



Thales CipherTrust Transparent Encryption Implementation Guide

リリース *1.0.0*

EDB

2022 年 06 月 24 日

目次

第 1 章	‘Partner Information’	3
第 2 章	‘Solution Summary’	5
第 3 章	‘Configuring the Integration’	7
3.1	前提条件	7
3.2	Postgres ホスト	7
3.3	CipherTrust マネージャー	7
3.4	CipherTrust Manager の設定	7
3.5	CTE エージェントのインストール	8
第 4 章	‘Using Thales CipherTrust Transparent Encryption’	9
4.1	サンプルユーザーシナリオ	9
第 5 章	‘Certification Environment’	11
第 6 章	‘Appendix’	13
6.1	サンプル config.yml ファイル	13

EDB GlobalConnect テクノロジーパートナー実装ガイド

< h3 style= “text-align: center;” >Thales CipherTrust の透過的暗号化

第1章 ‘Partner Information’

1	2
Partner Name	Thales
Partner Product	CipherTrust Transparent Encryption
Web Site	https://cpl.thalesgroup.com/encryption
Version & Platform	7.1.0, Available platforms: Windows, Linux
Product Description	CipherTrust Transparent Encryption (CTE) delivers data-at-rest encryption with centralized key management, privileged user access control, and detailed data access audit logging. This protects data wherever it resides: on-premises, across multiple clouds and within big data, and container environments.

第2章 ‘Solution Summary’

タレスの CipherTrust Transparent Encryption は、一元化されたキー管理、特権ユーザーアクセス制御、詳細なデータアクセス監査ログに裏打ちされたファイルシステムレベルの暗号化により、Postgres データベースとバックアップの保存データを保護します。CipherTrust Transparent Encryption を使用すると、お客様はコンプライアンス義務を満たしながら、オンプレミスとクラウドの両方の機密性の高い規制データに Postgres を採用できます。CipherTrust Transparent Encryption は、EDB Postgres Advanced Server、および BDR（双方向レプリケーション）クラスターの一部としての EDB Postgres Extended Server、および Barman で認定されています。

注釈: EDB Postgres Extended Server は、BDR（双方向レプリケーション）および Barman を備えた EDB Postgres Extended Server *を表します。

第3章 ‘Configuring the Integration’

CipherTrust 透過的暗号化（CTE）ソリューションを実装するには、次のコンポーネントが必要です。

- Postgres サーバーがインストールされ、動作していること。 - CipherTrust Manager がインストールされ、動作していること。 - CipherTrust Manager に登録されている Postgres ホストにインストールされている CTE エージェント。

次の図は、CTE ソリューションの基本的なフローを示しています。

3.1 前提条件

3.2 Postgres ホスト

- Postgres サーバーがインストールされ、実行されていることを確認します。
- CentOS 7 の場合、次のリポジトリをインストールする必要があります。

```
sudo yum install -y lsof
```

3.3 CipherTrust マネージャー

1. CipherTrust Manager がインストールされ実行されていることを確認します。

3.4 CipherTrust Manager の設定

1. CipherTrust Manager（CM）Web GUI にログオンし、次の手順を実行します。
 1. 登録トークンを作成します。
 - a. キーとアクセスの管理 に移動し、登録トークンを選択します。このトークンは、CM への CTE エージェントの登録に使用されます。
 - b. NewRegistrationToken を選択して、新しい登録トークンを作成します。

次のスクリーンショットは、**edb** という名前で作成された登録トークンを示しています。

2. ユーザーセットを作成します。
 - a. CTE に移動し、[ポリシー]、[ポリシー要素]、[ユーザーセット] の順に選択します。

- b. Create User Set を選択して、新しいユーザーセットを作成します。

次のスクリーンショットに示すように、Postgres、EnterpriseDB、および Barman のユーザーセットを作成します。

3. ポリシーを作成する

- a. ポリシー に戻り、 **ポリシーの作成** を選択します。

****次のスクリーンショットは、ライブデータ変換（LDT）ポリシーの postgres-policy、epas-policy、および barman-policy を示しています。**

注釈: ポリシーには、手順 2 でそれぞれ作成したユーザーセット ****Postgres****と ****EnterpriseDB****と、ポリシー用の同じキールールが含まれます。

3.5 CTE エージェントのインストール

Postgres ホストに CTE エージェントをインストールするには、Thales の次のガイドを参照してください。

[CTE Agent Quick Start Guide](#)

[CTE Agent Advanced Installation Guide](#)

注釈: インストール中に CipherTrust Manager の登録トークンとホストアドレスが必要になります。

CTE エージェントが正常にインストールされたら、Postgres ホストが CM に登録されていることを確認します。

1. CM Web GUI にログオンし、 **CTE** に移動します。
2. **クライアント**を選択します。以下に示すように、クライアントのステータスは **Healthy** と表示されます（ステータスが更新されるまで数秒かかる場合があります）。

次のスクリーンショットは、CipherTrust Manager に登録されたクライアントを示しています。

第4章 ‘Using Thales CipherTrust Transparent Encryption’

CTE は、ファイルレベルまたはストレージデバイスレベルでデータを保護します。(Postgres) ホストで実行されている CTE エージェントは、関連付けられたポリシーを適用することにより GuardPoint の背後にあるファイルを管理し、ログのために CipherTrust Manager にデータアクセスイベントを通信します。GuardPoint は通常、Linux マウントポイントまたは Windows ボリュームに関連付けられますが、ディレクトリサブツリーに関連付けられる場合もあります。

次の図は、CTE アーキテクチャを示しています。

4.1 サンプルユーザーシナリオ

このセクションでは、BDR ホストを備えた EDB Postgres Advanced Server および EDB Postgres Extended Server に CTE ソリューションを展開するサンプルユーザーシナリオについて説明します。

- **EDBPostgresAdvancedServer**
- **BDR を使用した EDBPostgres 拡張サーバー**

EDBPostgresAdvancedServer (シングルインスタンス)

1. Postgres ホストに CTE エージェントをインストールします。
2. Postgres ホストにログインし、postgres サーバーを停止します。
3. postgres ホストで **epas-policy** ポリシーを使用して、CM Web GUI を介して GuardPoint を作成します。EDB Postgres Advanced Server ホストで次のディレクトリを **保護パス**として設定します（ホストで PGDATA が `/var/lib/edb/as13/data` に設定されていると仮定します）。
4. Postgres ホストで Postgres サーバーをユーザー **enterprisedb** として再起動します。ssh（sudo ではなく）を使用してログインしていることを確認します。

BDR を使用した EDBPostgres 拡張サーバー-常にオン

次の図は、**BDR-Always-ON** アーキテクチャを示しています。詳細については、[BDR-Always-ON Architecture](#) のドキュメントを参照してください。

注釈: このドキュメントには EDB アクセス資格情報が必要です。

1. すべての postgres ノードと barman ノードに CTE エージェントをインストールします。

2. データセンター A (DC A) のバーマンノードのディレクトリ `/var/lib/barman/<server-name>` に `barman-policy` ポリシーを使用して、CM Web GUI を介して GuardPoint を作成します。次のスクリーンショットは、barman ノード用に作成された GuardPoint を示しています。

3. データセンター A のスタンバイノードにログインし、postgres サーバを停止します。

4. PGDATA ディレクトリ `/opt/postgres/data` で `postgres-policy` ポリシーを使用して、CM Web GUI を介してスタンバイノードに GuardPoint を作成します。

5. スタンバイノードでユーザー **postgres** として Postgres サーバーを再起動します。ssh (sudo ではなく) を使用してログインしていることを確認します。

6. データセンター A のシャドウマスターノードにログインし、postgres サーバーを停止します。

7. PGDATA ディレクトリ `/opt/postgres/data` で `postgres-policy` ポリシーを使用して、CM Web GUI を介してシャドウマスターノードに GuardPoint を作成します。

8. シャドウマスターノードで Postgres サーバーをユーザー **postgres** として再起動します。ssh (sudo ではなく) を使用してログインしていることを確認します。

9. データセンター A の Lead Master ノードにログインし、postgres サーバーを停止します。

10. PGDATA ディレクトリ `/opt/postgres/data` で `postgres-policy` ポリシーを使用して、CM Web GUI を介してリードマスターノードに GuardPoint を作成します。

11. ユーザー **postgres** として Lead Master ノードで Postgres サーバーを再起動します。ssh (sudo ではなく) を使用してログインしていることを確認します。

次のスクリーンショットは、データセンター A のリードマスター用に作成された GuardPoint を示しています。

12. データセンター B (DC B) の postgres と barman ノードに対して、手順 2~11 を繰り返します。

第5章 ‘Certification Environment’

1	2
Certification Test Date	May 19 2021
EDB Postgres Advanced Server	13.2.5
OS	CentOS Linux 7 (Core)
Memory	2G
Processor	Intel® Xeon® Processor SP Family (“Skylake”)
Cloud Platform	OpenStack (Kilo)
CPU(s)	1
Core(s) per socket	1
Socket(s)	1
Storage	80 GB
CipherTrust Transparent Encryption	7.0.0.99

1	2
Certification Test Date	May 19 2021
EDB Postgres Extended Server	11
Postgres-BDR	3.6.25
OS	CentOS Linux 7
Cloud Platform	AWS
Deployment Tool	tpaexec v20.11

注釈: デプロイメントの詳細については、付録の sample config.yml ファイルを参照してください。

第6章 ‘Appendix’

6.1 サンプル config.yml ファイル

```
- - -
architecture: BDR-Always-ON
cluster_name: bdrtest-always-on-aws
cluster_tags:
  Owner: Aziz Rahman
cluster_bucket: bdr-alwayson-azizrahman

cluster_rules:

-   cidr_ip: 0.0.0.0/0
    from_port: 22
    proto: tcp
    to_port: 22

-   cidr_ip: 10.35.104.176/24
    from_port: 0
    proto: tcp
    to_port: 65535

-   cidr_ip: 10.35.250.208/24
    from_port: 0
    proto: tcp
    to_port: 65535
ec2_ami:
  Name: RHEL-7.8_HVM_GA-20200225-x86_64-1-Hourly2-GP2
  Owner: 309956199498
ec2_instance_reachability: public
ec2_vpc:
  Name: BDR-Test
  cidr: 10.35.0.0/16

cluster_vars:
  bdr_database: bdrdb
  bdr_node_group: bdrgroup
  enable_harp: false
```

(次のページに続く)

(前のページからの続き)

```
extra_postgres_extensions:
- pglogical
postgres_coredump_filter: 0xff
preferred_python_version: python3
repmgr_failover: manual
tpa_2q_repositories:
- products/2ndqpostgres/release
packages:
  RedHat:
    - lsof

use_volatile_subscriptions: false

locations:

- Name: a
  az: us-east-1a
  region: us-east-1
  subnet: 10.35.104.0/24

- Name: b
  az: us-east-1b
  region: us-east-1
  subnet: 10.35.250.0/24

instance_defaults:
  default_volumes:
    - device_name: root
      encrypted: true
      volume_size: 16
      volume_type: gp2
    - device_name: /dev/xvdf
      encrypted: true
      vars:
        volume_for: postgres_data
        volume_size: 64
        volume_type: gp2
  platform: aws
  type: t3.micro
  vars:
    ansible_user: ec2-user

instances:

- Name: bdr1-a
```

(次のページに続く)

(前のページからの続き)

```
backup: barman-a
location: a
node: 1
role:
- primary
- bdr

- Name: bdr2-a
location: a
node: 2
role:
- primary
- bdr

- Name: bdr3-b
backup: barman-b
location: b
node: 3
role:
- primary
- bdr

- Name: bdr4-b
location: b
node: 4
role:
- primary
- bdr

- Name: standby-a
location: a
node: 5
role:
- primary
- bdr
- readonly
upstream: bdr2-a

- Name: standby-b
location: b
node: 6
role:
- primary
- bdr
- readonly
```

(次のページに続く)

(前のページからの続き)

```
upstream: bdr4-b

- Name: barman-a
  location: a
  node: 7
  role:
    - barman
  volumes:
    - device_name: /dev/xvdf
      encrypted: true
      vars:
        volume_for: barman_data
        volume_size: 128
        volume_type: gp2

- Name: barman-b
  location: b
  node: 8
  role:
    - barman
  volumes:
    - device_name: /dev/xvdf
      encrypted: true
      vars:
        volume_for: barman_data
        volume_size: 128
        volume_type: gp2

- Name: pgbouncer-haproxy-a
  location: a
  node: 9
  role:
    - pgbouncer
    - haproxy
  vars:
    haproxy_backend_servers:
      - bdr1-a
      - bdr2-a
  volumes:
    - device_name: /dev/xvdf
      volume_type: none

- Name: pgbouncer-haproxy-b
  location: b
  node: 10
```

(次のページに続く)

(前のページからの続き)

```
role:
- pgbouncer
- haproxy
vars:
  haproxy_backend_servers:
    - bdr3-b
    - bdr4-b
volumes:
- device_name: /dev/xvdf
  volume_type: none
```