# Oracle MySQL and Entrust KeyControl

## Integration Guide

10 May 2022

# Contents

1. Introduction . . . 3
    1.1. Requirements . . . 3
    1.2. High-availability considerations . . . 3
    1.3. Product configuration . . . 3
2. Procedures . . . 4
    2.1. Installation overview . . . 4
    2.2. Install the Entrust KeyControl Server . . . 4
    2.3. Create the KMIP Tenant for Oracle MySQL in KeyControl . . . 4
    2.4. Create the KMIP Certificates . . . 6
    2.5. Install the Oracle MySQL server . . . 7
    2.6. Install the keyring_okv plugin . . . 7
    2.7. Import the KeyControl KMIP Certificates to the keyring_okv plugin . . . 8
    2.8. Verify that the keyring_okv plugin is working . . . 9
    2.9. Use keyring_okv plugin to create encrypted tables . . . 10
    2.10. Test that encryption KeyControl is working . . . 11
    2.11. Secure the MySQL database . . . 13

# 1. Introduction

Oracle MySQL Enterprise Server is compatible with the Entrust KeyControl solution. This document describes the configuration of Oracle MySQL Enterprise Server 8.0.29 for integration with the Entrust KeyControl 5.5 key management solution. Entrust KeyControl can serve as a key manager MySQL encryption by using the open standard Key Management Interoperability Protocol (KMIP).

## 1.1. Requirements

- Entrust KeyControl version 5.5 or later.

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

- MySQL Enterprise Server 8.0.29 or later.

## 1.2. High-availability considerations

The Entrust KeyControl 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.

## 1.3. Product configuration

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

| Product | Version |
|---|---|
| CentOS Linux 8 | 4.18.0-383.el8.x86_64 |
| Oracle MySQL Enterprise Server | 8.0.29 |
| Entrust KeyControl | 5.5 |
| MySQL Keyring_okv library | 1.10 |

# 2. Procedures

## 2.1. Installation overview

Follow these steps to integrate Oracle MySQL Enterprise Server with Entrust KeyControl.

1. Install the Entrust KeyControl Server.
2. Create the KMIP Tenant for Oracle MySQL in KeyControl.
3. Create the KMIP Certificates.
4. Install the Oracle MySQL server.
5. Install the keyring_okv plugin.
6. Import the KeyControl KMIP Certificates to the keyring_okv plugin.
7. Verify that the keyring_okv plugin is working.
8. Use keyring_okv plugin to create encrypted tables.
9. Test that encryption KeyControl is working.
10. Secure the MySQL database.

## 2.2. Install the Entrust KeyControl Server

Follow the installation instructions for the Entrust KeyControl Server installation and configuration. These instructions can be found in the Entrust KeyControl Integration Guide located in the Entrust Documentation site. Search for the `Entrust KeyControl nShield HSM Integration Guide`, which documents the setup process. Set up the KeyControl server as a KMIP server according to the guide.

This solution uses external key management. The KeyControl server is the KMIP server and Oracle MySQL is the KMIP client.

## 2.3. Create the KMIP Tenant for Oracle MySQL in KeyControl

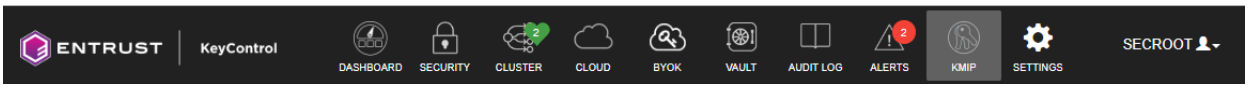
To use external key management, MySQL requires an external key management server such as the Entrust KeyControl server. Certificates are required to facilitate the KMIP communications from the KeyControl server to Oracle MySQL and conversely. To be able to create these certificates, you need to have the KMIP tenant for the application in KeyControl.

Now that Entrust KeyControl is installed, create a KMIP Tenant in KeyControl for Oracle MySQL:

1. Log in to the KeyControl instance.

   Point your browser to the KeyControl administration URL: **http://xx.xxx.xx.xxx**

2. Log in as **secroot** and go to the **KMIP** page of the application:




3. Select **Actions > Create a KMIP Tenant**.

   The **Create a KMIP Tenant** dialog appears.

4. In the **About** tab, enter the tenant name information:
   a. For **Name**, enter **OracleMySQL**.
   b. Optionally, enter a **Description**.
5. Select **Next**.
6. In the **Admin** tab, provide an Active Directory:
   a. For **Active Directory**, select **Other Active Directory**.
   b. In **Active Directory Domain**, select **"+"**.

      The **KMIP Active Directory Domain** dialog appears.

   c. For **Domain Name**, enter the name of the domain. For example, : **example.com**.
   d. In **Domain Controllers**, select **"+"**.

      The **Add Domain Controller** dialog appears.

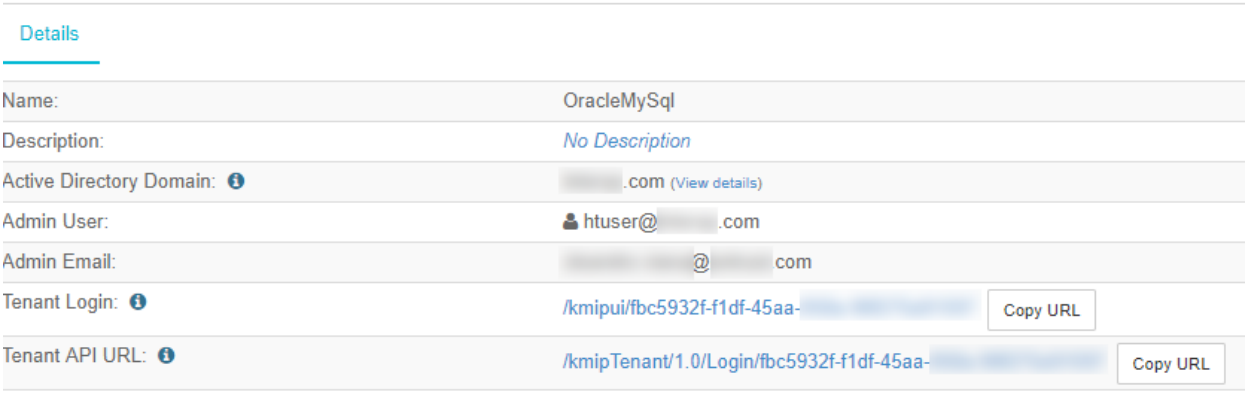
      For **Server URL**, select **LDAP** and enter the FQDN/IP of the Active directory server: **xx.xxx.xx.xx**.

      Then select **Save and Close**.

   e. Select **Save and Close** on the **KMIP Active Directory Domain** dialog.
   f. For **Admin**, select **User**.
   g. For **Name**, enter the name of the Active Directory user that will be the administrator of the tenant: **htuser@example.com**.
   h. For **Email**, enter the email of the administrator user.
   i. Select **Create**.

      The new KMIP tenant is created and appears in the list of tenants.

7. Select the tenant to see its details.




8. Copy the **Tenant Login** URL.

   The tenant login URL will be used to log in to the **Tenant Administration** page in KeyControl.

## 2.4. Create the KMIP Certificates

To be able to establish trust between the KeyControl and Oracle MySQL, you must create certificates in KeyControl and upload/import them into the configuration of Oracle MySQL.

> 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. Access the KeyControl web interface using the **Tenant Administration** URL you copied in the previous section.
2. Log in using the Tenant administrator user that you configured during the tenant creation process: **htuser@example.com**. Provide the user Active Directory password.
3. Select **Security > Client Certificates**.

   The **Manage Client Certificate** dialog appears.

4. Select **"+"** on the right to create a new certificate.

   The **Create Client Certificate** dialog appears.

5. In the **Create Client Certificate** dialog, enter the following information:

   a. For **Certificate Name**, enter a name for the certificate: **keyringokv**.

   b. For **Certificate Expiration**, set the date on which you want the certificate to expire.

c. Select **Create**.

The new certificate appears in the **Manage Client Certificate** dialog.

6. Select the certificate and select **Download** to download the certificate.

   The `certname_<datetimestamp>.zip` downloads. This file contains a user certification/key file called `certname.pem` and a server certification file called `cacert.pem`.

7. Unzip the file so that you have these files available to upload to the MySQL server.

After you create and download these certificates, you need to upload or import them into the MySQL server. First, Install the Oracle MySQL server.

## 2.5. 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.

## 2.6. 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 Entrust KeyControl. 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
```

## 2.7. 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 file named `okvclient.ora`. It should have 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
```

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
```

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.

## 2.8. 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 `/varlog/mysqld.log` logs to make sure there are no errors when connecting to KeyControl. 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)
```

## 2.9. 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 Tenant KeyControl web interface using the **Objects** page.

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:

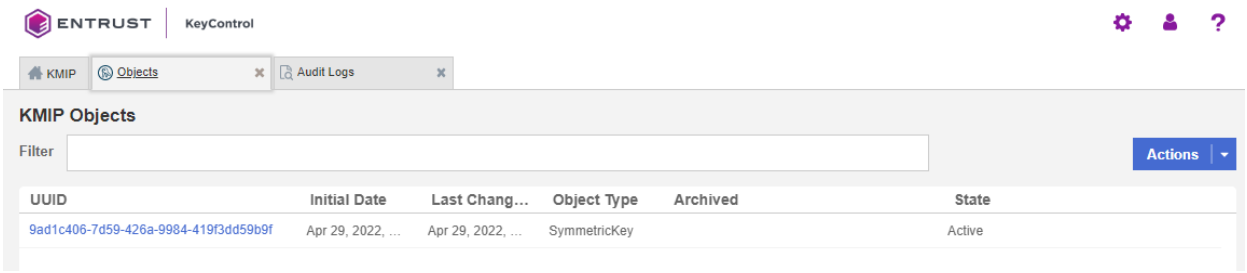
1. Log in into the MySQL database:

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

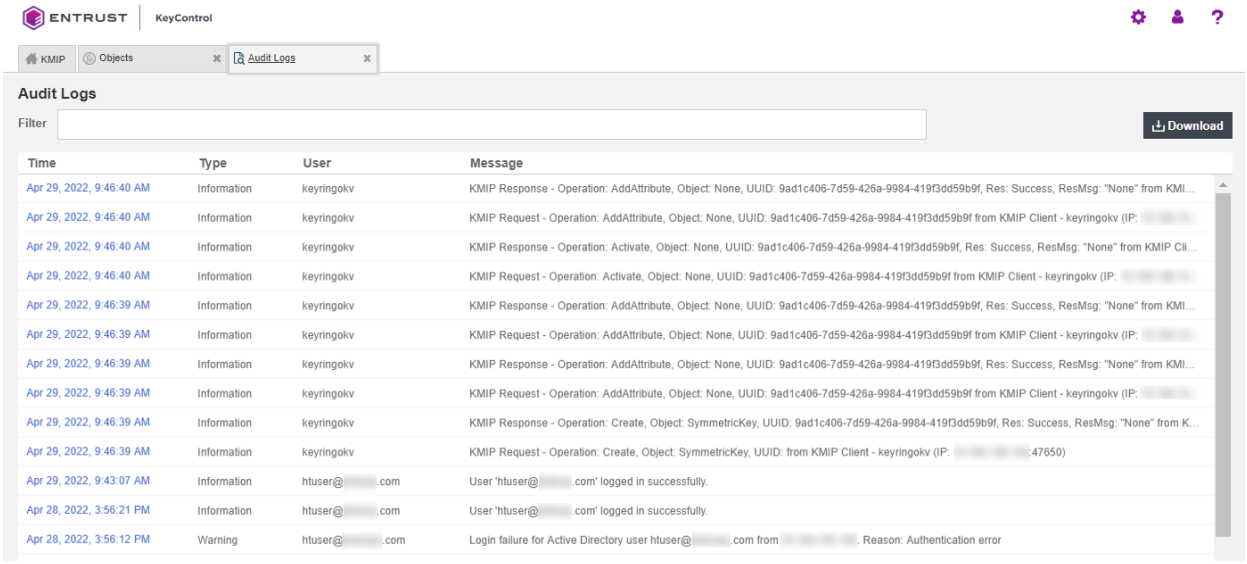
2. Create the encrypted table with the following SQL:

   ```
   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 Tenant KeyControl 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:




## 2.10. 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;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

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

mysql> select * from test_encryption;
+----+----------+
| id | name     |
+----+----------+
|  1 | cleandro |
+----+----------+
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 if you can read the encrypted table:

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

mysql> use MySQL_TDE_Test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> select * from test_encryption;
ERROR 3185 (HY000): Can't 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 you can view the encrypted table:

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

mysql> use MySQL_TDE_Test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

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

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

## 2.11. 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 (e.g., disk drive, 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`:

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