EDB Postgres and Entrust KeyControl

Integration Guide

2024-11-21

Table of Contents

Chapter 1. Introduction

Entrust KeyControl is a Key Management System (KMS) offers functionalities to create, manage, distribute, and safeguard cryptographic keys. It is deployed as a cluster of virtual appliances that integrate with FIPS 140-2-compliant third-party hardware security modules (HSM) to securely store keys.

Using EDB Postgres Advanced Server or EDB Postgres Extended Server TDE capabilities along with Entrust KeyControl for key management protect sensitive data wherever those data reside.

1.1. Documents to read first

- Entrust KeyControl Online Documentation Set
- Installing Postgres
- Transparent Data Encryption overview
- Transparent Data Encryption
- PyKMIP Installation

1.2. Product configuration

Product	Version	Notes
RedHat Linux	9.4	Red Hat Enterprise Linux release 9.4 (Plow)
EDB Postgres	16.4.1	PostgreSQL 16.4 (EnterpriseDB Advanced Server 16.4.1)
KeyControl Vault KMS	10.3.1	KMIP Vault installed and deployed per your environment
Python	3.9	Installed on the Linux server

Chapter 2. Procedures

2.1. Install Postgres on RedHat Linux 9 x86_64

2.1.1. Set up the EDB repository

1. Determine if your repository exists:

   ```
   % sudo dnf repolist | grep enterprisedb
   ```

   If no output is generated, the repository isn't installed.

2. Go to EDB repositories (https://www.enterprisedb.com/repos-downloads).
3. Select the button that provides access to the EDB repository.
4. Select the platform and software that you want to download.
   a. Platform: **RHEL 9 x86_64**
   b. Software: **EDB Postgres Advanced Server (Version 16)**
5. Follow the instructions for setting up the EDB repository.
   a. Setup the repository automatically

   ```
   % curl -1sLf
   'https://downloads.enterprisedb.com/amtZGfLa0ZhdjqNzh416xoa03f5knUmY/enterprise/setup.rpm.sh' | sudo
   -E bash
   ```

   ```
   Executing the  setup script for the 'enterprisedb/enterprise' repository ...

      OK: Checking for required executable 'curl' ...
      OK: Checking for required executable 'rpm' ...
      OK: Detecting your OS distribution and release using system methods ...
        : ... Detected/provided for your OS/distribution, version and architecture:
        :
        : ... distro=rhel  version=9.4  codename=Plow  arch=x86_64
        :
      OK: Importing 'enterprisedb/enterprise' repository GPG keys into rpm ...
      OK: Checking for available package manager (DNF/Microdnf/YUM/Zypper) ...
        : ... Detected package manager as 'dnf'
      OK: Checking for dnf dependency 'dnf-plugins-core' ...
      OK: Checking if upstream install config is OK ...
      OK: Fetching 'enterprisedb/enterprise' repository configuration ...
      OK: Installing 'enterprisedb/enterprise' repository via dnf ...
   Importing GPG key 0x9F1EF813:
    Userid     : "Cloudsmith Package (enterprisedb/enterprise) <support@cloudsmith.io>"
    Fingerprint: 31A4 CF09 0B3A E265 F131 58DE E71E B082 9F1E F813
    From       :
   https://downloads.enterprisedb.com/amtZGfLa0ZhdjqNzh416xoa03f5knUmY/enterprise/gpg.E71EB0829F1EF813.k
   ey
   Importing GPG key 0x9F1EF813:
    Userid     : "Cloudsmith Package (enterprisedb/enterprise) <support@cloudsmith.io>"
    Fingerprint: 31A4 CF09 0B3A E265 F131 58DE E71E B082 9F1E F813
   ```

```
 From       :
https://downloads.enterprisedb.com/amtZGfLa0ZhdjqNzh416xoa03f5knUmY/enterprise/gpg.E71EB0829F1EF813.k
ey
Importing GPG key 0x9F1EF813:
 Userid     : "Cloudsmith Package (enterprisedb/enterprise) <support@cloudsmith.io>"
 Fingerprint: 31A4 CF09 0B3A E265 F131 58DE E71E B082 9F1E F813
 From       :
https://downloads.enterprisedb.com/amtZGfLa0ZhdjqNzh416xoa03f5knUmY/enterprise/gpg.E71EB0829F1EF813.k
ey
   OK: Updating the dnf cache to fetch the new repository metadata ...
   OK: The repository has been installed successfully - You're ready to rock!
```

2.1.2. Install the software packages

Once the repository is configured, run this to install the software packages.

```
% sudo dnf -y install edb-as16-server
```

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:33 ago on Mon 09 Sep 2024 02:45:47 PM EDT.
Dependencies resolved.
================================================================================================
 Package                     Architecture  Version       Repository                        Size
================================================================================================
Installing:
 edb-as16-server             x86_64        16.4.1-1.el9  enterprisedb-enterprise          9.7 k
Installing dependencies:
 edb-as16-server-client      x86_64        16.4.1-1.el9  enterprisedb-enterprise          1.8 M
 edb-as16-server-contrib     x86_64        16.4.1-1.el9  enterprisedb-enterprise          774 k
 edb-as16-server-core        x86_64        16.4.1-1.el9  enterprisedb-enterprise          7.2 M
 edb-as16-server-devel       x86_64        16.4.1-1.el9  enterprisedb-enterprise          6.3 M
 edb-as16-server-docs        x86_64        16.4.1-1.el9  enterprisedb-enterprise           18 k
 edb-as16-server-libs        x86_64        16.4.1-1.el9  enterprisedb-enterprise          712 k
 edb-as16-server-plperl      x86_64        16.4.1-1.el9  enterprisedb-enterprise           71 k
 edb-as16-server-plpython3   x86_64        16.4.1-1.el9  enterprisedb-enterprise          110 k
 edb-as16-server-pltcl       x86_64        16.4.1-1.el9  enterprisedb-enterprise           48 k
 edb-as16-server-sqlprotect  x86_64        16.4.1-1.el9  enterprisedb-enterprise           96 k
 lz4                         x86_64        1.9.3-5.el9   rhel-9-for-x86_64-baseos-rpms     62 k

Transaction Summary
================================================================================================
Install  12 Packages

Total download size: 17 M
Installed size: 67 M
Downloading Packages:
(1/12): edb-as16-server-16.4.1-1.el9.x86_64.rpm                  7.1 kB/s | 9.7 kB     00:01
(2/12): edb-as16-server-contrib-16.4.1-1.el9.x86_64.rpm          412 kB/s | 774 kB     00:01
(3/12): edb-as16-server-client-16.4.1-1.el9.x86_64.rpm           913 kB/s | 1.8 MB     00:01
(4/12): edb-as16-server-docs-16.4.1-1.el9.x86_64.rpm              27 kB/s |  18 kB     00:00
(5/12): edb-as16-server-core-16.4.1-1.el9.x86_64.rpm             4.5 MB/s | 7.2 MB     00:01
(6/12): edb-as16-server-devel-16.4.1-1.el9.x86_64.rpm            4.3 MB/s | 6.3 MB     00:01
(7/12): edb-as16-server-libs-16.4.1-1.el9.x86_64.rpm             637 kB/s | 712 kB     00:01
(8/12): edb-as16-server-plperl-16.4.1-1.el9.x86_64.rpm            84 kB/s |  71 kB     00:00
(9/12): edb-as16-server-plpython3-16.4.1-1.el9.x86_64.rpm        128 kB/s | 110 kB     00:00
(10/12): edb-as16-server-pltcl-16.4.1-1.el9.x86_64.rpm            63 kB/s |  48 kB     00:00
(11/12): edb-as16-server-sqlprotect-16.4.1-1.el9.x86_64.rpm      110 kB/s |  96 kB     00:00
(12/12): lz4-1.9.3-5.el9.x86_64.rpm                               92 kB/s |  62 kB     00:00
------------------------------------------------------------------------------------------------
Total                                                            3.5 MB/s |  17 MB     00:04
Running transaction check
```

```
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                1/1
  Installing       : edb-as16-server-libs-16.4.1-1.el9.x86_64                       1/12
  Running scriptlet: edb-as16-server-libs-16.4.1-1.el9.x86_64                       1/12
  Installing       : edb-as16-server-client-16.4.1-1.el9.x86_64                     2/12
  Running scriptlet: edb-as16-server-client-16.4.1-1.el9.x86_64                     2/12
  Installing       : edb-as16-server-contrib-16.4.1-1.el9.x86_64                    3/12
  Running scriptlet: edb-as16-server-core-16.4.1-1.el9.x86_64                       4/12
  Installing       : edb-as16-server-core-16.4.1-1.el9.x86_64                       4/12
  Running scriptlet: edb-as16-server-core-16.4.1-1.el9.x86_64                       4/12
  Installing       : edb-as16-server-plperl-16.4.1-1.el9.x86_64                     5/12
  Installing       : edb-as16-server-plpython3-16.4.1-1.el9.x86_64                  6/12
  Installing       : edb-as16-server-pltcl-16.4.1-1.el9.x86_64                      7/12
  Installing       : edb-as16-server-sqlprotect-16.4.1-1.el9.x86_64                 8/12
  Running scriptlet: edb-as16-server-sqlprotect-16.4.1-1.el9.x86_64                 8/12
  Installing       : edb-as16-server-devel-16.4.1-1.el9.x86_64                      9/12
  Installing       : lz4-1.9.3-5.el9.x86_64                                        10/12
  Installing       : edb-as16-server-docs-16.4.1-1.el9.x86_64                      11/12
  Installing       : edb-as16-server-16.4.1-1.el9.x86_64                           12/12
  Running scriptlet: edb-as16-server-16.4.1-1.el9.x86_64                           12/12
  Verifying        : edb-as16-server-16.4.1-1.el9.x86_64                            1/12
  Verifying        : edb-as16-server-client-16.4.1-1.el9.x86_64                     2/12
  Verifying        : edb-as16-server-contrib-16.4.1-1.el9.x86_64                    3/12
  Verifying        : edb-as16-server-core-16.4.1-1.el9.x86_64                       4/12
  Verifying        : edb-as16-server-devel-16.4.1-1.el9.x86_64                      5/12
  Verifying        : edb-as16-server-docs-16.4.1-1.el9.x86_64                       6/12
  Verifying        : edb-as16-server-libs-16.4.1-1.el9.x86_64                       7/12
  Verifying        : edb-as16-server-plperl-16.4.1-1.el9.x86_64                     8/12
  Verifying        : edb-as16-server-plpython3-16.4.1-1.el9.x86_64                  9/12
  Verifying        : edb-as16-server-pltcl-16.4.1-1.el9.x86_64                     10/12
  Verifying        : edb-as16-server-sqlprotect-16.4.1-1.el9.x86_64                11/12
  Verifying        : lz4-1.9.3-5.el9.x86_64                                        12/12
Installed products updated.

Installed:
  edb-as16-server-16.4.1-1.el9.x86_64              edb-as16-server-client-16.4.1-1.el9.x86_64
  edb-as16-server-contrib-16.4.1-1.el9.x86_64      edb-as16-server-core-16.4.1-1.el9.x86_64
  edb-as16-server-devel-16.4.1-1.el9.x86_64        edb-as16-server-docs-16.4.1-1.el9.x86_64
  edb-as16-server-libs-16.4.1-1.el9.x86_64         edb-as16-server-plperl-16.4.1-1.el9.x86_64
  edb-as16-server-plpython3-16.4.1-1.el9.x86_64    edb-as16-server-pltcl-16.4.1-1.el9.x86_64
  edb-as16-server-sqlprotect-16.4.1-1.el9.x86_64   lz4-1.9.3-5.el9.x86_64

Complete!
```

2.1.3. Install the EPEL repository

```
% sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:04:36 ago on Mon 09 Sep 2024 02:45:47 PM EDT.
epel-release-latest-9.noarch.rpm                                 38 kB/s |  18 kB     00:00
Dependencies resolved.
================================================================================
 Package                 Architecture  Version       Repository              Size
================================================================================
Installing:
 epel-release            noarch        9-8.el9       @commandline             18 k
```

```
Transaction Summary
================================================================================================
Install  1 Package

Total size: 18 k
Installed size: 26 k
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                        1/1
  Installing       : epel-release-9-8.el9.noarch                                            1/1
  Running scriptlet: epel-release-9-8.el9.noarch                                            1/1
Many EPEL packages require the CodeReady Builder (CRB) repository.
It is recommended that you run /usr/bin/crb enable to enable the CRB repository.

  Verifying        : epel-release-9-8.el9.noarch                                            1/1
Installed products updated.

Installed:
  epel-release-9-8.el9.noarch

Complete!
```

2.1.4. Enable additional repositories to resolve dependencies

```
% ARCH=$( /bin/arch )
% subscription-manager repos --enable "codeready-builder-for-rhel-9-${ARCH}-rpms"
```

```
You are attempting to run "subscription-manager" which requires administrative
privileges, but more information is needed in order to do so.
Authenticating as "root"
Password:
Repository 'codeready-builder-for-rhel-9-x86_64-rpms' is enabled for this system.
```

2.1.5. Disable the built-in PostgreSQL module

```
% sudo dnf -qy module disable postgresql
```

2.1.6. Install the Postgres packages

```
% sudo dnf -y install postgresql<xx>-server postgresql<xx>-contrib
```

Where <xx> is the version of PostgreSQL you are installing. For example, if you are installing version 16, the package name would be `postgresql16-server postgresql16-contrib`.

```
% sudo dnf -y install postgresql16-server postgresql16-contrib

Updating Subscription Management repositories.
enterprisedb-enterprise                                          587  B/s | 659  B     00:01
enterprisedb-enterprise-noarch                                   776  B/s | 659  B     00:00
enterprisedb-enterprise-source                                   723  B/s | 659  B     00:00
Dependencies resolved.
================================================================================================
 Package                    Architecture  Version         Repository                       Size
================================================================================================
Installing:
 postgresql16-contrib       x86_64        16.4-1EDB.el9   enterprisedb-enterprise          724 k
 postgresql16-server        x86_64        16.4-1EDB.el9   enterprisedb-enterprise          6.7 M
Installing dependencies:
 postgresql16               x86_64        16.4-1EDB.el9   enterprisedb-enterprise          1.8 M
 postgresql16-libs          x86_64        16.4-1EDB.el9   enterprisedb-enterprise          334 k

Transaction Summary
================================================================================================
Install  4 Packages

Total download size: 9.5 M
Installed size: 42 M
Downloading Packages:
(1/4): postgresql16-libs-16.4-1EDB.el9.x86_64.rpm                 264 kB/s | 334 kB     00:01
(2/4): postgresql16-16.4-1EDB.el9.x86_64.rpm                      1.3 MB/s | 1.8 MB     00:01
(3/4): postgresql16-contrib-16.4-1EDB.el9.x86_64.rpm              518 kB/s | 724 kB     00:01
(4/4): postgresql16-server-16.4-1EDB.el9.x86_64.rpm               4.2 MB/s | 6.7 MB     00:01
------------------------------------------------------------------------------------------------
Total                                                             3.3 MB/s | 9.5 MB     00:02
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                        1/1
  Installing       : postgresql16-libs-16.4-1EDB.el9.x86_64                                 1/4
  Running scriptlet: postgresql16-libs-16.4-1EDB.el9.x86_64                                 1/4
  Installing       : postgresql16-16.4-1EDB.el9.x86_64                                      2/4
  Running scriptlet: postgresql16-16.4-1EDB.el9.x86_64                                      2/4
  Running scriptlet: postgresql16-server-16.4-1EDB.el9.x86_64                               3/4
  Installing       : postgresql16-server-16.4-1EDB.el9.x86_64                               3/4
  Running scriptlet: postgresql16-server-16.4-1EDB.el9.x86_64                               3/4
  Installing       : postgresql16-contrib-16.4-1EDB.el9.x86_64                              4/4
  Running scriptlet: postgresql16-contrib-16.4-1EDB.el9.x86_64                              4/4
  Verifying        : postgresql16-16.4-1EDB.el9.x86_64                                      1/4
  Verifying        : postgresql16-contrib-16.4-1EDB.el9.x86_64                              2/4
  Verifying        : postgresql16-libs-16.4-1EDB.el9.x86_64                                 3/4
  Verifying        : postgresql16-server-16.4-1EDB.el9.x86_64                               4/4
Installed products updated.

Installed:
  postgresql16-16.4-1EDB.el9.x86_64               postgresql16-contrib-16.4-1EDB.el9.x86_64
  postgresql16-libs-16.4-1EDB.el9.x86_64          postgresql16-server-16.4-1EDB.el9.x86_64

Complete!
```

2.2. Install PyKMIP on the Postgres Linux server

PyKMIP must be installed as `enterprisedb`.

1. Make sure enterprisedb user has sudo privileges.

```
% sudo usermod -aG wheel enterprisedb
```

2. Log in to the postgres Linux server as `enterprisedb`.
3. Check or install Python on the server:

```
% python
```

```
Python 3.9.18 (main, Aug 23 2024, 00:00:00)
[GCC 11.4.1 20231218 (Red Hat 11.4.1-3)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```

4. Install `pip`:

```
% sudo dnf install pip
```

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:04:54 ago on Mon 09 Sep 2024 02:59:02 PM EDT.
Dependencies resolved.
================================================================================================
 Package                 Architecture  Version         Repository                           Size
================================================================================================
Installing:
 python3-pip             noarch        21.2.3-8.el9    rhel-9-for-x86_64-appstream-rpms     2.0 M

Transaction Summary
================================================================================================
Install  1 Package

Total download size: 2.0 M
Installed size: 8.7 M
Is this ok [y/N]: y
Downloading Packages:
python3-pip-21.2.3-8.el9.noarch.rpm                                 1.8 MB/s | 2.0 MB     00:01
------------------------------------------------------------------------------------------------
Total                                                               1.8 MB/s | 2.0 MB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
1/1
  Installing       : python3-pip-21.2.3-8.el9.noarch
1/1
  Running scriptlet: python3-pip-21.2.3-8.el9.noarch
1/1
  Verifying        : python3-pip-21.2.3-8.el9.noarch
1/1
Installed products updated.

Installed:
  python3-pip-21.2.3-8.el9.noarch
```

```
Complete!
```

5. Upgrade `pip`:

```
% /usr/bin/python3.9 -m pip install --upgrade pip
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pip in /usr/lib/python3.9/site-packages (21.2.3)
Collecting pip
  Downloading pip-24.2-py3-none-any.whl (1.8 MB)
.
.
.
Successfully installed pip-24.2
```

6. Upgrade `setuptools`:

```
% sudo pip3 install --upgrade setuptools
```

```
Requirement already satisfied: setuptools in /usr/lib/python3.9/site-packages (53.0.0)
Collecting setuptools
  Downloading setuptools-74.1.2-py3-none-any.whl (1.3 MB)
     |                                                                | 1.3 MB 710 kB/s
Installing collected packages: setuptools
Successfully installed setuptools-74.1.2
```

7. Install `tox`:

```
% sudo pip install tox
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting tox
  Downloading tox-4.18.1-py3-none-any.whl.metadata (5.0 kB)
Collecting cachetools>=5.5 (from tox)
  Downloading cachetools-5.5.0-py3-none-any.whl.metadata (5.3 kB)
.
.
.
Successfully installed cachetools-5.5.0 chardet-5.2.0 colorama-0.4.6 distlib-0.3.8 filelock-3.16.0
packaging-24.1 platformdirs-4.3.2 pluggy-1.5.0 pyproject-api-1.7.1 tomli-2.0.1 tox-4.18.1 virtualenv-
20.26.4
```

8. Install `git`:

```
% sudo dnf install -y git
```

9. Install `cryptography`:

```
% sudo pip install cryptography
```

```
Collecting cryptography
  Downloading cryptography-43.0.1-cp39-abi3-manylinux_2_28_x86_64.whl (4.0 MB)
.
.
.
Successfully installed cffi-1.17.1 cryptography-43.0.1 pycparser-2.22
```

10. Build PyKMIP:

    a. Change directory to the enterprisedb user home directory.

    ```
    % cd ~
    ```

    b. Clone the `pykmip` repo.

    ```
    % git clone https://github.com/openkmip/pykmip.git
    ```

    ```
    Cloning into 'pykmip'...
    remote: Enumerating objects: 7476, done.
    remote: Counting objects: 100% (484/484), done.
    remote: Compressing objects: 100% (226/226), done.
    remote: Total 7476 (delta 266), reused 342 (delta 222), pack-reused 6992 (from 1)
    Receiving objects: 100% (7476/7476), 2.75 MiB | 9.41 MiB/s, done.
    Resolving deltas: 100% (5517/5517), done.
    ```

    c. Do the build.

    ```
    % sudo python3.9 pykmip/setup.py install
    ```

    ```
    running install
    .
    .
    .
    Using /usr/lib/python3.9/site-packages
    Searching for cffi==1.17.1
    Best match: cffi 1.17.1
    Adding cffi 1.17.1 to easy-install.pth file

    Using /usr/local/lib64/python3.9/site-packages
    Searching for pycparser==2.22
    Best match: pycparser 2.22
    Adding pycparser 2.22 to easy-install.pth file

    Using /usr/local/lib/python3.9/site-packages
    Finished processing dependencies for PyKMIP==0.11.0.dev1
    ```

11. Run the PyKMIP tests:

```
% cd ~/pykmip
% ./bin/run_tests.sh
```

```
pep8: install_deps> python -I -m pip install -r /home/xxxxx/pykmip/requirements.txt -r
/home/xxxxx/pykmip/test-requirements.txt
.
.
.
writing output... [100%] installation .. server
generating indices... genindex py-modindex done
writing additional pages... search done
dumping search index in English (code: en)... done
dumping object inventory... done
build succeeded.

The HTML pages are in ../.tox/docs/tmp/html.
  pep8: OK (37.05=setup[27.65]+cmd[9.41] seconds)
  py38: SKIP (0.24 seconds)
  py310: SKIP (0.01 seconds)
  bandit: OK (19.03=setup[13.74]+cmd[5.30] seconds)
  docs: OK (14.05=setup[11.38]+cmd[2.67] seconds)
  congratulations :) (70.52 seconds)
```

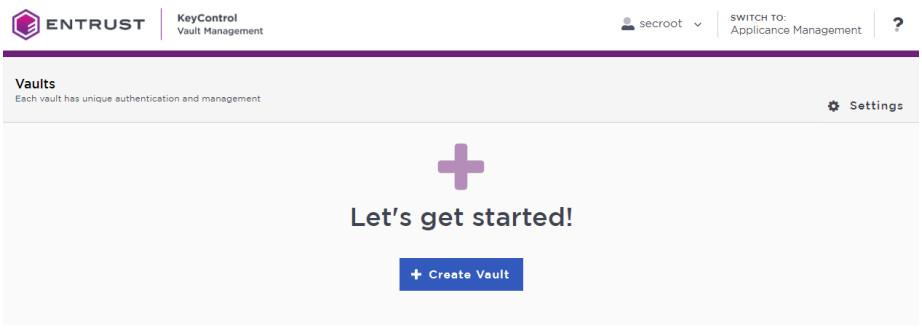
2.3. Install and configure KeyControl Vault

Follow the installation and setup instructions in the *Entrust KeyControl Vault nShield HSM Integration Guide*. You can access it from the Entrust Document Library and from the nShield Product Documentation website.

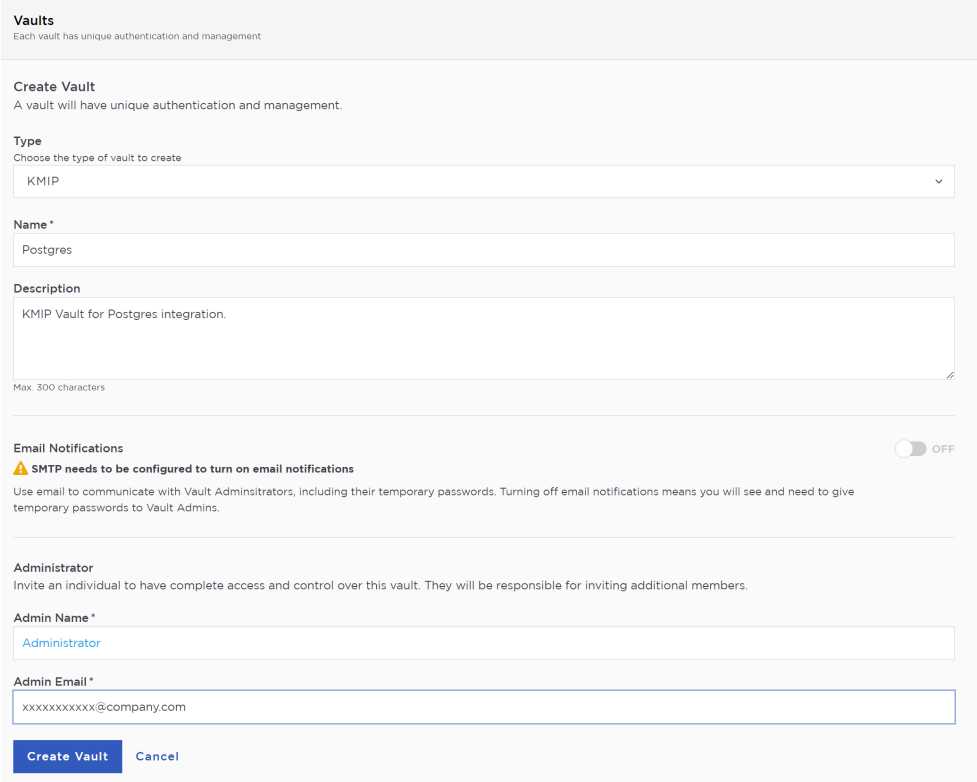
2.3.1. Create a KMIP vault in the KeyControl Vault Server

Refer to the Creating a Vault section of the admin guide for more details about it.

1. Log into the KeyControl Vault Server web user interface:
   a. Use your browser to access the IP address of the server.
   b. Sign in using the **secroot** credentials.
2. If you are not in the Vault Management interface, select **SWITCH TO: Manage Vaults** in the Menu Header.
3. In the KeyControl Vault Management interface, select **Create Vault**.



4. In the **Create Vault** page, create a **KMIP** Vault:
    a. For **Type**, select **KMIP**.
    b. For **Name**, enter the name of the Vault.
    c. For **Description**, enter the description of the Vault.
    d. For **Admin Name**, enter the name of the administrator of the Vault.
    e. For **Admin Email**, enter a valid email for the administrator.



    A temporary password will be emailed to the administrator's email address. This is the password that will be used to sign in for the first time to the KMIP Vaults space in KeyControl. In a closed gap environment where email is not available, the password for the user is displayed when you first create the vault. That can be copied and sent to the user.

5. Select **Create Vault**.

6. Select **Close** when the Vault creation completes.
7. The newly-created Vault is added to the Vault dashboard.
8. After the Vault has been created, the KMIP server settings on the appliance are **enabled**.

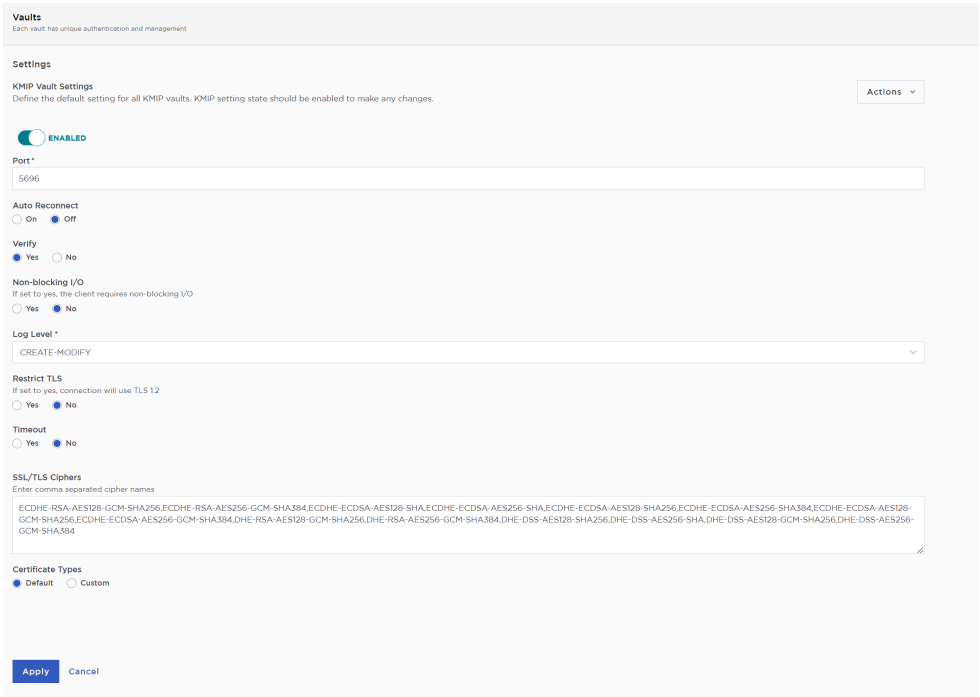
2.3.2. KMIP server settings

The KMIP server settings are set at the KeyControl appliance level and apply to all the KMIP vaults in the appliance. After a KMIP vault is created, they are automatically set to **ENABLED**.

To use external key management and configure the KeyControl Vault KMIP settings, refer to the KMIP Client and Server Configuration section of the admin guide.

When using external key management, as is the case in this solution, the KeyControl server is the KMIP server and the Postgres server is the KMIP client.

1. Select the **Settings** icon on the top right to view/change the KMIP settings.
   a. The defaults settings are appropriate for most applications.
   b. Make any changes necessary.



2. Select **Apply**.

2.3.3. View details for the vault

Select **View Details** when you hover over the vault.

2.4. Establishing trust between the KeyControl Vault Server and Postgres - KeyControl Certificates

Certificates are required to facilitate the KMIP communications from the KeyControl KMIP vault and the Postgres application and conversely. The built-in capabilities in the KeyControl KMIP vault are used to create and publish the certificates.

For more information on how to create a certificate bundle, refer to Establishing a Trusted Connection with a KeyControl Vault-Generated CSR.

2.4.1. Create a CSR for the Postgres Linux server

1. Change your working directory to the `$home` directory of `enterprisedb`:

```
% cd ~
```

2. Create a new certificate request good for 3 years:

```
% openssl req -new -nodes -keyout client1.key -out client1.csr -days 1095
```

```
Ignoring -days without -x509; not generating a certificate
.....................................................................................
+++++
.....................................................................................
+++++
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:FL
Locality Name (eg, city) [Default City]:Sunrise
Organization Name (eg, company) [Default Company Ltd]:Entrust
Organizational Unit Name (eg, section) []:Hurricanes
Common Name (eg, your name or your server's hostname) []:Postgres
Email Address []:xxxxx@company.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Verify that the CSR has been generated.

```
% ls client1.csr
```

4. Download the CSR file on the local machine. You will use it to generate the KMIP client certificate.

2.4.2. Create the certificate bundle to be used by Postgres.

1. Sign in to the KMIP vault that you created. Use the login URL and credentials provided to the administrator of the vault.
2. Select **Security**, then **Client Certificates**.

3. In the **Manage Client Certificate** page, select the **+** icon on the right to create a new certificate.
4. In the **Create Client Certificate** dialog box:
   a. Enter a name in the **Certificate Name** field. (**client1**)
   b. Set the date on which you want the certificate to expire in the **Certificate Expiration** field. (**3 years from today**)
   c. For **Certificate Signing Request (CSR)** upload the **client1** CSR file created earlier.
   d. Select **Create**.

   The new certificates are added to the **Manage Client Certificate** pane.

5. Select the certificate and select the **Download** icon to download the certificate.

   The webGUI downloads `certname_datetimestamp.zip`, which contains a user certification/key file called `certname.pem` and a server certification file called `cacert.pem`.

6. The download zip file contains the following:

    - A `certname.pem` file that includes both the client certificate and private key. In this example, this file is called `client1.pem`.

      The client certificate section of the `certname.pem` file includes the lines "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" and all text between them.

      The private key section of the `certname.pem` file includes the lines "-----BEGIN PRIVATE KEY-----" and "-----END PRIVATE KEY-----" and all text in between them.

    - A `cacert.pem` file which is the root certificate for the KMS cluster. It is always named `cacert.pem`.

    You will use these files in the Postgres KMIP configuration.

7. Transfer the zip file to the Postgres server.

    ```
    % scp client1_2024-09-10-14-24-31.zip enterprisedb@<keycontrol-server-ip-address>:/var/lib/edb/.
    ```

8. Login to the postgres server as `enterprisedb` user and unzip the file.

    ```
    % unzip client1_2024-09-10-14-24-31.zip
    ```

    ```
    Archive:  client1_2024-09-10-14-24-31.zip
      inflating: client1.pem
      inflating: cacert.pem
    ```

    These files will be used during the configuration described in the next section.

2.5. Configure PyKMIP on the Postgres Linux server

PyKMIP must be configured to use the client certificate generated by KeyControl.

1. Log in to the Linux server:

    ```
    $ ssh enterprisedb@<postgres-server-ip-address>
    ```

2. Install KMIP client and CA certificates:

    ```
    % sudo mkdir -p /etc/pykmip/certs
    % sudo cp client1.pem /etc/pykmip/certs/client_cert.pem
    ```

```
% sudo cp client1.key/etc/pykmip/certs/client_private_key.pem
% sudo cp cacert.pem /etc/pykmip/certs/server_ca_cert.pem
% sudo chmod a+r /etc/pykmip/certs/*
```

3. Create the `PYTHONPATH` variable: Edit the `enterprisedb` user's `.bash_profile` file and add the following line to it:

```
export PYTHONPATH=$HOME/pykmip
```

4. Source the file so the environment variable becomes available.

```
% source ~/.bash_profile
```

5. Check if the environment variable is set:

```
% echo $PYTHONPATH

/var/lib/edb/pykmip
```

6. Copy the PyKMIP `policy.json` file into `/etc/pykmip/policy.json`

```
% sudo cp ~/pykmip/examples/policy.json /etc/pykmip/policy.json
```

7. Copy the PyKMIP `pykmip.conf` file into `/etc/pykmip/pykmip.conf`

```
% sudo cp ~/pykmip/examples/pykmip.conf /etc/pykmip/pykmip.conf
```

8. Edit the `/etc/pykmip/pykmip.conf` file.

   Change the **host** parameter to IP address of the KeyControl Vault Server.

   Remember to comment out the last two lines.

```
[client]
host=<KEYCONTROL-VAULT-IP-ADDRESS>
port=5696
keyfile=/etc/pykmip/certs/client_private_key.pem
certfile=/etc/pykmip/certs/client_cert.pem
cert_reqs=CERT_REQUIRED
ssl_version=PROTOCOL_SSLv23
ca_certs=/etc/pykmip/certs/server_ca_cert.pem
do_handshake_on_connect=True
suppress_ragged_eofs=True
#username=example_username
#password=example_password
```

2.6. Test the KMIP service

1. Ping the KeyControl appliance:

```
% ping <keycontrol-vault-ip-address>
```

```
PING <keycontrol-vault-ip-address> (192.168.1.235) 56(84) bytes of data.
64 bytes from <keycontrol-vault-ip-address> (192.168.1.235): icmp_seq=1 ttl=64 time=0.701 ms
64 bytes from <keycontrol-vault-ip-address> (192.168.1.235): icmp_seq=2 ttl=64 time=0.437 ms
64 bytes from <keycontrol-vault-ip-address> (192.168.1.235): icmp_seq=3 ttl=64 time=0.421 ms
64 bytes from <keycontrol-vault-ip-address> (192.168.1.235): icmp_seq=4 ttl=64 time=0.405 ms
^C
--- <keycontrol-vault-ip-address> ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3044ms
```

2. Show information about the TLS certificate of the KMIP server (KMS):

```
% echo | \
  openssl s_client -servername <keycontrol-ip-address>-connect <keycontrol-ip-address>:443 2>/dev/null | \
  openssl x509 -noout -issuer -subject -dates -ext subjectAltName -fingerprint
```

```
issuer=C = US, O = HyTrust Inc., CN = HyTrust KeyControl Certificate Authority
subject=C = US, O = HyTrust Inc., CN = kcv-10p3p1-node-1.interop.local
notBefore=Jun  1 00:00:00 2011 GMT
notAfter=Dec 31 23:59:59 2049 GMT
X509v3 Subject Alternative Name:
    IP Address:<keycontrol-vault-ip-address>, DNS:kcv-10p3p1-node-1, DNS:kcv-10p3p1-node-1.interop.local
SHA1 Fingerprint=90:5C:34:AB:30:47:9F:1A:40:98:08:99:62:07:78:E7:9F:8D:C0:4B
```

3. Test the CA certificate:

```
% openssl s_client -CAfile /etc/pykmip/certs/server_ca_cert.pem -connect <keycontrol-vault-ip-address>:5696
```

```
CONNECTED(00000003)
.
.
.
--
SSL handshake has read 2800 bytes and written 433 bytes
Verification: OK
--
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 67C77D394B46DD966582BDF7E55B41F5B74A2DD8BE03FE3F6C76ADA8860CE414
    Session-ID-ctx:
    Master-Key:
71120791CFDDCA302C0C5A49AB999D8D5924233683A3D883BDFE177397E75444B097596BCC6D524C69338D88085119FA
```

```
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 70 bf ca 92 53 2d b5 29-41 c4 4c 44 5b e3 97 6a   p...S-.)A.LD[..j
    0010 - cf 7c 04 d6 9a a1 29 6f-60 70 d5 11 ab 41 7d 5b   .|....)o`p...A}[
    0020 - 35 d8 ee 96 0c 45 8a 6d-fe af ea 5b 5f 31 55 12   5....E.m...[_1U.
    0030 - 7f 90 74 5c d8 ec 17 69-42 09 38 22 20 03 39 9e   ..t\...iB.8" .9.
    0040 - e5 bd 4a 72 17 81 bd 68-66 e7 9f 6a 01 ec 75 79   ..Jr...hf..j..uy
    0050 - c5 e5 48 b3 85 60 0b 5e-ea c3 d9 c3 bd 5f c9 4f   ..H..`.^....._.O
    0060 - c9 e2 a7 ee 53 18 d2 ca-6c 13 1d 2c 8b 1c 1c dd   ....S...l..,....
    0070 - d2 6b 01 f1 56 1e 3b 6f-04 a0 e4 ba c5 03 1a 8d   .k..V.;o........
    0080 - 24 4f 1f 8f fe b3 7e cc-9f bc 59 87 7f 34 e5 a1   $O....~...Y..4..
    0090 - 49 2e 21 a9 27 3c a0 2e-03 7f 7e f7 53 cc 40 56   I.!.'<....~.S.@V

    Start Time: 1725981994
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
--
closed
```

4. Create an AES 256 AES key using pykmip:

```
% cd ~/pykmip/
% python3.9 ./kmip/demos/pie/create.py -a AES -l 256
```

```
2024-09-10 15:48:00,086 - demo - INFO - Successfully created symmetric key with ID: 74cf4f21-a006-467d-a64f-fbc3fa6ab6d1
```

   The key is created in the KeyControl KMIP vault.

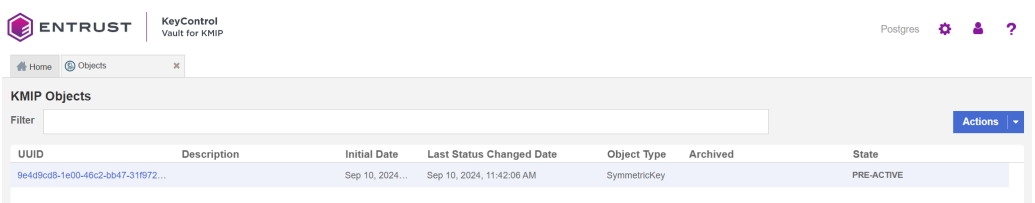
2.6.1. Activate the key

You must activate the key, otherwise it cannot be used for encryption.

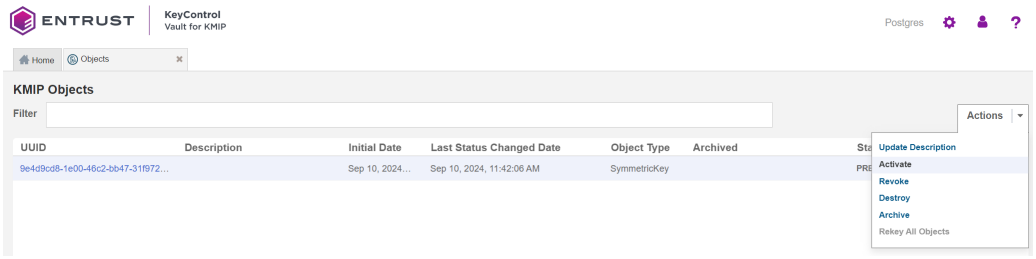
Sign in to the KeyControl KMIP vault and activate the key.

1. Check that the key exists in the KMIP vault.

   Select **Objects** in the **Home** tab in the KMIP vault. You should see the key created earlier listed.

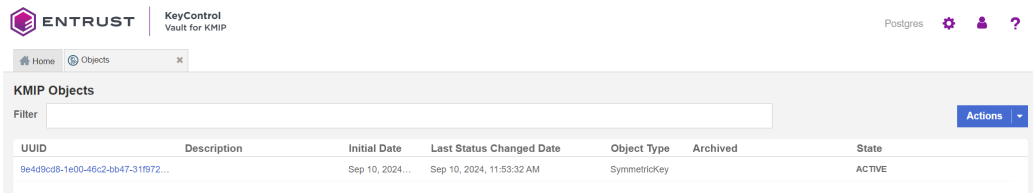


2. Activate the key. Select the key, then select **Actions > Activate**.



In the **Activate KMIP Object** window, select **Activate**.

The status of the key should be now **Active**.



2.7. Install edb-tde-kmip-client and check prerequisites

As the `enterprisedb` user, install the edb-tde-kmip client on your system.

1. Install `edb-tde-kmip-client` on your system:

```
% sudo dnf install -y edb-tde-kmip-client
```

```
Updating Subscription Management repositories.
enterprisedb-enterprise                                          529  B/s | 659  B     00:01
enterprisedb-enterprise-noarch                                   727  B/s | 659  B     00:00
enterprisedb-enterprise-source                                   842  B/s | 659  B     00:00
Dependencies resolved.
================================================================================================
 Package                 Architecture  Version         Repository                           Size
================================================================================================
Installing:
 edb-tde-kmip-client     noarch        1.0-1.el9       enterprisedb-enterprise-noarch        14 k

Transaction Summary
================================================================================================
Install  1 Package

Total download size: 14 k
Installed size: 20 k
Downloading Packages:
edb-tde-kmip-client-1.0-1.el9.noarch.rpm                          11 kB/s |  14 kB     00:01
------------------------------------------------------------------------------------------------
Total                                                             11 kB/s |  14 kB     00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

```
  Preparing        :                                                                1/1
  Installing       : edb-tde-kmip-client-1.0-1.el9.noarch                           1/1
  Verifying        : edb-tde-kmip-client-1.0-1.el9.noarch                           1/1
Installed products updated.

Installed:
  edb-tde-kmip-client-1.0-1.el9.noarch

Complete!
```

2. Locate `edb_tde_kmip_client.py` script:

```
% find / -name edb_tde_kmip_client.py
/usr/edb/kmip/client/edb_tde_kmip_client.py
```

3. Verify encryption with an active key using the `pykmip` variant:

```
% printf secret | python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt \
  --out-file=$HOME/test.bin \
  --pykmip-config-file=/etc/pykmip/pykmip.conf \
  --key-uid='74cf4f21-a006-467d-a64f-fbc3fa6ab6d1' \
  --variant=pykmip
```

4. Check if encryption took place.

```
% ls -al $HOME/test.bin

-rw-r--r--. 1 enterprisedb enterprisedb 32 Sep 10 12:06 test.bin
```

5. Verify decryption with the same key:

```
% python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt \
  --in-file=$HOME/test.bin \
  --pykmip-config-file=/etc/pykmip/pykmip.conf \
  --key-uid='74cf4f21-a006-467d-a64f-fbc3fa6ab6d1' \
  --variant=pykmip

secret
```

2.8. Perform the initial configuration and create an encrypted Postgres database

After you create the key and verify encryption and decryption, you can export `PGDATAKEYWRAPCMD` and `PGDATAKEYUNWRAPCMD` to wrap and unwrap your encryption key and initialize your database.

1. Sign in to your EDB Postgres distribution system as the database superuser. In this example, it's the `enterprisedb` user.

```
$ sudo su - enterprisedb
```

2. Navigate to the `/bin` directory of your executables:

   In this example, it's `/usr/edb/as16/bin`.

```
$ cd /usr/edb/as16/bin
```

3. Perform the initial configuration of the database:

```
% export PGDATAKEYWRAPCMD='python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt \
   --out-file=%p \
  --pykmip-config-file=/etc/pykmip/pykmip.conf \
  --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" \
  --variant=pykmip'

% export PGDATAKEYUNWRAPCMD='python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt \
  --pykmip-config-file=/etc/pykmip/pykmip.conf \
  --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" \
  --in-file=%p --variant=pykmip'
```

4. Check if variables are set.

```
% env | grep PGDATAKEY

PGDATAKEYWRAPCMD=python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt --out-file=%p --pykmip
-config-file=/etc/pykmip/pykmip.conf --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --variant=pykmip
PGDATAKEYUNWRAPCMD=python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --pykmip-config
-file=/etc/pykmip/pykmip.conf --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --in-file=%p
--variant=pykmip
```

5. Perform the initial configuration of the database.

```
% /usr/edb/as16/bin/initdb -D /var/lib/edb/as16/data -y
```

```
The files belonging to this database system will be owned by user "enterprisedb".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.
Transparent data encryption is enabled (128 bits).

creating directory /var/lib/edb/as16/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... America/New_York
creating configuration files ... ok
setting up data encryption ... /var/lib/edbok
```

```
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
creating edb sys ... ok
loading edb contrib modules ...
edb_redwood_bytea.sql
edb_redwood_date.sql
.
.
.
utl_raw_public.sql
utl_raw.plb
edb_gen_redwood.sql
waitstates.sql
installing extension edb_dblink_libpq ... ok
installing extension edb_dblink_oci ... ok
snap_tables.sql
snap_functions.sql
dblink_ora.sql
sys_stats.sql
ok
finalizing initial databases ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and --auth
-host, the next time you run initdb.

Success. You can now start the database server using:

    /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l logfile start
```

6. Start the database server.

```
% /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l $HOME/logfile start

waiting for server to start..... done
server started
```

7. Navigate to the data directory `/var/lib/edb/as16/data` to view the postgresql.conf file.

   Make sure that your `data_encryption_key_unwrap_command`, which you set with export **PGDATAUNWRAPCMD**, is present under the `Authentication` section.

```
# - Authentication -

#authentication_timeout = 1min          # 1s-600s
#password_encryption = scram-sha-256    # scram-sha-256 or md5
#scram_iterations = 4096
#db_user_namespace = off

# GSSAPI using Kerberos
#krb_server_keyfile = 'FILE:${sysconfdir}/krb5.keytab'
#krb_caseins_users = off
#gss_accept_delegation = off

# - SSL -

#ssl = off
#ssl_ca_file = ''
```

```
#ssl_cert_file = 'server.crt'
#ssl_crl_file = ''
#ssl_crl_dir = ''
#ssl_key_file = 'server.key'
#ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL' # allowed SSL ciphers
#ssl_prefer_server_ciphers = on
#ssl_ecdh_curve = 'prime256v1'
#ssl_min_protocol_version = 'TLSv1.2'
#ssl_max_protocol_version = ''
#ssl_dh_params_file = ''
#ssl_passphrase_command = ''
#ssl_passphrase_command_supports_reload = off

# - Data Encryption -

data_encryption_key_unwrap_command = 'python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt
--pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --in-file=%p
--variant=pykmip'
```

8. Ensure encryption is enabled.

   Run the following command and make sure the **Data encryption version** and **Data encryption key length** are set.

```
% /usr/edb/as16/bin/pg_controldata /var/lib/edb/as16/data
```

```
pg_control version number:            1501
Catalog version number:               202307071
.
.
.
Float8 argument passing:              by value
Data page checksum version:           0
Data encryption version:              1
Data encryption key length:           128
Mock authentication nonce:            ac2f2e58c2ace59d45af6642d3515e689891d8d1388c65c9d8bc827732710e59
```

9. Create a database for the enterprisedb user to do the testing.

```
% /usr/edb/as16/bin/createdb --owner enterprisedb hr
```

10. Connect to the `hr` database inside `psql`.

```
% /usr/edb/as16/bin/psql hr

psql (16.4)
Type "help" for help.

hr=#
```

11. Create columns to hold department numbers, unique department names, and locations:

```
hr=# CREATE TABLE public.dept (deptno numeric(2) NOT NULL CONSTRAINT dept_pk PRIMARY KEY, dname varchar(14)
CONSTRAINT dept_dname_uq UNIQUE, loc varchar(13));
CREATE TABLE
```

12. Insert values into the dept table.

```
hr=# INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');
INSERT 0 1
hr=# INSERT into dept VALUES (20,'RESEARCH','DALLAS');
INSERT 0 1
```

13. View the table data by selecting the values from the table.

```
hr=# SELECT * FROM dept;

 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
(2 rows)
```

2.9. Testing

The key used to encrypt the data in the database is fetched on startup of the database. We test the encryption of the data by the key stored in the KMIP vault by disabling the network in the Postgres Linux server and then attempt to start the database. The system should fail and report an error.

1. Stop the database.

```
% /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l $HOME/logfile stop

waiting for server to shut down.... done
server stopped
```

2. Kill the network connection on the server so it cannot contact the KeyControl Server.
3. Start the database service and see what happens.

```
% /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l $HOME/logfile start
waiting for server to start..... stopped waiting
pg_ctl: could not start server
Examine the log output.
```

4. Look at the log file.

```
% cat $HOME/logfile
```

```
2024-09-10 17:04:26 EDT LOG:  redirecting log output to logging collector process
2024-09-10 17:04:26 EDT HINT:  Future log output will appear in directory "log".
An error occurred while connecting to appliance <keycontrol-server-ip-address>: [Errno 101] Network is
unreachable
could not open client connection: [Errno 101] Network is unreachable
Traceback (most recent call last):
  File "/usr/edb/kmip/client/edb_tde_kmip_client.py", line 113, in <module>
    main()
  File "/usr/edb/kmip/client/edb_tde_kmip_client.py", line 70, in main
    with pykmip_client:
  File "/var/lib/edb/pykmip/kmip/pie/client.py", line 1753, in __enter__
    self.open()
  File "/var/lib/edb/pykmip/kmip/pie/client.py", line 173, in open
    self.proxy.open()
  File "/var/lib/edb/pykmip/kmip/services/kmip_client.py", line 285, in open
    six.reraise(*last_error)
  File "/usr/lib/python3.9/site-packages/six.py", line 709, in reraise
    raise value
  File "/var/lib/edb/pykmip/kmip/services/kmip_client.py", line 274, in open
    self.socket.connect((self.host, self.port))
  File "/usr/lib64/python3.9/ssl.py", line 1376, in connect
    self._real_connect(addr, False)
  File "/usr/lib64/python3.9/ssl.py", line 1363, in _real_connect
    super().connect(addr)
OSError: [Errno 101] Network is unreachable
2024-09-10 17:38:10 EDT FATAL:  could not run command "python3.9
/usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/pykmip.conf --key
-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --in-file=pg_encryption/key.bin --variant=pykmip": child
process exited with exit code 1
2024-09-10 17:38:10 EDT LOG:  database system is shut down
```

It cannot reach the KeyControl Server so it cannot start.

5. Re-establish the network connection and attempt start the database service again.

```
% /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l $HOME/logfile start
waiting for server to start..... done
server started
```

The database is able to start.

6. Connect to the database.

```
% /usr/edb/as16/bin/psql hr
```

7. Attempt to read the data in the dept table.

```
hr=# SELECT * FROM dept;

 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
```

```
     20 | RESEARCH   | DALLAS
(2 rows)
```

You can view the data again.

2.10. Key rotation

To change the master encryption key, manually run the unwrap command specifying the old key. Then feed the result into the wrap command specifying the new key. You can perform these operations while the database server is running. The wrapped data key in the file is used only on startup. It isn't used while the server is running.

Here is our Wrap Command:

```
python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt --out-file=%p --pykmip-config
-file=/etc/pykmip/pykmip.conf --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1"  --variant=pykmip
```

Here is our Unwrap Command:

```
python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/pykmip.conf --key
-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --in-file=%p --variant=pykmip
```

1. Create a new AES 256 AES key using `pykmip`.

   ```
   % cd ~/pykmip/
   % python3.9 ./kmip/demos/pie/create.py -a AES -l 256
   ```

   ```
   2024-09-11 10:05:39,278 - demo - INFO - Successfully created symmetric key with ID: 34816dab-24e5-4635-
   9990-00684b84a8c4
   ```

   The key is created in the KMIP vault.

2. Activate the key as documented earlier.
3. Change directory to the `pg_encryption` folder in the database.

   ```
   % cd $PGDATA/pg_encryption
   ```

4. Save the original `key.bin` file.

   ```
   % cp key.bin key.bin.original
   ```

5. Create the new `key.bin` file based on the new key we just created.

```
python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt --pykmip-config-file=/etc/pykmip/pykmip.conf
--key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --in-file=key.bin --variant=pykmip |
python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py encrypt --out-file=key.bin --pykmip-config
-file=/etc/pykmip/pykmip.conf --key-uid="34816dab-24e5-4635-9990-00684b84a8c4"  --variant=pykmip
```

   This is a single command, that is: **UNWRAPCMD** with the Old Key | **WRAPCMD** with the new key

   A new `key.bin` file should have been created.

6. Navigate to the data directory $PGDATA to view the `postgresql.conf` file.

   Edit the `postgresql.conf` file and change the key in `data_encryption_key_unwrap_command`, under the `Authentication` section.

```
#data_encryption_key_unwrap_command = 'python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt
--pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="74cf4f21-a006-467d-a64f-fbc3fa6ab6d1" --in-file=%p
--variant=pykmip'
data_encryption_key_unwrap_command = 'python3.9 /usr/edb/kmip/client/edb_tde_kmip_client.py decrypt
--pykmip-config-file=/etc/pykmip/pykmip.conf --key-uid="34816dab-24e5-4635-9990-00684b84a8c4" --in-file=%p
--variant=pykmip'
```

7. Go back to the $HOME directory

```
% cd
```

8. Stop the database.

```
% /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l $HOME/logfile stop

waiting for server to shut down.... done
server stopped
```

9. Star the database.

```
% /usr/edb/as16/bin/pg_ctl -D /var/lib/edb/as16/data -l $HOME/logfile start
waiting for server to start..... done
server started
```

10. Connect to the database.

```
% /usr/edb/as16/bin/psql hr
```

11. Attempt to read the data in the dept table.

```
hr=# SELECT * FROM dept;

 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
(2 rows)
```

As you can see we were able to rotate the key and view the data, which means the key rotation worked.

Chapter 3. Additional resources and related products

3.1. KeyControl

3.2. KeyControl as a Service

3.3. Entrust products

3.4. nShield product documentation