定することで、鍵の管理方式を変更するために必要な暗号鍵を入力したファイルパスから読み込みます。	任意

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 モードまたは標準 TDE モードから AWS KMS モードに変更する場合

図 5-4　透過的暗号化機能の利用モードの変更（簡易 TDE モードまたは標準 TDE モードから AWS KMS モードに変更）

1. コマンド実行

   - **【Transparent Data Encryption for PostgreSQL 全バージョン共通】**
     - 「--aws-kms」パラメータを指定して pgtde コマンドを実行します。

2. AWS アクセス情報入力

   セキュリティ管理者用の AWS アクセス情報(Access Key ID,Secret Access Key)を入力します。

3. 新しい KeyId の入力

   利用したい CMK の KeyId を入力します。

4. 現在使用している暗号鍵を入力

   現在使用している暗号鍵を入力します。

5. 実行決定

   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。

6. 暗号鍵ファイルの出力

   過的暗号化機能の利用モードの変更に成功すると、key_YYYYMMDDHHMMSS の書式で暗号鍵が記載されたファイルがコマンド実行時のカレントディレクトリに出力されます。

AWS KMS モードから簡易 TDE モードまたは標準 TDE モードに変更する場合

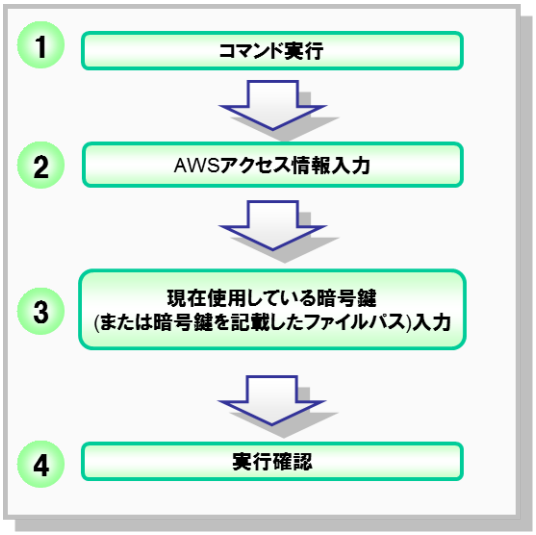

図 5-5　透過的暗号化機能の利用モードの変更（AWS KMS モードから簡易 TDE モードまたは標準 TDE モードに変更）

1. コマンド実行

   Transparent Data Encryption for PostgreSQL のバージョンごとに指定するパラメータが異なります。

   - **【Transparent Data Encryption for PostgreSQL V1.2.0 以降】**
     - 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。
   - **【Transparent Data Encryption for PostgreSQL V1.1.4 以前】**
     - 「`--tde-only`」パラメータを指定して `pgtde` コマンドを実行します。

2. AWS アクセス情報入力

   セキュリティ管理者用の AWS アクセス情報(Access Key ID,Secret Access Key)を入力します。

3. 現在使用している暗号鍵(暗号鍵ファイル)入力

   現在使用中の暗号鍵を入力します。「`--cipherfile`」パラメータを指定している場合は暗号鍵が記載されたファイルの絶対パスを入力します。

4. 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。

簡易 TDE モードと標準 TDE モードを切り替える場合

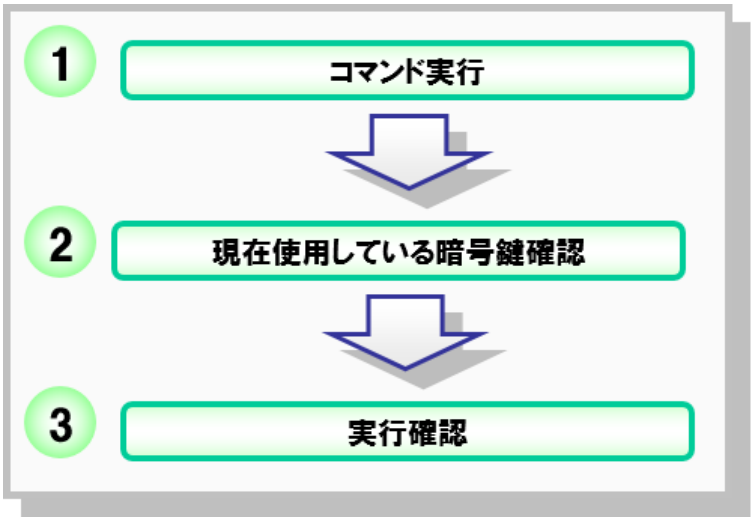

図 5-6　透過的暗号化機能の利用モードの変更（簡易 TDE モードと標準 TDE モードを切り替える）

1. コマンド実行

   - **【Transparent Data Encryption for PostgreSQL V1.2.0 以降】**
     - 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(暗号鍵ファイル)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。

制限事項

- 出力された暗号鍵ファイルの修正は禁止しています。

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 モードまたは標準 TDE モードから AWS KMS モードに切り替える際に、何らかの原因で失敗した場合、無効な暗号鍵ファイルが生成される可能性があります。この暗号鍵ファイルを残していてもセキュリティ上問題はありませんが、必要に応じて削除されることを推奨いたします。

- 簡易 TDE モードまたは標準 TDE モードから AWS KMS モードに切り替えた場合、自動的に新しい暗号鍵が透過的暗号化機能データベースに登録されます。AWS KMS モードから簡易 TDE モードまたは標準 TDE モードに切り替える場合は新しい暗号鍵は登録されません。また、簡易 TDE モードと標準 TDE モードを切り替えた場合も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

【Transparent Data Encryption for PostgreSQL 全バージョン共通】

- [Linux]
  - 次の例では、簡易 TDE モードまたは標準 TDE モードから AWS KMS モードに変更します（Transparent Data Encryption for PostgreSQL V1.1.4 共通 : AWS KMS 管理方式からローカル鍵管理方式に変更します）。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 --aws-kms -conf /tmp/pgtde_secuser.properties
Enter AWS Access Key ID: AACCCCEESSSSKKEEYY11
Enter AWS Secret Access Key: aaa+SeCrEt+AcCeSsKeY+cccddd
Enter KeyId: 8dkehald-akb3-ala9-9dkj-sl93kba03kd
Enter current data key:
Are you sure you want to switch key management mode of "tdedb"(DATABASE) to "AWS-KMS mode" ? (
Press Y(y) key to execute): y
New key version 3 is registered to tdedb

Switch key management mode to AWS-KMS mode is successfully.
Cipher key file name: /home/secuser/key_20180325220443
INFO: <I001> For security purposes please delete this cipher key file after used.
```

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [Linux]
  - 次の例では、AWS KMS モードから簡易 TDE モードに変更します（標準 TDE モードに変更する場合は「--simple-tde」を「--standard-tde」に読み替えてください）。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。また、AWS KMS モードで暗号鍵を更新する際には--cipherfile パラメータを利用します。

```
$ pgtde -m switch --simple-tde -conf /tmp/pgtde_secuser.properties --cipherfile
Enter AWS Access Key ID: AACCCCEESSSSKKEEYY11
Enter AWS Secret Access Key: aaa+SeCrEt+AcCeSsKeY+cccddd
Enter current cipher key file path: /home/secuser/key_20180325220443
Are you sure you want to switch key management mode of "tdedb"(DATABASE) to "Simple TDE mode"
? (Press Y(y) key to execute): y
New key version 3 is registered to tdedb

Switch key management mode to simple TDE mode is successfully.

Data key file path: /home/secuser/datakey_20180325220522
```

```
You need to manage this key to using in case of changing key management mode.
INFO: <I002> For security purposes please delete this data key file after used.
```

- [Windows]
  - 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「C:\Program Files\NEC\TDEforPG13\conf\pgtde_secuser.properties」を利用します。

```
CMD>pgtde.bat -m switch --standard-tde -conf "C:\Program Files\NEC\TDEforPG13\conf\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 2 is registered to tdedb

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

【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

- 次の例では、AWS KMS 管理方式からローカル鍵管理方式に変更します。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。また、AWS KMS モードで暗号鍵を更新する際には--cipherfile パラメータを利用します。

```
$ pgtde -m switch --tde-only -conf /tmp/pgtde_secuser.properties --cipherfile
Enter AWS Access Key ID: AACCCCEESSSSKKEEYY11
Enter AWS Secret Access Key: aaa+SeCrEt+AcCeSsKeY+cccddd
Enter current cipher key file path: /home/secuser/key_20180325220443
Are you sure you want to switch key management mode of "testdb"(DATABASE) to "TDE local mode" ? (Press Y(y) key to execute): y
Switch key management mode to TDE local mode is successfully.

Data key: +uKRqFUm+xisLEJx71sfkei93obkls83lks22ZJBzo=
```

5.1.3　最新の暗号鍵によるデータ再暗号化（-m cipher）

構文

【Transparent Data Encryption for PostgreSQL 全バージョン共通】

- [Linux]

```
pgtde -m cipher [--reset] [-i INTERVAL] [--cipherfile] [CONNECTION_OPTIONS]
```

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [Windows]

```
pgtde -m cipher [--reset] [-i INTERVAL] [CONNECTION_OPTIONS]
```

概要

最新の暗号鍵によるデータ再暗号化（-m cipher）では、接続対象データベースで利用する最新の暗号鍵を利用して暗号化されたデータを再暗号化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。

再暗号化実行後はユーザーデータの一括更新に伴いユーザーデータに多くの不要領域が発生するため、autovacuum 機能を使用していない場合は、VACUUM/VACUUM FULL の実行を推奨します。本-m オプションで実施する再暗号化の進捗状況は、PostgreSQL のログから確認することができます。ログに出力する進捗割合は、透過的暗号化機能が提供するパラメータ「進捗ログ出力設定」を使用することにより、変更することが可能です。詳細は「進捗ログ出力設定（70 ページ）」をご確認ください。

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、プライマリサーバー上で実行してください。実行結果はスタンバイサーバーにも反映されます。

パラメータ

表 5-9　再暗号化モード パラメーター覧

パラメータ	パラメータ内容	分類
--reset	透過的暗号化機能で利用する暗号鍵のリセットをすることができます。リセットを実施すると、最新の暗号鍵による再暗号化を実施後に最新の暗号鍵情報以外は削除されます。	任意
-i *INTERVAL*	再暗号化処理の一定件数毎の処理遅延時間を設定することができます。再暗号化処理は負荷がかかる処理であるため、このパラメータを設定することで 1000 件再暗号化処理を行う毎に指定した値(ms)の遅延を発生させることができます。このオプションを指定しない場合、再暗号化処理は遅延なしで実行されます。インターバル値にインターバル値に 2147483648 以上の値を指定することはできません。	任意
--cipherfile	AWS KMS モードの場合に、本パラメータを指定することで、再暗号化処理に必要な暗号鍵を入力したファイルパスから読み込みます。ファイルパスは現在の暗号鍵を入力するフェーズで指定します。	任意

使用方法

最新の暗号鍵によるデータ再暗号化の処理の流れについて説明します。

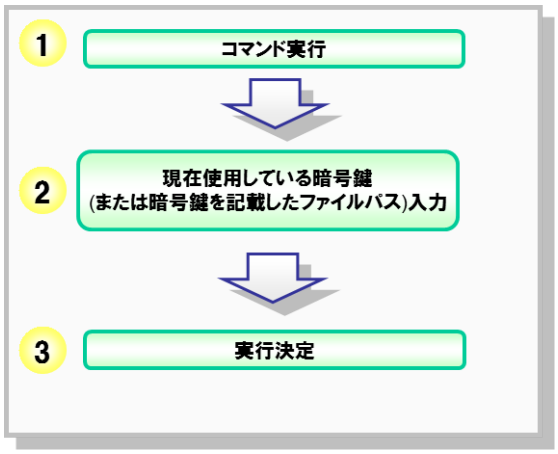

図 5-7　最新の暗号鍵によるデータ再暗号化

1. コマンド実行

   本コマンドを実行します。必要に応じて「--reset」「-i」「--cipherfile」パラメータを指定してください。

2. 現在使用している暗号鍵を入力

   透過的暗号化機能を利用するデータベースで使用している現在の暗号鍵を入力します。--cipherfile オプションを付与している場合は暗号鍵のみが記載されたファイルの絶対パスを入力します。

3. 実行決定

   上記までに行った設定でデータの再暗号化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

- 再暗号化を実行する際、暗号化列を定義したテーブルのルールまたはトリガは削除または無効化(ALTER TABLE … DISABLE RULE …)する必要があります。ルールまたはトリガが有効な状態では、再暗号化処理中にルールに定義した処理が実行されます。
- 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。
- 「--reset」パラメータを指定して再暗号化を実行している場合は、暗号鍵の登録・更新・削除等の操作は実行されず、再暗号化の完了を待ちます。
- 「--reset」パラメータを指定して再暗号化を実行する前から接続を開始している場合は、再暗号化処理完了後に再接続する必要があります。
- 「--reset」パラメータを指定して再暗号化を実行した後は、暗号化データ型を定義した全てのユーザーテーブルに ANALYZE を実施していただく必要があります。なお、ANALYZE の実施には対象のテーブルを所有する権限が必要となります。
- 暗号化列を持つ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。

実行例

【Transparent Data Encryption for PostgreSQL 全バージョン共通】

透過的暗号化機能を利用するデータベースに対して再暗号化を実行し、同時に最新の暗号鍵以外の古い暗号鍵情報を削除します。

次の例では、旧バージョンの暗号鍵の削除を伴うデータの再暗号化を実行します。

```
$ pgtde -m cipher -i 10 --reset
Enter current cipher key:
Are you sure you want to Reencrypt
"tdedb"(DATABASE) with Interval="10" and Reset="true"?
(Press Y(y) key to execute): y
All data in testdb are reencrypted
```

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

【Transparent Data Encryption for PostgreSQL 全バージョン共通】

```
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。また、AWS KMS の登録情報を確認することができます。Transparent Data Encryption for PostgreSQL V1.1.4 以前は AWS KMS の登録情報のみを確認することができます。

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

3. AWS KMS の情報を表示するか確認

   AWS KMS の情報を確認するかメッセージが表示されます

4. セキュリティ管理者用 AWS アクセス情報入力

   セキュリティ管理者用の AWS アクセス情報(Access Key ID,Secret Access Key)を入力します。

注意制限事項

- 表示される CMK の作成日付はシステムのタイムゾーンに合わせて表示されます。

実行例

【Transparent Data Encryption for PostgreSQL V1.2.0 以降】

- [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および AWS KMS の登録情報を確認しています。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
################################

Do you want to show AWS-KMS information(y/n)?: y
Enter AWS Access Key ID: AACCCCEESSSSKKEEYY11
Enter AWS Secret Access Key: aaa+SeCrEt+AcCeSsKeY+cccddd

State of using KeyIds in TDE (Endpoint: https://kms.ap-northeast-1.amazonaws.com)

KeyId           : 124ehald-akb3-ala9-9dkj-sl93kba03kd
Enabled Status  : true
TDE using       : Available
Creation Date   : 2011-03-23 21:28:37
AutoRotation    : true
Description     : my test key

KeyId           : 455ehald-akb3-ala9-9dkj-sl93kba03kd
Enabled Status  : true
TDE using       : Available
Creation Date   : 2011-08-06 00:28:51
AutoRotation    : false
Description     : Default key

KeyId           : 697ehald-akb3-ala9-9dkj-sl93kba03kd
Enabled Status  : true
TDE using       : using
Creation Date   : 2011-06-28 22:56:34
AutoRotation    : true
Description     : Key for Transparent Data Encryption For PostgreSQL
```

- [Windows]
  - 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「C:\Program Files\NEC\TDEforPG13\conf\pgtde_secuser.properties」を利用します。

```
CMD>pgtde -m show -conf "C:\Program Files\NEC\TDEforPG13\conf\pgtde_secuser.properties"

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

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

```
* Version: 5
################################
```

【Transparent Data Encryption for PostgreSQL V1.1.4 以前（Linux）】

- 次の例では、AWS KMS 利用状況を表示します。

```
$ pgtde -m show -conf /tmp/pgtde_secuser.properties
Enter AWS Access Key ID: AACCCCEESSSSKKEEYY11
Enter AWS Secret Access Key: aaa+SeCrEt+AcCeSsKeY+cccddd

State of using KeyIds in TDE (Endpoint: https://kms.ap-northeast-1.amazonaws.com)

KeyId           : 124ehald-akb3-ala9-9dkj-sl93kba03kd
Enabled Status  : true
TDE using       : Available
Creation Date   : 2011-03-23 21:28:37
AutoRotation    : true
Description     : my test key

KeyId           : 455ehald-akb3-ala9-9dkj-sl93kba03kd
Enabled Status  : true
TDE using       : Available
Creation Date   : 2011-08-06 00:28:51
AutoRotation    : false
Description     : Default key

KeyId           : 697ehald-akb3-ala9-9dkj-sl93kba03kd
Enabled Status  : true
TDE using       : using
Creation Date   : 2011-06-28 22:56:34
AutoRotation    : true
Description     : Key for Transparent Data Encryption For PostgreSQL

AWS KMS mode being used in this database
```

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 のユーザー定義関数を利用して機能を提供しています。提供するユーザー定義関数を「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、ユーザー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（59 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（60 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（62 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（63 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（65 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。AWS KMS モードの場合、ここで指定する暗号鍵は AWS KMS が管理しているデータを暗号化するための鍵を暗号化したものとなります。	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。
- 鍵管理に AWS KMS を利用している場合、「セッション開始」ファンクション実行中にユーザーからのキャンセル要求を受け取ると、以下のエラーメッセージが出力されますが、これは AWS KMS と処理を行うプロセスがユーザーからのキャンセル要求によって終了したことを示しており、異常ではありません。

```
ERROR: Child did not exit properly.
```

- `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
```

- AWS KMS モードの場合「セッション開始」ファンクション実行中は 1 セッションあたり 1 つの暗号鍵復号用プロセスが生成され、当該プロセスが最大約 100MB 程度のメモリを一時的に使用します。

- AWS KMS モードの場合、セッション開始完了までの時間は AWS KMS による暗号鍵の復号にかかる時間と AWS KMS との通信時間の合計に依存します。また本処理の実行時間と同時実行数には比例関係があり、弊社環境での測定結果は以下のとおりです。

表 6-5　弊社環境での暗号鍵復号処理時間(2015 年 7 月時点)

同時実行数	復号処理完了時間(s)
1	1.8
3	4.5
6	9.6

なお、簡易 TDE モードと標準 TDE モードでは AWS KMS に対して処理を要求することはありません。

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 6-6　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 6-7　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
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列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（69 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 6-8　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 のスーパーユーザーと cipher_setup.sh で作成したセキュリティユーザー

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
暗号化/復号制御	暗号化データ型に対する透過的な暗号化/復号処理の実行有無を制御することができます。	「暗号/復号制御（68 ページ）」
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（65 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（69 ページ）」
進捗ログ出力設定	透過的暗号化コマンドの再暗号化モードで出力される更新処理の進捗ログ出力設定を行うことができます。	「進捗ログ出力設定（70 ページ）」
暗号化データ型比較制御	暗号化データ型の列と比較する際の動作を制御することができます。	「暗号化データ型比較制御（71 ページ）」
AWS KMS 復号処理タイムアウト	AWS KMS 管理方式の場合、セッション開始ファンクションの暗号鍵復号処理にかかるタイムアウトを設定することが出来ます。	「AWS KMS 復号処理タイムアウト（72 ページ）」
透過的暗号化機能ライブラリパス	PostgreSQL が AWS KMS との通信に利用する透過的暗号化機能ライブラリのパスを設定します	「透過的暗号化機能ライブラリパス（73 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、「進捗ログ出力設定」の設定例を記載します。

[postgresql.conf 設定例]

```
encrypt.logoutpernum = 100
```

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　暗号化/復号制御（encrypt.enable）

構文

```
encrypt.enable = { ON | OFF }
```

説明

本パラメータで透過的暗号化機能による暗号化/復号処理の実行有無を制御することができます。 透過的暗号化機能では暗号化列の参照を行うと、透過的に復号されます。暗号化されたままのデータを取得したい場合には、本パラメータの設定を行うことで透過的な復号を行わず、暗号化されたままのデータを取得することが可能です。透過的暗号化機能による暗号化/復号処理の実行有無の動作については「暗号化/復号処理（16 ページ）」に記載しています。

設定値

表 7-2　暗号/復号設定

設定値	内容	種別
ON	暗号化データ型に対して暗号化/復号処理を行う	既定値
OFF	暗号化データ型に対して暗号化/復号処理を行わない	

注意制限事項

- バックアップ/リストア時を除き、本オプションを「OFF」に設定した状態で、暗号化データ型の列への挿入・更新を行わないでください。

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

次の例では透過的暗号化機能による暗号化/復号処理の実行を行わない状態に変更します。

```
=>SET encrypt.enable = OFF;
SET
```

次の例では暗号化/復号化処理の実行を行わない状態で pg_dump を実行します。

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

7.2　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 7-3　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 の起動ユーザー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 のスーパーユーザー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
```

7.3　進捗ログ出力設定（encrypt.logoutpernum）

構文

```
encrypt.logoutpernum = { 0 - 10000 }
```

説明

透過的暗号化コマンドの再暗号化モードで実行する再暗号化の進行状況はログに出力されます。本パラメータを設定することにより、ログに出力される進捗割合を変更することが可能です。

本パラメータに設定する件数毎に再暗号化の進捗をログへ出力をします。ただし、ログで出力される内容は目安であり正確な値を出力することを保証していません。設定した値は以下の法則により件数に置き換えられます。なお、本パラメータに「0」が設定された場合には、ログ出力を行いません。また、再暗号化モードにおいて、インターバル値を 0 に設定している場合も、ログは出力されません。

- ログ間隔件数

  カスタムパラメータに設定された値 × 10000 **件毎**

設定値

表 7-4　進捗ログ出力設定

設定値	内容	種別
1	1 万件単位でログに出力します。	既定値
0	ログ出力を行いません。	未出力
2 ～ 10000	『説明』のログ間隔件数の法則により、算出された単位でログを出力します。	出力有効範囲

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、進捗ログ出力設定の値を 100 に設定します。

```
=> SET encrypt.logoutpernum = 100;
SET
```

7.4　暗号化データ型比較制御（encrypt.diff_version）

構文

```
encrypt.encrypt.diff_version = { ON | OFF }
```

説明

暗号化データ型の等価比較を行う際には、暗号化された状態でデータの比較を実施します。暗号化に別のバージョンの暗号鍵を使用している場合、暗号化前のデータが一致していても認識することができません。そのため、データの比較を行う前に暗号鍵のバージョンを比較します。バージョンが不一致の場合には、暗号化データ型を復号してから比較することができますが復号処理が多くなると性能への影響が大きくなります。そこで、本パラメータを使用することで、バージョン不一致が存在する場合の動作を制御することができます。

設定値

表 7-5　暗号化データ型比較制御

設定値	内容	種別
ON	比較対象の暗号鍵バージョンに不一致が存在する場合には暗号化データ型を復号して比較を実行します。	既定値
OFF	比較対象の暗号鍵バージョンに不一致が存在する場合でも暗号化データのまま比較を実行します。比較対象が別のバージョンの暗号鍵を使用している場合、暗号化前のデータが一致していても認識することができません。	

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、暗号化データ型比較制御の設定を OFF に変更します。

```
=>SET encrypt.diff_version = OFF;
SET
```

7.5　AWS KMS 復号処理タイムアウト（encrypt.kms_timeout）

構文

```
encrypt.kms_timeout = { 0 - 3600 }
```

概要

透過的暗号化機能のセッション開始ファンクションが AWS KMS を利用して暗号鍵の復号処理を行う際、暗号鍵の復号処理が完了するまでの時間に対するタイムアウト値を本パラメータで提供します。

設定値

表 7-6　AWS KMS 接続タイムアウト

設定値	内容	種別
60	暗号鍵の復号処理が 60 秒以内に処理が完了しなければエラーとなる	既定値
1	暗号鍵の復号処理が 1 秒以内に処理が完了しなければエラーとなる	最小値
3600	暗号鍵の復号処理が 3600 秒以内に処理が完了しなければエラーとなる	最大値
0	暗号鍵の復号処理の際にタイムアウトを設定しない	無効値

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、AWS KMS 接続タイムアウトの設定を 120 秒に変更します。

```
=> SET encrypt.kms_timeout = 120;
SET
```

7.6　透過的暗号化機能ライブラリパス（encrypt.tde_rootpath）

構文

```
encrypt.tde_rootpath = '/opt/nec'
```

概要

既定値とは異なるディレクトリに Transparent Data Encryption for PostgreSQL をインストールした場合に、PostgreSQL が AWS KMS との通信に利用する透過的暗号化機能ライブラリパスを設定します。

設定値

表 7-7　透過的暗号化機能ライブラリパス

設定値	内容	種別
/opt/nec	Transparent Data Encryption for PostgreSQL が*/opt/nec* 配下にインストールされているとみなす。	既定値

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

透過的暗号化機能ライブラリパスを/tmp に変更する

```
=>SET encrypt.tde_rootpath = '/tmp';
SET
```

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

```
--Employee テーブル作成
CREATE TABLE Employee(
  EmployeeID Integer PRIMARY KEY,
  Name TEXT,
  Address ENCRYPT_TEXT,
  TelephoneNumber ENCRYPT_TEXT
);
```

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」、「INFO」の 4 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 keyid_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 サーバー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	E124	Password is not set	パスワード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	E143	Interval value is invalid	「-i」パラメータに指定している値を確認してください。
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。

レベル	エラーコード	エラーメッセージ	対処方法
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	E701	The specified KeyId is not valid.	アクセスしているリージョンに存在する有効な KeyId を指定してください。
ERROR	E702	The specified KeyId is in inactive status.	指定した KeyId に対応する CMK が無効化されている可能性があります。対象の CMK を有効化してから再度実行してください。
ERROR	E703	Could not connect to KMS server.	KMS にアクセス出来ませんでした。ネットワーク設定や kms_info.properties ファイルが正しく設定されいているか確認ください。
ERROR	E704	Invalid AWS Access Key ID or AWS Secret Access Key.	AWS Access Key ID、AWS Secret Access key の情報を正しく入力してください。
ERROR	E705	The specified KeyId could not confirm in KMS.	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	E706	Could not get 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	E709	The specified cipher key is not valid.	正しい暗号鍵を入力してください。
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	E715	Invalid option. Can not use "--aws-kms" and "--cipherfile" options simultaneously.	"--cipherkey"と"--cipherfile"オプションを同時に指定しないでください。
ERROR	E716	Can not specify "--keyid" without using KMS.	KMS を利用しない場合は--keyid を指定しないてください。
ERROR	E717	KeyId is not yet registered. Please try again with "--keyid" option.	KeyId を登録する必要があります。"--keyid" オプションを付けて、再度実行してください。
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"	フォルダーの権限を確認してください。
ERROR	E725	could not found accesskeyid or secretaccesskey in aws_info.properties file.	accesskeyid および secretaccesskey の情報を aws_info.properties ファイルに記載してください。
ERROR	E726	Invalid parameter protocol in kms_info.properties file. Only "http" and "https" are supported.	プロトコルは"http"または"https"を指定してください。
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。
ERROR	E731	"パラメータ名" parameter in kms_info.properties must be positive integer value.	該当するパラメータは負値は入力できません。0 以上の整数値を指定してください。
WARN	W206	The specified "--cipherfile" option is ignored.	暗号鍵の初回登録時であるため、"--cipherfile" オプションは無視されます。
WARN	W207	The specified "--keyid" option is ignored.	暗号鍵の初回登録時であるため、"--keyid"オプションは無視されます。

レベル	エラーコード	エラーメッセージ	対処方法
INFO	I001	For security purposes please delete this cipher key file after used.	セキュアな運用を行うため、出力された暗号鍵ファイルは使用後速やかに削除してください。
INFO	I002	For security purposes please delete this data key file after used.	セキュアな運用を行うため、出力されたデータ鍵ファイルは使用後速やかに削除してください。

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 32767 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、「--reset」オプションを指定して再暗号化を実施してください。
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	E0011	invalid interval time	不正な internal 値が入力されています。正しい interval 値を入力してください。
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。
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	E0018	could not decrypt data, because key was not found for version *バージョン*	表示されたバージョンの暗号鍵がすでに削除されています。
ERROR	E0019	Could not get data key	暗号鍵の復号に失敗しました。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	E0020	Failed to create pipe. errno: エラー番号	暗号鍵の復号に失敗しました。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。

レベル	エラーコード	エラーメッセージ	対処方法
ERROR	E0021	Failed to fork child process. errno: エラー番号	暗号鍵の復号に失敗しました。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	E0022	Failed to get datakey due to kms_timeout	kms_timeout で指定された時間内に暗号鍵を復号することが出来ませんでした。encrypt.kms_timeout パラメータの設定値を増加させることを検討してください。
ERROR	E0023	Unable to launch Java VM	暗号鍵の復号に失敗しました。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	E0024	Could not find PgtdeKmsAgent class in kms-agent.jar.	暗号鍵の復号に失敗しました。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	E0025	Could not locate Main method in PgtdeKmsAgent class.	暗号鍵の復号に失敗しました。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	E0026	Could not locate method getStatusCode in PgtdeKmsAgent class.	暗号鍵の復号に失敗しました。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	E0027	Could not locate method getDataKey in PgtdeKmsAgent class.	暗号鍵の復号に失敗しました。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	E0028	Internal Error	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	E0029	Could not read from pipe. errno: エラー番号	暗号鍵の復号に失敗しました。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	E0030	Could not write to pipe	暗号鍵の復号に失敗しました。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	E0031	Child process did not exit properly.	暗号鍵の復号に失敗しました。これはユーザーにより pgtde_begin_session 関数の実行がキャンセルされた場合にも発生する場合があります。この条件以外で発生し、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	E0032	applying modifier is not allowed if there is no valid key exists	E0018 のメッセージで表示されたバージョンの暗号鍵がないため、暗号化データ型に修飾子を適用することが出来ません
ERROR	E0033	applying modifier is not allowed if encrypt.enable is off	encrypt.enable パラメータは pg_dump, pg_restore を実行するとき以外は変更しないでください。
ERROR	E0034	type-casting is not allowed if there is no valid key exists	E0018 のメッセージで表示されたバージョンの暗号鍵がないため、暗号化データ型の型変換を実行することが出来ません
ERROR	E0035	type-casting is not allowed if encrypt.enable is off	encrypt.enable パラメータは pg_dump, pg_restore を実行するとき以外は変更しないでください。
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。

レベル	エラーコード	エラーメッセージ	対処方法
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、暗号化データ型の列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。
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 30000 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。
INFO	I0001	re-encryption of table "*スキーマ名*"."*テーブル名*" was started	再暗号化対象のテーブル処理開始時に出力されます。
INFO	I0002	re-encryption of table "*スキーマ名*"."*テーブル名*" was completed	再暗号化対象のテーブル処理完了時に出力されます。
INFO	I0003	re-encryption process done *処理済件数* rows (*処理済件数割合* %)	再暗号化の進捗率を表示します。

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	2015 年 6 月	• 初版作成
第二版	2015 年 11 月	• 第二版改訂
第三版	2017 年 3 月	• ENCRYPT_TIMESTAMP のデータサイズの記載修正（3.2.1 暗号鍵の管理) • バックアップ対象一覧を訂正(3.6.1 バックアップ対象について) • cipher_key_enable_log、cipher_key_disable_log の戻り値修正(第 5 章 ファンクションリファレンス) • GROUP BY 句の制限事項関連の修正 (付録 D.注意制限事項) • EXPLAIN ANALYZE に関する注意事項を追加(付録 D.注意制限事項) • 行セキュリティポリシー定義に関する注意事項を追加(付録 D.注意制限事項) • kms_info.properties ファイルの注意事項を追加(付録 D.注意制限事項)
第四版	2017 年 6 月	• psqlODBC の注意・制限事項記載
第五版	2017 年 9 月	• psqlODBC の対応記載 • libpq に関する注意事項追加 (付録 D.注意制限事項) • search_path に関する注意事項追加 (付録 D.注意制限事項) • データベースを削除する際の注意事項追加 (付録 D.注意制限事項) • サポート対象外の PostgreSQL の contrib パッケージ追加 (付録 D.注意制限事項) • サポート対象外の PostgreSQL の手続き言語 追加 (付録 D.注意制限事項) • サポート対象外の PostgreSQL のその他機能追加 (付録 D.注意制限事項)
第六版	2017 年 12 月	• PostgreSQL 9.6 に対応したことを追記(第 1 章 概要) • AES-NI に対応したことを追記(第 1 章 概要、第 3 章 運用、付録 D.注意制限事項) • サポート対象外の PostgreSQL の機能としてパラレルクエリを追加(付録 D.注意制限事項) • ENCRYPT_TIMESTAMP のサイズ低減したことを追記(第 3 章 運用)
第七版	2018 年 4 月	• 表記規則の追記 • 最新の情報の入手先の追記 • 章構成の変更 - インストールの概要を追記 - セットアップの方法として 1 つの章にまとめられていた新規インストール、再インストール、アップグレードインストールなどを章ごとに分割 - エラーメッセージを付録に変更

<table>
<tr><th>版数</th><th>発行日</th><th>改訂履歴</th></tr>
<tr><td></td><td></td><td>
- 注意事項の各項目を関連する箇所に記載。併せて注意事項の章を廃止
- 禁則文字の章を廃止
- ライセンスの章を『透過的暗号化機能 利用の手引』に移動

• 実行コマンドや実行例を修正（全体）
• メンテナンス機能の廃止に伴い関連する記述を削除（全体）
• PostgreSQL 10 に対応したことを追記(第 3 章動作環境の確認とインストール前の準備)
• Windows プラットフォームに対応したことを追記(第 3 章動作環境の確認とインストール前の準備)
• 暗号化データ型の整数型に対応したことを追記（第 1 章はじめに）
• 用語の変更（全体）
<table>
<tr><th>V1.1.4 まで</th><th>V1.2.0 以降</th></tr>
<tr><td>暗号鍵管理方式</td><td>モード</td></tr>
<tr><td>AWS KMS 管理方式</td><td>AWS KMS モード</td></tr>
<tr><td>ローカル鍵管理方式</td><td>標準 TDE モード</td></tr>
<tr><td>－</td><td>簡易 TDE モード</td></tr>
</table>
</td></tr>
<tr><td>第八版</td><td>2020 年 9 月</td><td>
• サポート対象外の PostgreSQL の機能として JIT コンパイルを追加(2.7 動作環境)
• OpenSSL 1.1.1 系（RHEL8）を追記(2.7.1.1 ハードウェア要件)
• PostgreSQL11 に対応したことを追記(2.7.1.2 ソフトウェア要件)
• Red Hat Enterprise Linux 8.1 以上に対応したことを追記(2.7.1.2 ソフトウェア要件、2.7.2.2 ソフトウェア要件)
• Windows Server 2019 に対応したことを追記(2.7.1.2 ソフトウェア要件、2.7.2.2 ソフトウェア要件)
• 論理バックアップについて簡易 TDE モードの記載を追加(4.5.4 論理バックアップについて)
</td></tr>
<tr><td>第九版</td><td>2021 年 4 月</td><td>
• PostgreSQL12 に対応したことを追記(2.7.1.2 ソフトウェア要件)
• インストール媒体の JDBC ドライバーファイル名、ODBC ドライバーファイル名を更新(2.7.2.2 ソフトウェア要件)
</td></tr>
<tr><td>第十版</td><td>2022 年 4 月</td><td>
• PostgreSQL13 に対応したことを追記(2.7.1.2 ソフトウェア要件)
• インストール媒体の JDBC ドライバーファイル名、ODBC ドライバーファイル名を更新(2.7.2.2 ソフトウェア要件)
</td></tr>
<tr><td>第十一版</td><td>2023 年 1 月</td><td>
• PostgreSQL12 、 PostgreSQL 13 に Windows Server 2016 、 Windows Server 2019 が対応したことを追記(2.7.1.2 ソフトウェア要件、2.7.2.2 ソフトウェア要件)
• 必要パッケージ（Windows）を更新(2.7.1.2 ソフトウェア要件)
• インストール媒体の JDBC ドライバーファイル名を更新(2.7.2.2 ソフトウェア要件)
• ライセンスから Java™ SE Runtime Environment を削除(付録 C. ライセンス)
</td></tr>
</table>

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
3	pgcrypto	PostgreSQL License
4	libpq	PostgreSQL License
5	The PostgreSQL JDBC Driver	BSD License
6	AWS SDK for Java	Apache Software License Ver2.0
7	Apache log4j	Apache Software License Ver2.0
8	Apache Commons Logging™	Apache Software License Ver2.0
9	Apache Commons Lang™	Apache Software License Ver2.0
10	Apache Commons Codec™	Apache Software License Ver2.0
11	Apache HttpComponents™	Apache Software License Ver2.0
12	Jackson 1.x	Apache Software License Ver2.0
13	Jackson 2.x	Apache Software License Ver2.0
14	Joda-Time	Apache Software License Ver2.0

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

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

    Portions Copyright (c) 1996-2022, 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
POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS
ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO
PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

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.

Copyright (c) 1997, PostgreSQL Global Development Group
Copyright (c) 2015-2022, NEC Corporation
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

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

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"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ー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。

■ 暗号化データ型

透過的暗号化機能が提供するデータを暗号化して保存するためのデータ型のことです。 text 型、bytea 型、numeric 型、timestamp 型、整数データ型（smallint 型、integer 型、bigint 型）に対応する encrypt_text 型、encrypt_bytea 型、 encrypt_numeric 型、encrypt_timestamp 型、encrypt_integer 型の総称です。

■ 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。 標準 TDE モードと簡易 TDE モードではデータを暗号化する実際の鍵と同一のものです。一方、AWS KMS モードを利用している場合は、 データの暗号化に使用するための鍵を CMK によって暗号化したものとなります。

＜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 AWS KMS を使用せずに独自に管理し、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。Transparent Data Encryption for PostgreSQL V1.1.4 までのローカル鍵管理方式のことです。

＜マ行＞

■ メンテナンス機能

Transparent Data Encryption for PostgreSQL のデータベース診断・復旧機能全体のことを指します。

＜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。

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 AWS KMS を使用せずに独自に管理する方式のことです。

＜ A-Z ＞

■ AWS KMS

Amazon Web Services が提供するデータの暗号化に使用される暗号化キーの作成と管理を容易にするマネージド型サービス である AWS Key Management Service の略称です。

■ AWS KMS 管理

透過的暗号化機能で暗号鍵を AWS KMS を使用してデータを暗号化する鍵を管理する方式のことです。 この方式を利用することで、よりセキュアにデータの暗号化を行う鍵を管理することができます。

■ CMK

AWS KMS が管理・提供する Customer Master Key の略称です。AWS KMS 上でデータを暗号化・復号する際に利用される鍵ですが、 本製品の透過的暗号化機能では、データを暗号化する鍵をさらに CMK を利用して暗号化することで データベースの鍵が漏えいすることを防いでいます。 CMK の詳細については AWS KMS のドキュメントをご参照ください。

■ Native 形式

メンテナンス機能が使用する独自データファイル形式のことです。 テーブルをサルベージした場合に出力されるデータのフォーマットとなります。

■ 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	ＯＳＳＤＢＴＤＥ０２－１１	貴社名 所 属	
マニュアル名	Transparent Data Encryption for PostgreSQL 列単位暗号化 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－Ａ０７０４）

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) SI 事業推進部門 データ活用基盤事業統括部 データモダナイズオファリング開発グループ宛

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
列単位暗号化 透過的暗号化機能 利用の手引

ＯＳＳＤＢＴＤＥ０２－１１

2023 年 1 月 第十一版 発行

日本電気株式会社

©NEC Corporation 2015-2023