



**ENTRUST**

# Oracle MySQL and Entrust KeyControl Vault

Integration Guide

2024-10-02

# Table of Contents

1. Introduction .....	1
1.1. Documents to read first .....	1
1.2. Requirements .....	1
1.3. High-availability considerations .....	1
1.4. Product configuration .....	2
2. Procedures .....	3
2.1. Install and configure KeyControl .....	3
2.2. Establish trust between the KeyControl Vault Server and Oracle MySQL with KeyControl Certificates .....	6
2.3. Install the Oracle MySQL server .....	8
2.4. Install the keyring_okv plugin .....	8
2.5. Import the KeyControl KMIP Certificates to the keyring_okv plugin .....	8
2.6. Verify that the keyring_okv plugin is working .....	10
2.7. Use keyring_okv plugin to create encrypted tables .....	11
2.8. Test that encryption KeyControl is working .....	12
2.9. Key rotation .....	14
2.10. Secure the MySQL database .....	16
3. Additional resources and related products .....	20
3.1. nShield Connect .....	20
3.2. nShield as a Service .....	20
3.3. KeyControl .....	20
3.4. Entrust digital security solutions .....	20
3.5. nShield product documentation .....	20

---

# Chapter 1. Introduction

This document describes the integration of Oracle MySQL Enterprise Server with the Entrust KeyControl Vault Solution. Entrust KeyControl Vault can serve as a KMS in Oracle MySQL using the open standard Key Management Interoperability Protocol (KMIP). 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.

## 1.1. Documents to read first

This guide describes how to configure the Entrust KeyControl Vault server as a KMS in Oracle MySQL.

To install and configure the Entrust KeyControl Vault server as a KMIP server, see the *Entrust KeyControl nShield HSM Integration Guide*. You can access it from the [Entrust Document Library](#) and from the [nShield Product Documentation website](#).

Also refer to the [Oracle MySQL online documentation](#).

## 1.2. Requirements

- Entrust KeyControl Vault version 10.3.1 or later.

An Entrust KeyControl Vault license is required for the installation. You can obtain this license from your Entrust KeyControl Vault and Oracle MySQL account team or through Entrust KeyControl Vault customer support.

- MySQL Enterprise Server 8.4.2 or later.



Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

## 1.3. High-availability considerations

The Entrust KeyControl Vault solution uses an active-active deployment, which provides high-availability capability to manage encryption keys. Entrust recommends this deployment configuration. In an active-active cluster, changes made to any KeyControl node in the cluster are automatically reflected on all nodes in the cluster. For information about the Entrust KeyControl solution, see

the [Entrust KeyControl Product Overview](#).

## 1.4. Product configuration

The integration between the Oracle MySQL Enterprise Server, Entrust KeyControl Vault, and nShield HSM has been successfully tested in the following configurations:

Product	Version
Red Hat Linux 9	9.4 (Plow) - 5.14.0-427.35.1.el9_4.x86_64
Oracle MySQL Enterprise Server	8.4.2
Entrust KeyControl	10.3.1
MySQL Keyring_okv library	1.11

---

# Chapter 2. Procedures

## 2.1. Install and configure KeyControl

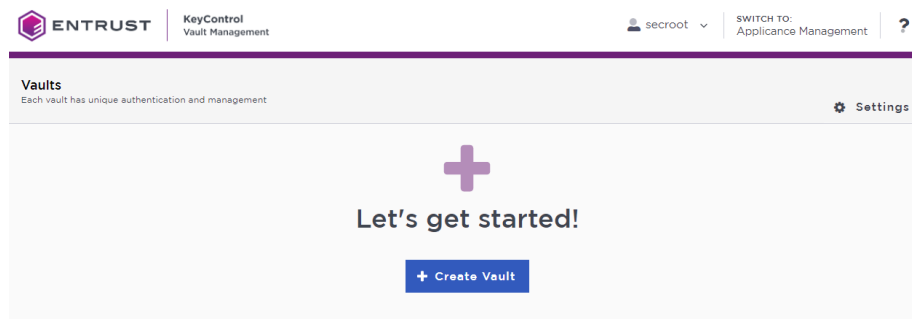
Follow the installation and setup instructions in the *Entrust KeyControl Vault nShield HSM Integration Guide*. You can access it from the [Entrust Document Library](#) and from the [nShield Product Documentation website](#).

### 2.1.1. Creating a KMIP Vault in the KeyControl Vault Management interface

1. Sign in to the KeyControl Vault Server web UI.

Use your browser to access the IP address of the server and sign in with the **secretroot** 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.

### Vaults

Each vault has unique authentication and management

---

#### Create Vault

A vault will have unique authentication and management.

**Type**  
Choose the type of vault to create

KMIP

**Name \***

OracleMySQL

**Description**

Oracle MySQL - KCV KMIP Vault Integration

Max. 300 characters

---

**Email Notifications**  OFF

**⚠ SMTP needs to be configured to turn on email notifications**

Use email to communicate with Vault Administrators, including their temporary passwords. Turning off email notifications means you will see and need to give temporary passwords to Vault Admins.

---

**Administrator**  
Invite an individual to have complete access and control over this vault. They will be responsible for inviting additional members.

**Admin Name \***

Administrator

**Admin Email \***

xxxxxx@company.com

**Create Vault** **Cancel**

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.

The newly vault appears on the vault dashboard and the KMIP server settings on the appliance are **enabled**.

## 2.1.2. KMIP server settings

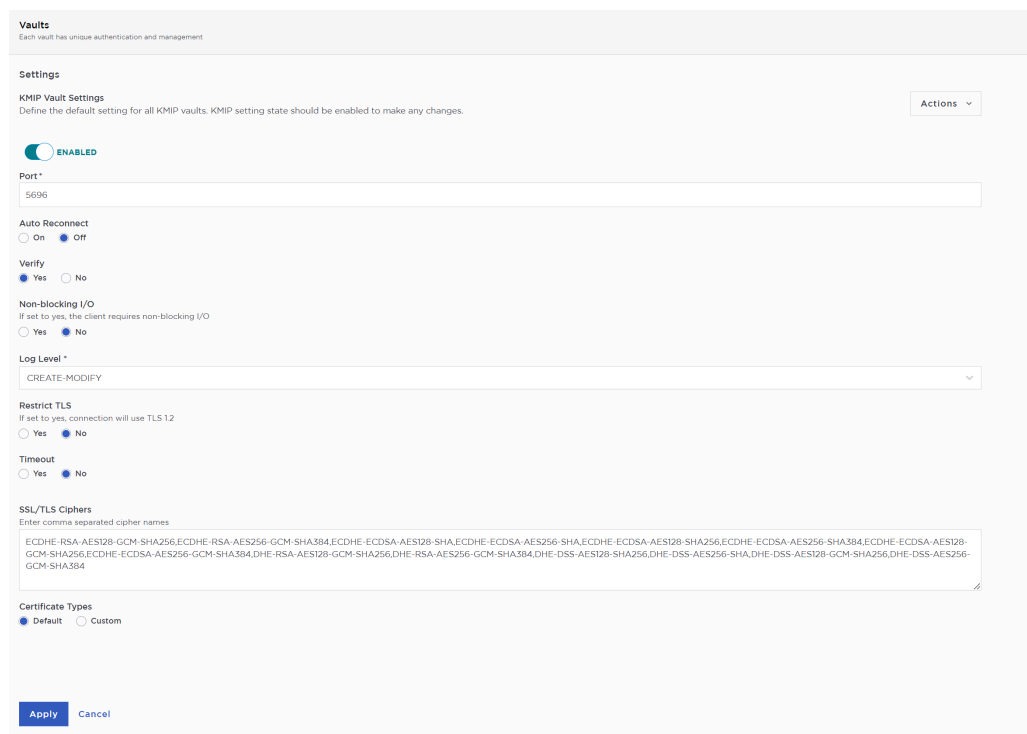
The KMIP server settings are set at the KeyControl appliance level and apply to all the KMIP vaults in the appliance.

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 you are using external key management, as is the case in this solution, the KeyControl server is the KMIP server and the Oracle MySQL server is the KMIP client.

1. Select the **Settings** icon on the top right to view/change the KMIP settings.

The defaults settings are appropriate for most applications. Make any changes necessary.



The screenshot shows the 'Vaults' management interface. Under the 'Settings' section, there is a 'KMIP Vault Settings' panel. The 'ENABLED' toggle is turned on. The 'Port' is set to 5696. 'Auto Reconnect' is set to 'Off'. 'Verify' is set to 'Yes'. 'Non-blocking I/O' is set to 'No'. 'Log Level' is set to 'CREATE-MODIFY'. 'Restrict TLS' is set to 'No'. 'Timeout' is set to 'No'. The 'SSL/TLS Ciphers' field contains a long list of cipher names. 'Certificate Types' is set to 'Default'. At the bottom, there are 'Apply' and 'Cancel' buttons.

2. Select **Apply**.

### 2.1.3. View details for the vault

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

## 2.2. Establish trust between the KeyControl Vault Server and Oracle MySQL with KeyControl Certificates

Certificates are required to facilitate the KMIP communications from the KeyControl KMIP vault and the Oracle MySQL application and conversely. The built-in capabilities in the KMIP Vault are used to create and publish the certificates. To be able to establish trust between the KeyControl and Oracle MySQL, you must create certificates in KeyControl and upload or import them into the configuration of Oracle MySQL.

For more information on how to create a certificate bundle, see [Establishing a Trusted Connection with a KeyControl Vault-Generated CSR](#).



Entrust tested using certificates without password protection. The MySQL online documentation describes the steps needed to use a password-protected keyring\_okv key, see [Password-Protecting the keyring\\_okv Key File](#).

1. Sign in to the KMIP vault that created earlier. Use the 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. (**oraclemysql**)



- 
- b. Set the date on which you want the certificate to expire in the **Certificate Expiration** field.
  - c. 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 web UI 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 `oraclemysql.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 Oracle MySQL KMIP configuration.

7. Transfer the zip file to the Oracle MySQL server.

```
% scp oraclemysql_2024-09-16-16-04-47.zip <youruserid>@<oraclemysql-server-ip-address>:/home/<youruserid>/.
```

8. Sign in to the Oracle MySQL server and unzip the file.

```
% unzip oraclemysql_2024-09-16-16-04-47.zip
```

```
Archive:  oraclemysql_2024-09-16-16-04-47.zip
  inflating: oraclemysql.pem
  inflating: cacert.pem
```

These files will be used in the configuration. First, [Install the Oracle MySQL server](#).

## 2.3. Install the Oracle MySQL server

The process for installing the Oracle MySQL Enterprise Edition depends on the operating system on which you are installing it. See the [Oracle online documentation](#) for details on how to install Oracle MySQL Enterprise Edition in your environment.

## 2.4. Install the `keyring_okv` plugin

The `keyring_okv` plugin is a KMIP 1.1 plugin for KMIP-compatible back-end keyring storage products, such as KeyControl Vault. It is available in MySQL Enterprise Edition distributions.

The configuration directory used by `keyring_okv` as the location for its support files should have a restrictive mode and be accessible only to the account used to run the MySQL server. For example, on Unix and Unix-like systems, to use the `/usr/local/mysql/mysql-keyring-okv` directory, the following commands, executed as `root`, create the directory and set its mode and ownership:

```
cd /usr/local
sudo mkdir -p mysql/mysql-keyring-okv/ssl
sudo chmod -R 750 mysql
sudo chown -R mysql mysql
sudo chgrp -R mysql mysql
```

To be usable during the server startup process, the `keyring_okv` plugin must be loaded using the `--early-plugin-load` option. Also, set the `keyring_okv_conf_dir` system variable to tell `keyring_okv` where to find its configuration directory. Edit the `/etc/my.cnf` file and add the plugin into the `mysqld` section:

```
[mysqld]
early-plugin-load=keyring_okv.so
keyring_okv_conf_dir=/usr/local/mysql/mysql-keyring-okv
```

## 2.5. Import the KeyControl KMIP Certificates to the `keyring_okv` plugin

The certificates must be installed before running the `keyring_okv` plugin, so that the plugin can be initialized.

1. Import the certificates into the configuration directory for the `keyring_okv` plugin.

---

The following files need to be imported:

- A `<cert_name>.pem` file that includes both the client certificate and private key. The administrator needs to open this single file and paste the two sections of the file into the `cert.pem` and `key.pem` files in the `/usr/local/mysql/mysql-keyring-okv/ssl` directory.

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

Open or create `/usr/local/mysql/mysql-keyring-okv/ssl/cert.pem` and paste `"-----BEGIN CERTIFICATE-----"` and `"-----END CERTIFICATE-----"` and all text between them into this file. Make sure it has a carriage return at the end of the file.

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

Open or create `/usr/local/mysql/mysql-keyring-okv/ssl/key.pem` and paste `"-----BEGIN CERTIFICATE-----"` and `"-----END CERTIFICATE-----"` and all text between them into this file. Make sure it has a carriage return at the end of the file.

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

This file needs to be copied to `/usr/local/mysql/mysql-keyring-okv/ssl/CA.pem`.

2. In the configuration directory, create a `okvclient.ora` file with the following format:

```
SERVER=xxx.xxx.xxx.xxx:5696
STANDBY_SERVER=xxx.xxx.xxx.xxx:5696
```

`STANDBY_SERVER` is optional.

For example:

```
SERVER=198.51.100.20:5696
STANDBY_SERVER=198.51.100.21:5696
```

These are the IP addresses of the KeyControl Vault servers.

### 3. Set the permissions on these files:

```
cd /usr/local/mysql/mysql-keyring-okv
sudo chmod -R 750 mysql .
sudo chown -R mysql .
sudo chgrp -R mysql .
```

### 4. If the firewall is running open up the firewall for port 5696.

As the root user on the mysql server:

```
% firewall-cmd --zone=public --add-port=5696/tcp --permanent
% firewall-cmd --zone=public --add-port=5696/udp --permanent
% firewall-cmd --reload
```

### 5. Disable SELinux the next time the server reboots.

To do this, in the `/etc/selinux/config` file set `SELINUX=disabled`.

To disable on the current shell:

```
% sudo setenforce 0
```



We are disabling SELinux for the purpose of the integration. Consult with your security team on how to handle SELinux in this case.

### 6. After completing the preceding procedure, restart the MySQL server:

```
% sudo systemctl restart mysqld
% sudo systemctl status mysqld
```

It loads the `keyring_okv` plugin, which uses the files in its configuration directory to communicate with KeyControl.

## 2.6. Verify that the `keyring_okv` plugin is working

After configuration is complete and you restarted MySQL to load the `keyring_okv` plugin, look in the `/var/log/mysqld.log` logs to make sure there are no errors when connecting to KeyControl. For example, the `mysqld.log` file could report the following errors if the plugin does not work:

```
2024-09-16T18:53:39.117806Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.4.2-commercial) starting
as process 8034
2024-09-16T18:53:39.123096Z 0 [ERROR] [MY-011398] [Server] Plugin keyring_okv reported: 'Error loading trust
```

```
store'
2024-09-16T18:53:39.123152Z 0 [ERROR] [MY-011386] [Server] Plugin keyring_okv reported: 'Could not initialize ssl
layer'
2024-09-16T18:53:39.123163Z 0 [ERROR] [MY-011377] [Server] Plugin keyring_okv reported: 'keyring_okv
initialization failure. Please check that the keyring_okv_conf_dir points to a readable directory and that the
directory contains Oracle Key Vault configuration file and ssl materials. Please also check that Oracle Key Vault
is up and running.'
2024-09-16T18:53:39.123170Z 0 [ERROR] [MY-010202] [Server] Plugin 'keyring_okv' init function returned error.
```

In this case, make sure the file permissions are correct in the keyring configuration directory and that the certificate files are named correctly.

To verify the plugin installation, with the MySQL server running, examine the `INFORMATION_SCHEMA.PLUGINS` table or use the `SHOW PLUGINS` statement. For example:

```
mysql> SELECT PLUGIN_NAME, PLUGIN_STATUS FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE 'keyring%';
+-----+-----+
| PLUGIN_NAME | PLUGIN_STATUS |
+-----+-----+
| keyring_okv | ACTIVE        |
+-----+-----+
1 row in set (0.00 sec)
```

## 2.7. Use keyring\_okv plugin to create encrypted tables

When you create the first encrypted table, InnoDB will ask `keyring_okv` to generate the primary key (AES-256) in KeyControl. This primary key is used to encrypt tablespace keys. You can check the primary key in the KeyControl KMIP Vault web interface by selecting **Objects** in the **Home** tab.

InnoDB also asks KeyControl to generate a key (AES-256) for the encrypting table. The tablespace key is wrapped using the primary key and stored alongside the encrypted table. For subsequent encrypted tables, only the tablespace key is generated and the same primary key is used to wrap the tablespace key.

With KeyControl, you will see a complete audit trail if every time the primary key or tablespace key is retrieved. You will have complete control on these keys. You can revoke access to a key or disable it, to lock down your data at rest.

To create an encrypted table:

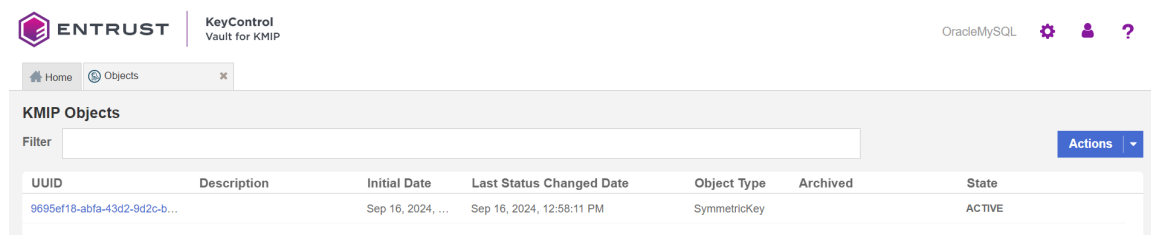
1. Sign in to the MySQL database:

```
% mysql -u root -p<password>
```

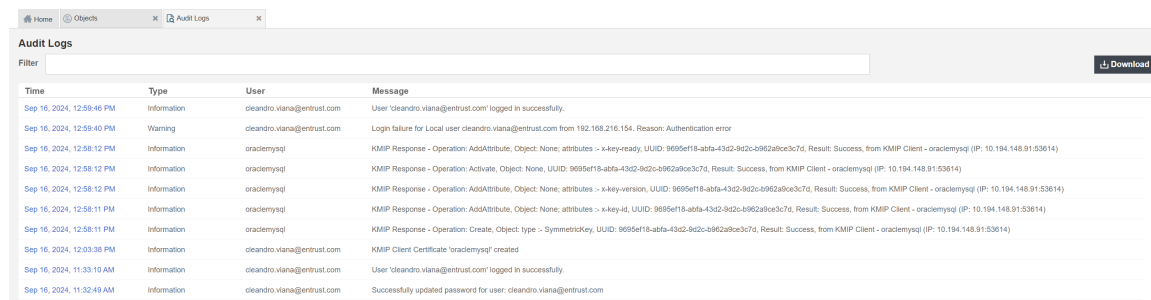
2. Create the encrypted table with the following SQL:

```
CREATE DATABASE MySQL_TDE_Test;
USE MySQL_TDE_Test;
CREATE TABLE `test_encryption` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(15) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1 ENCRYPTION = 'Y';
```

The **Objects** tab in the KeyControl KMIP Vault web UI shows the that the key was created. For example:



You can also check the **Audit Logs** tab. You should see all the KMIP operations that happened during that key creation process and retrieval. For example:



## 2.8. Test that encryption KeyControl is working

1. Log in into the MySQL database:

```
% mysql -u root -p<password>
```

2. Insert a record to the table that was created earlier:

```
mysql> USE MySQL_TDE_Test;
Database changed

mysql> INSERT INTO test_encryption VALUES (1, 'cLive');
Query OK, 1 row affected (0.00 sec)

mysql> select * from test_encryption;
+----+-----+
| id | name |
+----+-----+

```

```
+-----+
| 1 | clive |
+-----+
1 row in set (0.00 sec)
```

3. Edit the MySQL configuration file and disable the `keyring_okv` plugin:

```
% sudo vi /etc/my.cnf
#early-plugin-load=keyring_okv.so
#keyring_okv_conf_dir=/usr/local/mysql/mysql-keyring-okv
```

4. Restart MySQL:

```
% sudo systemctl restart mysqld
```

5. Check that you can read the encrypted table:

```
% mysql -u root -p<password>
mysql> use MySQL_TDE_Test;
Database changed
mysql> select * from test_encryption;
ERROR 3185 (HY000): Cannot find master key from keyring, please check in the server log if a keyring is
loaded and initialized successfully.
```

The table is not accessible because MySQL cannot get to the master key from the keyring.

6. Re-enable the keyring in the MySQL configuration file and remove the comments you added previously:

```
% sudo vi /etc/my.cnf
early-plugin-load=keyring_okv.so
keyring_okv_conf_dir=/usr/local/mysql/mysql-keyring-okv
```

7. Restart MySQL:

```
% sudo systemctl restart mysqld
```

8. Check that you can view the encrypted table:

```
% mysql -u root -p<password>
mysql> use MySQL_TDE_Test;
Database changed
mysql> select * from test_encryption;
+-----+
| id | name |
+-----+
```

```
+---+-----+
| 1 | clive |
+---+-----+
1 row in set (0.00 sec)
```

This shows that the configuration of the `keyring_okv` plugin using KeyControl is working.

## 2.9. Key rotation

Rotating the master key is the same thing as generating a new key to replace the old one. In MySQL, when you use InnoDB encrypted tables, there is a MASTER KEY. You can rotate this using the following command:

```
ALTER INSTANCE ROTATE INNODB MASTER KEY;
```

The new key is generated and the old one is replaced.

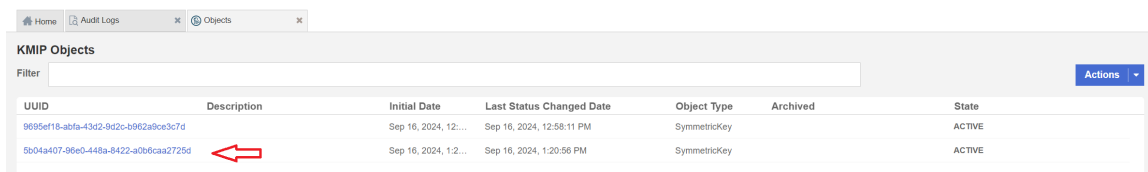
Let's give it a try and see what happens:

1. Sign in to MySQL and run the `ALTER INSTANCE ROTATE` command.

```
% mysql -u root -p<password>

mysql> ALTER INSTANCE ROTATE INNODB MASTER KEY;
Query OK, 0 rows affected (0.22 sec)
```

2. Check in the KeyControl KMIP Vault web UI under the **Objects** tab for the new key.



You will also be able to see the activity for the new key creation on the audit logs in the **Audit Logs** Tab.



Time	Type	User	Message
Sep 16, 2024, 1:20:56 PM	Information	oraclemysqj	KMIP Response - Operation: AddAttribute. Object: None; attributes -> x-key-ready, UUID: 5b04a407-96e0-448a-8422-a0b6caa2725d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:37992)
Sep 16, 2024, 1:20:56 PM	Information	oraclemysqj	KMIP Response - Operation: Activate. Object: None, UUID: 5b04a407-96e0-448a-8422-a0b6caa2725d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:37992)
Sep 16, 2024, 1:20:56 PM	Information	oraclemysqj	KMIP Response - Operation: AddAttribute. Object: None; attributes -> x-key-version, UUID: 5b04a407-96e0-448a-8422-a0b6caa2725d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:37992)
Sep 16, 2024, 1:20:56 PM	Information	oraclemysqj	KMIP Response - Operation: AddAttribute. Object: None; attributes -> x-key-id, UUID: 5b04a407-96e0-448a-8422-a0b6caa2725d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:37992)
Sep 16, 2024, 1:20:56 PM	Information	oraclemysqj	KMIP Response - Operation: Create. Object: type -> SymmetricKey, UUID: 5b04a407-96e0-448a-8422-a0b6caa2725d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:37992)
Sep 16, 2024, 12:59:46 PM	Information	cleandro.vian...	User 'cleandro.viana@entrust.com' logged in successfully.
Sep 16, 2024, 12:59:40 PM	Warning	cleandro.vian...	Login failure for Local user cleandro.viana@entrust.com from 192.168.216.154. Reason: Authentication error
Sep 16, 2024, 12:58:12 PM	Information	oraclemysqj	KMIP Response - Operation: AddAttribute. Object: None; attributes -> x-key-ready, UUID: 9695ef18-abfa-43d2-9d2c-b962a9ce3c7d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:53614)
Sep 16, 2024, 12:58:12 PM	Information	oraclemysqj	KMIP Response - Operation: Activate. Object: None, UUID: 9695ef18-abfa-43d2-9d2c-b962a9ce3c7d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:53614)
Sep 16, 2024, 12:58:12 PM	Information	oraclemysqj	KMIP Response - Operation: AddAttribute. Object: None; attributes -> x-key-version, UUID: 9695ef18-abfa-43d2-9d2c-b962a9ce3c7d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:53614)
Sep 16, 2024, 12:58:11 PM	Information	oraclemysqj	KMIP Response - Operation: AddAttribute. Object: None; attributes -> x-key-id, UUID: 9695ef18-abfa-43d2-9d2c-b962a9ce3c7d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:53614)
Sep 16, 2024, 12:58:11 PM	Information	oraclemysqj	KMIP Response - Operation: Create. Object: type -> SymmetricKey, UUID: 9695ef18-abfa-43d2-9d2c-b962a9ce3c7d, Result: Success, from KMIP Client - oraclemysqj (IP: 10.194.148.91:53614)
Sep 16, 2024, 12:03:38 PM	Information	cleandro.vian...	KMIP Client Certificate 'oraclemysqj' created
Sep 16, 2024, 11:33:10 AM	Information	cleandro.vian...	User 'cleandro.viana@entrust.com' logged in successfully.
Sep 16, 2024, 11:32:49 AM	Information	cleandro.vian...	Successfully updated password for user: cleandro.viana@entrust.com

### 3. Revoke the old key.

In the KeyControl KMIP Vault web UI under the **Objects** tab, select the old key and select **Actions**. In the dropdown menu select **Revoke** to revoke the key.

UUID	Description	Initial Date	Last Status Changed Date	Object Type	Archived	State	Update Description
9695ef18-abfa-43d2-9d2c-b962a9ce3c7d		Sep 16, 2024, 12:...	Sep 16, 2024, 12:58:11 PM	SymmetricKey		ACTIVE	Revoke
5b04a407-96e0-448a-8422-a0b6caa2725d		Sep 16, 2024, 1:2...	Sep 16, 2024, 1:20:56 PM	SymmetricKey		ACTIVE	Archive



We are just doing this to assure that MySQL cannot use the old key. In reality MySQL will just use the new key that has been created. The old key can stay active if you wish to do so.

### 4. Restart MySQL:

```
% sudo systemctl restart mysqld
```

### 5. Check if you can read the encrypted table:

```
% mysql -u root -p<password>

mysql> use MySQL_TDE_Test;
Database changed

mysql> select * from test_encryption;
+----+-----+
| id | name |
+----+-----+
| 1 | clive |
+----+-----+
1 row in set (0.00 sec)
```

6. To show that MySQL is using the new key that has been generated, issue the following command:

```
mysql> SELECT * FROM performance_schema.keyring_keys;
+-----+-----+-----+
| KEY_ID | KEY_OWNER | BACKEND_KEY_ID |
+-----+-----+-----+
| INNOBKKey-xxxxxxx-743c-11ef-8429-0050568b99be-2 | | 5b04a407-96e0-xxxx-yyyy-a0b6caa2725d |
+-----+-----+-----+
1 row in set (0.00 sec)
```

As you can see, the `BACKEND\_KEY\_ID` is the same as the new key that was created in the KMIP vault. This shows that the key rotation was successful.

## 2.10. Secure the MySQL database

The information below was taken from the following Security Technical Implementation Guides (STIG) page and can be used as guideline to address confidentiality and integrity of all information at rest in a MySQL database.

### Group Title

SRG-APP-000231-DB-000154

### Rule Title

The MySQL Database Server 8.0 must protect the confidentiality and integrity of all information at rest.

### Discussion

This control is intended to address the confidentiality and integrity of information at rest in non-mobile devices and covers user information and system information. Information at rest refers to the state of information when it is located on a secondary storage device, such as a disk drive or a tape drive, within an organizational information system. Applications and application users generate information throughout the course of their application use.

For more information, see [InnoDB Data-at-Rest Encryption](#) in the MySQL online documentation.

User-generated data, as well as application-specific configuration data, must be protected. Organizations may choose to employ different mechanisms to achieve confidentiality and integrity protections, as appropriate.

If the confidentiality and integrity of application data is not protected, the data will be open to compromise and unauthorized modification.

---

Apply appropriate controls to protect the confidentiality and integrity of data at rest in the database.

Using SQL, determine if all data-at-rest is encrypted:

1. Check `audit_log_encryption`:

```
SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM performance_schema.global_variables where variable_name = 'audit_log_encryption';
```

If `audit_log_encryption` is not set to `AES`, this is important.

2. Check `binlog_encryption`:

```
SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM performance_schema.global_variables where variable_name = 'binlog_encryption';
```

If `binlog_encrypt` is not set to `ON`, this is important.

3. Check `innodb_redo_log_encrypt`:

```
SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM performance_schema.global_variables where variable_name = 'innodb_redo_log_encrypt';
```

If `innodb_redo_log_encrypt` is not set to `ON`, this is important.

4. Check `innodb_undo_log_encrypt`:

```
SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM performance_schema.global_variables where variable_name = 'innodb_undo_log_encrypt';
```

If `innodb_undo_log_encrypt` is not set to `ON`, this is important.

5. Check `general_log`:

```
SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM performance_schema.global_variables
WHERE VARIABLE_NAME like 'general_log';
```

If `general_log` is not `OFF`, this is important.

Using SQL, find the encryption status for all MySQL table and tablespaces:

1. Check tablespaces:

```
SELECT
`INNODB_TABLESPACES`.`NAME`,
```

```
`INNODB_TABLESPACES`.`ENCRYPTION`  
FROM `information_schema`.`INNODB_TABLESPACES`;
```

If any tablespace does not have **ENCRYPTION** set to **Y (yes)**, this is important.

## 2. Check `innodb_redo_log_encrypt`:

```
SELECT VARIABLE_NAME, VARIABLE_VALUE  
FROM performance_schema.global_variables where variable_name = 'table_encryption_privilege_check';
```

If `innodb_redo_log_encrypt` is not set to **ON**, this is important.

Apply appropriate MySQL Database 8.0 controls to protect the confidentiality and integrity of data at rest in the database:

```
sudo vi /etc/my.cnf  
[mysqld]  
audit-log=FORCE_PLUS_PERMANENT  
audit-log-format=JSON  
audit-log-encryption=AES  
  
#Turn on binlog encryption  
set persist binlog_encryption=ON;  
  
#Turn on undo and redo log encryption  
set persist innodb_redo_log_encrypt=ON;  
set persist innodb_undo_log_encrypt=ON;
```

Enable encryption for a new file-per-table tablespace, **ENCRYPTION** option in a **CREATE TABLE** statement. The following example assumes that `innodb_file_per_table` is enabled:

```
mysql> CREATE TABLE t1 (c1 INT) ENCRYPTION='Y';
```

To enable encryption for an existing file-per-table tablespace, specify the **ENCRYPTION** option in an **ALTER TABLE** statement:

```
mysql> ALTER TABLE t1 ENCRYPTION='Y';
```

To disable encryption for file-per-table tablespace, set **ENCRYPTION='N'** using **ALTER TABLE**:

```
mysql> ALTER TABLE t1 ENCRYPTION='N';
```

To disable `general_log`:

---

```
mysql> SET PERSIST general_log = 'OFF';
```

## Chapter 3. Additional resources and related products

3.1. nShield Connect

3.2. nShield as a Service

3.3. KeyControl

3.4. Entrust digital security solutions

3.5. nShield product documentation