MySQL Connector/J Developer Guide

Abstract

This manual describes how to install, configure, and develop database applications using MySQL Connector/J 9.0, a JDBC and X DevAPI driver for communicating with MySQL servers.

MySQL Connector/J 9.0 supersedes the 8.4 series and is recommended for use on production systems. It is for use with MySQL Server 8.0 and up. Please upgrade to MySQL Connector/J 9.0.

For notes detailing the changes in each release of Connector/J, see MySQL Connector/J Release Notes.

For legal information, including licensing information, see the Preface and Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2024-11-05 (revision: 80192)

Table of Contents

Preface and Legal Notices

This manual describes how to install, configure, and develop database applications using MySQL Connector/J, the JDBC driver for communicating with MySQL servers.

Licensing information. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL Connector/J 9.0, see the MySQL Connector/J 9.0 Commercial License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL Connector/J 9.0, see the MySQL Connector/J 9.0 Community License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

License Restrictions

Warranty Disclaimer

Restricted Rights Notice

Hazardous Applications Notice

applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Use of This Documentation

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Chapter 1 Overview of MySQL Connector/J

MySQL provides connectivity for client applications developed in the Java programming language with MySQL Connector/J. Connector/J implements the Java Database Connectivity (JDBC) API, as well as a number of value-adding extensions of it. It also supports the new X DevAPI.

MySQL Connector/J is a JDBC Type 4 driver, implementing the JDBC 4.2 specification. The Type 4 designation means that the driver is a pure Java implementation of the MySQL protocol and does not rely on the MySQL client libraries. See Chapter 2, *Compatibility with MySQL and Java Versions* for compatibility information.

Connector/J 9.0 provides ease of development features including auto-registration with the Driver Manager, standardized validity checks, categorized SQLExceptions, support for large update counts, support for local and offset date-time variants from the `java.time` package, support for JDBC-4.x XML processing, support for per connection client information, and support for the `NCHAR`, `NVARCHAR` and `NCLOB` data types. See Chapter 2, *Compatibility with MySQL and Java Versions* for compatibility information.

For large-scale programs that use common design patterns of data access, consider using one of the popular persistence frameworks such as Hibernate, Spring's JDBC templates or MyBatis SQL Maps to reduce the amount of JDBC code for you to debug, tune, secure, and maintain.

Key Topics

- For installation instructions for Connector/J, see Chapter 4, *Connector/J Installation*.
- For help with connection strings, connection options, and setting up your connection through JDBC, see Chapter 6, *Connector/J Reference*.
- For information on connection pooling, see Chapter 8, *Connection Pooling with Connector/J*.
- For information on multi-host connections, see Chapter 9, *Multi-Host Connections*.
- For information on using the X DevAPI with Connector/J, see Chapter 10, *Using the X DevAPI with Connector/J: Special Topics*.

Chapter 2 Compatibility with MySQL and Java Versions

Here is some compatibility information for Connector/J 9.0:

- **JDBC versions**: Connector/J 9.0 implements JDBC 4.2. While Connector/J 9.0 works with libraries of higher JDBC versions, it returns a `SQLFeatureNotSupportedException` for any calls of methods supported only by JDBC 4.3 and higher.
- **MySQL Server versions**: Connector/J 9.0 supports MySQL 8.0 and up.
- **JRE versions**: Connector/J 9.0 supports JRE 8 or higher.
- **JDK Required for Compilation**: JDK 8.0 or higher is required for compiling Connector/J 9.0. Also, a customized JSSE provider might be required to use some later TLS versions and cipher suites when connecting to MySQL servers. For example, because Oracle's Java 8 releases before 8u261 were shipped with JSSE implementations that support TLS up to version 1.2 only, you need a customized JSSE implementation to use TLSv1.3 on those Java 8 platforms. Oracle Java 8u261 and above do support TLSv1.3, so no customized JSSE implementation is needed.

Chapter 3 What's New in Connector/J 9.0?

Version 9.0.0 is a new GA release version of the MySQL Connector/J. MySQL Connector/J 9.0.0 supersedes the 8.4 series and is recommended for use on production systems. This release can be used against MySQL Server version 8.0 and up. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

For notes detailling the changes in Connector/J 9.0, see MySQL Connector/J Release Notes

Chapter 4 Connector/J Installation

Table of Contents

You can install the Connector/J package using either a binary or source distribution. While the binary distribution provides the easiest method for installation, the source distribution lets you customize your installation. Both types of distributions are available from the Connector/J Download page. The source code for Connector/J is also available on GitHub at https://github.com/mysql/mysql-connector-j.

Connector/J is also available as a Maven artifact in the Central Repository. See Section 4.2, “Installing Connector/J Using Maven” for details.

If you are upgrading from a previous version, read the upgrade information in Section 4.4, “Upgrading from an Older Version” before continuing.

> **Important**
>
> *Third-party Libraries:* According to how you use Connector/J 9.0, you may also need to install the following third-party libraries on your system for it to work:
>
> - Protocol Buffers (`protobuf-java`) 4.26.1 is required for using X DevAPI
> - Oracle Cloud Infrastructure SDK for Java (`oci-java-sdk`) 3.41.2 is required to support OCI AIM authentication
> - Simple Logging Facade API (`slf4j-api`) 2.0.13 is required for using the logging capabilities provided by the default implementation of `org.slf4j.Logger.Slf4JLogger` by Connector/J
>
> These and other third-party libraries are required for building Connector/J from source—see the section for more information.

4.1 Installing Connector/J from a Binary Distribution

Obtaining and Using the Binary Distribution Packages

Different types of binary distribution packages for Connector/J are available from the Connector/J Download page. The following explains how to use each type of the packages to install Connector/J.

Using Platform-independent Archives: `.tar.gz` or `.zip` archives are available for installing Connector/J on any platform. Using the appropriate graphical or command-line utility (for example, `tar` for the `.tar.gz` archive and `WinZip` for the .zip archive), extract the JAR archive from the `.tar.gz` or `.zip` archive to a suitable location.

> **Note**
>
> Because there are potentially long file names in the distribution, the Connector/J archives use the GNU Tar archive format. Use GNU Tar or a compatible application to unpack the `.tar.gz` variant of the distribution.

Using Packages for Software Package Management Systems on Linux Platforms: RPM and Debian packages are available for installing Connector/J on a number of Linux distributions like Oracle Linux, Debian, Ubuntu, SUSE, and so on. Install these packages using your system's software package management system.

On Windows Platforms: You cannot install Connector/J on Windows platforms using the MySQL Installer for Windows. Notice that there are also no stand-alone Windows installer files (.msi) for installing Connector/J. Use the platform-independent archives instead for installations on Windows platforms.

Configuring the CLASSPATH

Once `mysql-connector-j-version.jar` has been extracted from the binary distribution package to the right place, finish installing the driver by placing the JAR archive in your Java classpath, either by adding its full file path to your `CLASSPATH` environment variable, or by directly specifying the file path with the command line switch `-cp` when starting the JVM.

For example, on Linux platforms, add the Connector/J driver to your `CLASSPATH` using one of the following forms, depending on your command shell:

```
# Bourne-compatible shell (sh, ksh, bash, zsh):
$> export CLASSPATH=/path/mysql-connector-j-ver.jar:$CLASSPATH

# C shell (csh, tcsh):
$> setenv CLASSPATH /path/mysql-connector-j-ver.jar:$CLASSPATH
```

You can also set the `CLASSPATH` environment variable in a profile file, either locally for a user within the user's `.profile`, `.login`, or other login file, or globally by editing the global `/etc/profile` file.

For Windows platforms, you set the environment variable through the System Control Panel.

> **Important**
>
> Remember to also add the locations of the third-party libraries required for using Connector/J to CLASSPATH.

Configuring Connector/J for Application Servers

To use MySQL Connector/J with an application server such as GlassFish or Tomcat, read your vendor's documentation for information on how to configure third-party class libraries, as most application servers ignore the `CLASSPATH` environment variable. For configuration examples for some J2EE application servers, see Chapter 8, *Connection Pooling with Connector/J*, Section 9.3, “Configuring Load Balancing with Connector/J”, and Section 9.5, “Advanced Load-balancing and Failover Configuration”. However, the authoritative source for JDBC connection pool configuration information is the documentation for your own application server.

If you are developing servlets or JSPs and your application server is J2EE-compliant, you can put the driver's `.jar` file in the `WEB-INF/lib` subdirectory of your web application, as this is a standard location for third-party class libraries in J2EE web applications. You can also use the `MysqlDataSource` or `MysqlConnectionPoolDataSource` classes in the `com.mysql.cj.jdbc` package, if your J2EE application server supports or requires them. The `javax.sql.XADataSource` interface is

implemented using the `com.mysql.cj.jdbc.MysqlXADataSource` class, which supports XA distributed transactions. The various `MysqlDataSource` classes support the following parameters (through standard set mutators):

- `user`
- `password`
- `serverName`
- `databaseName`
- `port`

4.2 Installing Connector/J Using Maven

You can also use Maven dependencies manager to install and configure the Connector/J library in your project. Connector/J is published in The Maven Central Repository with the following groupId and artifactId:

- groupId: `com.mysql`
- artifactId: `mysql-connector-j`

You can link the Connector/J library to your project by adding the following dependency in your `pom.xml` file:

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>x.y.z</version>
</dependency>
```

Notice that if you use Maven to manage your project dependencies, you do not need to explicitly refer to the library `protobuf-java` as it is resolved by dependency transitivity. However, if you do *not* want to use the X DevAPI features, you may also want to add a dependency exclusion to avoid linking the unneeded sub-library. For example:

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>x.y.z</version>
    <exclusions>
        <exclusion>
            <groupId>com.google.protobuf</groupId>
            <artifactId>protobuf-java</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

4.3 Installing from Source

> **Caution**
>
> You need to install Connector/J from source only if you want to build a customized version of Connector/J or if you are interested in helping us test our new code. To just get MySQL Connector/J up and running on your system, install Connector/J using a standard binary release distribution; see Section 4.1, “Installing Connector/J from a Binary Distribution” for instructions.

To install MySQL Connector/J from source, make sure that you have the following software on your system:

> **Tip**
>
> It is suggested that the latest versions available for the following software be used for compiling Connector/J; otherwise, some features might not be available.

- A Git client, if you want to check out the sources from our GitHub repository (available from http://git-scm.com/downloads).
- Apache Ant version 1.10.6 or newer (available from http://ant.apache.org/).
- JDK 1.8.x (available from https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).
- The following third-party libraries:
  - JUnit 5.10.2 (see installation and download information in the JUnit 5 User Guide). The following JAR files are required:
    - `junit-jupiter-api-5.10.2.jar` (available from, for example, https://central.sonatype.com/artifact/org.junit.jupiter/junit-jupiter-api/5.10.2/jar).
    - `junit-jupiter-engine-5.10.2.jar` (available from, for example, https://central.sonatype.com/artifact/org.junit.jupiter/junit-jupiter-engine/5.10.2/jar).
    - `junit-platform-commons-1.10.2.jar` (available from, for example, https://central.sonatype.com/artifact/org.junit.platform/junit-platform-commons/1.10.2/jar).
    - `junit-platform-engine-1.10.2.jar` (available from, for example, https://central.sonatype.com/artifact/org.junit.platform/junit-platform-engine/1.10.2/jar).
    - `junit-platform-launcher-1.10.2.jar` (available from, for example, https://central.sonatype.com/artifact/org.junit.platform/junit-platform-launcher/1.10.2/jar).
    - These additional JAR files, which JUnit 5 depends on:
      - `apiguardian-api-1.1.2.jar` (available from, for example, https://central.sonatype.com/artifact/org.apiguardian/apiguardian-api/1.1.2/jar).
      - `opentest4j-1.3.0.jar` (available from, for example, https://central.sonatype.com/artifact/org.opentest4j/opentest4j/1.3.0/jar).
  - Javassist 3.30.2 (`javassist-3.30.2-GA.jar`, available from, for example, https://central.sonatype.com/artifact/org.javassist/javassist/3.30.2-GA/bundle).
  - Protocol Buffers Java API 4.26.1 (`protobuf-java-4.26.1.jar`, available from, for example, https://central.sonatype.com/artifact/com.google.protobuf/protobuf-java/4.26.1/bundle).
  - C3P0 0.10.1 or newer (`c3p0-0.10.1.jar`, available from, for example, https://central.sonatype.com/artifact/com.mchange/c3p0/0.10.1/jar).
  - Simple Logging Facade API 2.0.13 or newer (`slf4j-api-2.0.13.jar`, available from, for example, https://central.sonatype.com/artifact/org.slf4j/slf4j-api/2.0.13/jar).
  - Java Hamcrest 2.2 or newer (`hamcrest-2.2.jar`, available from, for example, https://central.sonatype.com/artifact/org.hamcrest/hamcrest/2.2/jar).

- Oracle Cloud Infrastructure SDK for Java (`oci-java-sdk-common-3.41.2.jar`, available from, for example, https://central.sonatype.com/artifact/com.oracle.oci.sdk/oci-java-sdk-common/3.41.2/jar).
- OpenTelemetry API (`opentelemetry-api-1.38.0.jar`, available from, for example, https://central.sonatype.com/artifact/io.opentelemetry/opentelemetry-api/1.38.0).
- OpenTelemetry Context (`opentelemetry-context-1.38.0.jar`, available from, for example, https://central.sonatype.com/artifact/io.opentelemetry/opentelemetry-context/1.38.0).
- Open Test Alliance for the JVM (`opentest4j-1.3.0.jar`, available from, for example, https://central.sonatype.com/artifact/org.opentest4j/opentest4j/1.3.0).

To build MySQL Connector/J from source, follow these steps:

1. Make sure that you have JDK 1.8.x installed.
2. Obtain the sources for Connector/J by one of the following means:
   - Download the platform independent distribution archive (in `.tar.gz` or `.zip` format) for Connector/J, which contains the sources, from the Connector/J Download page. Extract contents of the archive into a folder named, for example, `mysql-connector-j`.
   - Download a source RPM package for Connector/J from Connector/J Download page and install it.
   - Check out the code from the source code repository for MySQL Connector/J located on GitHub at https://github.com/mysql/mysql-connector-j. The latest release of the Connector/J 9.0 series is on the `release/9.0` branch; use the following command to check it out:

     ```
     $> git clone --branch release/9.0 https://github.com/mysql/mysql-connector-j.git
     ```

     Under the current directory, the command creates a `mysql-connector-j` subdirectory , which contains the code you want.
3. Place all the required third-party libraries in a the directory called `lib` at the root of the source tree (that is, in `mysql-connector-j/lib`, if you have followed the steps above), or put them elsewhere and supply the location to Ant later (see Step 5 below).
4. Change your current working directory to the `mysql-connector-j` directory created in step 2 above.
5. In the directory, create a file named `build.properties` to indicate to Ant the location of the root directory for your JDK 1.8.x installation with the property `com.mysql.cj.build.jdk`, as well as the location for the extra libraries, if they are not in `mysql-connector-j/lib`, with the property `com.mysql.cj.extra.libs`. Here is a sample file with those properties set (replace the "*`path_to_*`*" parts with the appropriate file paths):

   ```
   com.mysql.cj.build.jdk=path_to_jdk_1.8
   com.mysql.cj.extra.libs=path_to_folder_for_extra_libraries
   ```

   Alternatively, you can set the values of those properties through the Ant `-D` options.

   > **Note**
   >
   > Going from Connector/J 5.1 to 8.0 and beyond, a number of Ant properties for building Connector/J have been renamed or removed; see Section 4.4.1.4, "Changes for Build Properties" for details.

6. Issue the following command to compile the driver and create a `.jar` file for Connector/J:

```
$> ant build
```

This creates a `build` directory in the current directory, where all the build output goes. A directory is created under the `build` directory, whose name includes the version number of the release you are building. That directory contains the sources, the compiled `.class` files, and a `.jar` file for deployment.

For information on all the build targets, including those that create a fully packaged distribution, issue the following command:

```
$> ant -projecthelp
```

7. Install the newly created `.jar` file for the JDBC driver as you would install a binary `.jar` file you download from MySQL by following the instructions given in Configuring the CLASSPATH or Configuring Connector/J for Application Servers.

4.4 Upgrading from an Older Version

This section has information for users who are upgrading from one version of Connector/J to another, or to a new version of the MySQL server that supports a more recent level of JDBC. A newer version of Connector/J might include changes to support new features, improve existing functionality, or comply with new standards.

Depending on the platform and the way you used to install Connector/J, upgrading can be performed by one of the following methods:

- Downloading a new platform-independent archive (`.tar`, `.tar.gz`, `.zip`, etc.) and overwriting with it your original installation created by an older archive.
- Updating the version of the Connector/J dependency in your Maven `.pom` file.
- Using the upgrade command of your Linux distro's package management system.
- Using the MySQL Installer for Windows, which can also perform automatic updates for Connector/J

See Chapter 4, *Connector/J Installation* for details on the installation and upgrade methods. You should also pay attention to any important changes in the new version like changes in 3rd-party dependencies, incompatibilities, etc.

4.4.1 Upgrading to MySQL Connector/J 9.0 from Connector/J 5.1

Upgrading an application developed for Connector/J 5.1 to use Connector/J 8.0 and beyond might require certain changes to your code or the environment in which it runs. Here are some changes for Connector/J going from 5.1 to 8.0 and beyond, for which adjustments might be required:

4.4.1.1 Running on the Java 8 Platform

Connector/J 8.0 and beyond is created specifically to run on the Java 8 platform. While Java 8 is known to be strongly compatible with earlier Java versions, incompatibilities do exist, and code designed to work on Java 7 might need to be adjusted before being run on Java 8. Developers should refer to the incompatibility information provided by Oracle.

4.4.1.2 Changes in Connection Properties

A complete list of Connector/J 9.0 connection properties are available in Section 6.3, “Configuration Properties”. The following are connection properties that have been changed (removed, added, have their names changed, or have their default values changed) going from Connector/J 5.1 to 8.0 and beyond.

Properties that have been removed (do not use them during connection):

- `useDynamicCharsetInfo`
- `useBlobToStoreUTF8OutsideBMP` , `utf8OutsideBmpExcludedColumnNamePattern`, and `utf8OutsideBmpIncludedColumnNamePattern`: MySQL 5.6 and later supports the utf8mb4 character set, which is the character set that should be used by Connector/J applications for supporting characters beyond the Basic Multilingual Plane (BMP) of Unicode Version 3.
- `useJvmCharsetConverters`: JVM character set conversion is now used in all cases
- The following date and time properties:
  - `dynamicCalendars`
  - `noTzConversionForTimeType`
  - `noTzConversionForDateType`
  - `cacheDefaultTimezone`
  - `useFastIntParsing`
  - `useFastDateParsing`
  - `useJDBCCompliantTimezoneShift`
  - `useLegacyDatetimeCode`
  - `useSSPSCompatibleTimezoneShift`
  - `useTimezone`
  - `useGmtMillisForDatetimes`
- `dumpMetadataOnColumnNotFound`
- `relaxAutoCommit`
- `strictFloatingPoint`
- `runningCTS13`
- `retainStatementAfterResultSetClose`
- `nullNamePatternMatchesAll`

Properties that have been added:

- `mysqlx.useAsyncProtocol` (deprecated)

Property that has its name changed:

- `com.mysql.jdbc.faultInjection.serverCharsetIndex` changed to `com.mysql.cj.testsuite.faultInjection.serverCharsetIndex`
- `loadBalanceEnableJMX` to `ha.enableJMX`
- `replicationEnableJMX` to `ha.enableJMX`

Properties that have their default values changed:

- `nullCatalogMeansCurrent` is now `false` by default

4.4.1.3 Changes in the Connector/J API

This section describes some of the more important changes to the Connector/J API going from version 5.1 to 8.0 and beyond. You might need to adjust your API calls accordingly:

- The name of the class that implements `java.sql.Driver` in MySQL Connector/J has changed from `com.mysql.jdbc.Driver` to `com.mysql.cj.jdbc.Driver`. The old class name has been deprecated.
- The names of these commonly-used classes and interfaces have also been changed:
  - ExceptionInterceptor: from `com.mysql.jdbc.ExceptionInterceptor` to `com.mysql.cj.exceptions.ExceptionInterceptor`
  - StatementInterceptor: from `com.mysql.jdbc.StatementInterceptorV2` to `com.mysql.cj.interceptors.QueryInterceptor`
  - ConnectionLifecycleInterceptor: from `com.mysql.jdbc.ConnectionLifecycleInterceptor` to `com.mysql.cj.jdbc.interceptors.ConnectionLifecycleInterceptor`
  - AuthenticationPlugin: from `com.mysql.jdbc.AuthenticationPlugin` to `com.mysql.cj.protocol.AuthenticationPlugin`
  - BalanceStrategy: from `com.mysql.jdbc.BalanceStrategy` to `com.mysql.cj.jdbc.ha.BalanceStrategy`
  - MysqlDataSource: from `com.mysql.jdbc.jdbc2.optional.MysqlDataSource` to `com.mysql.cj.jdbc.MysqlDataSource`
  - MysqlDataSourceFactory: from `com.mysql.jdbc.jdbc2.optional.MysqlDataSourceFactory` to `com.mysql.cj.jdbc.MysqlDataSourceFactory`
  - MysqlConnectionPoolDataSource: from `com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource` to `com.mysql.cj.jdbc.MysqlConnectionPoolDataSource`
  - MysqlXADataSource: from `com.mysql.jdbc.jdbc2.optional.MysqlXADataSource` to `com.mysql.cj.jdbc.MysqlXADataSource`
  - MysqlXid: from `com.mysql.jdbc.jdbc2.optional.MysqlXid` to `com.mysql.cj.jdbc.MysqlXid`

4.4.1.4 Changes for Build Properties

A number of Ant properties for building Connector/J from source have been renamed; see Table 4.1, “Changes with the Build Properties from Connector/J 5.1 to 8.0 and Beyond”

Table 4.1 Changes with the Build Properties from Connector/J 5.1 to 8.0 and Beyond

Old name	New name
`com.mysql.jdbc.extra.libs`	`com.mysql.cj.extra.libs`
`com.mysql.jdbc.jdk`	`com.mysql.cj.build.jdk`

Old name	New name
debug.enable	com.mysql.cj.build.addDebugInfo
com.mysql.jdbc.noCleanBetweenCompiles	com.mysql.cj.build.noCleanBetweenCompiles
com.mysql.jdbc.commercialBuild	com.mysql.cj.build.commercial
com.mysql.jdbc.filterLicense	com.mysql.cj.build.filterLicense
com.mysql.jdbc.noCryptoBuild	com.mysql.cj.build.noCrypto
com.mysql.jdbc.noSources	com.mysql.cj.build.noSources
com.mysql.jdbc.noMavenSources	com.mysql.cj.build.noMavenSources
major_version	com.mysql.cj.build.driver.version.major
minor_version	com.mysql.cj.build.driver.version.minor
subminor_version	com.mysql.cj.build.driver.version.subminor
version_status	com.mysql.cj.build.driver.version.status
extra.version	com.mysql.cj.build.driver.version.extra
snapshot.version	com.mysql.cj.build.driver.version.snapshot
version	com.mysql.cj.build.driver.version
full.version	com.mysql.cj.build.driver.version.full
prodDisplayName	com.mysql.cj.build.driver.displayName
prodName	com.mysql.cj.build.driver.name
fullProdName	com.mysql.cj.build.driver.fullName
buildDir	com.mysql.cj.build.dir
buildDriverDir	com.mysql.cj.build.dir.driver
mavenUploadDir	com.mysql.cj.build.dir.maven
distDir	com.mysql.cj.dist.dir
toPackage	com.mysql.cj.dist.dir.prepare
packageDest	com.mysql.cj.dist.dir.package
com.mysql.jdbc.docs.sourceDir	com.mysql.cj.dist.dir.prebuilt.docs

4.4.1.5 Change for Test Properties

A number of Ant properties for testing Connector/J have been renamed or removed; see Table 4.2, “Changes with the Test Properties from Connector/J 5.1 to 8.0 and Beyond”

Table 4.2 Changes with the Test Properties from Connector/J 5.1 to 8.0 and Beyond

Old name	New name
buildTestDir	com.mysql.cj.testsuite.build.dir
junit.results	com.mysql.cj.testsuite.junit.results
com.mysql.jdbc.testsuite.jvm	com.mysql.cj.testsuite.jvm
test	com.mysql.cj.testsuite.test.class
methods	com.mysql.cj.testsuite.test.methods
com.mysql.jdbc.testsuite.url	com.mysql.cj.testsuite.url
com.mysql.jdbc.testsuite.admin-url	com.mysql.cj.testsuite.url.admin

Old name	New name
com.mysql.jdbc.testsuite.ClusterUrl	com.mysql.cj.testsuite.url.cluster
com.mysql.jdbc.testsuite.url.sha256default	com.mysql.cj.testsuite.url.openssl
com.mysql.jdbc.testsuite.cantGrant	com.mysql.cj.testsuite.cantGrant
com.mysql.jdbc.testsuite.no-multi-hosts-tests	com.mysql.cj.testsuite.disable.multihost.tests
com.mysql.jdbc.test.ds.host	com.mysql.cj.testsuite.ds.host
com.mysql.jdbc.test.ds.port	com.mysql.cj.testsuite.ds.port
com.mysql.jdbc.test.ds.db	com.mysql.cj.testsuite.ds.db
com.mysql.jdbc.test.ds.user	com.mysql.cj.testsuite.ds.user
com.mysql.jdbc.test.ds.password	com.mysql.cj.testsuite.ds.password
com.mysql.jdbc.test.tabletype	com.mysql.cj.testsuite.loadstoreperf.tabletype
com.mysql.jdbc.testsuite.loadstoreperf.useBigResults	com.mysql.cj.testsuite.loadstoreperf.useBigResul
com.mysql.jdbc.testsuite.MiniAdminTest.runShutdown	com.mysql.cj.testsuite.miniAdminTest.runShutdown
com.mysql.jdbc.testsuite.noDebugOutput	com.mysql.cj.testsuite.noDebugOutput
com.mysql.jdbc.testsuite.retainArtifacts	com.mysql.cj.testsuite.retainArtifacts
com.mysql.jdbc.testsuite.runLongTests	com.mysql.cj.testsuite.runLongTests
com.mysql.jdbc.test.ServerController.basedir	com.mysql.cj.testsuite.serverController.basedir
com.mysql.jdbc.ReplicationConnection.isSlave	com.mysql.cj.testsuite.replicationConnection.isR
com.mysql.jdbc.test.isLocalHostnameReplacement	Removed
com.mysql.jdbc.testsuite.driver	Removed
com.mysql.jdbc.testsuite.url.default	Removed. No longer needed, as multi-JVM tests have been removed from the test suite.

4.4.1.6 Changes for Exceptions

Some exceptions have been removed from Connector/J going from version 5.1 to 8.0 and beyond. Applications that used to catch the removed exceptions should now catch the corresponding exceptions listed in Table 4.3 below.

> **Note**
>
> Some of these Connector/J 5.1 exceptions are duplicated in the com.mysql.jdbc.exception.jdbc4 package; that is indicated by “[jdbc4.]” in their names in Table 4.3.

Table 4.3 Changes for Exceptions from Connector/J 5.1 to 8.0 and Beyond

Removed Exception in Connector/J 5.1
com.mysql.jdbc.exceptions.jdbc4.CommunicationsException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLDataException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLIntegrityConstraintViolationException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLInvalidAuthorizationSpecException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLNonTransientConnectionException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLNonTransientException

Removed Exception in Connector/J 5.1
com.mysql.jdbc.exceptions.[jdbc4.]MySQLQueryInterruptedException
com.mysql.jdbc.exceptions.MySQLStatementCancelledException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLSyntaxErrorException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLTimeoutException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLTransactionRollbackException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLTransientConnectionException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLTransientException
com.mysql.jdbc.exceptions.[jdbc4.]MySQLIntegrityConstraintViolationException

4.4.1.7 Other Changes

Here are other changes with Connector/J 8.0 and beyond:

- Removed `ReplicationDriver`. Instead of using a separate driver, you can now obtain a connection for a replication setup just by using the `jdbc:mysql:replication://` scheme.
- See Chapter 4, *Connector/J Installation* for third-party libraries required for Connector/J 9.0 to work.
- *For Connector/J 8.0.22 and earlier:* Connector/J always performs time offset adjustments on date-time values, and the adjustments require one of the following to be true:
  - The MySQL server is configured with a canonical time zone that is recognizable by Java (for example, Europe/Paris, Etc/GMT-5, UTC, etc.)
  - The server's time zone is overridden by setting the Connector/J connection property `serverTimezone` (for example, `serverTimezone=Europe/Paris`).

    Note

    The Connector/J's behavior in this respect has changed since release 8.0.23. See Section 6.6.1, “Preserving Time Instants” for details. `serverTimezone` is now an alias for the connection property `connectionTimeZone`, which has replaced `serverTimezone`.

4.5 Testing Connector/J

The Connector/J source code repository or packages that are shipped with source code include an extensive test suite, containing test cases that can be executed independently. The test cases are divided into the following categories:

- *Unit tests:* They are methods located in packages aligning with the classes that they test.
- *Functional tests:* Classes from the package `testsuite.simple`. Include test code for the main features of Connector/J.
- *Performance tests:* Classes from the package `testsuite.perf`. Include test code to make measurements for the performance of Connector/J.
- *Regression tests:* Classes from the package `testsuite.regression`. Includes code for testing bug and regression fixes.
- *X DevAPI and X Protocol tests:* Classes from the package `testsuite.x` for testing X DevAPI and X Protocol functionality.

The bundled Ant build file contains targets like `test`, which can facilitate the process of running the Connector/J tests; see the target descriptions in the build file for details. To run the tests, in addition to fulfilling the requirements described in Section 4.3, “Installing from Source”, you must also set the following properties in the `build.properties` file or through the Ant `-D` options:

- `com.mysql.cj.testsuite.jvm`: the JVM to be used for the tests. If the property is not set, the JVM supplied with `com.mysql.cj.build.jdk` will be used.
- `com.mysql.cj.testsuite.url`: it specifies the JDBC URL for connection to a MySQL test server; see Section 6.2, “Connection URL Syntax”.
- `com.mysql.cj.testsuite.url.openssl`: (*for release 8.0.26 and earlier only*) it specifies the JDBC URL for connection to a MySQL test server compiled with OpenSSL; see Section 6.2, “Connection URL Syntax”.
- `com.mysql.cj.testsuite.mysqlx.url`: it specifies the X DevAPI URL for connection to a MySQL test server; see Section 6.2, “Connection URL Syntax”.
- `com.mysql.cj.testsuite.mysqlx.url.openssl`: (*for release 8.0.26 and earlier only*) it specifies the X DevAPI URL for connection to a MySQL test server compiled with OpenSSL; see Section 6.2, “Connection URL Syntax”.

After setting these parameters, run the tests with Ant in the following ways:

- Building the `test` target with `ant test` runs all test cases by default on a single server instance. If you want to run a particular test case, put the test's fully qualified class names in the `com.mysql.cj.testsuite.test.class` variable; for example:

  ```
  shell > ant -Dcom.mysql.cj.testsuite.test.class=testsuite.simple.StringUtilsTest test
  ```

  You can also run individual tests in a test case by specifying the names of the corresponding methods in the `com.mysql.cj.testsuite.test.methods` variable, separating multiple methods by commas; for example:

  ```
  shell > ant -Dcom.mysql.cj.testsuite.test.class=testsuite.simple.StringUtilsTest \
  -Dcom.mysql.cj.testsuite.test.methods=testIndexOfIgnoreCase,testGetBytes test
  ```

While the test results are partially reported by the console, complete reports in HTML and XML formats are provided. View the HTML report by opening `buildtest/junit/report/index.html`. XML version of the reports are located in the folder `buildtest/junit`.

> **Note**
>
> Going from Connector/J 5.1 to 8.0 and beyond, a number of Ant properties for testing Connector/J have been renamed or removed; see Section 4.4.1.5, “Change for Test Properties” for details.

Chapter 5 Connector/J Examples

Examples of using Connector/J are located throughout this document. This section provides a summary and links to these examples.

- Example 7.1, “Connector/J: Obtaining a connection from the `DriverManager`”
- Example 7.2, “Connector/J: Using java.sql.Statement to execute a `SELECT` query”
- Example 7.3, “Connector/J: Calling Stored Procedures”
- Example 7.4, “Connector/J: Using `Connection.prepareCall()`”
- Example 7.5, “Connector/J: Registering output parameters”
- Example 7.6, “Connector/J: Setting `CallableStatement` input parameters”
- Example 7.7, “Connector/J: Retrieving results and output parameter values”
- Example 7.8, “Connector/J: Retrieving `AUTO_INCREMENT` column values using `Statement.getGeneratedKeys()`”
- Example 7.9, “Connector/J: Retrieving `AUTO_INCREMENT` column values using `SELECT LAST_INSERT_ID()`”
- Example 7.10, “Connector/J: Retrieving `AUTO_INCREMENT` column values in `Updatable ResultSets`”
- Example 8.1, “Connector/J: Using a connection pool with a J2EE application server”
- Example 16.1, “Connector/J: Example of transaction with retry logic”

Chapter 6 Connector/J Reference

Table of Contents

This section of the manual contains reference material for MySQL Connector/J.

6.1 Driver/Datasource Class Name

The name of the class that implements `java.sql.Driver` in MySQL Connector/J is `com.mysql.cj.jdbc.Driver`.

6.2 Connection URL Syntax

This section explains the syntax of the URLs for connecting to MySQL.

This is the generic format of the connection URL:

```
protocol//[hosts][/database][?properties]
```

The URL consists of the following parts:

> **Important**
>
> Any reserved characters for URLs (for example, `/`, `:`, `@`, `(`, `)`, `[`, `]`, `&`, `#`, `=`, `?`, and space) that appear in any part of the connection URL must be percent encoded.

protocol

There are the possible protocols for a connection:

- `jdbc:mysql:` is for ordinary and basic JDBC failover connections.
- `jdbc:mysql:loadbalance:` is for load-balancing JDBC connections. See Section 9.3, “Configuring Load Balancing with Connector/J” for details.
- `jdbc:mysql:replication:` is for JDBC replication connections. See Section 9.4, “Configuring Source/Replica Replication with Connector/J” for details.
- `mysqlx:` is for X DevAPI connections.
- `jdbc:mysql+srv:` is for ordinary and basic failover JDBC connections that make use of DNS SRV records.
- `jdbc:mysql+srv:loadbalance:` is for load-balancing JDBC connections that make use of DNS SRV records.
- `jdbc:mysql+srv:replication:` is for replication JDBC connections that make use of DNS SRV records.
- `mysqlx+srv:` is for X DevAPI connections that make use of DNS SRV records.

hosts

Depending on the situation, the *hosts* part may consist simply of a host name, or it can be a complex structure consisting of various elements like multiple host names, port numbers, host-specific properties, and user credentials.

- Single host:
  - Single-host connections without adding host-specific properties:

- The *hosts* part is written in the format of *host*:*port*. This is an example of a simple single-host connection URL:

```
jdbc:mysql://host1:33060/sakila
```

- *host* can be an IPv4 or an IPv6 host name string, and in the latter case it must be put inside square brackets, for example "[1000:2000::abcd]." When *host* is not specified, the default value of `localhost` is used.
- *port* is a standard port number, i.e., an integer between 1 and 65535. The default port number for an ordinary MySQL connection is 3306, and it is 33060 for a connection using the X Protocol. If *port* is not specified, the corresponding default is used.

- Single-host connections adding host-specific properties:
  - In this case, the host is defined as a succession of *key=value* pairs. Keys are used to identify the host, the port, as well as any host-specific properties. There are two alternate formats for specifying keys:
    - The "address-equals" form:

```
address=(host=host_or_ip)(port=port)(key1=value1)(key2=value2)...(keyN=valueN)
```

      Here is a sample URL using the"address-equals" form :

```
jdbc:mysql://address=(host=myhost)(port=1111)(key1=value1)/db
```

    - The "key-value" form:

```
(host=host,port=port,key1=value1,key2=value2,...,keyN=valueN)
```

      Here is a sample URL using the "key-value" form :

```
jdbc:mysql://(host=myhost,port=1111,key1=value1)/db
```

  - The host and the port are identified by the keys *host* and *port*. The descriptions of the format and default values of *host* and *port* in Single host without host-specific properties [22] above also apply here.
  - Other keys that can be added include `user`, `password`, `protocol`, and so on. They override the global values set in the *properties* part of the URL. Limit the overrides to user, password, network timeouts, and statement and metadata cache sizes; the effects of other per-host overrides are not defined.
  - Different protocols may require different keys. For example, the `mysqlx:` scheme uses two special keys, *address* and *priority*. *address* is a *host*:*port* pair and *priority* an integer. For example:

```
mysqlx://(address=host:1111,priority=1,key1=value1)/db
```

  - *key* is case-sensitive. Two keys differing in case only are considered conflicting, and there are no guarantees on which one will be used.

- Multiple hosts

  There are two formats for specifying multiple hosts:

  - List hosts in a comma-separated list:

```
host1,host2,...,hostN
```

Each host can be specified in any of the three ways described in Single host [22] above. Here are some examples:

```
jdbc:mysql://myhost1:1111,myhost2:2222/db
jdbc:mysql://address=(host=myhost1)(port=1111)(key1=value1),address=(host=myhost2)(port=2222)(key2=value2)
jdbc:mysql://(host=myhost1,port=1111,key1=value1),(host=myhost2,port=2222,key2=value2)/db
jdbc:mysql://myhost1:1111,(host=myhost2,port=2222,key2=value2)/db
mysqlx://(address=host1:1111,priority=1,key1=value1),(address=host2:2222,priority=2,key2=value2)/db
```

- List hosts in a comma-separated list, and then encloses the list by square brackets:

```
[host1,host2,...,hostN]
```

This is called the host sublist form, which allows sharing of the user credentials by all hosts in the list as if they are a single host. Each host in the list can be specified in any of the three ways described in Single host [22] above. Here are some examples:

```
jdbc:mysql://sandy:secret@[myhost1:1111,myhost2:2222]/db
jdbc:mysql://sandy:secret@[address=(host=myhost1)(port=1111)(key1=value1),address=(host=myhost2)(port=2222
jdbc:mysql://sandy:secret@[myhost1:1111,address=(host=myhost2)(port=2222)(key2=value2)]/db
```

While it is not possible to write host sublists recursively, a host list may contain host sublists as its member hosts.

- User credentials

User credentials can be set outside of the connection URL—for example, as arguments when getting a connection from the `java.sql.DriverManager` (see Section 6.3, “Configuration Properties” for details). When set with the connection URL, there are several ways to specify them:

- Prefix the a single host, a host sublist (see Multiple hosts [23]), or any host in a list of hosts with the user credentials with an `@`:

```
user:password@host_or_host_sublist
```

For example:

```
mysqlx://sandy:secret@[(address=host1:1111,priority=1,key1=value1),(address=host2:2222,priority=2,key2=val
```

- Use the keys `user` and `password` to specify credentials for each host:

```
(user=sandy)(password=mypass)
```

For example:

```
jdbc:mysql://[(host=myhost1,port=1111,user=sandy,password=secret),(host=myhost2,port=2222,user=finn,passwo
jdbc:mysql://address=(host=myhost1)(port=1111)(user=sandy)(password=secret),address=(host=myhost2)(port=22
```

In both forms, when multiple user credentials are specified, the one to the left takes precedence—that is, going from left to right in the connection string, the first one found that is applicable to a host is the one that is used.

Inside a host sublist, no host can have user credentials in the @ format, but individual host can have user credentials specified in the key format.

database

The default database or catalog to open. If the database is not specified, the connection is made with no default database. In this case, either call the `setCatalog()` method on the `Connection` instance, or specify table names using the database name (that is, `SELECT dbname.tablename.colname FROM dbname.tablename...`) in your SQL statements. Opening a connection without specifying the database to use is, in general, only useful when building tools that work with multiple databases, such as GUI database managers.

> **Note**
>
> Always use the `Connection.setCatalog()` method to specify the desired database in JDBC applications, rather than the `USE database` statement.

properties

A succession of global properties applying to all hosts, preceded by `?` and written as `key=value` pairs separated by the symbol "`&`." Here are some examples:

```
jdbc:mysql://(host=myhost1,port=1111),(host=myhost2,port=2222)/db?key1=value1&key2=value2&key3=value3
```

The following are true for the key-value pairs:

- `key` and `value` are just strings. Proper type conversion and validation are performed internally in Connector/J.
- `key` is case-sensitive. Two keys differing in case only are considered conflicting, and it is uncertain which one will be used.
- Any host-specific values specified with key-value pairs as explained in Single host with host-specific properties [23] and Multiple hosts [23] above override the global values set here.

See Section 6.3, "Configuration Properties" for details about the configuration properties.

6.3 Configuration Properties

Configuration properties define how Connector/J will make a connection to a MySQL server. Unless otherwise noted, properties can be set for a `DataSource` object or for a `Connection` object.

Configuration properties can be set in one of the following ways:

- Using the `set*()` methods on MySQL implementations of `java.sql.DataSource` (which is the preferred method when using implementations of `java.sql.DataSource`):
  - `com.mysql.cj.jdbc.MysqlDataSource`
  - `com.mysql.cj.jdbc.MysqlConnectionPoolDataSource`
- As a key-value pair in the `java.util.Properties` instance passed to `DriverManager.getConnection()` or `Driver.connect()`
- As a JDBC URL parameter in the URL given to `java.sql.DriverManager.getConnection()`, `java.sql.Driver.connect()` or the MySQL implementations of the `javax.sql.DataSource setURL()` method. If you specify a configuration property in the URL without providing a value for it, nothing will be set; for example, adding `useServerPrepStmts` alone to the URL does not make Connector/J use server-side prepared statements; you need to add `useServerPrepStmts=true`.

> **Note**
>
> If the mechanism you use to configure a JDBC URL is XML-based, use the XML character literal `&` to separate configuration parameters, as the ampersand is a reserved character for XML.

The properties are listed by categories in the following tables and then in the subsections that follow. Click on a property name in the tables to see its full description in the subsections.

Table 6.1 Authentication Properties

Name	Default Value	Since Version
`user`	-	all versions
`password`	-	all versions
`password1`	-	8.0.28
`password2`	-	8.0.28
`password3`	-	8.0.28
`authenticationPlugins`	-	5.1.19
`disabledAuthenticationPlugins`	-	5.1.19
`defaultAuthenticationPlugin`	caching_sha2_password	5.1.19
`ldapServerHostname`	-	8.0.23
`ociConfigFile`	-	8.0.27
`ociConfigProfile`	DEFAULT	8.0.33
`authenticationWebAuthnCallbackHandler`	-	8.2.0

Table 6.2 Connection Properties

Name	Default Value	Since Version
`connectionAttributes`	-	5.1.25
`connectionLifecycleInterceptors`	-	5.1.4
`useConfigs`	-	3.1.5
`clientInfoProvider`	com.mysql.cj.jdbc.CommentClientInfoProvider	5.1.0
`createDatabaseIfNotExist`	false	3.1.9
`databaseTerm`	CATALOG	8.0.17
`detectCustomCollations`	false	5.1.29
`disconnectOnExpiredPasswords`	true	5.1.23
`interactiveClient`	false	3.1.0
`passwordCharacterEncoding`	-	5.1.7
`propertiesTransform`	-	3.1.4
`rollbackOnPooledClose`	true	3.0.15
`useAffectedRows`	false	5.1.7

Table 6.3 Session Properties

Name	Default Value	Since Version
`sessionVariables`	-	3.1.8

Name	Default Value	Since Version
characterEncoding	-	1.1g
characterSetResults	-	3.0.13
connectionCollation	-	3.0.13
customCharsetMapping	-	8.0.26
trackSessionState	false	8.0.26

Table 6.4 Networking Properties

Name	Default Value	Since Version
socksProxyHost	-	5.1.34
socksProxyPort	1080	5.1.34
socketFactory	com.mysql.cj.protocol.StandardSocketFactory	3.0.3
connectTimeout	0	3.0.1
socketTimeout	0	3.0.1
dnsSrv	false	8.0.19
localSocketAddress	-	5.0.5
maxAllowedPacket	65535	5.1.8
socksProxyRemoteDns	false	8.0.29
tcpKeepAlive	true	5.0.7
tcpNoDelay	true	5.0.7
tcpRcvBuf	0	5.0.7
tcpSndBuf	0	5.0.7
tcpTrafficClass	0	5.0.7
useCompression	false	3.0.17
useUnbufferedInput	true	3.0.11

Table 6.5 Security Properties

Name	Default Value	Since Version
paranoid	false	3.0.1
serverRSAPublicKeyFile	-	5.1.31
allowPublicKeyRetrieval	false	5.1.31
sslMode	PREFERRED	8.0.13
trustCertificateKeyStoreUrl	-	5.1.0
trustCertificateKeyStoreType	JKS	5.1.0
trustCertificateKeyStorePassword	-	5.1.0
fallbackToSystemTrustStore	true	8.0.22
clientCertificateKeyStoreUrl	-	5.1.0
clientCertificateKeyStoreType	JKS	5.1.0
clientCertificateKeyStorePassword	-	5.1.0
fallbackToSystemKeyStore	true	8.0.22

Name	Default Value	Since Version
tlsCiphersuites	-	5.1.35
tlsVersions	-	8.0.8
fipsCompliantJsse	false	8.1.0
KeyManagerFactoryProvider	-	8.1.0
trustManagerFactoryProvider	-	8.1.0
keyStoreProvider	-	8.1.0
sslContextProvider	-	8.1.0
allowLoadLocalInfile	false	3.0.3
allowLoadLocalInfileInPath	-	8.0.22
allowMultiQueries	false	3.1.1
allowUrlInLocalInfile	false	3.1.4
requireSSL	false	3.1.0
useSSL	true	3.0.2
verifyServerCertificate	false	5.1.6

Table 6.6 Statements Properties

Name	Default Value	Since Version
cacheDefaultTimeZone	true	8.0.20
continueBatchOnError	true	3.0.3
dontTrackOpenResources	false	3.1.7
queryInterceptors	-	8.0.7
queryTimeoutKillsConnection	false	5.1.9

Table 6.7 Prepared Statements Properties

Name	Default Value	Since Version
allowNanAndInf	false	3.1.5
autoClosePStmtStreams	false	3.1.12
compensateOnDuplicateKeyUpdateCounts	false	5.1.7
emulateUnsupportedPstmts	true	3.1.7
generateSimpleParameterMetadata	false	5.0.5
processEscapeCodesForPrepStmts	true	3.1.12
useServerPrepStmts	false	3.1.0
useStreamLengthsInPrepStmts	true	3.0.2

Table 6.8 Result Sets Properties

Name	Default Value	Since Version
clobberStreamingResults	false	3.0.9
emptyStringsConvertToZero	true	3.1.8
holdResultsOpenOverStatementClose	false	3.1.7

Name	Default Value	Since Version
jdbcCompliantTruncation	true	3.1.2
maxRows	-1	all versions
netTimeoutForStreamingResults	600	5.1.0
padCharsWithSpace	false	5.0.6
populateInsertRowWithDefaultValues	false	5.0.5
scrollTolerantForwardOnly	false	8.0.24
strictUpdates	true	3.0.4
tinyInt1isBit	true	3.0.16
transformedBitIsBoolean	false	3.1.9

Table 6.9 Metadata Properties

Name	Default Value	Since Version
getProceduresReturnsFunctions	true	5.1.26
noAccessToProcedureBodies	false	5.0.3
nullDatabaseMeansCurrent	false	3.1.8
useHostsInPrivileges	true	3.0.2
useInformationSchema	false	5.0.0

Table 6.10 BLOB/CLOB processing Properties

Name	Default Value	Since Version
blobSendChunkSize	1048576	3.1.9
blobsAreStrings	false	5.0.8
clobCharacterEncoding	-	5.0.0
emulateLocators	false	3.1.0
functionsNeverReturnBlobs	false	5.0.8
locatorFetchBufferSize	1048576	3.2.1

Table 6.11 Datetime types processing Properties

Name	Default Value	Since Version
connectionTimeZone	-	3.0.2
forceConnectionTimeZoneToSession	false	8.0.23
noDatetimeStringSync	false	3.1.7
preserveInstants	true	8.0.23
sendFractionalSeconds	true	5.1.37
sendFractionalSecondsForTime	true	8.0.23
treatMysqlDatetimeAsTimestamp	false	8.2.0
treatUtilDateAsTimestamp	true	5.0.5
yearIsDateType	true	3.1.9
zeroDateTimeBehavior	EXCEPTION	3.1.4

Table 6.12 High Availability and Clustering Properties

Name	Default Value	Since Version
autoReconnect	false	1.1
autoReconnectForPools	false	3.1.3
failOverReadOnly	true	3.0.12
maxReconnects	3	1.1
reconnectAtTxEnd	false	3.0.10
retriesAllDown	120	5.1.6
initialTimeout	2	1.1
queriesBeforeRetrySource	50	3.0.2
secondsBeforeRetrySource	30	3.0.2
allowReplicaDownConnections	false	6.0.2
allowSourceDownConnections	false	5.1.27
ha.enableJMX	false	5.1.27
loadBalanceHostRemovalGracePeriod	15000	6.0.3
readFromSourceWhenNoReplicas	false	6.0.2
selfDestructOnPingMaxOperations	0	5.1.6
selfDestructOnPingSecondsLifetime	0	5.1.6
ha.loadBalanceStrategy	random	5.0.6
loadBalanceAutoCommitStatementRegex	-	5.1.15
loadBalanceAutoCommitStatementThreshold	0	5.1.15
loadBalanceBlocklistTimeout	0	5.1.0
loadBalanceConnectionGroup	-	5.1.13
loadBalanceExceptionChecker	com.mysql.cj.jdbc.ha.StandardLoadBalanceExceptionChecker	5.1.13
loadBalancePingTimeout	0	5.1.13
loadBalanceSQLExceptionSubclassFailover	-	5.1.13
loadBalanceSQLStateFailover	-	5.1.13
loadBalanceValidateConnectionOnSwapServer	false	5.1.13
pinGlobalTxToPhysicalConnection	false	5.0.1
replicationConnectionGroup	-	8.0.7
resourceId	-	5.0.1
serverAffinityOrder	-	8.0.8

Table 6.13 Performance Extensions Properties

Name	Default Value	Since Version
callableStmtCacheSize	100	3.1.2
metadataCacheSize	50	3.1.1
useLocalSessionState	false	3.1.7
useLocalTransactionState	false	5.1.7
prepStmtCacheSize	25	3.0.10

Name	Default Value	Since Version
prepStmtCacheSqlLimit	256	3.0.10
queryInfoCacheFactory	com.mysql.cj.PerConnectionLRUFactory	5.1.1
serverConfigCacheFactory	com.mysql.cj.util.PerVmServerConfigCacheFactory	5.1.1
alwaysSendSetIsolation	true	3.1.7
maintainTimeStats	true	3.1.9
useCursorFetch	false	5.0.0
cacheCallableStmts	false	3.1.2
cachePrepStmts	false	3.0.10
cacheResultSetMetadata	false	3.1.1
cacheServerConfiguration	false	3.1.5
defaultFetchSize	0	3.1.9
dontCheckOnDuplicateKeyUpdateInSQL	false	5.1.32
elideSetAutoCommits	false	3.1.3
enableEscapeProcessing	true	6.0.1
enableQueryTimeouts	true	5.0.6
largeRowSizeThreshold	2048	5.1.1
readOnlyPropagatesToServer	true	5.1.35
rewriteBatchedStatements	false	3.1.13
useReadAheadInput	true	3.1.5

Table 6.14 Debugging/Profiling Properties

Name	Default Value	Since Version
logger	com.mysql.cj.log.StandardLogger	3.1.1
profilerEventHandler	com.mysql.cj.log.LoggingProfilerEventHandler	5.1.6
useNanosForElapsedTime	false	5.0.7
maxQuerySizeToLog	2048	3.1.3
maxByteArrayAsHex	1024	8.0.31
profileSQL	false	3.1.0
logSlowQueries	false	3.1.2
slowQueryThresholdMillis	2000	3.1.2
slowQueryThresholdNanos	0	5.0.7
autoSlowLog	true	5.1.4
explainSlowQueries	false	3.1.2
gatherPerfMetrics	false	3.1.2
reportMetricsIntervalMillis	30000	3.1.2
logXaCommands	false	5.0.5
traceProtocol	false	3.1.2
enablePacketDebug	false	3.1.3
packetDebugBufferSize	20	3.1.3

Name	Default Value	Since Version
useUsageAdvisor	false	3.1.1
resultSetSizeThreshold	100	5.0.5
autoGenerateTestcaseScript	false	3.1.9
openTelemetry	PREFERRED	8.4.0

Table 6.15 Exceptions/Warnings Properties

Name	Default Value	Since Version
dumpQueriesOnException	false	3.1.3
exceptionInterceptors	-	5.1.8
ignoreNonTxTables	false	3.0.9
includeInnodbStatusInDeadlockExceptions	false	5.0.7
includeThreadDumpInDeadlockExceptions	false	5.1.15
includeThreadNamesAsStatementComment	false	5.1.15
useOnlyServerErrorMessages	true	3.0.15

Table 6.16 Tunes for integration with other products Properties

Name	Default Value	Since Version
overrideSupportsIntegrityEnhancementFacility	false	3.1.12
ultraDevHack	false	2.0.3

Table 6.17 JDBC compliance Properties

Name	Default Value	Since Version
useColumnNamesInFindColumn	false	5.1.7
pedantic	false	3.0.0
useOldAliasMetadataBehavior	false	5.0.4

Table 6.18 X Protocol and X DevAPI Properties

Name	Default Value	Since Version
xdevapi.auth	PLAIN	8.0.8
xdevapi.compression	PREFERRED	8.0.20
xdevapi.compression-algorithms	zstd_stream,lz4_message,deflate_stream	8.0.22
xdevapi.compression-extensions	-	8.0.22
xdevapi.connect-timeout	10000	8.0.13
xdevapi.connection-attributes	-	8.0.16
xdevapi.dns-srv	false	8.0.19
xdevapi.fallback-to-system-keystore	true	8.0.22
xdevapi.fallback-to-system-truststore	true	8.0.22

Name	Default Value	Since Version
xdevapi.ssl-keystore	-	8.0.22
xdevapi.ssl-keystore-password	-	8.0.22
xdevapi.ssl-keystore-type	JKS	8.0.22
xdevapi.ssl-mode	REQUIRED	8.0.7
xdevapi.ssl-truststore	-	6.0.6
xdevapi.ssl-truststore-password	-	6.0.6
xdevapi.ssl-truststore-type	JKS	6.0.6
xdevapi.tls-ciphersuites	-	8.0.19
xdevapi.tls-versions	-	8.0.19

6.3.1 Authentication

- user

  The user to connect as. If none is specified, it is authentication plugin dependent what user name is used. Built-in authentication plugins default to the session login user name.

Since Version	all versions

- password

  The password to use when authenticating the user.

Since Version	all versions

- password1

  The password to use in the first phase of a Multi-Factor Authentication workflow. It is a synonym of the connection property 'password' and can also be set with user credentials in the connection string.

Since Version	8.0.28

- password2

  The password to use in the second phase of a Multi-Factor Authentication workflow.

Since Version	8.0.28

- password3

  The password to use in the third phase of a Multi-Factor Authentication workflow.

Since Version	8.0.28

- authenticationPlugins

  Comma-delimited list of classes that implement the interface 'com.mysql.cj.protocol.AuthenticationPlugin'. These plugins will be loaded at connection initialization

and can be used together with their sever-side counterparts for authenticating users, unless they are disabled in the connection property 'disabledAuthenticationPlugins'.

Since Version	5.1.19

- `disabledAuthenticationPlugins`

  Comma-delimited list of authentication plugins client-side protocol names or classes implementing the interface 'com.mysql.cj.protocol.AuthenticationPlugin'. The authentication plugins listed will not be used for authenticating users and, if anyone of them is required during the authentication exchange, the connection fails. The default authentication plugin specified in the property 'defaultAuthenticationPlugin' cannot be disabled.

Since Version	5.1.19

- `defaultAuthenticationPlugin`

  The default authentication plugin client-side protocol name or a fully qualified name of a class that implements the interface 'com.mysql.cj.protocol.AuthenticationPlugin'. The specified authentication plugin must be either one of the built-in authentication plugins or one of the plugins listed in the property 'authenticationPlugins'. Additionally, the default authentication plugin cannot be disabled with the property 'disabledAuthenticationPlugins'. Neither an empty nor unknown plugin name or class can be set for this property.

  By default, Connector/J honors the server-side default authentication plugin, which is known after receiving the initial handshake packet, and falls back to this property's default value if that plugin cannot be used. However, when a value is explicitly provided to this property, Connector/J then overrides the server-side default authentication plugin and always tries first the plugin specified with this property.

Default Value	caching_sha2_password
Since Version	5.1.19

- `ldapServerHostname`

  When using MySQL's LDAP pluggable authentication with GSSAPI/Kerberos authentication method, allows setting the LDAP service principal hostname as configured in the Kerberos KDC. If this property is not set, Connector/J takes the system property 'java.security.krb5.kdc' and extracts the hostname (short name) from its value and uses it. If neither is set, the connection fails with an exception.

Since Version	8.0.23

- `ociConfigFile`

  The location of the OCI configuration file as required by the OCI SDK for Java. Default value is "~/.oci/config" for Unix-like systems and "%HOMEDRIVE%%HOMEPATH%.oci\config" for Windows.

Since Version	8.0.27

- `ociConfigProfile`

  The profile in the OCI configuration file specified in 'ociConfigFile', from where the configuration to use in the 'authentication_oci_client' authentication plugin is to be read.

Default Value	DEFAULT

Since Version	8.0.33

- `authenticationWebAuthnCallbackHandler`

  Fully-qualified class name of a class implementing the interface 'com.mysql.cj.callback.MysqlCallbackHandler'. This class will be used by the WebAuthn authentication plugin to obtain the authenticator data and signature required for the FIDO authentication process. See the documentation of com.mysql.cj.callback.WebAuthnAuthenticationCallback for more details.

Since Version	8.2.0

6.3.2 Connection

- `connectionAttributes`

  A comma-delimited list of user-defined "key:value" pairs, in addition to standard MySQL-defined "key:value" pairs, to be passed to MySQL Server for display as connection attributes in the 'PERFORMANCE_SCHEMA' tables 'session_account_connect_attrs' and 'session_connect_attrs'. Example usage: "connectionAttributes=key1:value1,key2:value2" This functionality is available for use with MySQL Server version 5.6 or later only. Earlier versions of MySQL Server do not support connection attributes, causing this configuration option to be ignored. Setting "connectionAttributes=none" will cause connection attribute processing to be bypassed for situations where Connection creation/initialization speed is critical.

Since Version	5.1.25

- `connectionLifecycleInterceptors`

  A comma-delimited list of classes that implement 'com.mysql.cj.jdbc.interceptors.ConnectionLifecycleInterceptor' that should be notified of connection lifecycle events (creation, destruction, commit, rollback, setting the current database and changing the autocommit mode) and potentially alter the execution of these commands. 'ConnectionLifecycleInterceptors' are stackable, more than one interceptor may be specified via the configuration property as a comma-delimited list, with the interceptors executed in order from left to right.

Since Version	5.1.4

- `useConfigs`

  Load the comma-delimited list of configuration properties for specifying combinations of options for particular scenarios. These properties are loaded before parsing the URL or applying user-specified properties. Allowed values are "3-0-Compat", "clusterBase", "coldFusion", "fullDebug", "maxPerformance", "maxPerformance-8-0" and "solarisMaxPerformance", and they correspond to properties files shipped within the Connector/J jar file, under "com/mysql/cj/configurations".

Since Version	3.1.5

- `clientInfoProvider`

  The name of a class that implements the 'com.mysql.cj.jdbc.ClientInfoProvider' interface in order to support JDBC-4.0's 'Connection.get/setClientInfo()' methods.

Default Value	com.mysql.cj.jdbc.CommentClientInfoProvider
Since Version	5.1.0

- `createDatabaseIfNotExist`

  Creates the database given in the URL if it doesn't yet exist. Assumes the configured user has permissions to create databases.

Default Value	false
Since Version	3.1.9

- `databaseTerm`

  MySQL uses the term "schema" as a synonym of the term "database," while Connector/J historically takes the JDBC term "catalog" as synonymous to "database". This property sets for Connector/J which of the JDBC terms "catalog" and "schema" is used in an application to refer to a database. The property takes one of the two values "CATALOG" or "SCHEMA" and uses it to determine (1) which Connection methods can be used to set/get the current database (e.g. 'setCatalog()' or 'setSchema()'?), (2) which arguments can be used within the various 'DatabaseMetaData' methods to filter results (e.g. the catalog or 'schemaPattern' argument of 'getColumns()'?), and (3) which fields in the result sets returned by 'DatabaseMetaData' methods contain the database identification information (i.e., the 'TABLE_CAT' or 'TABLE_SCHEM' field in the result set returned by 'getTables()'?).

  If "databaseTerm=CATALOG", 'schemaPattern' for searches are ignored and calls of schema methods (like 'setSchema()' or get 'Schema()') become no-ops, and vice versa.

Default Value	CATALOG
Since Version	8.0.17

- `detectCustomCollations`

  Should the driver detect custom charsets/collations installed on server? If this option set to "true" the driver gets actual charsets/collations from the server each time a connection establishes. This could slow down connection initialization significantly.

Default Value	false
Since Version	5.1.29

- `disconnectOnExpiredPasswords`

  If 'disconnectOnExpiredPasswords' is set to "false" and password is expired then server enters sandbox mode and sends 'ERR(08001, ER_MUST_CHANGE_PASSWORD)' for all commands that are not needed to set a new password until a new password is set.

Default Value	true
Since Version	5.1.23

- `interactiveClient`

  Set the 'CLIENT_INTERACTIVE' flag, which tells MySQL to timeout connections based on 'interactive_timeout' instead of 'wait_timeout'.

Default Value	false
Since Version	3.1.0

- `passwordCharacterEncoding`

Instructs the server to use the default character set for the specified Java encoding during the authentication phase. If this property is not set, Connector/J falls back to the collation name specified in the property 'connectionCollation' or to the Java encoding specified in the property 'characterEncoding', in that order of priority. The default collation of the character set utf8mb4 is used if none of the properties is set.

Since Version	5.1.7

- `propertiesTransform`

  An implementation of 'com.mysql.cj.conf.ConnectionPropertiesTransform' that the driver will use to modify connection string properties passed to the driver before attempting a connection.

Since Version	3.1.4

- `rollbackOnPooledClose`

  Should the driver issue a 'rollback()' when the logical connection in a pool is closed?

Default Value	true
Since Version	3.0.15

- `useAffectedRows`

  Don't set the 'CLIENT_FOUND_ROWS' flag when connecting to the server. Note that this is not JDBC-compliant and it will break most applications that rely on "found" rows vs. "affected rows" for DML statements, but does cause correct update counts from "INSERT ... ON DUPLICATE KEY UPDATE" statements to be returned by the server.

Default Value	false
Since Version	5.1.7

6.3.3 Session

- `sessionVariables`

  A comma or semicolon separated list of "name=value" pairs to be sent as "SET [SESSION] ..." to the server when the driver connects.

Since Version	3.1.8

- `characterEncoding`

  Instructs the server to set session system variables 'character_set_client' and 'character_set_connection' to the default character set supported by MySQL for the specified Java character encoding and set 'collation_connection' to the default collation for this character set. If neither this property nor the property 'connectionCollation' is set:

  For Connector/J 8.0.25 and earlier, the driver will try to use the server's default character set;

  For Connector/J 8.0.26 and later, the driver will use "utf8mb4".

Since Version	1.1g

- `characterSetResults`

  Instructs the server to return the data encoded with the default character set for the specified Java encoding. If not set or set to "null", the server will send data in its original character set and the driver will decode it according to the result metadata.

Since Version	3.0.13

- `connectionCollation`

  Instructs the server to set session system variable 'collation_connection' to the specified collation name and set 'character_set_client' and 'character_set_connection' to a corresponding character set. This property overrides the value of 'characterEncoding' with the default character set this collation belongs to, if and only if 'characterEncoding' is not configured or is configured with a character set that is incompatible with the collation. That means 'connectionCollation' may not always correct a mismatch of character sets. For example, if 'connectionCollation' is set to "latin1_swedish_ci", the corresponding character set is "latin1" for MySQL, which maps it to the Java character set "windows-1252"; so if 'characterEncoding' is not set,"windows-1252" is the character set that will be used; but if 'characterEncoding' has been set to, e.g. "ISO-8859-1", that is compatible with "latin1_swedish_ci", so the character encoding setting is left unchanged; and if client is actually using "windows-1252" (which is similar but different from "ISO-8859-1"), errors would occur for some characters. If neither this property nor the property 'characterEncoding' is set:

  For Connector/J 8.0.25 and earlier, the driver will try to use the server's default character set;

  For Connector/J 8.0.26 and later, the driver will use utf8mb4's default collation.

Since Version	3.0.13

- `customCharsetMapping`

  A comma-delimited list of custom "charset:java encoding" pairs.

  In case the MySQL server is configured with custom character sets and "detectCustomCollations=true", Connector/J needs to know which Java character encoding to use for the data represented by these character sets. Example usage: "customCharsetMapping=charset1:UTF-8,charset2:Cp1252".

Since Version	8.0.26

- `trackSessionState`

  Receive server session state changes on query results. These changes are accessible via 'MysqlConnection.getServerSessionStateController()'.

Default Value	false
Since Version	8.0.26

6.3.4 Networking

- `socksProxyHost`

  Name or IP address of a SOCKS host to connect through.

Since Version	5.1.34

- `socksProxyPort`

  Port of the SOCKS server.

Default Value	1080
Since Version	5.1.34

- `socketFactory`

  The name of the class that the driver should use for creating socket connections to the server. This class must implement the interface 'com.mysql.cj.protocol.SocketFactory' and have a public no-args constructor.

Default Value	com.mysql.cj.protocol.StandardSocketFactory
Since Version	3.0.3

- `connectTimeout`

  Timeout for socket connect (in milliseconds), with 0 being no timeout.

Default Value	0
Since Version	3.0.1

- `socketTimeout`

  Timeout, specified in milliseconds, on network socket operations. Value "0" means no timeout.

Default Value	0
Since Version	3.0.1

- `dnsSrv`

  Should the driver use the given host name to lookup for DNS SRV records and use the resulting list of hosts in a multi-host failover connection? Note that a single host name and no port must be provided when this option is enabled.

Default Value	false
Since Version	8.0.19

- `localSocketAddress`

  Hostname or IP address given to explicitly configure the interface that the driver will bind the client side of the TCP/IP connection to when connecting.

Since Version	5.0.5

- `maxAllowedPacket`

  Maximum allowed packet size to send to server. If not set, the value of system variable 'max_allowed_packet' will be used to initialize this upon connecting. This value will not take effect if set

larger than the value of 'max_allowed_packet'. Also, due to an internal dependency with the property 'blobSendChunkSize', this setting has a minimum value of "8203" if 'useServerPrepStmts' is set to "true".

Default Value	65535
Since Version	5.1.8

- `socksProxyRemoteDns`

  When using a SOCKS proxy, whether the DNS lookup for the database host should be performed locally or through the SOCKS proxy.

Default Value	false
Since Version	8.0.29

- `tcpKeepAlive`

  If connecting using TCP/IP, should the driver set 'SO_KEEPALIVE'?

Default Value	true
Since Version	5.0.7

- `tcpNoDelay`

  If connecting using TCP/IP, should the driver set 'SO_TCP_NODELAY', disabling the Nagle Algorithm?

Default Value	true
Since Version	5.0.7

- `tcpRcvBuf`

  If connecting using TCP/IP, should the driver set 'SO_RCV_BUF' to the given value? The default value of "0", means use the platform default value for this property.

Default Value	0
Since Version	5.0.7

- `tcpSndBuf`

  If connecting using TCP/IP, should the driver set 'SO_SND_BUF' to the given value? The default value of "0", means use the platform default value for this property.

Default Value	0
Since Version	5.0.7

- `tcpTrafficClass`

  If connecting using TCP/IP, should the driver set traffic class or type-of-service fields? See the documentation for 'java.net.Socket.setTrafficClass()' for more information.

Default Value	0
Since Version	5.0.7

- `useCompression`

Use zlib compression when communicating with the server?

Default Value	false
Since Version	3.0.17

- `useUnbufferedInput`

  Don't use 'BufferedInputStream' for reading data from the server.

Default Value	true
Since Version	3.0.11

6.3.5 Security

- `paranoid`

  Take measures to prevent exposure sensitive information in error messages and clear data structures holding sensitive data when possible?

Default Value	false
Since Version	3.0.1

- `serverRSAPublicKeyFile`

  File path to the server RSA public key file for 'sha256_password' authentication. If not specified, the public key will be retrieved from the server.

Since Version	5.1.31

- `allowPublicKeyRetrieval`

  Allows special handshake round-trip to get an RSA public key directly from server.

Default Value	false
Since Version	5.1.31

- `sslMode`

  By default, network connections are SSL encrypted; this property permits secure connections to be turned off, or a different levels of security to be chosen. The following values are allowed: "DISABLED" - Establish unencrypted connections; "PREFERRED" - Establish encrypted connections if the server enabled them, otherwise fall back to unencrypted connections; "REQUIRED" - Establish secure connections if the server enabled them, fail otherwise; "VERIFY_CA" - Like "REQUIRED" but additionally verify the server TLS certificate against the configured Certificate Authority (CA) certificates; "VERIFY_IDENTITY" - Like "VERIFY_CA", but additionally verify that the server certificate matches the host to which the connection is attempted.

  This property replaced the deprecated legacy properties 'useSSL', 'requireSSL', and 'verifyServerCertificate', which are still accepted but translated into a value for 'sslMode' if 'sslMode' is not explicitly set: "useSSL=false" is translated to "sslMode=DISABLED"; {"useSSL=true", "requireSSL=false", "verifyServerCertificate=false"} is translated to "sslMode=PREFERRED"; {"useSSL=true", "requireSSL=true", "verifyServerCertificate=false"} is translated to "sslMode=REQUIRED"; {"useSSL=true", "verifyServerCertificate=true"} is translated to

"sslMode=VERIFY_CA". There is no equivalent legacy settings for "sslMode=VERIFY_IDENTITY". Note that, for all server versions, the default setting of 'sslMode' is "PREFERRED", and it is equivalent to the legacy settings of "useSSL=true", "requireSSL=false", and "verifyServerCertificate=false", which are different from their default settings for Connector/J 8.0.12 and earlier in some situations. Applications that continue to use the legacy properties and rely on their old default settings should be reviewed.

The legacy properties are ignored if 'sslMode' is set explicitly. If none of 'sslMode' or 'useSSL' is set explicitly, the default setting of "sslMode=PREFERRED" applies.

Default Value	PREFERRED
Since Version	8.0.13

- `trustCertificateKeyStoreUrl`

  URL for the trusted root certificates key store.

  If not specified, the property 'fallbackToSystemTrustStore' determines if system-wide trust store is used.

Since Version	5.1.0

- `trustCertificateKeyStoreType`

  Key store type for trusted root certificates.

  Null or empty means use the default, which is "JKS". Standard key store types supported by the JVM are "JKS" and "PKCS12", your environment may have more available depending on what security providers are installed and available to the JVM.

Default Value	JKS
Since Version	5.1.0

- `trustCertificateKeyStorePassword`

  Password for the trusted root certificates key store.

Since Version	5.1.0

- `fallbackToSystemTrustStore`

  Whether the absence of setting a value for 'trustCertificateKeyStoreUrl' falls back to using the system-wide default trust store or one defined through the system properties 'javax.net.ssl.trustStore*'.

Default Value	true
Since Version	8.0.22

- `clientCertificateKeyStoreUrl`

  URL for the client certificate KeyStore.

  If not specified, the property 'fallbackToSystemKeyStore' determines if system-wide key store is used.

Since Version	5.1.0

- `clientCertificateKeyStoreType`

  Key store type for client certificates.

  Null or empty means use the default, which is "JKS". Standard key store types supported by the JVM are "JKS" and "PKCS12", your environment may have more available depending on what security providers are installed and available to the JVM.

Default Value	JKS
Since Version	5.1.0

- `clientCertificateKeyStorePassword`

  Password for the client certificates key store.

Since Version	5.1.0

- `fallbackToSystemKeyStore`

  Whether the absence of setting a value for 'clientCertificateKeyStoreUrl' falls back to using the system-wide key store defined through the system properties 'javax.net.ssl.keyStore*'.

Default Value	true
Since Version	8.0.22

- `tlsCiphersuites`

  When establishing secure connections, overrides the cipher suites enabled for use on the underlying SSL sockets. This may be required when using external JSSE providers or to specify cipher suites compatible with both MySQL server and used JVM. Prior to version 8.0.28, this property was named 'enabledSSLCipherSuites', which remains as an alias.

Since Version	5.1.35

- `tlsVersions`

  List of TLS protocols to allow when establishing secure connections. Overrides the TLS protocols enabled in the underlying SSL sockets. This can be used to restrict connections to specific TLS versions and, by doing that, avoid TLS negotiation fallback. Allowed and default values are "TLSv1.2" and "TLSv1.3". Prior to version 8.0.28, this property was named 'enabledTLSProtocols', which remains as an alias.

Since Version	8.0.8

- `fipsCompliantJsse`

  Enables Connector/J to be compatible to JSSE operating in FIPS mode. Should be set to "true" if the JSSE is configured to operate in FIPS mode and Connector/J receives the error "FIPS mode: only SunJSSE TrustManagers may be used" when creating secure connections. If set to "true" then, when establishing secure connections, the driver operates as if the 'sslMode' was set to "VERIFY_CA" or "VERIFY_IDENTITY", i.e., all secure connections require at least server certificate validation, for which a trust store must be configured or fall back to the system-wide trust store must be enabled.

Default Value	false

Since Version	8.1.0

- `KeyManagerFactoryProvider`

  The name of the a Java Security Provider that provides a 'javax.net.ssl.KeyManagerFactory' implementation. If none is specified then the default one is used.

Since Version	8.1.0

- `trustManagerFactoryProvider`

  The name of the a Java Security Provider that provides a 'javax.net.ssl.TrustManagerFactory' implementation. If none is specified then the default one is used.

Since Version	8.1.0

- `keyStoreProvider`

  The name of the a Java Security Provider that provides a 'java.security.KeyStore' implementation that supports the key stores types specified with 'clientCertificateKeyStoreType' and 'trustCertificateKeyStoreType'. If none is specified then the default one is used.

Since Version	8.1.0

- `sslContextProvider`

  The name of the a Java Security Provider that provides a 'javax.net.ssl.SSLContext' implementation. If none is specified then the default one is used.

Since Version	8.1.0

- `allowLoadLocalInfile`

  Should the driver allow use of "LOAD DATA LOCAL INFILE ..."?

  Setting to "true" overrides whatever path is set in 'allowLoadLocalInfileInPath', allowing uploading files from any location.

Default Value	false
Since Version	3.0.3

- `allowLoadLocalInfileInPath`

  Enables "LOAD DATA LOCAL INFILE ..." statements, but only allows loading files from the specified path. Files within sub-directories are also allowed, but relative paths or symlinks that fall outside this path are forbidden.

Since Version	8.0.22

- `allowMultiQueries`

  Allow the use of ";" to delimit multiple queries during one statement. This option does not affect the 'addBatch()' and 'executeBatch()' methods, which rely on 'rewriteBatchStatements' instead.

Default Value	false

Since Version	3.1.1

- `allowUrlInLocalInfile`

  Should the driver allow URLs in "LOAD DATA LOCAL INFILE ..." statements?

Default Value	false
Since Version	3.1.4

- `requireSSL`

  DEPRECATED: See 'sslMode' property description for details.

  For 8.0.12 and earlier: Require server support of SSL connection if "useSSL=true".

Default Value	false
Since Version	3.1.0

- `useSSL`

  DEPRECATED: See 'sslMode' property description for details.

  For 8.0.12 and earlier: Use SSL when communicating with the server, default is "true" when connecting to MySQL 5.5.45+, 5.6.26+ or 5.7.6+, otherwise default is "false".

  For 8.0.13 and later: Default is "true".

Default Value	true
Since Version	3.0.2

- `verifyServerCertificate`

  DEPRECATED: See 'sslMode' property description for details.

  For 8.0.12 and earlier: If 'useSSL' is set to "true", should the driver verify the server's certificate? When using this feature, the key store parameters should be specified by the 'clientCertificateKeyStore*' properties, rather than system properties. Default is "false" when connecting to MySQL 5.5.45+, 5.6.26+ or 5.7.6+ and 'useSSL' was not explicitly set to "true". Otherwise default is "true".

  For 8.0.13 and later: Default is "false".

Default Value	false
Since Version	5.1.6

6.3.6 Statements

- `cacheDefaultTimeZone`

  Caches client's default time zone. This results in better performance when dealing with time zone conversions in Date and Time data types, however it won't be aware of time zone changes if they happen at runtime.

Default Value	true

Since Version	8.0.20

- `continueBatchOnError`

  Should the driver continue processing batch commands if one statement fails. The JDBC spec allows either way.

Default Value	true
Since Version	3.0.3

- `dontTrackOpenResources`

  The JDBC specification requires the driver to automatically track and close resources, however if your application doesn't do a good job of explicitly calling 'close()' on statements or result sets this can cause memory leakage. Setting this property to "true" relaxes this constraint, and can be more memory efficient for some applications. Also the automatic closing of the statement and current result set in 'Statement.closeOnCompletion()' and 'Statement.getMoreResults([Statement.CLOSE_CURRENT_RESULT | Statement.CLOSE_ALL_RESULTS])', respectively, ceases to happen. This property automatically sets "holdResultsOpenOverStatementClose=true".

Default Value	false
Since Version	3.1.7

- `queryInterceptors`

  A comma-delimited list of classes that implement 'com.mysql.cj.interceptors.QueryInterceptor' that intercept query executions and are able influence the results. Query iterceptors are chainable: the results returned by the current interceptor will be passed on to the next in the chain, from left-to-right in the order specified in this property.

Since Version	8.0.7

- `queryTimeoutKillsConnection`

  If the timeout given in 'Statement.setQueryTimeout()' expires, should the driver forcibly abort the connection instead of attempting to abort the query?

Default Value	false
Since Version	5.1.9

6.3.7 Prepared Statements

- `allowNanAndInf`

  Should the driver allow NaN or +/- INF values in 'PreparedStatement.setDouble()'?

Default Value	false
Since Version	3.1.5

- `autoClosePStmtStreams`

  Should the driver automatically call the method 'close()' on streams/readers passed as arguments via 'set*()' methods?

Default Value	false
Since Version	3.1.12

- `compensateOnDuplicateKeyUpdateCounts`

  Should the driver compensate for the update counts of "INSERT ... ON DUPLICATE KEY UPDATE" statements (2 = 1, 0 = 1) when using prepared statements?

Default Value	false
Since Version	5.1.7

- `emulateUnsupportedPstmts`

  Should the driver detect prepared statements that are not supported by the server, and replace them with client-side emulated versions?

Default Value	true
Since Version	3.1.7

- `generateSimpleParameterMetadata`

  Should the driver generate simplified parameter metadata for prepared statements when no metadata is available either because the server couldn't support preparing the statement, or server-side prepared statements are disabled?

Default Value	false
Since Version	5.0.5

- `processEscapeCodesForPrepStmts`

  Should the driver process escape codes in queries that are prepared? Default escape processing behavior in non-prepared statements must be defined with the property 'enableEscapeProcessing'.

Default Value	true
Since Version	3.1.12

- `useServerPrepStmts`

  Use server-side prepared statements if the server supports them? The server may limit the number of prepared statements with 'max_prepared_stmt_count' or disable them altogether. In case of not being possible to prepare new server-side prepared statements, it depends on the value of 'emulateUnsupportedPstmts' to whether return an error or fall back to client-side emulated prepared statements.

Default Value	false
Since Version	3.1.0

- `useStreamLengthsInPrepStmts`

  Honor stream length parameter in 'PreparedStatement/ResultSet.set*Stream()' method calls?

Default Value	true

Since Version	3.0.2

6.3.8 Result Sets

- `clobberStreamingResults`

  This will cause a streaming result set to be automatically closed, and any outstanding data still streaming from the server to be discarded if another query is executed before all the data has been read from the server.

Default Value	false
Since Version	3.0.9

- `emptyStringsConvertToZero`

  Should the driver allow conversions from empty string fields to numeric values of "0"?

Default Value	true
Since Version	3.1.8

- `holdResultsOpenOverStatementClose`

  Should the driver close result sets on 'Statement.close()' as required by the JDBC specification?

Default Value	false
Since Version	3.1.7

- `jdbcCompliantTruncation`

  Should the driver throw 'java.sql.DataTruncation' exceptions when data is truncated as is required by the JDBC specification? This property has no effect if the server sql-mode includes 'STRICT_TRANS_TABLES'.

Default Value	true
Since Version	3.1.2

- `maxRows`

  The maximum number of rows to return. The default "0" means return all rows.

Default Value	-1
Since Version	all versions

- `netTimeoutForStreamingResults`

  What value should the driver automatically set the server setting 'net_write_timeout' to when the streaming result sets feature is in use? Value has unit of seconds, the value "0" means the driver will not try and adjust this value.

Default Value	600
Since Version	5.1.0

- `padCharsWithSpace`

If a result set column has the CHAR type and the value does not fill the amount of characters specified in the DDL for the column, should the driver pad the remaining characters with space (for ANSI compliance)?

Default Value	false
Since Version	5.0.6

- `populateInsertRowWithDefaultValues`

  When using result sets that are 'CONCUR_UPDATABLE', should the driver pre-populate the insert row with default values from the DDL for the table used in the query so those values are immediately available for 'ResultSet' accessors? This functionality requires a call to the database for metadata each time a result set of this type is created. If disabled, the default values will be populated by the an internal call to 'refreshRow()' which pulls back default values and/or values changed by triggers.

Default Value	false
Since Version	5.0.5

- `scrollTolerantForwardOnly`

  Should the driver contradict the JDBC API and tolerate and support backward and absolute cursor movement on result sets of type 'ResultSet.TYPE_FORWARD_ONLY'?

  Regardless of this setting, cursor-based and row streaming result sets cannot be navigated in the prohibited directions.

Default Value	false
Since Version	8.0.24

- `strictUpdates`

  Should the driver do strict checking, i.e. all primary keys selected, of updatable result sets?

Default Value	true
Since Version	3.0.4

- `tinyInt1isBit`

  Since the MySQL server silently converts BIT to TINYINT(1) when creating tables, should the driver treat the datatype TINYINT(1) as the BIT type?

Default Value	true
Since Version	3.0.16

- `transformedBitIsBoolean`

  If the driver converts TINYINT(1) to a different type, should it use BOOLEAN instead of BIT?

Default Value	false
Since Version	3.1.9

6.3.9 Metadata

- `getProceduresReturnsFunctions`

  Pre-JDBC4 'DatabaseMetaData' API has only the 'getProcedures()' and 'getProcedureColumns()' methods, so they return metadata info for both stored procedures and functions. JDBC4 was extended with the 'getFunctions()' and 'getFunctionColumns()' methods and the expected behaviours of previous methods are not well defined. For JDBC4 and higher, default "true" value of the option means that calls of 'DatabaseMetaData.getProcedures()' and 'DatabaseMetaData.getProcedureColumns()' return metadata for both procedures and functions as before, keeping backward compatibility. Setting this property to "false" decouples Connector/J from its pre-JDBC4 behaviours for 'DatabaseMetaData.getProcedures()' and 'DatabaseMetaData.getProcedureColumns()', forcing them to return metadata for procedures only.

Default Value	true
Since Version	5.1.26

- `noAccessToProcedureBodies`

  When determining procedure parameter types for 'CallableStatement', and the connected user can't access procedure bodies through "SHOW CREATE PROCEDURE" or SELECT on mysql.proc should the driver instead create basic metadata, with all parameters reported as INOUT VARCHARs, instead of throwing an exception?

Default Value	false
Since Version	5.0.3

- `nullDatabaseMeansCurrent`

  In 'DatabaseMetaData' methods that take a 'catalog' or 'schema' parameter, does the value "null" mean to use the current database? See also the property 'databaseTerm'.

Default Value	false
Since Version	3.1.8

- `useHostsInPrivileges`

  Add '@hostname' to users in 'DatabaseMetaData.getColumn/TablePrivileges()'.

Default Value	true
Since Version	3.0.2

- `useInformationSchema`

  Should the driver use the INFORMATION_SCHEMA to derive information used by 'DatabaseMetaData'? Default is "true" when connecting to MySQL 8.0.3+, otherwise default is "false".

Default Value	false
Since Version	5.0.0

6.3.10 BLOB/CLOB processing

- `blobSendChunkSize`

Chunk size to use when sending BLOB/CLOBs via server-prepared statements. Note that this value cannot exceed the value of 'maxAllowedPacket' and, if that is the case, then this value will be corrected automatically.

Default Value	1048576
Since Version	3.1.9

- `blobsAreStrings`

  Should the driver always treat BLOBs as Strings - specifically to work around dubious metadata returned by the server for GROUP BY clauses?

Default Value	false
Since Version	5.0.8

- `clobCharacterEncoding`

  The character encoding to use for sending and retrieving TEXT, MEDIUMTEXT and LONGTEXT values instead of the configured connection 'characterEncoding'.

Since Version	5.0.0

- `emulateLocators`

  Should the driver emulate 'java.sql.Blob' with locators? With this feature enabled, the driver will delay loading the actual Blob data until the one of the retrieval methods ('getInputStream()', 'getBytes()', and so forth) on the blob data stream has been accessed. For this to work, you must use a column alias with the value of the column to the actual name of the Blob. The feature also has the following restrictions: The SELECT that created the result set must reference only one table, the table must have a primary key; the SELECT must alias the original blob column name, specified as a string, to an alternate name; the SELECT must cover all columns that make up the primary key.

Default Value	false
Since Version	3.1.0

- `functionsNeverReturnBlobs`

  Should the driver always treat data from functions returning BLOBs as Strings - specifically to work around dubious metadata returned by the server for "GROUP BY" clauses?

Default Value	false
Since Version	5.0.8

- `locatorFetchBufferSize`

  If 'emulateLocators' is configured to "true", what size buffer should be used when fetching BLOB data for 'getBinaryInputStream()'?

Default Value	1048576
Since Version	3.2.1

6.3.11 Datetime types processing

- `connectionTimeZone`

  Configures the connection time zone which is used by Connector/J if conversion between the JVM default and a target time zone is needed when preserving instant temporal values.

  Accepts a geographic time zone name or a time zone offset from Greenwich/UTC, using a syntax 'java.time.ZoneId' is able to parse, or one of the two logical values "LOCAL" and "SERVER". Default is "LOCAL". If set to an explicit time zone then it must be one that either the JVM or both the JVM and MySQL support. If set to "LOCAL" then the driver assumes that the connection time zone is the same as the JVM default time zone. If set to "SERVER" then the driver attempts to detect the session time zone from the values configured on the MySQL server session variables 'time_zone' or 'system_time_zone'. The time zone detection and subsequent mapping to a Java time zone may fail due to several reasons, mostly because of time zone abbreviations being used, in which case an explicit time zone must be set or a different time zone must be configured on the server.

  This option itself does not set MySQL server session variable 'time_zone' to the given value. To do that the 'forceConnectionTimeZoneToSession' connection option must be set to "true".

  Please note that setting a value to 'connectionTimeZone' in conjunction with "forceConnectionTimeZoneToSession=false" and "preserveInstants=false" has no effect since, in this case, neither this option is used to change the session time zone nor used for time zone conversions of time-based data.

  Former connection option 'serverTimezone' is still valid as an alias of this one but may be deprecated in the future.

  See also 'forceConnectionTimeZoneToSession' and 'preserveInstants' for more details.

Since Version	3.0.2

- `forceConnectionTimeZoneToSession`

  If enabled, sets the time zone value determined by 'connectionTimeZone' connection property to the current server session 'time_zone' variable. If the time zone value is given as a geographical time zone, then Connector/J sets this value as-is in the server session, in which case the time zone system tables must be populated beforehand (consult the MySQL Server documentation for further details); but, if the value is given as an offset from Greenwich/UTC in any of the supported syntaxes, then the server session time zone is set as a numeric offset from UTC.

  With that no intermediate conversion between JVM default time zone and connection time zone is needed to store correct milliseconds value of instant Java objects such as 'java.sql.Timestamp' or 'java.time.OffsetDateTime' when stored in TIMESTAMP columns.

  Note that it also affects the result of MySQL functions such as 'NOW()', 'CURTIME()' or 'CURDATE()'.

  This option has no effect if used in conjunction with "connectionTimeZone=SERVER" since, in this case, the session is already set with the required time zone.

  See also 'connectionTimeZone' and 'preserveInstants' for more details.

Default Value	false
Since Version	8.0.23

- `noDatetimeStringSync`

  Don't ensure that 'ResultSet.getTimestamp().toString().equals(ResultSet.getString())'.

Default Value	false
Since Version	3.1.7

- `preserveInstants`

  If enabled, Connector/J does its best to preserve the instant point on the time-line for Java instant-based objects such as 'java.sql.Timestamp' or 'java.time.OffsetDateTime' instead of their original visual form. Otherwise, the driver always uses the JVM default time zone for rendering the values it sends to the server and for constructing the Java objects from the fetched data.

  MySQL uses implied time zone conversion for TIMESTAMP values: they are converted from the session time zone to UTC for storage, and back from UTC to the session time zone for retrieval. So, to store the correct correct UTC value internally, the driver converts the value from the original time zone to the session time zone before sending to the server. On retrieval, Connector/J converts the received value from the session time zone to the JVM default one.

  When storing, the conversion is performed only if the target 'SQLType', either the explicit one or the default one, is TIMESTAMP. When retrieving, the conversion is performed only if the source column has the TIMESTAMP, DATETIME or character type and the target class is an instant-based one, like 'java.sql.Timestamp' or 'java.time.OffsetDateTime'.

  Note that this option has no effect if used in conjunction with "connectionTimeZone=LOCAL" since, in this case, the source and target time zones are the same. Though, in this case, it's still possible to store a correct instant value if set together with "forceConnectionTimeZoneToSession=true".

  See also 'connectionTimeZone' and 'forceConnectionTimeZoneToSession' for more details.

Default Value	true
Since Version	8.0.23

- `sendFractionalSeconds`

  If set to "false", the fractional seconds will always be truncated before sending any data to the server. This option applies only to prepared statements, callable statements or updatable result sets.

Default Value	true
Since Version	5.1.37

- `sendFractionalSecondsForTime`

  If set to "false", the fractional seconds of 'java.sql.Time' will be ignored as required by JDBC specification. If set to "true", its value is rendered with fractional seconds allowing to store milliseconds into MySQL TIME column. This option applies only to prepared statements, callable statements or updatable result sets. It has no effect if "sendFractionalSeconds=false".

Default Value	true
Since Version	8.0.23

- `treatMysqlDatetimeAsTimestamp`

  Should the driver treat the MySQL DATETIME type as TIMESTAMP in 'ResultSet.getObject()'? Enabling this option changes the default MySQL data type to Java type mapping for DATETIME from 'java.time.LocalDateTime' to 'java.sql.Timestamp'. Given the nature of the DATETIME type and its inability to represent instant values, it is not advisable to enable this option unless the driver is used with a framework or API that expects exclusively objects following the default MySQL data types to Java types mapping, which is the case of, for example, 'javax.sql.rowset.CachedRowSet'.

Default Value	false
Since Version	8.2.0

- `treatUtilDateAsTimestamp`

  Should the driver treat 'java.util.Date' as a TIMESTAMP in 'PreparedStatement.setObject()'?

Default Value	true
Since Version	5.0.5

- `yearIsDateType`

  Should the JDBC driver treat the MySQL type YEAR as a 'java.sql.Date', or as a SHORT?

Default Value	true
Since Version	3.1.9

- `zeroDateTimeBehavior`

  What should happen when the driver encounters DATETIME values that are composed entirely of zeros - used by MySQL to represent invalid dates? Valid values are "EXCEPTION", "ROUND" and "CONVERT_TO_NULL".

Default Value	EXCEPTION
Since Version	3.1.4

6.3.12 High Availability and Clustering

- `autoReconnect`

  Should the driver try to re-establish stale and/or dead connections? If enabled the driver will throw an exception for queries issued on a stale or dead connection, which belong to the current transaction, but will attempt reconnect before the next query issued on the connection in a new transaction. The use of this feature is not recommended, because it has side effects related to session state and data consistency when applications don't handle SQLExceptions properly, and is only designed to be used when you are unable to configure your application to handle SQLExceptions resulting from dead and stale connections properly. Alternatively, as a last option, investigate setting the MySQL server variable 'wait_timeout' to a high value, rather than the default of 8 hours.

Default Value	false
Since Version	1.1

- `autoReconnectForPools`

Use a reconnection strategy appropriate for connection pools?

Default Value	false
Since Version	3.1.3

- `failOverReadOnly`

  When failing over in 'autoReconnect' mode, should the connection be set to 'read-only'?

Default Value	true
Since Version	3.0.12

- `maxReconnects`

  Maximum number of reconnects to attempt if 'autoReconnect' is "true".

Default Value	3
Since Version	1.1

- `reconnectAtTxEnd`

  If 'autoReconnect' is set to "true", should the driver attempt reconnections at the end of every transaction?

Default Value	false
Since Version	3.0.10

- `retriesAllDown`

  When using load balancing or failover, the number of times the driver should cycle through available hosts, attempting to connect. Between cycles, the driver will pause for 250 ms if no servers are available.

Default Value	120
Since Version	5.1.6

- `initialTimeout`

  If 'autoReconnect' is enabled, the initial time to wait between re-connect attempts (in seconds, defaults to "2").

Default Value	2
Since Version	1.1

- `queriesBeforeRetrySource`

  When using multi-host failover, the number of queries to issue before falling back to the primary host when failed over. Whichever condition is met first, 'queriesBeforeRetrySource' or 'secondsBeforeRetrySource' will cause an attempt to be made to reconnect to the primary host. Setting both properties to "0" disables the automatic fall back to the primary host at transaction boundaries.

Default Value	50
Since Version	3.0.2

- `secondsBeforeRetrySource`

  How long, in seconds, should the driver wait when failed over, before attempting to reconnect to the primary host? Whichever condition is met first, 'queriesBeforeRetrySource' or 'secondsBeforeRetrySource' will cause an attempt to be made to reconnect to the source host. Setting both properties to "0" disables the automatic fall back to the primary host at transaction boundaries.

Default Value	30
Since Version	3.0.2

- `allowReplicaDownConnections`

  By default, a replication-aware connection will fail to connect when configured replica hosts are all unavailable at initial connection. Setting this property to "true" allows to establish the initial connection. It won't prevent failures when switching to replicas i.e. by setting the replication connection to read-only state. The property 'readFromSourceWhenNoReplicas' should be used for this purpose.

Default Value	false
Since Version	6.0.2

- `allowSourceDownConnections`

  By default, a replication-aware connection will fail to connect when configured source hosts are all unavailable at initial connection. Setting this property to "true" allows to establish the initial connection, by failing over to the replica servers, in read-only state. It won't prevent subsequent failures when switching back to the source hosts i.e. by setting the replication connection to read/write state.

Default Value	false
Since Version	5.1.27

- `ha.enableJMX`

  Enables JMX-based management of load-balanced connection groups, including live addition/removal of hosts from load-balancing pool. Enables JMX-based management of replication connection groups, including live replica promotion, addition of new replicas and removal of source or replica hosts from load-balanced source and replica connection pools.

Default Value	false
Since Version	5.1.27

- `loadBalanceHostRemovalGracePeriod`

  Sets the grace period to wait for a host being removed from a load-balanced connection, to be released when it is currently the active host.

Default Value	15000
Since Version	6.0.3

- `readFromSourceWhenNoReplicas`

  Replication-aware connections distribute load by using the source hosts when in read/write state and by using the replica hosts when in read-only state. If, when setting the connection to read-only state, none of the replica hosts are available, an 'SQLException' is thrown back. Setting this property to "true" allows

to fail over to the source hosts, while setting the connection state to read-only, when no replica hosts are available at switch instant.

Default Value	false
Since Version	6.0.2

- `selfDestructOnPingMaxOperations`

  If set to a non-zero value, the driver will report close the connection and report failure when 'com.mysql.cj.jdbc.JdbcConnection.ping()' or 'java.sql.Connection.isValid(int)' is called if the connection's count of commands sent to the server exceeds this value.

Default Value	0
Since Version	5.1.6

- `selfDestructOnPingSecondsLifetime`

  If set to a non-zero value, the driver will close the connection and report failure when 'com.mysql.cj.jdbc.JdbcConnection.ping()' or 'java.sql.Connection.isValid(int)' is called if the connection's lifetime exceeds this value, specified in milliseconds.

Default Value	0
Since Version	5.1.6

- `ha.loadBalanceStrategy`

  If using a load-balanced connection to connect to SQL servers in a MySQL Cluster configuration (by using the URL prefix "jdbc:mysql:loadbalance://"), which load balancing algorithm should the driver use: (1) "random" - the driver will pick a random host for each request. This tends to work better than round-robin, as the randomness will somewhat account for spreading loads where requests vary in response time, while round-robin can sometimes lead to overloaded nodes if there are variations in response times across the workload. (2) "bestResponseTime" - the driver will route the request to the host that had the best response time for the previous transaction. (3) "serverAffinity" - the driver initially attempts to enforce server affinity while still respecting and benefiting from the fault tolerance aspects of the load-balancing implementation. The server affinity ordered list is provided using the property 'serverAffinityOrder'. If none of the servers listed in the affinity list is responsive, the driver then refers to the "random" strategy to proceed with choosing the next server.

Default Value	random
Since Version	5.0.6

- `loadBalanceAutoCommitStatementRegex`

  When load-balancing is enabled for auto-commit statements (via 'loadBalanceAutoCommitStatementThreshold'), the statement counter will only increment when the SQL matches the regular expression. By default, every statement issued matches.

Since Version	5.1.15

- `loadBalanceAutoCommitStatementThreshold`

  When auto-commit is enabled, the number of statements which should be executed before triggering load-balancing to rebalance. Default value of "0" causes load-balanced connections to only rebalance

when exceptions are encountered, or auto-commit is disabled and transactions are explicitly committed or rolled back.

Default Value	0
Since Version	5.1.15

- `loadBalanceBlocklistTimeout`

  Time in milliseconds between checks of servers which are unavailable, by controlling how long a server lives in the global blocklist.

Default Value	0
Since Version	5.1.0

- `loadBalanceConnectionGroup`

  Logical group of load-balanced connections within a class loader, used to manage different groups independently. If not specified, live management of load-balanced connections is disabled.

Since Version	5.1.13

- `loadBalanceExceptionChecker`

  Fully-qualified class name of custom exception checker. The class must implement 'com.mysql.cj.jdbc.ha.LoadBalanceExceptionChecker' interface, and is used to inspect 'SQLException' exceptions and determine whether they should trigger fail-over to another host in a load-balanced deployment.

Default Value	com.mysql.cj.jdbc.ha.StandardLoadBalanceExceptionChecker
Since Version	5.1.13

- `loadBalancePingTimeout`

  Time in milliseconds to wait for ping responses from each of load-balanced physical connections when using a load-balanced connection.

Default Value	0
Since Version	5.1.13

- `loadBalanceSQLExceptionSubclassFailover`

  Comma-delimited list of classes/interfaces used by default load-balanced exception checker to determine whether a given 'SQLException' should trigger a failover. The comparison is done using 'Class.isInstance(SQLException)' using the 'SQLException' thrown.

Since Version	5.1.13

- `loadBalanceSQLStateFailover`

  Comma-delimited list of 'SQLState' codes used by the default load-balanced exception checker to determine whether a given 'SQLException' should trigger a failover. The 'SQLState' of a given

'SQLException' is evaluated to determine whether it begins with any of the values specified in the comma-delimited list.

Since Version	5.1.13

- `loadBalanceValidateConnectionOnSwapServer`

  Should the load-balanced connection explicitly check whether the connection is live when swapping to a new physical connection at commit/rollback?

Default Value	false
Since Version	5.1.13

- `pinGlobalTxToPhysicalConnection`

  When using XA connections, should the driver ensure that operations on a given XID are always routed to the same physical connection? This allows the 'XAConnection' to support "XA START ... JOIN" after "XA END" has been called.

Default Value	false
Since Version	5.0.1

- `replicationConnectionGroup`

  Logical group of replication connections within a class loader, used to manage different groups independently. If not specified, live management of replication connections is disabled.

Since Version	8.0.7

- `resourceId`

  A globally unique name that identifies the resource that this data source or connection is connected to, used for 'XAResource.isSameRM()' when the driver can't determine this value based on hostnames used in the URL.

Since Version	5.0.1

- `serverAffinityOrder`

  A comma separated list containing the host/port pairs that are to be used in load-balancing "serverAffinity" strategy. Only the sub-set of the hosts enumerated in the main hosts section in this URL will be used and they must be identical in case and type, i.e., can't use an IP address in one place and the corresponding host name in the other.

Since Version	8.0.8

6.3.13 Performance Extensions

- `callableStmtCacheSize`

  If 'cacheCallableStmts' is enabled, how many callable statements should be cached?

Default Value	100
Since Version	3.1.2

- `metadataCacheSize`

  The number of queries to cache 'ResultSetMetadata' for if 'cacheResultSetMetaData' is set to "true".

Default Value	50
Since Version	3.1.1

- `useLocalSessionState`

  Should the driver refer to the internal values of auto-commit and transaction isolation that are set by 'Connection.setAutoCommit()' and 'Connection.setTransactionIsolation()' and transaction state as maintained by the protocol, rather than querying the database or blindly sending commands to the database for 'commit()' or 'rollback()' method calls?

Default Value	false
Since Version	3.1.7

- `useLocalTransactionState`

  Should the driver use the in-transaction state provided by the MySQL protocol to determine if a 'commit()' or 'rollback()' should actually be sent to the database?

Default Value	false
Since Version	5.1.7

- `prepStmtCacheSize`

  If prepared statement caching is enabled, how many prepared statements should be cached?

Default Value	25
Since Version	3.0.10

- `prepStmtCacheSqlLimit`

  If prepared statement caching is enabled, what's the largest SQL the driver will cache the parsing for?

Default Value	256
Since Version	3.0.10

- `queryInfoCacheFactory`

  Name of a class implementing 'com.mysql.cj.CacheAdapterFactory', which will be used to create caches for the parsed representation of prepared statements. Prior to version 8.0.29, this property was named 'parseInfoCacheFactory', which remains as an alias.

Default Value	com.mysql.cj.PerConnectionLRUFactory
Since Version	5.1.1

- `serverConfigCacheFactory`

  Name of a class implementing 'com.mysql.cj.CacheAdapterFactory', which will be used to create caches for MySQL server configuration values.

Default Value	com.mysql.cj.util.PerVmServerConfigCacheFactory
Since Version	5.1.1

- `alwaysSendSetIsolation`

  Should the driver always communicate with the database when 'Connection.setTransactionIsolation()' is called? If set to "false", the driver will only communicate with the database when the requested transaction isolation is different than the whichever is newer, the last value that was set via 'Connection.setTransactionIsolation()', or the value that was read from the server when the connection was established. Note that "useLocalSessionState=true" will force the same behavior as "alwaysSendSetIsolation=false", regardless of how 'alwaysSendSetIsolation' is set.

Default Value	true
Since Version	3.1.7

- `maintainTimeStats`

  Should the driver maintain various internal timers to enable idle time calculations as well as more verbose error messages when the connection to the server fails? Setting this property to false removes at least two calls to 'System.getCurrentTimeMillis()' per query.

Default Value	true
Since Version	3.1.9

- `useCursorFetch`

  Should the driver use cursor-based fetching to retrieve rows? If set to "true" and 'defaultFetchSize' is set to a value higher than zero or 'setFetchSize()' with a value higher than zero is called on a statement, then the cursor-based result set will be used. Please note that 'useServerPrepStmts' is automatically set to "true" in this case because cursor functionality is available only for server-side prepared statements.

Default Value	false
Since Version	5.0.0

- `cacheCallableStmts`

  Should the driver cache the parsing stage of CallableStatements?

Default Value	false
Since Version	3.1.2

- `cachePrepStmts`

  Should the driver cache the parsing stage of PreparedStatements of client-side prepared statements, the "check" for suitability of server-side prepared and server-side prepared statements themselves?

Default Value	false
Since Version	3.0.10

- `cacheResultSetMetadata`

  Should the driver cache 'ResultSetMetaData' for statements and prepared statements?

Default Value	false
Since Version	3.1.1

- `cacheServerConfiguration`

  Should the driver cache the results of "SHOW VARIABLES" and "SHOW COLLATION" on a per-URL basis?

Default Value	false
Since Version	3.1.5

- `defaultFetchSize`

  The driver will call 'setFetchSize(n)' with this value on all newly-created statements.

Default Value	0
Since Version	3.1.9

- `dontCheckOnDuplicateKeyUpdateInSQL`

  Stops checking if every INSERT statement contains the "ON DUPLICATE KEY UPDATE" clause. As a side effect, obtaining the statement's generated keys information will return a list where normally it would not. Also be aware that, in this case, the list of generated keys returned may not be accurate. The effect of this property is canceled if set simultaneously with "rewriteBatchedStatements=true".

Default Value	false
Since Version	5.1.32

- `elideSetAutoCommits`

  Should the driver only issue 'set autocommit=n' queries when the server's state doesn't match the requested state by 'Connection.setAutoCommit(boolean)'?

Default Value	false
Since Version	3.1.3

- `enableEscapeProcessing`

  Sets the default escape processing behavior for Statement objects. The method 'Statement.setEscapeProcessing()' can be used to specify the escape processing behavior for an individual statement object. Default escape processing behavior in prepared statements must be defined with the property 'processEscapeCodesForPrepStmts'.

Default Value	true
Since Version	6.0.1

- `enableQueryTimeouts`

  When enabled, query timeouts set via 'Statement.setQueryTimeout()' use a shared 'java.util.Timer' instance for scheduling. Even if the timeout doesn't expire before the query is processed, there will be memory used by the 'TimerTask' for the given timeout which won't be reclaimed until the time the timeout would have expired if it hadn't been cancelled by the driver. High-load environments might want to consider disabling this functionality.

Default Value	true
Since Version	5.0.6

- `largeRowSizeThreshold`

  What size result set row should the JDBC driver consider large, and thus use a more memory-efficient way of representing the row internally?

Default Value	2048
Since Version	5.1.1

- `readOnlyPropagatesToServer`

  Should the driver issue appropriate statements to implicitly set the transaction access mode on server side when 'Connection.setReadOnly()' is called? Setting this property to "true" enables InnoDB read-only potential optimizations but also requires an extra roundtrip to set the right transaction state. Even if this property is set to "false", the driver will do its best effort to prevent the execution of database-state-changing queries.

Default Value	true
Since Version	5.1.35

- `rewriteBatchedStatements`

  Should the driver use multi-queries, regardless of the setting of 'allowMultiQueries', as well as rewriting of prepared statements for INSERT and REPLACE queries into multi-values clause statements when 'executeBatch()' is called?

  Notice that this might allow SQL injection when using plain statements and the provided input is not properly sanitized. Also notice that for prepared statements, if the stream length is not specified when using 'PreparedStatement.set*Stream()', the driver would not be able to determine the optimum number of parameters per batch and might return an error saying that the resultant packet is too large.

  'Statement.getGeneratedKeys()', for statements that are rewritten only works when the entire batch consists of INSERT or REPLACE statements.

  Be aware that when using "rewriteBatchedStatements=true" with "INSERT ... ON DUPLICATE KEY UPDATE" for rewritten statements, the server returns only one value for all affected (or found) rows in the batch, and it is not possible to map it correctly to the initial statements; in this case the driver returns "0" as the result for each batch statement if total count was zero, and 'Statement.SUCCESS_NO_INFO' if total count was above zero.

Default Value	false
Since Version	3.1.13

- `useReadAheadInput`

Use optimized non-blocking buffered input stream when reading from the server?

Default Value	true
Since Version	3.1.5

6.3.14 Debugging/Profiling

- `logger`

  The name of a class that implements 'com.mysql.cj.log.Log' that will be used to log messages to. (default is 'com.mysql.cj.log.StandardLogger', which logs to STDERR).

Default Value	com.mysql.cj.log.StandardLogger
Since Version	3.1.1

- `profilerEventHandler`

  Name of a class that implements the interface 'com.mysql.cj.log.ProfilerEventHandler' that will be used to handle profiling/tracing events.

Default Value	com.mysql.cj.log.LoggingProfilerEventHandler
Since Version	5.1.6

- `useNanosForElapsedTime`

  For profiling/debugging functionality that measures elapsed time, should the driver try to use nanoseconds resolution?

Default Value	false
Since Version	5.0.7

- `maxQuerySizeToLog`

  Controls the maximum length of the part of a query that will get logged when profiling or tracing.

Default Value	2048
Since Version	3.1.3

- `maxByteArrayAsHex`

  Maximum size for a byte array parameter in a prepared statement that is converted to a hexadecimal literal when interpolated by 'JdbcPreparedStatement.toString()'. Any byte arrays larger than this value are interpolated generically as "** BYTE ARRAY DATA **".

Default Value	1024
Since Version	8.0.31

- `profileSQL`

  Trace queries and their execution/fetch times to the configured 'profilerEventHandler'.

Default Value	false

Since Version	3.1.0

- `logSlowQueries`

  Should queries that take longer than 'slowQueryThresholdMillis' or detected by the 'autoSlowLog' monitoring be reported to the registered 'profilerEventHandler'?

Default Value	false
Since Version	3.1.2

- `slowQueryThresholdMillis`

  If 'logSlowQueries' is enabled, how long, in milliseconds, should a query take before it is logged as slow?

Default Value	2000
Since Version	3.1.2

- `slowQueryThresholdNanos`

  If 'logSlowQueries' is enabled, 'useNanosForElapsedTime' is set to "true", and this property is set to a non-zero value, the driver will use this threshold, in nanosecond units, to determine if a query was slow.

Default Value	0
Since Version	5.0.7

- `autoSlowLog`

  Instead of using 'slowQueryThreshold*' to determine if a query is slow enough to be logged, maintain statistics that allow the driver to determine queries that are outside the 99th percentile?

Default Value	true
Since Version	5.1.4

- `explainSlowQueries`

  If 'logSlowQueries' is enabled, should the driver automatically issue an 'EXPLAIN' on the server and send the results to the configured logger at a WARN level?

Default Value	false
Since Version	3.1.2

- `gatherPerfMetrics`

  Should the driver gather performance metrics, and report them via the configured logger every 'reportMetricsIntervalMillis' milliseconds?

Default Value	false
Since Version	3.1.2

- `reportMetricsIntervalMillis`

  If 'gatherPerfMetrics' is enabled, how often should they be logged (in milliseconds)?

Default Value	30000
Since Version	3.1.2

- `logXaCommands`

  Should the driver log XA commands sent by 'MysqlXaConnection' to the server, at the DEBUG level of logging?

Default Value	false
Since Version	5.0.5

- `traceProtocol`

  Should the network protocol be logged at the TRACE level?

Default Value	false
Since Version	3.1.2

- `enablePacketDebug`

  When enabled, a ring-buffer of 'packetDebugBufferSize' packets will be kept, and dumped when exceptions are thrown in key areas in the driver's code.

Default Value	false
Since Version	3.1.3

- `packetDebugBufferSize`

  The maximum number of packets to retain when 'enablePacketDebug' is "true".

Default Value	20
Since Version	3.1.3

- `useUsageAdvisor`

  Should the driver issue usage warnings advising proper and efficient usage of JDBC and MySQL Connector/J to the 'profilerEventHandler'?

Default Value	false
Since Version	3.1.1

- `resultSetSizeThreshold`

  If 'useUsageAdvisor' is "true", how many rows should a result set contain before the driver warns that it is suspiciously large?

Default Value	100
Since Version	5.0.5

- `autoGenerateTestcaseScript`

  Should the driver dump the SQL it is executing, including server-side prepared statements to STDERR?

Default Value	false
Since Version	3.1.9

- `openTelemetry`

  Should the driver generate OpenTelemetry traces and handle context propagation to the MySQL Server? This option accepts the values "REQUIRED", "PREFERRED", and "DISABLED". If set to "REQUIRED", an OpenTelemetry library must be available at run time, or connections to the MySQL Server will fail. Setting it to "DISABLED" turns off generating OpenTelemetry instrumentation by Connector/J. Setting it to "PREFERRED" enables generating OpenTelemetry instrumentation provided that an OpenTelemetry library is available at run time, and a warning is issued otherwise. Not setting a value for the property is equivalent to setting it as "PREFERRED", but no warning is issued when no OpenTelmetry library is available at run time. Connector/J relies entirely on the OpenTelemetry exporters configured in the calling application and does not provide any means of configuring its own exporters.

Default Value	PREFERRED
Since Version	8.4.0

6.3.15 Exceptions/Warnings

- `dumpQueriesOnException`

  Should the driver dump the contents of the query sent to the server in the message for SQLExceptions?

Default Value	false
Since Version	3.1.3

- `exceptionInterceptors`

  Comma-delimited list of classes that implement the interface 'com.mysql.cj.exceptions.ExceptionInterceptor'. These classes will be instantiated one per 'Connection' instance, and all 'SQLException' exceptions thrown by the driver will be allowed to be intercepted by these interceptors, in a chained fashion, with the first class listed as the head of the chain.

Since Version	5.1.8

- `ignoreNonTxTables`

  Ignore non-transactional table warning for rollback?

Default Value	false
Since Version	3.0.9

- `includeInnodbStatusInDeadlockExceptions`

  Include the output of "SHOW ENGINE INNODB STATUS" in exception messages when deadlock exceptions are detected?

Default Value	false

Since Version	5.0.7

- `includeThreadDumpInDeadlockExceptions`

  Include current Java thread dump in exception messages when deadlock exceptions are detected?

Default Value	false
Since Version	5.1.15

- `includeThreadNamesAsStatementComment`

  Include the name of the current thread as a comment visible in "SHOW PROCESSLIST", or in Innodb deadlock dumps, useful in correlation with "includeInnodbStatusInDeadlockExceptions=true" and "includeThreadDumpInDeadlockExceptions=true".

Default Value	false
Since Version	5.1.15

- `useOnlyServerErrorMessages`

  Don't prepend standard 'SQLState' error messages to error messages returned by the server.

Default Value	true
Since Version	3.0.15

6.3.16 Tunes for integration with other products

- `overrideSupportsIntegrityEnhancementFacility`

  Should the driver return "true" for 'DatabaseMetaData.supportsIntegrityEnhancementFacility()' even if the database doesn't support it to workaround applications that require this method to return "true" to signal support of foreign keys, even though the SQL specification states that this facility contains much more than just foreign key support (one such application being OpenOffice)?

Default Value	false
Since Version	3.1.12

- `ultraDevHack`

  Create prepared statements for 'prepareCall()' when required, because UltraDev is broken and issues a 'prepareCall()' for all statements?

Default Value	false
Since Version	2.0.3

6.3.17 JDBC compliance

- `useColumnNamesInFindColumn`

  Prior to JDBC-4.0, the JDBC specification had a bug related to what could be given as a column name to result set methods like 'findColumn()', or getters that took a String property. JDBC-4.0 clarified "column name" to mean the label, as given in an "AS" clause and returned by

'ResultSetMetaData.getColumnLabel()', and if no "AS" clause is specified, the column name. Setting this property to "true" will result in a behavior that is congruent to JDBC-3.0 and earlier versions of the JDBC specification, but which could have unexpected results. This property is preferred over 'useOldAliasMetadataBehavior' unless in need of the specific behavior that it provides with respect to 'ResultSetMetadata'.

Default Value	false
Since Version	5.1.7

- `pedantic`

  Follow the JDBC specification to the letter.

Default Value	false
Since Version	3.0.0

- `useOldAliasMetadataBehavior`

  Should the driver use the legacy behavior for "AS" clauses on columns and tables, and only return aliases ,if any, for 'ResultSetMetaData.getColumnName()' or 'ResultSetMetaData.getTableName()' rather than the original column/table name?

Default Value	false
Since Version	5.0.4

6.3.18 X Protocol and X DevAPI

- `xdevapi.auth`

  Authentication mechanism to use with the X Protocol. Allowed values are "SHA256_MEMORY", "MYSQL41", "PLAIN", and "EXTERNAL". Value is case insensitive. If the property is not set, the mechanism is chosen depending on the connection type: "PLAIN" is used for TLS connections and "SHA256_MEMORY" or "MYSQL41" is used for unencrypted connections.

Default Value	PLAIN
Since Version	8.0.8

- `xdevapi.compression`

  X DevAPI-specific network traffic compression. This option accepts one of the three values: "PREFERRED", "REQUIRED", and "DISABLED". Setting this option to "PREFERRED" or "REQUIRED" enables compression algorithm negotiation between Connector and Server, and turns on compression of large X Protocol packets, as long as a consensus is reached between client and server regarding the compression algorithm to use. If a consensus cannot be reached, connection fails if the option is set to "REQUIRED" and continues without compression if the option is set to "PREFERRED". Setting this option as "DISABLED" skips the compression negotiation phase and forbids the interchange of compressed messages between client and server.

Default Value	PREFERRED
Since Version	8.0.20

- `xdevapi.compression-algorithms`

A comma-delimited list of compression algorithms, each one identified by its name and operating mode, (e.g. "lz4_message"; consult the description for the MySQL global variable 'mysqlx_compression_algorithms' for a list of supported and enabled algorithms), that defines the order and which algorithms will be attempted when negotiating connection compression with the server.

The compression algorithm 'deflate_stream' is supported natively. Additional compression algorithms require using third-party libraries and enabling them with the connection property 'xdevapi.compression-extensions'.

This option is meaningful only when network traffic compression is enabled using the connection property 'xdevapi.compression'.

As an alternative to the default algorithm names, that contain a reference to the compression operation mode, the aliases "zstd", "lz4", and "deflate" can be used instead of "zstd_stream", "lz4_message", and "deflate_stream".

Default Value	zstd_stream,lz4_message,deflate_stream
Since Version	8.0.22

- `xdevapi.compression-extensions`

A comma-delimited list of triplets, with their elements delimited by colon, that enables the support for additional compression algorithms. Each triplet must contain: first, an algorithm name and operating mode (e.g. "lz4_message"; consult the description for the MySQL global variable 'mysqlx_compression_algorithms' for a list of supported and enabled algorithms); second, a fully-qualified class name of a class implementing the interface 'java.io.InputStream' that will be used to inflate data compressed with the named algorithm; third, a fully-qualified class name of a class implementing the interface 'java.io.OutputStream' that will be used to deflate data using the named algorithm. Along with this setting, the library containing implementations of the designated classes must be available in the application's class path.

Any number of triplets defining compression algorithms and their inflater and deflater implementations can be provided but only the ones supported and enabled on the MySQL Server can be used.

The compression algorithm 'deflate_stream' is supported natively. Additional compression algorithms require using third-party libraries.

This option is meaningful only when network traffic compression is enabled using the connection property 'xdevapi.compression'.

As an alternative to the default algorithm names, that contain a reference to the compression operation mode, the aliases "zstd", "lz4", and "deflate" can be used instead of "zstd_stream", "lz4_message", and "deflate_stream".

Since Version	8.0.22

- `xdevapi.connect-timeout`

X DevAPI-specific timeout, in milliseconds, for socket connect, with "0" being no timeout. If 'xdevapi.connect-timeout' is not set explicitly and 'connectTimeout' is, 'xdevapi.connect-timeout' takes up the value of 'connectTimeout'.

Default Value	10000

Since Version	8.0.13

- `xdevapi.connection-attributes`

  An X DevAPI-specific comma-delimited list of user-defined "key=value" pairs, in addition to standard X Protocol-defined "key=value" pairs, to be passed to MySQL Server for display as connection attributes in the 'PERFORMANCE_SCHEMA' tables 'session_account_connect_attrs' and 'session_connect_attrs'. Example usage: "xdevapi.connection-attributes=key1=value1,key2=value2" or "xdevapi.connection-attributes=[key1=value1,key2=value2]". This functionality is available for use with MySQL Server version 8.0.16 or later only. Earlier versions of X Protocol do not support connection attributes, causing this configuration option to be ignored. For situations where Session creation/initialization speed is critical, setting "xdevapi.connection-attributes=false" will cause connection attribute processing to be bypassed.

Since Version	8.0.16

- `xdevapi.dns-srv`

  X DevAPI-specific option for instructing the driver use the given host name to lookup for DNS SRV records and use the resulting list of hosts in a multi-host failover connection. Note that a single host name and no port must be provided when this option is enabled.

Default Value	false
Since Version	8.0.19

- `xdevapi.fallback-to-system-keystore`

  X DevAPI-specific switch to specify whether in the absence of a set value for 'xdevapi.ssl-keystore' (or 'clientCertificateKeyStoreUrl'), Connector/J falls back to using the system-wide key store defined through the system properties 'javax.net.ssl.keyStore*'. If not specified, the value of 'fallbackToSystemKeyStore' is used.

Default Value	true
Since Version	8.0.22

- `xdevapi.fallback-to-system-truststore`

  X DevAPI-specific switch to specify whether in the absence of a set value for 'xdevapi.ssl-truststore' (or 'trustCertificateKeyStoreUrl'), Connector/J falls back to using the system-wide default trust store or one defined through the system properties 'javax.net.ssl.trustStore*'. If not specified, the value of 'fallbackToSystemTrustStore' is used.

Default Value	true
Since Version	8.0.22

- `xdevapi.ssl-keystore`

  X DevAPI-specific URL for the client certificate key store. If not specified, use 'clientCertificateKeyStoreUrl' value.

Since Version	8.0.22

- `xdevapi.ssl-keystore-password`

  X DevAPI-specific password for the client certificate key store. If not specified, use 'clientCertificateKeyStorePassword' value.

Since Version	8.0.22

- `xdevapi.ssl-keystore-type`

  X DevAPI-specific type of the client certificate key store. If not specified, use 'clientCertificateKeyStoreType' value.

Default Value	JKS
Since Version	8.0.22

- `xdevapi.ssl-mode`

  X DevAPI-specific SSL mode setting. If not specified, use 'sslMode'. Because the "PREFERRED" mode is not applicable to X Protocol, if 'xdevapi.ssl-mode' is not set and 'sslMode' is set to "PREFERRED", 'xdevapi.ssl-mode' is set to "REQUIRED".

Default Value	REQUIRED
Since Version	8.0.7

- `xdevapi.ssl-truststore`

  X DevAPI-specific URL for the trusted CA certificates key store. If not specified, use 'trustCertificateKeyStoreUrl' value.

Since Version	6.0.6

- `xdevapi.ssl-truststore-password`

  X DevAPI-specific password for the trusted CA certificates key store. If not specified, use 'trustCertificateKeyStorePassword' value.

Since Version	6.0.6

- `xdevapi.ssl-truststore-type`

  X DevAPI-specific type of the trusted CA certificates key store. If not specified, use 'trustCertificateKeyStoreType' value.

Default Value	JKS
Since Version	6.0.6

- `xdevapi.tls-ciphersuites`

  X DevAPI-specific property overriding the cipher suites enabled for use on the underlying SSL sockets. If not specified, the value of 'enabledSSLCipherSuites' is used.

Since Version	8.0.19

- `xdevapi.tls-versions`

X DevAPI-specific property that takes a list of TLS protocols to allow when creating secure sessions. Overrides the TLS protocols enabled in the underlying SSL socket. If not specified, then the value of 'tlsVersions' is used instead. Allowed and default values are "TLSv1.2" and "TLSv1.3".

Since Version	8.0.19

6.4 JDBC API Implementation Notes

MySQL Connector/J, as a rigorous implementation of the JDBC API, passes all of the tests in the publicly available version of Oracle's JDBC compliance test suite. The JDBC specification is flexible on how certain functionality should be implemented. This section gives details on an interface-by-interface level about implementation decisions that might affect how you code applications with MySQL Connector/J.

- **BLOB**

  You can emulate BLOBs with locators by adding the property `emulateLocators=true` to your JDBC URL. Using this method, the driver will delay loading the actual BLOB data until you retrieve the other data and then use retrieval methods (`getInputStream()`, `getBytes()`, and so forth) on the BLOB data stream.

  You must use a column alias with the value of the column to the actual name of the BLOB, for example:

  ```
  SELECT id, 'data' as blob_data from blobtable
  ```

  You must also follow these rules:

  - The `SELECT` must reference only one table. The table must have a primary key.
  - The `SELECT` must alias the original BLOB column name, specified as a string, to an alternate name.
  - The `SELECT` must cover all columns that make up the primary key.

  The BLOB implementation does not allow in-place modification (they are copies, as reported by the `DatabaseMetaData.locatorsUpdateCopies()` method). Because of this, use the corresponding `PreparedStatement.setBlob()` or `ResultSet.updateBlob()` (in the case of updatable result sets) methods to save changes back to the database.

- **Connection**

  The `isClosed()` method does not ping the server to determine if it is available. In accordance with the JDBC specification, it only returns true if `closed()` has been called on the connection. If you need to determine if the connection is still valid, issue a simple query, such as `SELECT 1`. The driver will throw an exception if the connection is no longer valid.

- **DatabaseMetaData**

  Foreign key information (`getImportedKeys()`/`getExportedKeys()` and `getCrossReference()`) is only available from `InnoDB` tables. The driver uses `SHOW CREATE TABLE` to retrieve this information, so if any other storage engines add support for foreign keys, the driver would transparently support them as well.

- **PreparedStatement**

  Two variants of prepared statements are implemented by Connector/J, the client-side and the server-side prepared statements. Client-side prepared statements are used by default because early MySQL versions did not support the prepared statement feature or had problems with its implementation. Server-

side prepared statements and binary-encoded result sets are used when the server supports them. To enable usage of server-side prepared statements, set `useServerPrepStmts=true`.

Be careful when using a server-side prepared statement with **large** parameters that are set using `setBinaryStream()`, `setAsciiStream()`, `setUnicodeStream()`, `setCharacterStream()`, `setNCharacterStream()`, `setBlob()`, `setClob()`, or `setNCLob()`. To re-execute the statement with any large parameter changed to a nonlarge parameter, call `clearParameters()` and set all parameters again. The reason for this is as follows:

- During both server-side prepared statements and client-side emulation, large data is exchanged only when `PreparedStatement.execute()` is called.
- Once that has been done, the stream used to read the data on the client side is closed (as per the JDBC spec), and cannot be read from again.
- If a parameter changes from large to nonlarge, the driver must reset the server-side state of the prepared statement to allow the parameter that is being changed to take the place of the prior large value. This removes all of the large data that has already been sent to the server, thus requiring the data to be re-sent, using the `setBinaryStream()`, `setAsciiStream()`, `setUnicodeStream()`, `setCharacterStream()`, `setNCharacterStream()`, `setBlob()`, `setClob()`, or `setNCLob()` method.

Consequently, to change the type of a parameter to a nonlarge one, you must call `clearParameters()` and set all parameters of the prepared statement again before it can be re-executed.

- **ResultSet**

  By default, ResultSets are completely retrieved and stored in memory. In most cases this is the most efficient way to operate and, due to the design of the MySQL network protocol, is easier to implement. If you are working with ResultSets that have a large number of rows or large values and cannot allocate heap space in your JVM for the memory required, you can tell the driver to stream the results back one row at a time.

  To enable this functionality, create a `Statement` instance in the following manner:

```
stmt = conn.createStatement(java.sql.ResultSet.TYPE_FORWARD_ONLY,
              java.sql.ResultSet.CONCUR_READ_ONLY);
stmt.setFetchSize(Integer.MIN_VALUE);
```

  The combination of a forward-only, read-only result set, with a fetch size of `Integer.MIN_VALUE` serves as a signal to the driver to stream result sets row-by-row. After this, any result sets created with the statement will be retrieved row-by-row.

  There are some caveats with this approach. You must read all of the rows in the result set (or close it) before you can issue any other queries on the connection, or an exception will be thrown.

  The earliest the locks these statements hold can be released (whether they be `MyISAM` table-level locks or row-level locks in some other storage engine such as `InnoDB`) is when the statement completes.

  If the statement is within scope of a transaction, then locks are released when the transaction completes (which implies that the statement needs to complete first). As with most other databases, statements are not complete until all the results pending on the statement are read or the active result set for the statement is closed.

  Therefore, if using streaming results, process them as quickly as possible if you want to maintain concurrent access to the tables referenced by the statement producing the result set.

Another alternative is to use cursor-based streaming to retrieve a set number of rows each time. This can be done by setting the connection property `useCursorFetch` to true, and then calling `setFetchSize(int)` with `int` being the desired number of rows to be fetched each time:

```
conn = DriverManager.getConnection("jdbc:mysql://localhost/?useCursorFetch=true", "user", "s3cr3t");
stmt = conn.createStatement();
stmt.setFetchSize(100);
rs = stmt.executeQuery("SELECT * FROM your_table_here");
```

- **Statement**

  Connector/J includes support for both `Statement.cancel()` and `Statement.setQueryTimeout()`. Both require a separate connection to issue the `KILL QUERY` statement. In the case of `setQueryTimeout()`, the implementation creates an additional thread to handle the timeout functionality.

  > **Note**
  >
  > Failures to cancel the statement for `setQueryTimeout()` may manifest themselves as `RuntimeException` rather than failing silently, as there is currently no way to unblock the thread that is executing the query being cancelled due to timeout expiration and have it throw the exception instead.

  MySQL does not support SQL cursors, and the JDBC driver does not emulate them, so `setCursorName()` has no effect.

  Connector/J also supplies two additional methods:

  - `setLocalInfileInputStream()` sets an `InputStream` instance that will be used to send data to the MySQL server for a `LOAD DATA LOCAL INFILE` statement rather than a `FileInputStream` or `URLInputStream` that represents the path given as an argument to the statement.

    This stream will be read to completion upon execution of a `LOAD DATA LOCAL INFILE` statement, and will automatically be closed by the driver, so it needs to be reset before each call to `execute*()` that would cause the MySQL server to request data to fulfill the request for `LOAD DATA LOCAL INFILE`.

    If this value is set to `NULL`, the driver will revert to using a `FileInputStream` or `URLInputStream` as required.

  - `getLocalInfileInputStream()` returns the `InputStream` instance that will be used to send data in response to a `LOAD DATA LOCAL INFILE` statement.

    This method returns `NULL` if no such stream has been set using `setLocalInfileInputStream()`.

6.5 Java, JDBC, and MySQL Types

MySQL Connector/J is flexible in the way it handles conversions between MySQL data types and Java data types.

In general, any MySQL data type can be converted to a `java.lang.String`, and any numeric type can be converted to any of the Java numeric types, although round-off, overflow, or loss of precision may occur.

Connector/J issues warnings or throws `DataTruncation` exceptions as is required by the JDBC specification, unless the connection was configured not to do so by using the property `jdbcCompliantTruncation` and setting it to `false`.

The conversions that are always guaranteed to work are listed in the following table. The first column lists one or more MySQL data types, and the second column lists one or more Java types to which the MySQL types can be converted.

Table 6.19 Possible Conversions Between MySQL and Java Data Types

These MySQL Data Types	Can always be converted to these Java types
`CHAR, VARCHAR, BLOB, TEXT, ENUM, and SET`	`java.lang.String, java.io.InputStream, java.io.Reader, java.sql.Blob, java.sql.Clob`
`FLOAT, REAL, DOUBLE PRECISION, NUMERIC, DECIMAL, TINYINT, SMALLINT, MEDIUMINT, INTEGER, BIGINT`	`java.lang.String, java.lang.Short, java.lang.Integer, java.lang.Long, java.lang.Double, java.math.BigDecimal`
`DATE, TIME, DATETIME, TIMESTAMP`	`java.lang.String, java.sql.Date, java.sql.Timestamp`

Note

Round-off, overflow or loss of precision may occur if you choose a Java numeric data type that has less precision or capacity than the MySQL data type you are converting to/from.

The `ResultSet.getObject()` method uses the type conversions between MySQL and Java types, following the JDBC specification where appropriate. The values returned by `ResultSetMetaData.GetColumnTypeName()` and `ResultSetMetaData.GetColumnClassName()` are shown in the table below. For more information on the JDBC types, see the reference on the java.sql.Types class.

Table 6.20 MySQL Types and Return Values for ResultSetMetaData.GetColumnTypeName()and ResultSetMetaData.GetColumnClassName()

MySQL Type Name	Return value of `GetColumnTypeName`	Return value of `GetColumnClassName`
`BIT(1)`	`BIT`	`java.lang.Boolean`
`BIT( > 1)`	`BIT`	`byte[]`
`TINYINT(1) SIGNED, BOOLEAN`	If `tinyInt1isBit=true` and `transformedBitIsBoolean=false` `BIT` If `tinyInt1isBit=true` and `transformedBitIsBoolean=true:` `BOOLEAN` If `tinyInt1isBit=false:` `TINYINT`	If `tinyInt1isBit=true` and `transformedBitIsBoolean=false`: `java.lang.Boolean` If `tinyInt1isBit=true` and `transformedBitIsBoolean=true`: `java.lang.Boolean` If `tinyInt1isBit=false`: `java.lang.Integer`

MySQL Type Name	Return value of `GetColumnTypeName`	Return value of `GetColumnClassName`
`TINYINT( > 1) SIGNED`	`TINYINT`	`java.lang.Integer`
`TINYINT( any ) UNSIGNED`	`TINYINT UNSIGNED`	`java.lang.Integer`
`SMALLINT[(M)] [UNSIGNED]`	`SMALLINT [UNSIGNED]`	`java.lang.Integer` (regardless of whether it is `UNSIGNED` or not)
`MEDIUMINT[(M)] [UNSIGNED]`	`MEDIUMINT [UNSIGNED]`	`java.lang.Integer` (regardless of whether it is `UNSIGNED` or not)
`INT,INTEGER[(M)]`	`INTEGER`	`java.lang.Integer`
`INT,INTEGER[(M)] UNSIGNED`	`INTEGER UNSIGNED`	`java.lang.Long`
`BIGINT[(M)]`	`BIGINT`	`java.lang.Long`
`BIGINT[(M)] UNSIGNED`	`BIGINT UNSIGNED`	`java.math.BigInteger`
`FLOAT[(M,D)]`	`FLOAT`	`java.lang.Float`
`DOUBLE[(M,B)] [UNSIGNED]`	`DOUBLE`	`java.lang.Double` (regardless of whether it is `UNSIGNED` or not)
`DECIMAL[(M[,D])] [UNSIGNED]`	`DECIMAL`	`java.math.BigDecimal` (regardless of whether it is `UNSIGNED` or not)
`DATE`	`DATE`	`java.sql.Date`
`DATETIME`	`DATETIME`	`java.time.LocalDateTime`
`TIMESTAMP[(M)]`	`TIMESTAMP`	`java.sql.Timestamp`
`TIME`	`TIME`	`java.sql.Time`
`YEAR[(2|4)]`	`YEAR`	If `yearIsDateType` configuration property is set to `false`, then the returned object type is `java.sql.Short`. If set to `true` (the default), then the returned object is of type `java.sql.Date`.
`CHAR(M)`	`CHAR`	`java.lang.String`
`VARCHAR(M)`	`VARCHAR`	`java.lang.String`
`BINARY(M)`, `CHAR(M) BINARY`	`BINARY`	`byte[]`
`VARBINARY(M)`, `VARCHAR(M) BINARY`	`VARBINARY`	`byte[]`
`BLOB`	`BLOB`	`byte[]`
`TINYBLOB`	`TINYBLOB`	`byte[]`
`MEDIUMBLOB`	`MEDIUMBLOB`	`byte[]`
`LONGBLOB`	`LONGBLOB`	`byte[]`
`TEXT`	`TEXT`	`java.lang.String`
`TINYTEXT`	`TINYTEXT`	`java.lang.String`
`MEDIUMTEXT`	`MEDIUMTEXT`	`java.lang.String`
`LONGTEXT`	`LONGTEXT`	`java.lang.String`

MySQL Type Name	Return value of `GetColumnTypeName`	Return value of `GetColumnClassName`
`JSON`	`JSON`	`java.lang.String`
`GEOMETRY`	`GEOMETRY`	`byte[]`
`VECTOR(M)` (only supported when available with MySQL Enterprise Server)	`VECTOR`	`byte[]`
`ENUM('value1','value2',...)`	`CHAR`	`java.lang.String`
`SET('value1','value2',...)`	`CHAR`	`java.lang.String`

6.6 Handling of Date-Time Values

6.6.1 Preserving Time Instants

Background

A time instant is a specific moment on a time-line. A time instant is said to be preserved when it always refers to the same point in time when its value is being stored to or retrieved from a database, no matter what time zones the database server and the clients are operating in.

`TIMESTAMP` is the only MySQL data type designed to store instants. To preserve time instants, the server applies time zone conversions in incoming or outgoing time values when needed. Incoming values are converted by server from the connection session's time zone to Coordinated Universal Time (UTC) for storage, and outgoing values are converted from UTC to the session time zone. Starting from MySQL 8.0.19, you can also specify a time zone offset when storing `TIMESTAMP` values (see The DATE, DATETIME, and TIMESTAMP Types for details), in which case the `TIMESTAMP` values are converted to the UTC from the specified offset instead of the session time zone. But, once stored, the original offset information is no longer preserved.

The situation is less straightforward with the `DATETIME` data type: it does not represent an instant and, when no time zone offset is specified, there is no time zone conversion for `DATETIME` values, so they are stored and retrieved as they are. However, with a specified time zone offset, the input value is converted to the session time zone before it is stored; the result is that, when retrieved in a different session with a different time zone offset as the specified one, the `DATETIME` value becomes different from the original input value.

Because MySQL data types other than `TIMESTAMP` (and the Java wrapper classes for those other MySQL data types) do not represent true time instants; mixing up instant-representing and non-instant-representing date-time types when storing and retrieving values might give rise to unexpected results. For example:

- When storing `java.sql.Timestamp` to, for example, a `DATETIME` column, you might not get back the same instant value when retrieving it into a client that is in a different time zone than the one the client was in when storing the value.
- When storing, for example, a `java.time.LocalDateTime` to a `TIMESTAMP` column, you might not be storing the correct UTC-based value for it, because the time zone for the value is actually undefined.

Therefore, do not pass instant date-time types (`java.util.Calendar`, `java.util.Date`, `java.time.OffsetDateTime`, `java.sql.Timestamp`) to non-instant date-time types (for example, `java.sql.DATE`, `java.time.LocalDate`, `java.time.LocalTime`, `java.time.OffsetTime`) or vice versa, when working with the server.

The rest of the section discusses how to preserve time instants when working with Connector/J.

Preserving Instants with Connector/J

The scenario: Let us assume that an application is running on a certain application server and is connecting to a MySQL server using Connector/J. Certain events take place in a connection session, for which timestamps are generated, and the event timestamps are associated with the JVM time zone of the application server. These timestamps are to be stored onto a MySQL Server, and are also to be retrieved from it later.

The challenge: The timestamps' instant values need to be preserved when they are saved onto or retrieved from the server using Connector/J. Because the MySQL Server always assumes implicitly that a time instant value references to the connection session time zone (which is set by the session `time_zone` variable) when being saved to or retrieved form the server, a time instant value is properly preserved only in the following situations:

1. When Connector/J is running in the same time zone as the MySQL Server (i.e., the server's session time zone is the same as the JVM's time zone), time instants are naturally preserved, and no time zone conversion is needed. Note that in this case, time instants are really preserved only if the server and the JVM continue to run always in the same time zone in the future.

2. When Connector/J is running in a different time zone from that of the MySQL Server (i.e., the JVM's time zone is different from the server's session time zone), Connector/.J performs one of the following:

   a. Queries the value of the session time zone from the server, and converts the event timestamps between the session time zone and the JVM time zone.

   b. Changes the server's session time zone to that of the JVM time zone, after which no time zone conversion will be required.

   c. Changes the server session time zone to a desired time zone specified by the user, and then converts the timestamps between the JVM time zone and the user-specified time zone.

We identify the above solutions for time instant preservation as Solution 1, 2a, 2b, and 2c. To achieve these solutions, the following connection properties have been introduced in Connector/J since release 8.0.23:

- `preserveInstants={true|false}`: Whether to attempt to preserve time instant values by adjusting timestamps.
  - When it is `false`, no conversions are attempted; a timestamp is sent to the server as-is for storage, and its visual presentation, not the actual time instant is preserved. When it is retrieved from the server by Connector/J, different time zones might be associated with it, as the retrieval might happen in different JVM time zones. For example: For example:
    - Time zones: UTC for JVM, UTC+1 for server session
    - Original timestamp from client (in UTC): `2020-01-01 01:00:00`
    - Timestamp sent to server by Connector/J: `2020-01-01 01:00:00` (no conversion)
    - Timestamp values stored internally on the server: `2020-01-01 00:00:00 UTC` (after internal conversion of `2020-01-01 00:00:00 UTC+1` to UTC)
    - Timestamp value retrieved later into a server section (in UTC+1): `2020-01-01 01:00:00` (after internal conversion of `2020-01-01 00:00:00` from UTC to UTC+1)

    - Timestamp values constructed by Connector/J in some other JVM time zone then before (say, in UTC+3): `2020-01-01 01:00:00`
    - Comment: Time instant is not preserved

  - When it is `true`, Connector/J attempts to preserve the time instants by performing the conversions in a manner defined by the connection properties `connectionTimeZone` and `forceConnectionTimeZoneToSession`.

    When storing a value, the conversion is performed only if the target data type, either the explicit one or the default one, is `TIMESTAMP`. When retrieving a value, the conversion is performed only if the source column has the `TIMESTAMP`, `DATETIME`, or a character data type and the target class is an instant-preserving one, like `java.sql.Timestamp` or `java.time.OffsetDateTime`.

- `connectionTimeZone={LOCAL|SERVER|`*`user-defined-time-zone`*`}`: Specifies how the server's session time zone (in reference to which the timestamps are saved onto the server) is to be determined by Connector/J. It takes on one of the following values:

  - `LOCAL`: Connector/J assumes that the server's session time zone either (a) is the same as the JVM time zone for Connector/J, or (b) should be set as the same as the JVM time zone for Connector/J. Connector/J takes the situation as (a) or (b) depending on the value of the connection property `forceConnectionTimeZoneToSession`.

  - SERVER: Connector/J should query the session's time zone from the server, instead of making any assumptions about it. If the session time zone actually turns out to be different from Connector/J's JVM time zone and `preserveInstants=true`, Connector/J performs time zone conversion between the session time zone and the JVM time zone.

  - *`user-defined-time-zone`*: Connector/J assumes that the server's session time zone either (a) is the same as the user-defined time zone, or (b) should be set as the user-defined time zone. Connector/J takes the situation as (a) or (b) depending on the value of the connection property `forceConnectionTimeZoneToSession`.

    > **Note**
    >
    > For Connector/J 8.0.23 and later, `serverTimezone` is an alias for `connectionTimeZone`. For Connector/J 8.0.22 and earlier, `serverTimezone` was used to override the session time zone setting on the server.

- `forceConnectionTimeZoneToSession={true|false}`: Controls whether the session `time_zone` variable is to be set to the value specified in `connectionTimeZone`.

Now, here are the connection properties values to be used for achieving the Solutions defined above for preserving time instants:

- Solution 1: Use either **preserveInstants=false** or **connectionTimeZone=LOCAL& forceConnectionTimeZoneToSession=false**. Because it can be safely assumed that the server session time zone is the same as Connector/J' s JVM timezone, no query of the server's session time zone occurs, and no time zone conversion occurs. For example:

  - Time zones: UTC+1 for both the JVM and the server session
  - Original timestamp from client (in UTC+1): `2020-01-01 01:00:00`
  - Timestamp sent to server by Connector/J: `2020-01-01 01:00:00` (no conversion needed)

- Timestamp values stored internally on the server: `2020-01-01 00:00:00 UTC` (after internal conversion from UTC+1 to UTC)
- Timestamp value retrieved later into a server time session in UTC+1 that Connector/J connects to: `2020-01-01 01:00:00` (after internal conversion from UTC to UTC+1)
- Timestamp value constructed by Connector/J in the same JVM time zone as before (UTC+1) and returned to an application: `2020-01-01 01:00:00`
- Comment: Time instant is preserved without conversion.

> **Note**
>
> This setting corresponds to the default behavior of Connector/J 5.1

- Solution 2a: Use **preserveInstants=true&connectionTimeZone=SERVER** . Connector/J then queries the value of the session time zone from the server, and converts the event timestamps between the session time zone and the JVM time zone. For example:
  - Time zones: UTC+2 for JVM, UTC+1 for server session
  - Original timestamp from client (in UTC+2): `2020-01-01 02:00:00`
  - Timestamp sent to server by Connector/J: `2020-01-01 01:00:00` (after conversion from UTC+2 to UTC+1)
  - Timestamp value stored internally on the server: `2020-01-01 00:00:00 UTC` (after internal conversion from UTC+1 to UTC)
  - Timestamp value retrieved later into a server session in UTC+1: `2020-01-01 01:00:00` (after internal conversion from UTC to UTC+1)
  - Timestamp values constructed by Connector/J in the same JVM time zone as before (UTC+2) and returned to an application: `2020-01-01 02:00:00` (after conversion from UTC+1 to UTC+2)
  - Timestamp values constructed by Connector/J in another JVM time zone (say, UTC+3) and returned to an application: `2020-01-01 03:00:00` (after conversion from UTC+1 to UTC+3)
  - Comment: Time instant is preserved.

> **Notes**
>
> - This setting corresponds to the default behavior of Connector/J 8.0.22 and before and to the behavior of Connector/J 5.1 with `useLegacyDatetimeCode=false`.

- Solution 2b: Use **connectionTimeZone=LOCAL& forceConnectionTimeZoneToSession=true**. Connector/J then changes the server's session time zone to that of the JVM time zone, after which no timezone conversions are required when storing or achieving the timestamps. For example:
  - Time zones: UTC+1 for JVM, UTC+2 for server session originally, but now modified to UTC+1 by Connector/J
  - Original timestamp from client (in UTC+1): `2020-01-01 01:00:00`
  - Timestamp sent to server by Connector/J: `2020-01-01 01:00:00` (no conversion)
  - Timestamp values stored internally on the server: `2020-01-01 00:00:00` (after internal conversion from UTC+1 to UTC)
  - Timestamp values retrieved later into a server session (in UTC+1, as set by Connector/J): `2020-01-01 01:00:00` (after internal conversion from UTC to UTC+1)
  - Timestamp value constructed by Connector/J in the same JVM time zone as before (UTC+1): `2020-01-01 01:00:00` (no conversion needed)
  - Timestamp values retrieved later into a server session (time zone modified to, say, UTC+3, by Connector/J): `2020-01-01 03:00:00` (after internal conversion from UTC to UTC+3)
  - Timestamp value constructed by Connector/J in the JVM time zone of UTC+3: `2020-01-01 03:00:00` (no conversion needed)
  - Comment: Time instant is preserved without conversion by Connector/J, because the session time zone is changed by Connector/J to its JVM's value.

    Warnings

    - Altering the session time zone affects the results of MySQL functions such as `NOW()`, `CURTIME()`, or `CURDATE()`—if you do not want those functions to be affected, do not use this setting.
    - If you use this setting on different clients in different time zones, the clients are going to modify their connection session's time zones to different values; if you want to keep the same visual date-time value representation for the same time instant for all the clients and in all their sessions, store the values to a `DATETIME` instead of a `TIMESTAMP` column and use non-instant Java classes for them, for example, `java.time.LocalDateTime`.
- Solution 2c: Use **preserveInstants=true&connectionTimeZone=*`user-defined-time-zone`*& forceConnectionTimeZoneToSession=true**. Connector/J then changes the server's session time zone to the user-defined time zone, and converts the timestamps between the user-defined time zone and the JVM time zone. A typical use case for this setting is when the session time zone value on the server is known to be unrecognizable by Connector/J (e.g., `CST` or `CEST`). For example:
  - Time zones: UTC+2 for JVM, `CET` for server session originally, but now modified to user-specified `Europe/Berlin` by Connector/J
  - Original timestamp from client (in UTC+2): `2020-01-01 02:00:00`
  - Timestamp sent to server by Connector/J: `2020-01-01 01:00:00` (after conversion between JVM time zone (UTC+2) and user-defined time zone (`Europe/Berlin`=UTC+1))

- Timestamp values stored internally on the server: `2020-01-01 00:00:00` (after internal conversion from UTC+1 to UTC)
- Timestamp value retrieved into a server session (time zone modified to `Europe/Berlin` (=UTC+1) by Connector/J): `2020-01-01 01:00:00` (after internal conversion from UTC to UTC+1)
- Timestamp value constructed by Connector/J in the same JVM time zone as before (UTC+2) and returned to an application: `2020-01-01 02:00:00` (after conversion between user-defined time zone (UTC+1) and JVM time zone (UTC+2)).
- Comment: Time instant is preserved with conversion and with the session time zone being changed by Connector/J according to a user-defined value.

As an alternative to this solution, the user might want the same conversion of the timestamps between the JVM time zone and the user-defined time zone as described above, without actually correcting the unrecognizable time zone value on the server. To do so, use, `preserveInstants=true&connectionTimeZone=user-defined-time-zone&forceConnectionTimeZoneToSession=false`. This achieves the same result of preserving the time instant.

> **Warnings**
>
> See the warnings above for Solution 2b.

6.6.2 Fractional Seconds

While a `java.sql.TIME` instance, according to the JDBC specification, is not supposed to contain fractional seconds by design, because `java.sql.TIME` is a wrapper around `java.util.Date`, it is possible to store fractional seconds in a `java.sql.TIME` instance. However, when Connector/J inserted a `java.sql.TIME` into the server as a MySQL `TIME` value, the fractional seconds were always truncated. To allow the fractional seconds to be sent to the server, a connection property, `sendFractionalSecondsForTime`, has been introduced in release 8.0.23: when the property is `true` (which is the default value), the fractional seconds for `java.sql.TIME` are sent to the server; otherwise, the fractional seconds are truncated.

Also, the connection property `sendFractionalSeconds` has become a global control for the sending of fractional seconds for ALL date-time types since release 8.0.23. As a result, if `sendFractionalSeconds=false`, fractional seconds are not sent irrespective of the value of `sendFractionalSecondsForTime`.

6.6.3 Handling of YEAR Values

How a value in a MySQL YEAR column is handled is controlled by the connection property yearIsDateType:

- If yearIsDateType is `true` (the default), YEAR is mapped to the Java data type `java.sql.Date`.
- If yearIsDateType is `false`, YEAR is mapped to the Java data type `java.sql.Short`.

Connector/J follows the same rules that govern how values are inserted by a `mysql` client; see explanations in The YEAR Type for details.

Connector/J handles the retrieval of zero values from a YEAR column differently than a `mysql` client. Treatments of zero values depend on whether they are strings or numbers, and on the value of yearIsDateType:

- If a string value of `'0'`, `'00'`, or `'000'` is entered into a YEAR column, when retrieved by Connector/J:
  - If yearIsDateType is true, the retrieved value is equivalent to January 1, 2000 00:00:00.000.
  - If yearIsDateType is false, the retrieved value is `2000`
- If a numeric value of `0`, `00`, `000`, or `0000` is entered into a YEAR column, when retrieved by Connector/J,
  - If yearIsDateType is true, the retrieved value is equivalent to January 1, 2000 00:00:00.000.
  - If yearIsDateType is false, the retrieved value is `0`

6.7 Using Character Sets and Unicode

All strings sent from the JDBC driver to the server are converted automatically from native Java Unicode form to the connection's character encoding, including all queries sent using `Statement.execute()`, `Statement.executeUpdate()`, and `Statement.executeQuery()`, as well as all `PreparedStatement` and `CallableStatement` parameters, *excluding* parameters set using the following methods:

- `setBlob()`
- `setBytes()`
- `setClob()`
- `setNClob()`
- `setAsciiStream()`
- `setBinaryStream()`
- `setCharacterStream()`
- `setNCharacterStream()`
- `setUnicodeStream()`

Number of Encodings Per Connection

Connector/J supports a single character encoding between the client and the server, and any number of character encodings for data returned by the server to the client in `ResultSets`.

Setting the Character Encoding

For Connector/J 8.0.25 and earlier: The character encoding between the client and the server is automatically detected upon connection (provided that the Connector/J connection properties `characterEncoding` and `connectionCollation` are not set). The encoding on the server is specified using the system variable `character_set_server` (for more information, see Server Character Set and Collation), and the driver automatically uses the encoding. For example, to use the 4-byte UTF-8 character set with Connector/J, configure the MySQL server with `character_set_server=utf8mb4`, and leave `characterEncoding` and `connectionCollation` out of the Connector/J connection string. Connector/J will then autodetect the UTF-8 setting. To override the automatically detected encoding on the client side, use the `characterEncoding` property in the connection URL to the server.

For Connector/J 8.0.26 and later: There are two phases during the connection initialization in which the character encoding and collation are set.

- *Pre-Authentication Phase:* In this phase, the character encoding between the client and the server is determined by the settings of the Connector/J connection properties, in the following order of priority:
  - `passwordCharacterEncoding`
  - `connectionCollation`
  - `characterEncoding`
  - Set to `UTF8` (corresponds to `utf8mb4` on MySQL servers), if none of the properties above is set
- *Post-Authentication Phase*: In this phase, the character encoding between the client and the server for the rest of the session is determined by the settings of the Connector/J connection properties, in the following order of priority:
  - `connectionCollation`
  - `characterEncoding`
  - Set to `UTF8` (corresponds to `utf8mb4` on MySQL servers), if none of the properties above is set

  This means Connector/J needs to issue a SET NAMES Statement to change the character set and collation that were established in the pre-authentication phase only if `passwordCharacterEncoding` is set, but its setting is different from that of `connectionCollation`, or different from that of `characterEncoding` (when `connectionCollation` is not set), or different from `utf8mb4` (when both `connectionCollation` and `characterEncoding` are not set).

Custom Character Sets and Collations

To support the use of custom character sets and collations on the server, set the Connector/J connection property `detectCustomCollations` to `true`, and provide the mapping between the custom character sets and the Java character encodings by supplying the `customCharsetMapping` connection property with a comma-delimited list of *`custom_charset:java_encoding`* pairs (for example: `customCharsetMapping=charset1:UTF-8,charset2:Cp1252`).

MySQL to Java Encoding Name Translations

Use Java-style names when specifying character encodings. The following table lists MySQL character set names and their corresponding Java-style names:

Table 6.21 MySQL to Java Encoding Name Translations

MySQL Character Set Name	Java-Style Character Encoding Name
`ascii`	`US-ASCII`
`big5`	`Big5`
`gbk`	`GBK`
`sjis`	`SJIS or Cp932`
`cp932`	`Cp932 or MS932`
`gb2312`	`EUC_CN`

MySQL Character Set Name	Java-Style Character Encoding Name
ujis	EUC_JP
euckr	EUC_KR
latin1	Cp1252
latin2	ISO8859_2
greek	ISO8859_7
hebrew	ISO8859_8
cp866	Cp866
tis620	TIS620
cp1250	Cp1250
cp1251	Cp1251
cp1257	Cp1257
macroman	MacRoman
macce	MacCentralEurope
utf8mb4	UTF-8
ucs2	UnicodeBig

Notes

- When `UTF-8` is used for `characterEncoding` in the connection string, it maps to the MySQL character set name `utf8mb4`.
- If the connection option `connectionCollation` is also set alongside `characterEncoding` and is incompatible with it, `characterEncoding` will be overridden with the encoding corresponding to `connectionCollation`.
- Because there is no Java-style character set name for `utfmb3` that you can use with the connection option `charaterEncoding`, the only way to use `utf8mb3` as your connection character set is to use a `utf8mb3` collation (for example, `utf8_general_ci`) for the connection option `connectionCollation`, which forces a `utf8mb3` character set to be used, as explained in the last bullet.

Warning

Do not issue the query SET NAMES with Connector/J, as the driver will not detect that the character set has been changed by the query, and will continue to use the character set configured when the connection was first set up.

6.8 Using Query Attributes

Connector/J supports Query Attributes when it has been enabled on the server by installing the `query_attributes` component (see Prerequisites for Using Query Attributes for details).

Attributes are set for a query by using the `setAttribute()` method of the `JdbcStatement` interface. Here is the method's signature:

```
JdbcStatement.setAttribute(String name, Object value)
```

Here is an example of using the query attributes with a `JdbcStatement`:

Example 6.1 Using Query Attributes with a Plain Statement

```
conn = DriverManager.getConnection("jdbc:mysql://localhost/test", "myuser", "password");

Statement stmt = conn.createStatement();

JdbcStatement jdbcStmt = (JdbcStatement) stmt;

jdbcStmt.executeUpdate("CREATE TABLE t11 (c1 CHAR(20), c2 CHAR(20))");

jdbcStmt.setAttribute("attr1", "cat");
jdbcStmt.setAttribute("attr2", "mat");
jdbcStmt.executeUpdate("INSERT INTO t11 (c1, c2) VALUES(\n" +
    " mysql_query_attribute_string('attr1'),\n" +
    " mysql_query_attribute_string('attr2')\n" +
    " );");

ResultSet rs = stmt.executeQuery("SELECT * from t11");

while(rs.next()) {
    String col1 = rs.getString(1);
    String col2 = rs.getString(2);
    System.out.println("The "+col1+" is on the "+col2);
    }
```

While query attributes are cleared on the server after each query, they are kept on the side of Connector/J, so they can be resent for the next query. To clear the attributes, use the `clearAttributes()` method of the `JdbcStatement` interface:

```
JdbcStatement.clearAttributes()
```

The following example (a continuation of the code in Example 6.1, “Using Query Attributes with a Plain Statement”) shows how the attributes are preserved for a statement until it is cleared :

Example 6.2 Preservation of Query Attributes

```
/* Continuing from the code in the last example, where query attributes have
already been set and used */

rs = stmt.executeQuery("SELECT c2 FROM t11 where " +
                "c1 = mysql_query_attribute_string('attr1')");

        if (rs.next()) {
            String col1 = rs.getString(1);
            System.out.println("It is on the "+col1);
        }

        // Prints "It is on the mat"

        jdbcStmt.clearAttributes();
        rs = stmt.executeQuery("SELECT c2 FROM t11 where " +
                "c1 = mysql_query_attribute_string('attr1')");

        if (rs.next()) {
            String col1 = rs.getString(1);
            System.out.println("It is on the "+col1);
        }

        else {
            System.out.println("No results!");
        }
        // Prints "No results!" as attribute string attr1 is empty
```

Attributes can also be set for client-side and server-side prepared statements, using the `setAttribute()` method:

Example 6.3 Using Query Attributes with a Prepared Statement

```
conn = DriverManager.getConnection("jdbc:mysql://localhost/test", "myuser", "password");

PreparedStatement ps = conn.prepareStatement(
                "select ?, c2 from t11 where c1 = mysql_query_attribute_string('attr1')");
ps.setString(1, "It is on a ");

JdbcStatement jdbcPs = (JdbcStatement) ps;
jdbcPs.setAttribute("attr1", "cat");
rs = ps.executeQuery();
if (rs.next()) {
            System.out.println(rs.getString(1)+" "+ rs.getString(2));
}
```

Not all MySQL data types are supported by the `setAttribute()` method; only the following MySQL data types are supported and are directly mapped to from specific Java objects or their subclasses:

Table 6.22 Data Type Mappings for Query Attributes

MySQL Data Type	Java Object
`MYSQL_TYPE_STRING`	`java.lang.String`
`MYSQL_TYPE_TINY`	`java.lang.Boolean`, `java.lang.Byte`
`MYSQL_TYPE_SHORT`	`java.lang.Short`
`MYSQL_TYPE_LONG`	`java.lang.Integer`
`MYSQL_TYPE_LONGLONG`	`java.lang.Long`, `java.math.BigInteger`
`MYSQL_TYPE_FLOAT`	`java.lang.Float`
`MYSQL_TYPE_DOUBLE`	`java.lang.Double`, `java.math.BigDecimal`
`MYSQL_TYPE_DATE`	`java.sql.Date`, `java.time.LocalDate`
`MYSQL_TYPE_TIME`	`java.sql.Time`, `java.time.LocalTime`, `java.time.OffsetTime`, `java.time.Duration`
`MYSQL_TYPE_DATETIME`	`java.time.LocalDateTime`
`MYSQL_TYPE_TIMESTAMP`	`java.sql.Timestamp`, `java.time.Instant`, `java.time.OffsetDateTime`, `java.time.ZonedDateTime`, `java.util.Date`, `java.util.Calendar`

When there is no direct mapping from a Java object type to any MySQL data type, the attribute is set with a string value that comes from converting the supplied object to a `String` using the `.toString()` method.

6.9 Connecting Securely Using SSL

Connector/J can encrypt all data communicated between the JDBC driver and the server (except for the initial handshake) using SSL. There is a performance penalty for enabling connection encryption, the severity of which depends on multiple factors including (but not limited to) the size of the query, the amount of data returned, the server hardware, the SSL library used, the network bandwidth, and so on.

The system works through two Java keystore files: one file contains the certificate information for the server (`truststore` in the examples below), and another contains the keys and certificate for the client (`keystore` in the examples below). All Java keystore files are protected by the password supplied to the `keytool` when you created the files. You need the file names and the associated passwords to create an SSL connection.

For SSL support to work, you must have the following:

- A MySQL server that supports SSL, and compiled and configured to do so. For more information, see Using Encrypted Connections and Configuring SSL Library Support.
- A signed client certificate, if using mutual (two-way) authentication.

By default, Connector/J establishes secure connections with the MySQL servers. Note that MySQL servers 5.7 and up, when compiled with OpenSSL, can automatically generate missing SSL files at startup and configure the SSL connection accordingly.

For 8.0.12 and earlier: As long as the server is correctly configured to use SSL, there is no need to configure anything on the Connector/J client to use encrypted connections (the exception is when Connector/J is connecting to very old server versions like 5.6.25 and earlier or 5.7.5 and earlier, in which case the client must set the connection property `useSSL=true` in order to use encrypted connections). The client can demand SSL to be used by setting the connection property `requireSSL=true`; the connection then fails if the server is not configured to use SSL. Without `requireSSL=true`, the connection just falls back to non-encrypted mode if the server is not configured to use SSL.

For 8.0.13 and later: As long as the server is correctly configured to use SSL, there is no need to configure anything on the Connector/J client to use encrypted connections. The client can demand SSL to be used by setting the connection property `sslMode=REQUIRED`, `VERIFY_CA`, or `VERIFY_IDENTITY`; the connection then fails if the server is not configured to use SSL. With `sslMode=PREFERRED`, the connection just falls back to non-encrypted mode if the server is not configured to use SSL. For X-Protocol connections, the connection property `xdevapi.ssl-mode` specifies the SSL Mode setting, just like `sslMode` does for MySQL-protocol connections (except that `PREFERRED` is not supported by X Protocol); if not explicitly set, `xdevapi.ssl-mode` takes up the value of `sslMode` (if `xdevapi.ssl-mode` is not set and `sslMode` is set to `PREFERRED`, `xdevapi.ssl-mode` is set to `REQUIRED`).

For additional security, you can setup the client for a one-way (server or client) or two-way (server and client) SSL authentication, allowing the client or the server to authenticate each other's identity.

TLS versions: The allowable versions of TLS protocol can be restricted using the connection properties `tlsVersions` and, for X DevAPI connections and for release 8.0.19 and later, `xdevapi.tls-versions` (when `xdevapi.tls-versions` is not specified, it takes up the value of `tlsVersions`). If no such restrictions have been specified, Connector/J attempts to connect to the server with the TLSv1.2 and TLSv1.3.

Notes

- *Since Connector/J 8.0.28,* the connection property `enabledTLSProtocols` has been renamed to `tlsVersions`, and `enabledSSLCipherSuites` has been renamed to `tlsCiphersuites`; the original names remain as aliases.
- *For Connector/J 8.0.26 and later:* TLSv1 and TLSv1.1 were deprecated in Connector/J 8.0.26 and removed in release 8.0.28; the removed values are considered invalid for use with connection options and session settings. Connections can be made using the more-secure TLSv1.2 and TLSv1.3 protocols. Using TLSv1.3 requires that the server be compiled with OpenSSL 1.1.1 or higher and Connector/J be run with a JVM that supports TLSv1.3 (for example, Oracle Java 8u261 and above).
- *For Connector/J 8.0.18 and earlier when connecting to MySQL Community Server 5.6 and 5.7 using the JDBC API:* Due to compatibility issues with MySQL Server compiled with yaSSL, Connector/J does not enable connections

> with TLSv1.2 and higher by default. When connecting to servers that restrict connections to use those higher TLS versions, enable them explicitly by setting the Connector/J connection property `enabledTLSProtocols` (e.g., set `enabledTLSProtocols=TLSv1.2,TLSv1.3`).

Cipher Suites: Since release 8.0.19, the cipher suites usable by Connector/J are pre-restricted by a properties file that can be found at `src/main/resources/com/mysql/cj/TlsSettings.properties` inside the `src` folder on the source tree or in the platform-independent distribution archive (in `.tar.gz` or `.zip` format) for Connector/J. The file contains four sections, listing in each the mandatory, approved, deprecated, and unacceptable ciphers. Only suites listed in the first three sections can be used. The last section (unacceptable) defines patterns or masks that blocklist unsafe cipher suites. Practically, with the allowlist already given in the first three sections, the blocklist patterns in the forth section are redundant; but they are there as an extra safeguard against unwanted ciphers. The allowlist and blocklist of cipher suites apply to both JDBC and X DevAPI connections.

The allowable cipher suites for SSL connections can be restricted using the connection properties `tlsCiphersuites` and, for X DevAPI connections and for release 8.0.19 and later, `xdevapi.tls-ciphersuites` (when `xdevapi.tls-ciphersuites` is not specified, it takes up the value of `tlsCiphersuites`). If no such restrictions have been specified, Connector/J attempts to establish SSL connections with any allowlisted cipher suites that the server accepts.

6.9.1 Setting up Server Authentication

Server authentication via server certificate verification is enabled when the Connector/J connection property `sslMode` is set to `VERIFY_CA` or `VERIFY_IDENTITY`. If `sslMode` is not set, server authentication via server certificate verification is enabled when the legacy properties `useSSL` AND `verifyServerCertificate` are both true.

Certificates signed by a trusted CA. When server authentication via server certificate verification is enabled, if no additional configurations are made regarding server authentication, Java verifies the server certificate using its default trusted CA certificates, usually from `$JAVA_HOME/lib/security/cacerts`.

Using self-signed certificates. It is pretty common though for MySQL server certificates to be self-signed or signed by a self-signed CA certificate; the auto-generated certificates and keys created by the MySQL server are based on the latter—that is, the server generates all required keys and a self-signed CA certificate that is used to sign a server and a client certificate. The server then configures itself to use the CA certificate and the server certificate. Although the client certificate file is placed in the same directory, it is not used by the server.

To verify the server certificate, Connector/J needs to be able to read the certificate that signed it, that is, the server certificate that signed itself or the self-signed CA certificate. This can be accomplished by either importing the certificate (`ca.pem` or any other certificate) into the Java default truststore (although tampering the default truststore is not recommended) or by importing it into a custom Java truststore file and configuring the Connector/J driver accordingly. Use Java's keytool (typically located in the `bin` subdirectory of your JDK or JRE installation) to import the server certificates:

```
$> keytool -importcert -alias MySQLCACert -file ca.pem \
    -keystore truststore -storepass mypassword
```

Supply the proper arguments for the command options. If the truststore file does not already exist, a new one will be created; otherwise the certificate will be added to the existing file. Interaction with `keytool` looks like this:

```
Owner: CN=MySQL_Server_8.4.0_Auto_Generated_CA_Certificate
Issuer: CN=MySQL_Server_8.4.0_Auto_Generated_CA_Certificate
```

```
Serial number: 1
Valid from: Thu Mar 07 11:37:33 WET 2024 until: Sun Mar 05 11:37:33 WET 2034
Certificate fingerprints:
  SHA1: 43:12:0F:96:1A:09:1C:D2:5B:62:7A:2A:55:6C:62:6A:84:5F:78:E4
  SHA256: 7D:86:18:FF:06:A7:DF:A7:7C:D0:07:AB:96:1A:51:FD:02:4F:32:BF:1C:51:35:42:27:81:53:0A:8F:D3:56:39
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

The output of the command shows all details about the imported certificate. Make sure you remember the password you have supplied. Also, be mindful that the password will have to be written as plain text in your Connector/J configuration file or application source code.

The next step is to configure Java or Connector/J to read the truststore you just created or modified. This can be done by using one of the following three methods:

1. Using the Java command line arguments:

   ```
   -Djavax.net.ssl.trustStore=path_to_truststore_file
   -Djavax.net.ssl.trustStorePassword=mypassword
   ```

2. Setting the system properties directly in the client code:

   ```
   System.setProperty("javax.net.ssl.trustStore","path_to_truststore_file");
   System.setProperty("javax.net.ssl.trustStorePassword","mypassword");
   ```

3. Setting the Connector/J connection properties:

   ```
   trustCertificateKeyStoreUrl=file:path_to_truststore_file
   trustCertificateKeyStorePassword=mypassword
   ```

Notice that when used together, the connection properties override the values set by the other two methods. Also, whatever values set with connection properties are used in that connection only, while values set using the system-wide values are used for all connections (unless overridden by the connection properties). Setting the connection property `fallbackToSystemTrustStore` to `false` prevents Connector/J from falling back to the system-wide truststore setup you created using method (1) or (2) when method (3) is not used.

With the above setup and the server authentication enabled, all connections established are going to be SSL-encrypted, with the server being authenticated in the SSL handshake process, and the client can now safely trust the server it is connecting to.

For X-Protocol connections, the connection properties `xdevapi.ssl-truststore`, `xdevapi.ssl-truststore-type`, `xdevapi.ssl-truststore-password`, and `xdevapi.ssl-fallbackToSystemTrustStore` specify the truststore settings, just like `trustCertificateKeyStoreUrl`, `trustCertificateKeyStoreType`, `trustCertificateKeyStorePassword`, and `fallbackToSystemTrustStore` do for MySQL-protocol connections; if not explicitly set, `xdevapi.ssl-truststore`, `xdevapi.ssl-truststore-type`, `xdevapi.ssl-truststore-password`, and `xdevapi.ssl-fallbackToSystemTrustStore`

take up the values of `trustCertificateKeyStoreUrl`, `trustCertificateKeyStoreType`, `trustCertificateKeyStorePassword`, and `fallbackToSystemTrustStore` respectively.

Service Identity Verification. Beyond server authentication via server certificate verification, when `sslMode` is set to `VERIFY_IDENTITY`, Connector/J also performs host name identity verification by checking whether the host name that it uses for connecting matches the Common Name value in the server certificate.

6.9.2 Setting up Client Authentication

The server may want to authenticate a client and require the client to provide an SSL certificate to it, which it verifies against its known certificate authorities or performs additional checks on the client identity if needed (see CREATE USER SSL/TLS Options for details). In that case, Connector/J needs to have access to the client certificate, so it can be sent to the server while establishing new database connections. This is done using the Java keystore files.

To allow client authentication, the client connecting to the server must have its own set of keys and an SSL certificate. The client certificate must be signed so that the server can verify it. While you can have the client certificates signed by official certificate authorities, it is more common to use an intermediate, private, CA certificate to sign client certificates. Such an intermediate CA certificate may be self-signed or signed by a trusted root CA. The requirement is that the server knows a CA certificate that is capable of validating the client certificate.

Some MySQL server builds are able to generate SSL keys and certificates for communication encryption, including a certificate and a private key (contained in the `client-cert.pem` and `client-key.pem` files), which can be used by any client. This SSL certificate is already signed by the self-signed CA certificate `ca.pem`, which the server may have already been configured to use.

If you do not want to use the client keys and certificate files generated by the server, you can also generate new ones using the procedures described in Creating SSL and RSA Certificates and Keys. Notice that, according to the setup of the server, you may have to reuse the already existing CA certificate the server is configured to work with to sign the new client certificate, instead of creating a new one.

Once you have the client private key and certificate files you want to use, you need to import them into a Java keystore so that they can be used by the Java SSL library and Connector/J. The following instructions explain how to create the keystore file:

- Convert the client key and certificate files to a PKCS #12 archive:

```
$> openssl pkcs12 -export -in client-cert.pem -inkey client-key.pem \
  -name "mysqlclient" -passout pass:mypassword -out client-keystore.p12
```

- Import the client key and certificate into a Java keystore:

```
$> keytool -importkeystore -srckeystore client-keystore.p12 -srcstoretype pkcs12 \
 -srcstorepass mypassword -destkeystore keystore -deststoretype JKS -deststorepass mypassword
```

  Supply the proper arguments for the command options. If the keystore file does not already exist, a new one will be created; otherwise the certificate will be added to the existing file. Output by `keytool` looks like this:

```
Entry for alias mysqlclient successfully imported.
Import command completed:  1 entries successfully imported, 0 entries failed or cancelled
```

  Make sure you remember the password you have chosen. Also, be mindful that the password will have to be written as plain text in your Connector/J configuration file or application source code.

After the step, you can delete the PKCS #12 archive (*`client-keystore.p12`* in the example).

The next step is to configure Java or Connector/J so that it reads the keystore you just created or modified. This can be done by using one of the following three methods:

1. Using the Java command line arguments:

```
-Djavax.net.ssl.keyStore=path_to_keystore_file
-Djavax.net.ssl.keyStorePassword=mypassword
```

2. Setting the system properties directly in the client code:

```
System.setProperty("javax.net.ssl.keyStore","path_to_keystore_file");
System.setProperty("javax.net.ssl.keyStorePassword","mypassword");
```

3. Through Connector/J connection properties:

```
clientCertificateKeyStoreUrl=file:path_to_truststore_file
clientCertificateKeyStorePassword=mypassword
```

Notice that when used together, the connection properties override the values set by the other two methods. Also, whatever values set with connection properties are used in that connection only, while values set using the system-wide values are used for all connections (unless overridden by the connection properties). Setting the connection property `fallbackToSystemKeyStore` to `false` prevents Connector/J from falling back to the system-wide keystore setup you created using method (1) or (2) when method (3) is not used.

With the above setups, all connections established are going to be SSL-encrypted with the client being authenticated in the SSL handshake process, and the server can now safely trust the client that is requesting a connection to it.

For X-Protocol connections, the connection properties `xdevapi.ssl-keystore`, `xdevapi.ssl-keystore-type`, `xdevapi.ssl-keystore-password`, and `xdevapi.ssl-fallbackToSystemKeyStore` specify the keystore settings, just like `trustCertificateKeyStoreUrl`, `trustCertificateKeyStoreType`, `trustCertificateKeyStorePassword`, and `fallbackToSystemTKeyStore` do for MySQL-protocol connections; if not explicitly set, `xdevapi.ssl-keystore`, `xdevapi.ssl-keystore-type`, `xdevapi.ssl-keystore-password`, and `xdevapi.ssl-fallbackToSystemKeyStore` take up the values of `clientCertificateKeyStoreUrl`, `clientCertificateKeyStoreType`, `clientCertificateKeyStorePassword`, and `fallbackToSystemKeyStore` respectively.

6.9.3 Setting up 2-Way Authentication

Apply the steps outlined in both Section 6.9.1, “Setting up Server Authentication” and Section 6.9.2, “Setting up Client Authentication” to set up a mutual, two-way authentication process in which the server and the client authenticate each other before establishing a connection.

Although the typical setup described above uses the same CA certificate in both ends for mutual authentication, it does not have to be the case. The only requirements are that the CA certificate configured in the server must be able to validate the client certificate and the CA certificate imported into the client truststore must be able to validate the server certificate; the two CA certificates used on the two ends can be distinct.

6.9.4 JSSE in FIPS Mode

When using a Java 8 to 12 JREs, if JSSE is configured to use FIPS mode, attempts to connect to a MySQL Server may fail in some cases with a `KeyManagementException`, complaining that "FIPS mode: only SunJSSE `TrustManagers` may be used." This happens because, in that case, a custom

`TrustManager` implemented by Connector/J that supports the different `sslMode` options is invoked but is eventually rejected by the default implementation of SunJSSE.

The issue can be overcome by telling Connector/J not to use its custom `TrustManager` implementation, but use your own security providers instead. This can be done by setting the following connection properties:

- `fipsCompliantJsse`: Set to `true` to overcome the above-mentioned issue with FIPS mode.

  > **Note**
  >
  > When set to true, Connector/J always performs server certificate validation (even if `sslMode` is set to `PREFERRED` or `REQUIRED`), which means a truststore must be configured with the connection proprieties described below, or the fallback system-wide truststore must be enabled.

- `KeyManagerFactoryProvider`: The name of the a Java Security Provider that provides a `javax.net.ssl.KeyManagerFactory` implementation.
- `trustManagerFactoryProvider`: The name of the a Java Security Provider that provides a `javax.net.ssl.TrustManagerFactory` implementation.
- `keyStoreProvider`: The name of the a Java Security Provider that provides a `java.security.KeyStore` implementation, supporting the key stores types specified with `clientCertificateKeyStoreType` and `trustCertificateKeyStoreType`.

6.9.5 Debugging an SSL Connection

JSSE provides debugging information to `stdout` when you set the system property `-Djavax.net.debug=all`. Java then tells you what keystores and truststores are being used, as well as what is going on during the SSL handshake and certificate exchange. That will be helpful when you are trying to debug a failed SSL connection.

6.10 Connecting Using Unix Domain Sockets

Connector/J does not natively support connections to MySQL Servers with Unix domain sockets. However, there is provision for using 3rd-party libraries that supply the function via a pluggable socket factory. Such a custom factory should implement the `com.mysql.cj.protocol.SocketFactory` interface or the legacy `com.mysql.jdbc.SocketFactory` interface of Connector/J. Follow these requirements when you use such a custom socket factory for Unix sockets :

- The MySQL Server must be configured with the system variable `--socket` (for native protocol connections using the JDBC API) or `--mysqlx-socket` (for X Protocol connections using the X DevAPI), which must contain the file path of the Unix socket file.
- The fully-qualified class name of the custom factory should be passed to Connector/J via the connection property `socketFactory`. For example, with the junixsocket library, set:

  ```
  socketFactory=org.newsclub.net.mysql.AFUNIXDatabaseSocketFactory
  ```

  You might also need to pass other parameters to the custom factory as connection properties. For example, for the junixsocket library, provide the file path of the socket file with the property `junixsocket.file`:

  ```
  junixsocket.file=path_to_socket_file
  ```

6.11 Connecting Using Named Pipes

> **Important**
>
> Minimal permissions on named pipes are granted to clients that use them to connect to the server. Connector/J, however, can only use named pipes when granted full access on them. As a workaround, the MySQL Server that Connector/J wants to connect to must be started with the system variable `named_pipe_full_access_group`, which specifies a Windows local group containing the user by which the client application JVM (and thus Connector/J) is being executed; see the description for `named_pipe_full_access_group` for more details.

> **Note**
>
> Support for named pipes is not available for X Protocol connections.

Connector/J also supports access to MySQL using named pipes on Windows platforms with the `NamedPipeSocketFactory` as a plugin-sockets factory. If you do not use a `namedPipePath` property, the default of `'\\.\pipe\MySQL'` is used. If you use the `NamedPipeSocketFactory`, the host name and port number values in the JDBC URL are ignored. To enable this feature, set the `socketFactory` property:

```
socketFactory=com.mysql.cj.protocol.NamedPipeSocketFactory
```

Set this property, as well as the path of the named pipe, with the following connection URL:

```
jdbc:mysql:///test?socketFactory=com.mysql.cj.protocol.NamedPipeSocketFactory&namedPipePath=\\.\pipe\MySQL
```

To create your own socket factories, follow the sample code in `com.mysql.cj.protocol.NamedPipeSocketFactory` or `com.mysql.cj.protocol.StandardSocketFactory`.

An alternate approach is to use the following two properties in connection URLs for establishing named pipe connections on Windows platforms:

- `(protocol=pipe)` for named pipes (default value for the property is `tcp`).
- `(path=`*`path_to_pipe`*`)` for path of named pipes. Default value for the path is `\\.\pipe\MySQL`.

The “address-equals” or “key-value” form of host specification (see Single host [22] for details) greatly simplifies the URL for a named pipe connection on Windows. For example, to use the default named pipe of “`\\.\pipe\MySQL`,” just specify:

```
jdbc:mysql://address=(protocol=pipe)/test
```

To use the custom named pipe of “`\\.\pipe\MySQL80`” :

```
jdbc:mysql://address=(protocol=pipe)(path=\\.\pipe\MySQL80)/test
```

With `(protocol=pipe)`, the `NamedPipeSocketFactory` is automatically selected.

Named pipes only work when connecting to a MySQL server on the same physical machine where the JDBC driver is running. In simple performance tests, named pipe access is between 30%-50% faster than the standard TCP/IP access. However, this varies per system, and named pipes are slower than TCP/IP in many Windows configurations.

6.12 Connecting Using Various Authentication Methods

6.12.1 Connecting Using PAM Authentication

Java applications using Connector/J can connect to MySQL servers that use the pluggable authentication module (PAM) authentication scheme.

For PAM authentication to work, you must have the following:

- A MySQL server that supports PAM authentication. See PAM Pluggable Authentication for more information. Connector/J implements the same cleartext authentication method as in Client-Side Cleartext Pluggable Authentication.
- SSL capability, as explained in Section 6.9, “Connecting Securely Using SSL”. Because the PAM authentication scheme sends the original password to the server, the connection to the server must be encrypted.

PAM authentication support is enabled by default in Connector/J 9.0, so no extra configuration is needed.

To disable the PAM authentication feature, specify `mysql_clear_password` (the method) or `com.mysql.cj.protocol.a.authentication.MysqlClearPasswordPlugin` (the class name) in the comma-separated list of arguments for the `disabledAuthenticationPlugins` connection option. See Section 6.3, “Configuration Properties” for details about that connection option.

6.12.2 Connecting Using Kerberos

Kerberos is a ticket-based server-client mutual authentication protocol that is supported by the MySQL Server (commercial versions only) .

Support for Kerberos is implemented by Connector/J using the GSS-API, JAAS API, and JCA API; providers for each of these APIs must be available on the Java Virtual Machine running your application that uses Kerberos authentication. Using non-default providers can lead to unexpected results.

Kerberos Authentication Workflow

The main usage of Kerberos authentication in MySQL is to allow users to create connections without having to specify a user name and password in the connection string. For that to work, Connector/J must be configured with the connection property setting `defaultAuthenticationPlugin=authentication_kerberos_client` and then the MySQL user name may be extracted from the Kerberos principal associated to the locally cached Ticket-Granting Ticket (TGT). Notice that a MySQL user name differs from a Kerberos principal in not containing a realm part; therefore, Connector/J cuts all the characters in the principle after the “@” sign and uses it as the MySQL user name.

If there is no TGT available in the local Kerberos cache, Connector/J uses the OS login user name as the MySQL user name. A user name specified in the connection string always takes precedence over names obtained by any other means for the MySQL user.

The MySQL user name is then sent to the MySQL server for validation. Non-existing users cause the server to return an error. Existing users are allowed to proceed with the authentication process, and the authentication mechanism that follows depends on how the MySQL user was created:

- For users created with the authentication plugin `authentication_kerberos`, MySQL server sends the corresponding Kerberos realm back to Connector/J, which, in turn, uses it to construct the Kerberos principal that identifies the user on the Kerberos server. One of three things may then happen:
  - The newly constructed Kerberos principal matches the Kerberos principal associated to the locally cached TGT; this TGT is then sent to the Kerberos server to obtain the desired MySQL Service Ticket, and the authentication proceeds.

  - The newly constructed Kerberos principal does not match the Kerberos principal associated to the locally cached TGT, or there is no local Kerberos cache; this Kerberos principal, as well as the password that may have been specified in the connection string (or an empty string if none was specified), is sent to the Kerberos server to obtain first a valid TGT, and then the desired MySQL Service Ticket; and the authentication proceeds.
  - An error is thrown if Connector/J is unable to obtain the correct Kerberos configurations, unable to communicate with the Kerberos server, or unable to perform either of the two steps above.
- For users defined with a plugin different from `authentication_kerberos`, the server requests Connector/J to use another authentication method.

Client-side Kerberos configurations

In order to operate properly with the Kerberos server, Connector/J requires either a system-wide Kerberos configuration, or these local system property settings for the JVM:

- `-Djava.security.krb5.kdc=`*`[the KDC host name]`*
- `-Djava.security.krb5.realm=`*`[the default Kerberos realm]`*

Debug Information

The process of configuring Connector/J to use Kerberos authentication is not always straightforward. Enabling logging in the internal Java providers can help find potential problems. That can be done by setting these system properties:

- `-Dsun.security.krb5.debug=true`
- `-Dsun.security.jgss.debug=true`

6.12.3 Connecting Using Multifactor Authentication

Multifactor authentication (MFA) is the use of multiple authentication factors during an authentication process. MySQL Server supports MFA for up to three authentication factors.

Connection to MySQL Server with MFA is supported by Connector/J. When authenticating user accounts that require multiple passwords, up to three passwords can be specified using the Connector/J connection properties `password1`, `password2`, and `password3` . This is a sample connection string that uses the three connection properties for passwords:

```
jdbc:mysql://localhost/db?user=johndoe&password1=password&passsword2=password&password3=password
```

The following apply when using the connection properties for passwords:

- `password1`, `password2`, and `password3` are passwords for authentication factors 1, 2, and 3, respectively, as described in Getting Started with Multifactor Authentication.
- If any of the authentication factors (say, factor *`N`*) does not require a password, the corresponding password (`password`*`N`*) is ignored, even if supplied.
- Not specifying the corresponding password for an authentication factor that requires a password is equivalent to supplying an empty password for the factor.
- password and `password1` are taken as synonyms except when both are supplied, in which case `password1` overrides password.

6.12.4 Connecting Using Web Authentication (WebAuthn) Authentication

Web Authentication (WebAuthn) enables user authentication for MySQL Server using devices such as smart cards, security keys, and biometric readers. WebAuthn enables passwordless authentication, and can be used for MySQL accounts that use multifactor authentication. It is supported by MySQL Enterprise Edition and Connector/J since release 8.2.0—see WebAuthn Pluggable Authentication for details.

The following explains how to use WebAuthn authentication with Connector/J. It assumes there is a MySQL server running and configured to support WebAuthn authentication, with the authentication plugin `authentication_webauthn` loaded and the system variable `authentication_webauthn_rp_id` properly configured. Although not always the case, FIDO authentication often works with multifactor authentication, so additional configuration might be necessary but, typically, a default MySQL installation is multifactor authentication ready.

Create a MySQL User

Create the MySQL user to be linked to the FIDO device. Use the `mysql` client with a root user:

```
mysql > CREATE USER 'johndoe'@'%' IDENTIFIED WITH caching_sha2_password BY 's3cr3t' AND IDENTIFIED WITH authen
Query OK, 0 rows affected (0,02 sec)
```

Register the FIDO device by the user you just created. This is accomplished by running the `mysql` client on the same system the device is installed, which might require installing the `mysql` client in your working machine or moving the FIDO device to the system where the MySQL Server is running. In either case, issue the following command (additional command options to connect to the right server might be needed):

```
$ mysql --user=johndoe --password1 --register-factor=2
Enter password: <type "s3cr3t">
Please insert FIDO device and follow the instruction.  Depending on the device, you may have to perform gestur
1. Perform gesture action (Skip this step if you are prompted to enter device PIN).
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.2.0-commercial MySQL Enterprise Server - Commercial

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql >
```

Get 3rd-party Dependencies

MySQL Connector/J is a JDBC Type 4 driver, which is a 100% pure Java implementation, However, there is no pure Java library supporting the authentication devices that Connector/J can use. Therefore, developers need to implement the code that handles the interaction with the authentication devices, for which the following 3rd-party libraries are needed.

- The `libfido2` native library, which must be installed in the system where the application will run.
- Some Java bindings, for example Java Native Interface (JNI) or Java Native Access (JNA). In the following example, Java Native Access (JNA) is used to implement our minimal Java bindings over the `libfido2` library.

Implement the Native Bindings

Create a simple class (called `FidoAssertion` below) that implements the minimal set of bindings between Java and the `libfido2` native library (consult the `libfido2` manuals if needed):

```
import com.sun.jna.Library;
import com.sun.jna.Native;
import com.sun.jna.Pointer;
import com.sun.jna.PointerType;
import com.sun.jna.ptr.IntByReference;
import com.sun.jna.ptr.PointerByReference;

public class FidoAssertion {

    private interface LibFido2 extends Library {
        public static int FIDO_OK = 0;
        static class FidoAssertT extends PointerType {}
        static class FidoDevInfoT extends PointerType {}
        static class FidoDevT extends PointerType {}
        LibFido2 INSTANCE = Native.load("fido2", LibFido2.class);
        int fido_assert_allow_cred(FidoAssertT assrt, byte[] ptr, int len);
        int fido_assert_authdata_len(FidoAssertT assrt, int idx);
        Pointer fido_assert_authdata_ptr(FidoAssertT assrt, int idx);
        void fido_assert_free(PointerByReference assrt);
        FidoAssertT fido_assert_new();
        int fido_assert_count(FidoAssertT assrt);
        int fido_assert_set_clientdata_hash(FidoAssertT assrt, byte[] ptr, int len);
        int fido_assert_set_rp(FidoAssertT assrt, String id);
        int fido_assert_sig_len(FidoAssertT assrt, int idx);
        Pointer fido_assert_sig_ptr(FidoAssertT assrt, int idx);
        int fido_dev_close(FidoDevT dev);
        void fido_dev_free(PointerByReference dev);
        int fido_dev_get_assert(FidoDevT dev, FidoAssertT assrt, String pin);
        void fido_dev_info_free(PointerByReference devlist, int n);
        int fido_dev_info_manifest(FidoDevInfoT devlist, int ilen, IntByReference olen);
        FidoDevInfoT fido_dev_info_new(int n);
        String fido_dev_info_path(FidoDevInfoT di);
        FidoDevInfoT fido_dev_info_ptr(FidoDevInfoT devList, int size);
        FidoDevT fido_dev_new();
        int fido_dev_open(FidoDevT dev, String path);
        boolean fido_dev_supports_credman(FidoDevT dev);
        void fido_init(int flags);
    }

    private LibFido2.FidoAssertT fidoAssert;
    private LibFido2.FidoDevT fidoDev;
    private byte[] clientDataHash;
    private String relyingPartyId;
    private byte[] credentialId;
    private boolean supportsCredMan = false;

    public FidoAssertion() {
        LibFido2.INSTANCE.fido_init(0);
        initializeFidoDevice();
    }

    private void initializeFidoDevice() {
        LibFido2.FidoDevInfoT fidoDevInfo = LibFido2.INSTANCE.fido_dev_info_new(1);
        IntByReference olen = new IntByReference();
        int r = LibFido2.INSTANCE.fido_dev_info_manifest(fidoDevInfo, 1, olen);
        if (r != LibFido2.FIDO_OK) {
            throw new RuntimeException("Failed locating FIDO devices.");
        }
        LibFido2.FidoDevInfoT dev = LibFido2.INSTANCE.fido_dev_info_ptr(fidoDevInfo, 0);
        String path = LibFido2.INSTANCE.fido_dev_info_path(dev);
```

```
        LibFido2.INSTANCE.fido_dev_info_free(new PointerByReference(fidoDevInfo.getPointer()), 1);

        this.fidoDev = LibFido2.INSTANCE.fido_dev_new();
        r = LibFido2.INSTANCE.fido_dev_open(this.fidoDev, path);
        if (r != LibFido2.FIDO_OK) {
            throw new RuntimeException("Failed opening the FIDO device.");
        }

        this.supportsCredMan = LibFido2.INSTANCE.fido_dev_supports_credman(this.fidoDev);
    }

    boolean supportsCredentialManagement() {
        return this.supportsCredMan;
    }

    void setClienDataHash(byte[] clientDataHash) {
        this.clientDataHash = clientDataHash;
    }

    void setRelyingPartyId(String relyingPartyId) {
        this.relyingPartyId = relyingPartyId;
    }

    void setCredentialId(byte[] credentialId) {
        this.credentialId = credentialId;
    }

    void computeAssertions() {
        int r;
        this.fidoAssert = LibFido2.INSTANCE.fido_assert_new();

        // Set the Relying Party Id.
        r = LibFido2.INSTANCE.fido_assert_set_rp(this.fidoAssert, this.relyingPartyId);
        if (r != LibFido2.FIDO_OK) {
            throw new RuntimeException("Failed setting the relying party id.");
        }

        // Set the Client Data Hash.
        r = LibFido2.INSTANCE.fido_assert_set_clientdata_hash(this.fidoAssert, this.clientDataHash, this.clien
        if (r != LibFido2.FIDO_OK) {
            throw new RuntimeException("Failed setting the client data hash.");
        }

        // Set the Credential Id. Not applicable when resident keys are used.
        if (this.credentialId.length > 0) {
            r = LibFido2.INSTANCE.fido_assert_allow_cred(this.fidoAssert, this.credentialId, this.credentialId
            if (r != LibFido2.FIDO_OK) {
                throw new RuntimeException("Failed setting the credential id.");
            }
        }

        // Obtain the assertion(s) from the FIDO device.
        r = LibFido2.INSTANCE.fido_dev_get_assert(this.fidoDev, this.fidoAssert, null);
        if (r != LibFido2.FIDO_OK) {
            throw new RuntimeException("Failed obtaining the assertion(s) from the FIDO device.");
        }
    }

    public int getAssertCount() {
        int assertCount = LibFido2.INSTANCE.fido_assert_count(this.fidoAssert);
        return assertCount;
    }

    public byte[] getAuthenticatorData(int idx) {
        int authDataLen = LibFido2.INSTANCE.fido_assert_authdata_len(this.fidoAssert, idx);
        Pointer authData = LibFido2.INSTANCE.fido_assert_authdata_ptr(this.fidoAssert, idx);
        byte[] authenticatorData = authData.getByteArray(0, authDataLen);
```

```
        return authenticatorData;
    }

    public byte[] getSignature(int idx) {
        int sigLen = LibFido2.INSTANCE.fido_assert_sig_len(this.fidoAssert, idx);
        Pointer sigData = LibFido2.INSTANCE.fido_assert_sig_ptr(this.fidoAssert, idx);
        byte[] signature = sigData.getByteArray(0, sigLen);
        return signature;
    }

    public void freeResources() {
        LibFido2.INSTANCE.fido_dev_close(this.fidoDev);
        LibFido2.INSTANCE.fido_dev_free(new PointerByReference(this.fidoDev.getPointer()));
        LibFido2.INSTANCE.fido_assert_free(new PointerByReference(this.fidoAssert.getPointer()));
    }
}
```

Compile the class with a Java 8 compiler (or above).

```
$ javac -classpath *:. FidoAssertion.java
```

Implement the Authentication Callback

MySQL Connector/J uses a pluggable callback class that exchanges data between the authentication process and the interaction with the authentication device. This class must be an instance of the interface `com.mysql.cj.callback.MysqlCallbackHandler`, which defines one single method: `void handle(MysqlCallback cb);`. The `MysqlCallback` argument this method takes is an instance of `com.mysql.cj.callback.WebAuthnAuthenticationCallback` and it contains all the data required by the FIDO assertion code implemented earlier. Likewise, it also takes the output from the FIDO device (authenticator data and signatures) to the running authentication process.

Here is one possible implementation of the `WebAuthnAuthenticationCallback`.

```
import com.mysql.cj.callback.MysqlCallback;
import com.mysql.cj.callback.MysqlCallbackHandler;
import com.mysql.cj.callback.WebAuthnAuthenticationCallback;

public class AuthenticationWebAuthnCallbackHandler implements MysqlCallbackHandler {
    @Override
    public void handle(MysqlCallback cb) {
        if (!WebAuthnAuthenticationCallback.class.isAssignableFrom(cb.getClass())) {
            return;
        }

        WebAuthnAuthenticationCallback webAuthnAuthCallback = (WebAuthnAuthenticationCallback) cb;

        FidoAssertion libFido2Assertion = new FidoAssertion();
        webAuthnAuthCallback.setSupportsCredentialManagement(libFido2Assertion.supportsCredentialManagement

        libFido2Assertion.setClienDataHash(webAuthnAuthCallback.getClientDataHash());
        libFido2Assertion.setRelyingPartyId(webAuthnAuthCallback.getRelyingPartyId());
        libFido2Assertion.setCredentialId(webAuthnAuthCallback.getCredentialId());

        System.out.println("Please perform the gesture action on your FIDO device.");
        libFido2Assertion.computeAssertions();

        for (int i = 0; i < libFido2Assertion.getAssertCount(); i++) {
            webAuthnAuthCallback.addAuthenticatorData(libFido2Assertion.getAuthenticatorData(i));
            webAuthnAuthCallback.addSignature(libFido2Assertion.getSignature(i));
        }

        libFido2Assertion.freeResources();
    }
}
```

Notice how this implementation is responsible for asking the user to perform the gesture action. In a real use case, this would eventually trigger an event that would, for example, open a pop-up message to the user.

Compile this code:

```
$ javac -classpath *:.  AuthenticationWebAuthnCallbackHandler.java
```

The name of this class must be supplied to Connector/J through the connection property `authenticationWebAuthnCallbackHandler`.

Implement the Application

Implement the client application. The following implementation is just a proof of concept that creates a MySQL connection to the MySQL server with the user created earlier and checks if the connection was established successfully. Notice that FIDO authentication requires some sort of human interactions, so this is not a solution to apply for a typical three-tier architecture, where there is usually a single database user configured in the application server and connections to the database are established from a remote machine.

Here is a simple client application code:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.Properties;

import com.mysql.cj.conf.PropertyKey;

public class AuthenticationWebAuthnApp {
    private static final String HOST = "localhost";
    private static final String PORT = "3306";
    private static final String USER = "johndoe";
    private static final String PASS = "s3cr3t";

    public static void main(String[] args) throws Exception {
        Properties props = new Properties();
        props.setProperty(PropertyKey.authenticationWebAuthnCallbackHandler.getKeyName(), AuthenticationWebAut

        String url = "jdbc:mysql://" + USER + ":" + PASS + "@" + HOST + ":" + PORT + "/";

        try (Connection conn = DriverManager.getConnection(url, props)) {
            ResultSet rs = conn.createStatement().executeQuery("SELECT CURRENT_USER()");
            rs.next();
            System.out.println(rs.getString(1) + " AUTHENTICATED SUCCESSFULLY!");
        }
    }
}
```

Compile the code:

```
$ javac -classpath *:.  AuthenticationWebAuthnApp.java
```

Run the code:

```
$ /usr/lib/jvm/jdk-17/bin/java -classpath *:. AuthenticationWebAuthnApp
Please perform the gesture action on your FIDO device.
johndoe@% AUTHENTICATED SUCCESSFULLY!
```

6.12.5 Connecting Using OpenID Connect Authentication

OpenID Connect is an authentication protocol based on the OAuth 2.0 framework, providing a simplified and interoperable ways of authentication for enhanced security. It is supported by MySQL Enterprise Edition 9.1.0 and later.

Connector/J supports authentication for users created on a MySQL server using the `authentication_openid_connect` plugin. The authentication requires:

- The connection to the server must be secure by SSL encryption.
- An Identity Token that must be obtained from some external process and provided to Connector/J through a customizable callback handler.
- The callback handler must be an implementation of the Connector/J interface `com.mysql.cj.callback.MysqlCallbackHandler`.
- The class name of the callback handler must be provided to Connector/J via the connection property `authenticationOpenidConnectCallbackHandler`.

Default Implementation

Connector/J provides a default implementation of the callback handler named `com.mysql.cj.callback.MysqlCallbackHandler`, and that name is the default value of the connection property `authenticationOpenidConnectCallbackHandler`. This implementation requires an Identity Token file, and the following must be true for it:

- The absolute path of the file is provided to Connector/J through the connection property `idTokenFile`.
- The file specified by `idTokenFile` must exist and must be readable during runtime, or authentication will fail.
- The file must be 10K or smaller in size, or it will be taken as an invalid file.

If no server user name is specified in the connection string or during the creation of the `Connection` object, the implementation takes the OS user name as the user to be authenticated with the server.

6.13 Using Source/Replica Replication with ReplicationConnection

See Section 9.4, “Configuring Source/Replica Replication with Connector/J” for details on the topic.

6.14 Support for DNS SRV Records

Connector/J supports the use of DNS SRV records for connections. For information about DNS SRV support in MySQL, see Connecting to the Server Using DNS SRV Records.

When multiple MySQL instances provide the same service for your applications, DNS SRV records can be used to provide failover, load balancing, and replication services. They eliminate the need for clients to identify each possible host in the connection string, or for connections to be handled by an additional software component. Here is a summary for Connector/J's support for DNS SRV records:

- These new schemas in the connection URLs enable DNS SRV record support:
  - `jdbc:mysql+srv:` For ordinary and basic failover JDBC connections that make use of DNS SRV records.
  - `jdbc:mysql+srv:loadbalance:` For load-balancing JDBC connections that make use of DNS SRV records.

  - `jdbc:mysql+srv:replication:` For replication JDBC connections that make use of DNS SRV records.
  - `mysqlx+srv:` For X DevAPI connections that make use of DNS SRV records.
- Besides using the new schemas in the connection URLs, DNS SRV record support can be enabled or disabled using the two new connection properties, `dnsSrv` and `xdevapi.dns-srv`, for JDBC and X DevAPI connections respectively. For example, this connection URL enables DNS SRV record support:

```
mysqlx://johndoe:secret@_mysql._tcp.mycompany.local/db?xdevapi.dns-srv=true
```

  However, using the DNS SRV schema with the DNS SRV connection properties set to `false` results in an error; for example:

```
mysqlx+srv://johndoe:secret@_mysql._tcp.mycompany.local/db?xdevapi.dns-srv=false
# The connection URL causes Connector/J to throw an error
```

Here are some requirements and restrictions on the DNS SRV record support by Connector/J:

- Connector/J throws an exception if multiple hosts are specified in the connection URL for a DNS SRV connection (except for a replication set up, created using `jdbc:mysql+srv:replication`, which requires exactly one source and one replica server to be specified).
- Connector/J throws an exception if a port number is specified in the connection URL for a DNS SRV connection.
- DNS SRV records are supported only for TCP/IP connections. Connector/J throws an exception if you attempt to enable DNS SRV record support Windows named pipe connections.

DNS SRV Record Support for Load Balancing and Failover. For load-balancing and failover connections, Connector/J uses the `priority` field of the DNS SRV records to decide on the priorities for connection attempts for hosts.

DNS SRV Record Support for Connection Pooling. In an X DevAPI connection pooling setup, Connector/J re-queries the DNS SRV records regularly and phases out gracefully any connections whose hosts no longer appear in the records, and readmits the connections into the pool when their hosts reappear in the records.

Looking up DNS SRV Records. It is the users' responsibility to provide a full service host name; Connector/J does not append any prefix nor validate the host name structure. The following are examples of valid service host name patterns:

- `foo.domain.local`
- `_mysql._tcp.foo.domain.local`
- `_mysqlx._tcp.foo.domain.local`
- `_readonly._tcp.foo.domain.local`
- `_readwrite._tcp.foo.domain.local`

See *Connections Using DNS SRV Records* in the X DevAPI User Guide for details.

6.15 Client Session State Tracker

Connector/J can receive information on client session state changes tracked by the server if the tracking has been enabled on the server. The reception of the information is enabled by setting the Connector/J connection property `trackSessionState` to `true` (default value is `false` for the property).

When the function is enabled, information on session state changes received from the server are stored inside the `SessionStateChanges` object, accessible through a `ServerSessionStateController` and its `getSessionStateChanges()` method:

```
ServerSessionStateChanges ssc =
  MysqlConnection.getServerSessionStateController().getSessionStateChanges();
```

In `SessionStateChanges` is a list of `SessoinStateChange` objects, accessible by the `getSessionStateChangesList()` method:

```
List<SessionStateChange> sscList = ssc.getSessionStateChangesList();
```

Each `SessionStateChange` has the fields `type` and `values`, accessible by the `getType()` and `getValues()` methods. The types and their corresponding values are described below:

Table 6.23 SessionStateChange Type and Values

Type	Number of Values in the value List	Values
`SESSION_TRACK_SYSTEM_VARIABLES`	2	The name of the changed system variable and its new value
`SESSION_TRACK_SCHEMA`	1	The new schema name
`SESSION_TRACK_STATE_CHANGE`	1	"1" or "0"
`SESSION_TRACK_GTIDS`	1	List of GTIDs as reported by server
`SESSION_TRACK_TRANSACTION_CHARACTERISTICS`	1	Transaction characteristics statement
`SESSION_TRACK_TRANSACTION_STATE`	1	Transaction state record

Connector/J receives changes only from the most recent OK packet sent by the server. With `getSessionStateChanges()`, some changes returned by the intermediate queries issued by Connector/J could be missed. However, the session state change information can also be received using a `SessionStateChangesListener`, which has to be registered with a `ServerSessionStateController` using the `addSessionStateChangesListener()` method. The following example implements `SessionStateChangesListener` in a class, which also provides a method to print the change information:

```
class SSCListener implements SessionStateChangesListener {
        ServerSessionStateChanges changes = null;

        public void handleSessionStateChanges(ServerSessionStateChanges ch) {
            this.changes = ch;
            for (SessionStateChange change : ch.getSessionStateChangesList()) {
                printChange(change);
            }
        }

        private void printChange(SessionStateChange change) {
            System.out.print(change.getType() + " == > ");
            int pos = 0;
            if (change.getType() == ServerSessionStateController.SESSION_TRACK_SYSTEM_VARIABLES) {
                // There are two values with this change type, the system variable name and its new value
                System.out.print(change.getValues().get(pos++) + "=");
            }
            System.out.println(change.getValues().get(pos));
        }
    }
SessionStateChangesListener listener = new SSCListener();
MysqlConnection.getServerSessionStateController().addSessionStateChangesListener(listener);
```

With a registered `SessionStateChangesListener`, users have access to all intermediate results, though the listener might slow down the delivery of query results. That is because the listener is invoked immediately after the OK packet is consumed by Connector/J, before the `ResultSet` is constructed.

6.16 Mapping MySQL Error Numbers to JDBC SQLState Codes

The table below provides a mapping of the MySQL error numbers to JDBC `SQLState` values.

Table 6.24 Mapping of MySQL Error Numbers to SQLStates

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1002	ER_NO	HY000
1003	ER_YES	HY000
1004	ER_CANT_CREATE_FILE	HY000
1005	ER_CANT_CREATE_TABLE	HY000
1006	ER_CANT_CREATE_DB	HY000
1007	ER_DB_CREATE_EXISTS	HY000
1008	ER_DB_DROP_EXISTS	HY000
1010	ER_DB_DROP_RMDIR	HY000
1012	ER_CANT_FIND_SYSTEM_REC	HY000
1013	ER_CANT_GET_STAT	HY000
1015	ER_CANT_LOCK	HY000
1016	ER_CANT_OPEN_FILE	HY000
1017	ER_FILE_NOT_FOUND	HY000
1018	ER_CANT_READ_DIR	HY000
1020	ER_CHECKREAD	HY000
1022	ER_DUP_KEY	23000
1024	ER_ERROR_ON_READ	HY000
1025	ER_ERROR_ON_RENAME	HY000
1026	ER_ERROR_ON_WRITE	HY000
1027	ER_FILE_USED	HY000
1030	ER_GET_ERRNO	HY000
1031	ER_ILLEGAL_HA	HY000
1032	ER_KEY_NOT_FOUND	HY000
1033	ER_NOT_FORM_FILE	HY000
1034	ER_NOT_KEYFILE	HY000
1035	ER_OLD_KEYFILE	HY000
1036	ER_OPEN_AS_READONLY	HY000
1037	ER_OUTOFMEMORY	HY001
1038	ER_OUT_OF_SORTMEMORY	HY001
1040	ER_CON_COUNT_ERROR	08004

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1041	ER_OUT_OF_RESOURCES	HY000
1042	ER_BAD_HOST_ERROR	08S01
1043	ER_HANDSHAKE_ERROR	08S01
1044	ER_DBACCESS_DENIED_ERROR	42000
1045	ER_ACCESS_DENIED_ERROR	28000
1046	ER_NO_DB_ERROR	3D000
1047	ER_UNKNOWN_COM_ERROR	08S01
1048	ER_BAD_NULL_ERROR	23000
1049	ER_BAD_DB_ERROR	42000
1050	ER_TABLE_EXISTS_ERROR	42S01
1051	ER_BAD_TABLE_ERROR	42S02
1052	ER_NON_UNIQ_ERROR	23000
1053	ER_SERVER_SHUTDOWN	08S01
1054	ER_BAD_FIELD_ERROR	42S22
1055	ER_WRONG_FIELD_WITH_GROUP	42000
1056	ER_WRONG_GROUP_FIELD	42000
1057	ER_WRONG_SUM_SELECT	42000
1058	ER_WRONG_VALUE_COUNT	21S01
1059	ER_TOO_LONG_IDENT	42000
1060	ER_DUP_FIELDNAME	42S21
1061	ER_DUP_KEYNAME	42000
1062	ER_DUP_ENTRY	23000
1063	ER_WRONG_FIELD_SPEC	42000
1064	ER_PARSE_ERROR	42000
1065	ER_EMPTY_QUERY	42000
1066	ER_NONUNIQ_TABLE	42000
1067	ER_INVALID_DEFAULT	42000
1068	ER_MULTIPLE_PRI_KEY	42000
1069	ER_TOO_MANY_KEYS	42000
1070	ER_TOO_MANY_KEY_PARTS	42000
1071	ER_TOO_LONG_KEY	42000
1072	ER_KEY_COLUMN_DOES_NOT_EXITS	42000
1073	ER_BLOB_USED_AS_KEY	42000
1074	ER_TOO_BIG_FIELDLENGTH	42000
1075	ER_WRONG_AUTO_KEY	42000
1076	ER_READY	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1079	ER_SHUTDOWN_COMPLETE	HY000
1080	ER_FORCING_CLOSE	08S01
1081	ER_IPSOCK_ERROR	08S01
1082	ER_NO_SUCH_INDEX	42S12
1083	ER_WRONG_FIELD_TERMINATORS	42000
1084	ER_BLOBS_AND_NO_TERMINATED	42000
1085	ER_TEXTFILE_NOT_READABLE	HY000
1086	ER_FILE_EXISTS_ERROR	HY000
1087	ER_LOAD_INFO	HY000
1088	ER_ALTER_INFO	HY000
1089	ER_WRONG_SUB_KEY	HY000
1090	ER_CANT_REMOVE_ALL_FIELDS	42000
1091	ER_CANT_DROP_FIELD_OR_KEY	42000
1092	ER_INSERT_INFO	HY000
1093	ER_UPDATE_TABLE_USED	HY000
1094	ER_NO_SUCH_THREAD	HY000
1095	ER_KILL_DENIED_ERROR	HY000
1096	ER_NO_TABLES_USED	HY000
1097	ER_TOO_BIG_SET	HY000
1098	ER_NO_UNIQUE_LOGFILE	HY000
1099	ER_TABLE_NOT_LOCKED_FOR_WRITE	HY000
1100	ER_TABLE_NOT_LOCKED	HY000
1101	ER_BLOB_CANT_HAVE_DEFAULT	42000
1102	ER_WRONG_DB_NAME	42000
1103	ER_WRONG_TABLE_NAME	42000
1104	ER_TOO_BIG_SELECT	42000
1105	ER_UNKNOWN_ERROR	HY000
1106	ER_UNKNOWN_PROCEDURE	42000
1107	ER_WRONG_PARAMCOUNT_TO_PROCEDURE	42000
1108	ER_WRONG_PARAMETERS_TO_PROCEDURE	HY000
1109	ER_UNKNOWN_TABLE	42S02
1110	ER_FIELD_SPECIFIED_TWICE	42000
1111	ER_INVALID_GROUP_FUNC_USE	HY000
1112	ER_UNSUPPORTED_EXTENSION	42000
1113	ER_TABLE_MUST_HAVE_COLUMNS	42000
1114	ER_RECORD_FILE_FULL	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1115	ER_UNKNOWN_CHARACTER_SET	42000
1116	ER_TOO_MANY_TABLES	HY000
1117	ER_TOO_MANY_FIELDS	HY000
1118	ER_TOO_BIG_ROWSIZE	42000
1119	ER_STACK_OVERRUN	HY000
1120	ER_WRONG_OUTER_JOIN_UNUSED	42000
1121	ER_NULL_COLUMN_IN_INDEX	42000
1122	ER_CANT_FIND_UDF	HY000
1123	ER_CANT_INITIALIZE_UDF	HY000
1124	ER_UDF_NO_PATHS	HY000
1125	ER_UDF_EXISTS	HY000
1126	ER_CANT_OPEN_LIBRARY	HY000
1127	ER_CANT_FIND_DL_ENTRY	HY000
1128	ER_FUNCTION_NOT_DEFINED	HY000
1129	ER_HOST_IS_BLOCKED	HY000
1130	ER_HOST_NOT_PRIVILEGED	HY000
1131	ER_PASSWORD_ANONYMOUS_USER	42000
1132	ER_PASSWORD_NOT_ALLOWED	42000
1133	ER_PASSWORD_NO_MATCH	42000
1134	ER_UPDATE_INFO	HY000
1135	ER_CANT_CREATE_THREAD	HY000
1136	ER_WRONG_VALUE_COUNT_ON_ROW	21S01
1137	ER_CANT_REOPEN_TABLE	HY000
1138	ER_INVALID_USE_OF_NULL	22004
1139	ER_REGEXP_ERROR	42000
1140	ER_MIX_OF_GROUP_FUNC_AND_FIELDS	42000
1141	ER_NONEXISTING_GRANT	42000
1142	ER_TABLEACCESS_DENIED_ERROR	42000
1143	ER_COLUMNACCESS_DENIED_ERROR	42000
1144	ER_ILLEGAL_GRANT_FOR_TABLE	42000
1145	ER_GRANT_WRONG_HOST_OR_USER	42000
1146	ER_NO_SUCH_TABLE	42S02
1147	ER_NONEXISTING_TABLE_GRANT	42000
1148	ER_NOT_ALLOWED_COMMAND	42000
1149	ER_SYNTAX_ERROR	42000
1152	ER_ABORTING_CONNECTION	08S01

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1153	ER_NET_PACKET_TOO_LARGE	08S01
1154	ER_NET_READ_ERROR_FROM_PIPE	08S01
1155	ER_NET_FCNTL_ERROR	08S01
1156	ER_NET_PACKETS_OUT_OF_ORDER	08S01
1157	ER_NET_UNCOMPRESS_ERROR	08S01
1158	ER_NET_READ_ERROR	08S01
1159	ER_NET_READ_INTERRUPTED	08S01
1160	ER_NET_ERROR_ON_WRITE	08S01
1161	ER_NET_WRITE_INTERRUPTED	08S01
1162	ER_TOO_LONG_STRING	42000
1163	ER_TABLE_CANT_HANDLE_BLOB	42000
1164	ER_TABLE_CANT_HANDLE_AUTO_INCREMENT	42000
1166	ER_WRONG_COLUMN_NAME	42000
1167	ER_WRONG_KEY_COLUMN	42000
1168	ER_WRONG_MRG_TABLE	HY000
1169	ER_DUP_UNIQUE	23000
1170	ER_BLOB_KEY_WITHOUT_LENGTH	42000
1171	ER_PRIMARY_CANT_HAVE_NULL	42000
1172	ER_TOO_MANY_ROWS	42000
1173	ER_REQUIRES_PRIMARY_KEY	42000
1175	ER_UPDATE_WITHOUT_KEY_IN_SAFE_MODE	HY000
1176	ER_KEY_DOES_NOT_EXITS	42000
1177	ER_CHECK_NO_SUCH_TABLE	42000
1178	ER_CHECK_NOT_IMPLEMENTED	42000
1179	ER_CANT_DO_THIS_DURING_AN_TRANSACTION	25000
1180	ER_ERROR_DURING_COMMIT	HY000
1181	ER_ERROR_DURING_ROLLBACK	HY000
1182	ER_ERROR_DURING_FLUSH_LOGS	HY000
1184	ER_NEW_ABORTING_CONNECTION	08S01
1188	ER_SOURCE	HY000
1189	ER_SOURCE_NET_READ	08S01
1190	ER_SOURCE_NET_WRITE	08S01
1191	ER_FT_MATCHING_KEY_NOT_FOUND	HY000
1192	ER_LOCK_OR_ACTIVE_TRANSACTION	HY000
1193	ER_UNKNOWN_SYSTEM_VARIABLE	HY000
1194	ER_CRASHED_ON_USAGE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1195	ER_CRASHED_ON_REPAIR	HY000
1196	ER_WARNING_NOT_COMPLETE_ROLLBACK	HY000
1197	ER_TRANS_CACHE_FULL	HY000
1199	ER_REPLICA_NOT_RUNNING	HY000
1200	ER_BAD_REPLICA	HY000
1201	ER_CONNECTION_METADATA	HY000
1202	ER_REPLICA_THREAD	HY000
1203	ER_TOO_MANY_USER_CONNECTIONS	42000
1204	ER_SET_CONSTANTS_ONLY	HY000
1205	ER_LOCK_WAIT_TIMEOUT	40001
1206	ER_LOCK_TABLE_FULL	HY000
1207	ER_READ_ONLY_TRANSACTION	25000
1210	ER_WRONG_ARGUMENTS	HY000
1211	ER_NO_PERMISSION_TO_CREATE_USER	42000
1213	ER_LOCK_DEADLOCK	40001
1214	ER_TABLE_CANT_HANDLE_FT	HY000
1215	ER_CANNOT_ADD_FOREIGN	HY000
1216	ER_NO_REFERENCED_ROW	23000
1217	ER_ROW_IS_REFERENCED	23000
1218	ER_CONNECT_TO_SOURCE	08S01
1220	ER_ERROR_WHEN_EXECUTING_COMMAND	HY000
1221	ER_WRONG_USAGE	HY000
1222	ER_WRONG_NUMBER_OF_COLUMNS_IN_SELECT	21000
1223	ER_CANT_UPDATE_WITH_READLOCK	HY000
1224	ER_MIXING_NOT_ALLOWED	HY000
1225	ER_DUP_ARGUMENT	HY000
1226	ER_USER_LIMIT_REACHED	42000
1227	ER_SPECIFIC_ACCESS_DENIED_ERROR	42000
1228	ER_LOCAL_VARIABLE	HY000
1229	ER_GLOBAL_VARIABLE	HY000
1230	ER_NO_DEFAULT	42000
1231	ER_WRONG_VALUE_FOR_VAR	42000
1232	ER_WRONG_TYPE_FOR_VAR	42000
1233	ER_VAR_CANT_BE_READ	HY000
1234	ER_CANT_USE_OPTION_HERE	42000
1235	ER_NOT_SUPPORTED_YET	42000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1236	ER_SOURCE_FATAL_ERROR_READING_BINLOG	HY000
1237	ER_REPLICA_IGNORED_TABLE	HY000
1238	ER_INCORRECT_GLOBAL_LOCAL_VAR	HY000
1239	ER_WRONG_FK_DEF	42000
1240	ER_KEY_REF_DO_NOT_MATCH_TABLE_REF	HY000
1241	ER_OPERAND_COLUMNS	21000
1242	ER_SUBQUERY_NO_1_ROW	21000
1243	ER_UNKNOWN_STMT_HANDLER	HY000
1244	ER_CORRUPT_HELP_DB	HY000
1246	ER_AUTO_CONVERT	HY000
1247	ER_ILLEGAL_REFERENCE	42S22
1248	ER_DERIVED_MUST_HAVE_ALIAS	42000
1249	ER_SELECT_REDUCED	01000
1250	ER_TABLENAME_NOT_ALLOWED_HERE	42000
1251	ER_NOT_SUPPORTED_AUTH_MODE	08004
1252	ER_SPATIAL_CANT_HAVE_NULL	42000
1253	ER_COLLATION_CHARSET_MISMATCH	42000
1256	ER_TOO_BIG_FOR_UNCOMPRESS	HY000
1257	ER_ZLIB_Z_MEM_ERROR	HY000
1258	ER_ZLIB_Z_BUF_ERROR	HY000
1259	ER_ZLIB_Z_DATA_ERROR	HY000
1260	ER_CUT_VALUE_GROUP_CONCAT	HY000
1261	ER_WARN_TOO_FEW_RECORDS	01000
1262	ER_WARN_TOO_MANY_RECORDS	01000
1263	ER_WARN_NULL_TO_NOTNULL	22004
1264	ER_WARN_DATA_OUT_OF_RANGE	22003
1265	WARN_DATA_TRUNCATED	01000
1266	ER_WARN_USING_OTHER_HANDLER	HY000
1267	ER_CANT_AGGREGATE_2COLLATIONS	HY000
1269	ER_REVOKE_GRANTS	HY000
1270	ER_CANT_AGGREGATE_3COLLATIONS	HY000
1271	ER_CANT_AGGREGATE_NCOLLATIONS	HY000
1272	ER_VARIABLE_IS_NOT_STRUCT	HY000
1273	ER_UNKNOWN_COLLATION	HY000
1274	ER_REPLICA_IGNORED_SSL_PARAMS	HY000
1276	ER_WARN_FIELD_RESOLVED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1277	ER_BAD_REPLICA_UNTIL_COND	HY000
1278	ER_MISSING_SKIP_REPLICA	HY000
1279	ER_UNTIL_COND_IGNORED	HY000
1280	ER_WRONG_NAME_FOR_INDEX	42000
1281	ER_WRONG_NAME_FOR_CATALOG	42000
1283	ER_BAD_FT_COLUMN	HY000
1284	ER_UNKNOWN_KEY_CACHE	HY000
1285	ER_WARN_HOSTNAME_WONT_WORK	HY000
1286	ER_UNKNOWN_STORAGE_ENGINE	42000
1287	ER_WARN_DEPRECATED_SYNTAX	HY000
1288	ER_NON_UPDATABLE_TABLE	HY000
1289	ER_FEATURE_DISABLED	HY000
1290	ER_OPTION_PREVENTS_STATEMENT	HY000
1291	ER_DUPLICATED_VALUE_IN_TYPE	HY000
1292	ER_TRUNCATED_WRONG_VALUE	22007
1294	ER_INVALID_ON_UPDATE	HY000
1295	ER_UNSUPPORTED_PS	HY000
1296	ER_GET_ERRMSG	HY000
1297	ER_GET_TEMPORARY_ERRMSG	HY000
1298	ER_UNKNOWN_TIME_ZONE	HY000
1299	ER_WARN_INVALID_TIMESTAMP	HY000
1300	ER_INVALID_CHARACTER_STRING	HY000
1301	ER_WARN_ALLOWED_PACKET_OVERFLOWED	HY000
1302	ER_CONFLICTING_DECLARATIONS	HY000
1303	ER_SP_NO_RECURSIVE_CREATE	2F003
1304	ER_SP_ALREADY_EXISTS	42000
1305	ER_SP_DOES_NOT_EXIST	42000
1306	ER_SP_DROP_FAILED	HY000
1307	ER_SP_STORE_FAILED	HY000
1308	ER_SP_LILABEL_MISMATCH	42000
1309	ER_SP_LABEL_REDEFINE	42000
1310	ER_SP_LABEL_MISMATCH	42000
1311	ER_SP_UNINIT_VAR	01000
1312	ER_SP_BADSELECT	0A000
1313	ER_SP_BADRETURN	42000
1314	ER_SP_BADSTATEMENT	0A000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1315	ER_UPDATE_LOG_DEPRECATED_IGNORED	42000
1316	ER_UPDATE_LOG_DEPRECATED_TRANSLATED	42000
1317	ER_QUERY_INTERRUPTED	70100
1318	ER_SP_WRONG_NO_OF_ARGS	42000
1319	ER_SP_COND_MISMATCH	42000
1320	ER_SP_NORETURN	42000
1321	ER_SP_NORETURNEND	2F005
1322	ER_SP_BAD_CURSOR_QUERY	42000
1323	ER_SP_BAD_CURSOR_SELECT	42000
1324	ER_SP_CURSOR_MISMATCH	42000
1325	ER_SP_CURSOR_ALREADY_OPEN	24000
1326	ER_SP_CURSOR_NOT_OPEN	24000
1327	ER_SP_UNDECLARED_VAR	42000
1328	ER_SP_WRONG_NO_OF_FETCH_ARGS	HY000
1329	ER_SP_FETCH_NO_DATA	02000
1330	ER_SP_DUP_PARAM	42000
1331	ER_SP_DUP_VAR	42000
1332	ER_SP_DUP_COND	42000
1333	ER_SP_DUP_CURS	42000
1334	ER_SP_CANT_ALTER	HY000
1335	ER_SP_SUBSELECT_NYI	0A000
1336	ER_STMT_NOT_ALLOWED_IN_SF_OR_TRG	0A000
1337	ER_SP_VARCOND_AFTER_CURSHNDLR	42000
1338	ER_SP_CURSOR_AFTER_HANDLER	42000
1339	ER_SP_CASE_NOT_FOUND	20000
1340	ER_FPARSER_TOO_BIG_FILE	HY000
1341	ER_FPARSER_BAD_HEADER	HY000
1342	ER_FPARSER_EOF_IN_COMMENT	HY000
1343	ER_FPARSER_ERROR_IN_PARAMETER	HY000
1344	ER_FPARSER_EOF_IN_UNKNOWN_PARAMETER	HY000
1345	ER_VIEW_NO_EXPLAIN	HY000
1347	ER_WRONG_OBJECT	HY000
1348	ER_NONUPDATEABLE_COLUMN	HY000
1350	ER_VIEW_SELECT_CLAUSE	HY000
1351	ER_VIEW_SELECT_VARIABLE	HY000
1352	ER_VIEW_SELECT_TMPTABLE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1353	ER_VIEW_WRONG_LIST	HY000
1354	ER_WARN_VIEW_MERGE	HY000
1355	ER_WARN_VIEW_WITHOUT_KEY	HY000
1356	ER_VIEW_INVALID	HY000
1357	ER_SP_NO_DROP_SP	HY000
1359	ER_TRG_ALREADY_EXISTS	HY000
1360	ER_TRG_DOES_NOT_EXIST	HY000
1361	ER_TRG_ON_VIEW_OR_TEMP_TABLE	HY000
1362	ER_TRG_CANT_CHANGE_ROW	HY000
1363	ER_TRG_NO_SUCH_ROW_IN_TRG	HY000
1364	ER_NO_DEFAULT_FOR_FIELD	HY000
1365	ER_DIVISION_BY_ZERO	22012
1366	ER_TRUNCATED_WRONG_VALUE_FOR_FIELD	HY000
1367	ER_ILLEGAL_VALUE_FOR_TYPE	22007
1368	ER_VIEW_NONUPD_CHECK	HY000
1369	ER_VIEW_CHECK_FAILED	HY000
1370	ER_PROCACCESS_DENIED_ERROR	42000
1371	ER_RELAY_LOG_FAIL	HY000
1373	ER_UNKNOWN_TARGET_BINLOG	HY000
1374	ER_IO_ERR_LOG_INDEX_READ	HY000
1375	ER_BINLOG_PURGE_PROHIBITED	HY000
1376	ER_FSEEK_FAIL	HY000
1377	ER_BINLOG_PURGE_FATAL_ERR	HY000
1378	ER_LOG_IN_USE	HY000
1379	ER_LOG_PURGE_UNKNOWN_ERR	HY000
1380	ER_RELAY_LOG_INIT	HY000
1381	ER_NO_BINARY_LOGGING	HY000
1382	ER_RESERVED_SYNTAX	HY000
1390	ER_PS_MANY_PARAM	HY000
1391	ER_KEY_PART_0	HY000
1392	ER_VIEW_CHECKSUM	HY000
1393	ER_VIEW_MULTIUPDATE	HY000
1394	ER_VIEW_NO_INSERT_FIELD_LIST	HY000
1395	ER_VIEW_DELETE_MERGE_VIEW	HY000
1396	ER_CANNOT_USER	HY000
1397	ER_XAER_NOTA	XAE04

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1398	ER_XAER_INVAL	XAE05
1399	ER_XAER_RMFAIL	XAE07
1400	ER_XAER_OUTSIDE	XAE09
1401	ER_XAER_RMERR	XAE03
1402	ER_XA_RBROLLBACK	XA100
1403	ER_NONEXISTING_PROC_GRANT	42000
1404	ER_PROC_AUTO_GRANT_FAIL	HY000
1405	ER_PROC_AUTO_REVOKE_FAIL	HY000
1406	ER_DATA_TOO_LONG	22001
1407	ER_SP_BAD_SQLSTATE	42000
1408	ER_STARTUP	HY000
1409	ER_LOAD_FROM_FIXED_SIZE_ROWS_TO_VAR	HY000
1410	ER_CANT_CREATE_USER_WITH_GRANT	42000
1411	ER_WRONG_VALUE_FOR_TYPE	HY000
1412	ER_TABLE_DEF_CHANGED	HY000
1413	ER_SP_DUP_HANDLER	42000
1414	ER_SP_NOT_VAR_ARG	42000
1415	ER_SP_NO_RETSET	0A000
1416	ER_CANT_CREATE_GEOMETRY_OBJECT	22003
1418	ER_BINLOG_UNSAFE_ROUTINE	HY000
1419	ER_BINLOG_CREATE_ROUTINE_NEED_SUPER	HY000
1421	ER_STMT_HAS_NO_OPEN_CURSOR	HY000
1422	ER_COMMIT_NOT_ALLOWED_IN_SF_OR_TRG	HY000
1423	ER_NO_DEFAULT_FOR_VIEW_FIELD	HY000
1424	ER_SP_NO_RECURSION	HY000
1425	ER_TOO_BIG_SCALE	42000
1426	ER_TOO_BIG_PRECISION	42000
1427	ER_M_BIGGER_THAN_D	42000
1428	ER_WRONG_LOCK_OF_SYSTEM_TABLE	HY000
1429	ER_CONNECT_TO_FOREIGN_DATA_SOURCE	HY000
1430	ER_QUERY_ON_FOREIGN_DATA_SOURCE	HY000
1431	ER_FOREIGN_DATA_SOURCE_DOESNT_EXIST	HY000
1432	ER_FOREIGN_DATA_STRING_INVALID_CANT_CREATE	HY000
1433	ER_FOREIGN_DATA_STRING_INVALID	HY000
1435	ER_TRG_IN_WRONG_SCHEMA	HY000
1436	ER_STACK_OVERRUN_NEED_MORE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1437	ER_TOO_LONG_BODY	42000
1438	ER_WARN_CANT_DROP_DEFAULT_KEYCACHE	HY000
1439	ER_TOO_BIG_DISPLAYWIDTH	42000
1440	ER_XAER_DUPID	XAE08
1441	ER_DATETIME_FUNCTION_OVERFLOW	22008
1442	ER_CANT_UPDATE_USED_TABLE_IN_SF_OR_TRG	HY000
1443	ER_VIEW_PREVENT_UPDATE	HY000
1444	ER_PS_NO_RECURSION	HY000
1445	ER_SP_CANT_SET_AUTOCOMMIT	HY000
1447	ER_VIEW_FRM_NO_USER	HY000
1448	ER_VIEW_OTHER_USER	HY000
1449	ER_NO_SUCH_USER	HY000
1450	ER_FORBID_SCHEMA_CHANGE	HY000
1451	ER_ROW_IS_REFERENCED_2	23000
1452	ER_NO_REFERENCED_ROW_2	23000
1453	ER_SP_BAD_VAR_SHADOW	42000
1454	ER_TRG_NO_DEFINER	HY000
1455	ER_OLD_FILE_FORMAT	HY000
1456	ER_SP_RECURSION_LIMIT	HY000
1458	ER_SP_WRONG_NAME	42000
1459	ER_TABLE_NEEDS_UPGRADE	HY000
1460	ER_SP_NO_AGGREGATE	42000
1461	ER_MAX_PREPARED_STMT_COUNT_REACHED	42000
1462	ER_VIEW_RECURSIVE	HY000
1463	ER_NON_GROUPING_FIELD_USED	42000
1464	ER_TABLE_CANT_HANDLE_SPKEYS	HY000
1465	ER_NO_TRIGGERS_ON_SYSTEM_SCHEMA	HY000
1466	ER_REMOVED_SPACES	HY000
1467	ER_AUTOINC_READ_FAILED	HY000
1468	ER_USERNAME	HY000
1469	ER_HOSTNAME	HY000
1470	ER_WRONG_STRING_LENGTH	HY000
1471	ER_NON_INSERTABLE_TABLE	HY000
1472	ER_ADMIN_WRONG_MRG_TABLE	HY000
1473	ER_TOO_HIGH_LEVEL_OF_NESTING_FOR_SELECT	HY000
1474	ER_NAME_BECOMES_EMPTY	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1475	ER_AMBIGUOUS_FIELD_TERM	HY000
1476	ER_FOREIGN_SERVER_EXISTS	HY000
1477	ER_FOREIGN_SERVER_DOESNT_EXIST	HY000
1478	ER_ILLEGAL_HA_CREATE_OPTION	HY000
1479	ER_PARTITION_REQUIRES_VALUES_ERROR	HY000
1480	ER_PARTITION_WRONG_VALUES_ERROR	HY000
1481	ER_PARTITION_MAXVALUE_ERROR	HY000
1484	ER_PARTITION_WRONG_NO_PART_ERROR	HY000
1485	ER_PARTITION_WRONG_NO_SUBPART_ERROR	HY000
1486	ER_WRONG_EXPR_IN_PARTITION_FUNC_ERROR	HY000
1488	ER_FIELD_NOT_FOUND_PART_ERROR	HY000
1490	ER_INCONSISTENT_PARTITION_INFO_ERROR	HY000
1491	ER_PARTITION_FUNC_NOT_ALLOWED_ERROR	HY000
1492	ER_PARTITIONS_MUST_BE_DEFINED_ERROR	HY000
1493	ER_RANGE_NOT_INCREASING_ERROR	HY000
1494	ER_INCONSISTENT_TYPE_OF_FUNCTIONS_ERROR	HY000
1495	ER_MULTIPLE_DEF_CONST_IN_LIST_PART_ERROR	HY000
1496	ER_PARTITION_ENTRY_ERROR	HY000
1497	ER_MIX_HANDLER_ERROR	HY000
1498	ER_PARTITION_NOT_DEFINED_ERROR	HY000
1499	ER_TOO_MANY_PARTITIONS_ERROR	HY000
1500	ER_SUBPARTITION_ERROR	HY000
1501	ER_CANT_CREATE_HANDLER_FILE	HY000
1502	ER_BLOB_FIELD_IN_PART_FUNC_ERROR	HY000
1503	ER_UNIQUE_KEY_NEED_ALL_FIELDS_IN_PF	HY000
1504	ER_NO_PARTS_ERROR	HY000
1505	ER_PARTITION_MGMT_ON_NONPARTITIONED	HY000
1506	ER_FOREIGN_KEY_ON_PARTITIONED	HY000
1507	ER_DROP_PARTITION_NON_EXISTENT	HY000
1508	ER_DROP_LAST_PARTITION	HY000
1509	ER_COALESCE_ONLY_ON_HASH_PARTITION	HY000
1510	ER_REORG_HASH_ONLY_ON_SAME_NO	HY000
1511	ER_REORG_NO_PARAM_ERROR	HY000
1512	ER_ONLY_ON_RANGE_LIST_PARTITION	HY000
1513	ER_ADD_PARTITION_SUBPART_ERROR	HY000
1514	ER_ADD_PARTITION_NO_NEW_PARTITION	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1515	ER_COALESCE_PARTITION_NO_PARTITION	HY000
1516	ER_REORG_PARTITION_NOT_EXIST	HY000
1517	ER_SAME_NAME_PARTITION	HY000
1518	ER_NO_BINLOG_ERROR	HY000
1519	ER_CONSECUTIVE_REORG_PARTITIONS	HY000
1520	ER_REORG_OUTSIDE_RANGE	HY000
1521	ER_PARTITION_FUNCTION_FAILURE	HY000
1523	ER_LIMITED_PART_RANGE	HY000
1524	ER_PLUGIN_IS_NOT_LOADED	HY000
1525	ER_WRONG_VALUE	HY000
1526	ER_NO_PARTITION_FOR_GIVEN_VALUE	HY000
1527	ER_FILEGROUP_OPTION_ONLY_ONCE	HY000
1528	ER_CREATE_FILEGROUP_FAILED	HY000
1529	ER_DROP_FILEGROUP_FAILED	HY000
1530	ER_TABLESPACE_AUTO_EXTEND_ERROR	HY000
1531	ER_WRONG_SIZE_NUMBER	HY000
1532	ER_SIZE_OVERFLOW_ERROR	HY000
1533	ER_ALTER_FILEGROUP_FAILED	HY000
1534	ER_BINLOG_ROW_LOGGING_FAILED	HY000
1537	ER_EVENT_ALREADY_EXISTS	HY000
1539	ER_EVENT_DOES_NOT_EXIST	HY000
1542	ER_EVENT_INTERVAL_NOT_POSITIVE_OR_TOO_BIG	HY000
1543	ER_EVENT_ENDS_BEFORE_STARTS	HY000
1544	ER_EVENT_EXEC_TIME_IN_THE_PAST	HY000
1551	ER_EVENT_SAME_NAME	HY000
1553	ER_DROP_INDEX_FK	HY000
1554	ER_WARN_DEPRECATED_SYNTAX_WITH_VER	HY000
1556	ER_CANT_LOCK_LOG_TABLE	HY000
1557	ER_FOREIGN_DUPLICATE_KEY_OLD_UNUSED	23000
1558	ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE	HY000
1560	ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_FORMAT	HY000
1562	ER_PARTITION_NO_TEMPORARY	HY000
1563	ER_PARTITION_CONST_DOMAIN_ERROR	HY000
1564	ER_PARTITION_FUNCTION_IS_NOT_ALLOWED	HY000
1566	ER_NULL_IN_VALUES_LESS_THAN	HY000
1567	ER_WRONG_PARTITION_NAME	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1568	ER_CANT_CHANGE_TX_CHARACTERISTICS	25001
1569	ER_DUP_ENTRY_AUTOINCREMENT_CASE	HY000
1571	ER_EVENT_SET_VAR_ERROR	HY000
1572	ER_PARTITION_MERGE_ERROR	HY000
1575	ER_BASE64_DECODE_ERROR	HY000
1576	ER_EVENT_RECURSION_FORBIDDEN	HY000
1578	ER_ONLY_INTEGERS_ALLOWED	HY000
1579	ER_UNSUPORTED_LOG_ENGINE	HY000
1580	ER_BAD_LOG_STATEMENT	HY000
1581	ER_CANT_RENAME_LOG_TABLE	HY000
1582	ER_WRONG_PARAMCOUNT_TO_NATIVE_FCT	42000
1583	ER_WRONG_PARAMETERS_TO_NATIVE_FCT	42000
1584	ER_WRONG_PARAMETERS_TO_STORED_FCT	42000
1585	ER_NATIVE_FCT_NAME_COLLISION	HY000
1586	ER_DUP_ENTRY_WITH_KEY_NAME	23000
1587	ER_BINLOG_PURGE_EMFILE	HY000
1588	ER_EVENT_CANNOT_CREATE_IN_THE_PAST	HY000
1589	ER_EVENT_CANNOT_ALTER_IN_THE_PAST	HY000
1591	ER_NO_PARTITION_FOR_GIVEN_VALUE_SILENT	HY000
1592	ER_BINLOG_UNSAFE_STATEMENT	HY000
1593	ER_BINLOG_FATAL_ERROR	HY000
1598	ER_BINLOG_LOGGING_IMPOSSIBLE	HY000
1599	ER_VIEW_NO_CREATION_CTX	HY000
1600	ER_VIEW_INVALID_CREATION_CTX	HY000
1602	ER_TRG_CORRUPTED_FILE	HY000
1603	ER_TRG_NO_CREATION_CTX	HY000
1604	ER_TRG_INVALID_CREATION_CTX	HY000
1605	ER_EVENT_INVALID_CREATION_CTX	HY000
1606	ER_TRG_CANT_OPEN_TABLE	HY000
1609	ER_NO_FORMAT_DESCRIPTION_EVENT_BEFORE_BINLOG_STATEMENT	HY000
1610	ER_REPLICA_CORRUPT_EVENT	HY000
1612	ER_LOG_PURGE_NO_FILE	HY000
1613	ER_XA_RBTIMEOUT	XA106
1614	ER_XA_RBDEADLOCK	XA102
1615	ER_NEED_REPREPARE	HY000
1617	WARN_NO_CONNECTION_METADATA	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1618	WARN_OPTION_IGNORED	HY000
1619	ER_PLUGIN_DELETE_BUILTIN	HY000
1620	WARN_PLUGIN_BUSY	HY000
1621	ER_VARIABLE_IS_READONLY	HY000
1622	ER_WARN_ENGINE_TRANSACTION_ROLLBACK	HY000
1624	ER_REPLICA_HEARTBEAT_VALUE_OUT_OF_RANGE	HY000
1625	ER_NDB_REPLICATION_SCHEMA_ERROR	HY000
1626	ER_CONFLICT_FN_PARSE_ERROR	HY000
1627	ER_EXCEPTIONS_WRITE_ERROR	HY000
1628	ER_TOO_LONG_TABLE_COMMENT	HY000
1629	ER_TOO_LONG_FIELD_COMMENT	HY000
1630	ER_FUNC_INEXISTENT_NAME_COLLISION	42000
1631	ER_DATABASE_NAME	HY000
1632	ER_TABLE_NAME	HY000
1633	ER_PARTITION_NAME	HY000
1634	ER_SUBPARTITION_NAME	HY000
1635	ER_TEMPORARY_NAME	HY000
1636	ER_RENAMED_NAME	HY000
1637	ER_TOO_MANY_CONCURRENT_TRXS	HY000
1638	WARN_NON_ASCII_SEPARATOR_NOT_IMPLEMENTED	HY000
1639	ER_DEBUG_SYNC_TIMEOUT	HY000
1640	ER_DEBUG_SYNC_HIT_LIMIT	HY000
1641	ER_DUP_SIGNAL_SET	42000
1642	ER_SIGNAL_WARN	01000
1643	ER_SIGNAL_NOT_FOUND	02000
1644	ER_SIGNAL_EXCEPTION	HY000
1645	ER_RESIGNAL_WITHOUT_ACTIVE_HANDLER	0K000
1646	ER_SIGNAL_BAD_CONDITION_TYPE	HY000
1647	WARN_COND_ITEM_TRUNCATED	HY000
1648	ER_COND_ITEM_TOO_LONG	HY000
1649	ER_UNKNOWN_LOCALE	HY000
1650	ER_REPLICA_IGNORE_SERVER_IDS	HY000
1652	ER_SAME_NAME_PARTITION_FIELD	HY000
1653	ER_PARTITION_COLUMN_LIST_ERROR	HY000
1654	ER_WRONG_TYPE_COLUMN_VALUE_ERROR	HY000
1655	ER_TOO_MANY_PARTITION_FUNC_FIELDS_ERROR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1656	ER_MAXVALUE_IN_VALUES_IN	HY000
1657	ER_TOO_MANY_VALUES_ERROR	HY000
1658	ER_ROW_SINGLE_PARTITION_FIELD_ERROR	HY000
1659	ER_FIELD_TYPE_NOT_ALLOWED_AS_PARTITION_FIELD	HY000
1660	ER_PARTITION_FIELDS_TOO_LONG	HY000
1661	ER_BINLOG_ROW_ENGINE_AND_STMT_ENGINE	HY000
1662	ER_BINLOG_ROW_MODE_AND_STMT_ENGINE	HY000
1663	ER_BINLOG_UNSAFE_AND_STMT_ENGINE	HY000
1664	ER_BINLOG_ROW_INJECTION_AND_STMT_ENGINE	HY000
1665	ER_BINLOG_STMT_MODE_AND_ROW_ENGINE	HY000
1666	ER_BINLOG_ROW_INJECTION_AND_STMT_MODE	HY000
1667	ER_BINLOG_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE	HY000
1668	ER_BINLOG_UNSAFE_LIMIT	HY000
1670	ER_BINLOG_UNSAFE_SYSTEM_TABLE	HY000
1671	ER_BINLOG_UNSAFE_AUTOINC_COLUMNS	HY000
1672	ER_BINLOG_UNSAFE_UDF	HY000
1673	ER_BINLOG_UNSAFE_SYSTEM_VARIABLE	HY000
1674	ER_BINLOG_UNSAFE_SYSTEM_FUNCTION	HY000
1675	ER_BINLOG_UNSAFE_NONTRANS_AFTER_TRANS	HY000
1676	ER_MESSAGE_AND_STATEMENT	HY000
1678	ER_REPLICA_CANT_CREATE_CONVERSION	HY000
1679	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_FORMAT	HY000
1680	ER_PATH_LENGTH	HY000
1681	ER_WARN_DEPRECATED_SYNTAX_NO_REPLACEMENT	HY000
1682	ER_WRONG_NATIVE_TABLE_STRUCTURE	HY000
1683	ER_WRONG_PERFSCHEMA_USAGE	HY000
1684	ER_WARN_I_S_SKIPPED_TABLE	HY000
1685	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_BINLOG_DIRECT	HY000
1686	ER_STORED_FUNCTION_PREVENTS_SWITCH_BINLOG_DIRECT	HY000
1687	ER_SPATIAL_MUST_HAVE_GEOM_COL	42000
1688	ER_TOO_LONG_INDEX_COMMENT	HY000
1689	ER_LOCK_ABORTED	HY000
1690	ER_DATA_OUT_OF_RANGE	22003
1692	ER_BINLOG_UNSAFE_MULTIPLE_ENGINES_AND_SELF_LOGGING_ENGINE	HY000
1693	ER_BINLOG_UNSAFE_MIXED_STATEMENT	HY000
1694	ER_INSIDE_TRANSACTION_PREVENTS_SWITCH_SQL_LOG_BIN	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1695	ER_STORED_FUNCTION_PREVENTS_SWITCH_SQL_LOG_BIN	HY000
1696	ER_FAILED_READ_FROM_PAR_FILE	HY000
1697	ER_VALUES_IS_NOT_INT_TYPE_ERROR	HY000
1698	ER_ACCESS_DENIED_NO_PASSWORD_ERROR	28000
1701	ER_TRUNCATE_ILLEGAL_FK	42000
1702	ER_PLUGIN_IS_PERMANENT	HY000
1703	ER_REPLICA_HEARTBEAT_VALUE_OUT_OF_RANGE_MIN	HY000
1704	ER_REPLICA_HEARTBEAT_VALUE_OUT_OF_RANGE_MAX	HY000
1705	ER_STMT_CACHE_FULL	HY000
1706	ER_MULTI_UPDATE_KEY_CONFLICT	HY000
1707	ER_TABLE_NEEDS_REBUILD	HY000
1708	WARN_OPTION_BELOW_LIMIT	HY000
1709	ER_INDEX_COLUMN_TOO_LONG	HY000
1710	ER_ERROR_IN_TRIGGER_BODY	HY000
1711	ER_ERROR_IN_UNKNOWN_TRIGGER_BODY	HY000
1712	ER_INDEX_CORRUPT	HY000
1713	ER_UNDO_RECORD_TOO_BIG	HY000
1714	ER_BINLOG_UNSAFE_INSERT_IGNORE_SELECT	HY000
1715	ER_BINLOG_UNSAFE_INSERT_SELECT_UPDATE	HY000
1716	ER_BINLOG_UNSAFE_REPLACE_SELECT	HY000
1717	ER_BINLOG_UNSAFE_CREATE_IGNORE_SELECT	HY000
1718	ER_BINLOG_UNSAFE_CREATE_REPLACE_SELECT	HY000
1719	ER_BINLOG_UNSAFE_UPDATE_IGNORE	HY000
1720	ER_PLUGIN_NO_UNINSTALL	HY000
1721	ER_PLUGIN_NO_INSTALL	HY000
1722	ER_BINLOG_UNSAFE_WRITE_AUTOINC_SELECT	HY000
1723	ER_BINLOG_UNSAFE_CREATE_SELECT_AUTOINC	HY000
1724	ER_BINLOG_UNSAFE_INSERT_TWO_KEYS	HY000
1725	ER_TABLE_IN_FK_CHECK	HY000
1726	ER_UNSUPPORTED_ENGINE	HY000
1727	ER_BINLOG_UNSAFE_AUTOINC_NOT_FIRST	HY000
1728	ER_CANNOT_LOAD_FROM_TABLE_V2	HY000
1729	ER_SOURCE_DELAY_VALUE_OUT_OF_RANGE	HY000
1730	ER_ONLY_FD_AND_RBR_EVENTS_ALLOWED_IN_BINLOG_STATEMENT	HY000
1731	ER_PARTITION_EXCHANGE_DIFFERENT_OPTION	HY000
1732	ER_PARTITION_EXCHANGE_PART_TABLE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1733	ER_PARTITION_EXCHANGE_TEMP_TABLE	HY000
1734	ER_PARTITION_INSTEAD_OF_SUBPARTITION	HY000
1735	ER_UNKNOWN_PARTITION	HY000
1736	ER_TABLES_DIFFERENT_METADATA	HY000
1737	ER_ROW_DOES_NOT_MATCH_PARTITION	HY000
1738	ER_BINLOG_CACHE_SIZE_GREATER_THAN_MAX	HY000
1739	ER_WARN_INDEX_NOT_APPLICABLE	HY000
1740	ER_PARTITION_EXCHANGE_FOREIGN_KEY	HY000
1742	ER_RPL_INFO_DATA_TOO_LONG	HY000
1745	ER_BINLOG_STMT_CACHE_SIZE_GREATER_THAN_MAX	HY000
1746	ER_CANT_UPDATE_TABLE_IN_CREATE_TABLE_SELECT	HY000
1747	ER_PARTITION_CLAUSE_ON_NONPARTITIONED	HY000
1748	ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET	HY000
1750	ER_CHANGE_RPL_INFO_REPOSITORY_FAILURE	HY000
1751	ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_CREATED_TEMP_TABLE	HY000
1752	ER_WARNING_NOT_COMPLETE_ROLLBACK_WITH_DROPPED_TEMP_TABLE	HY000
1753	ER_MTA_FEATURE_IS_NOT_SUPPORTED	HY000
1754	ER_MTA_UPDATED_DBS_GREATER_MAX	HY000
1755	ER_MTA_CANT_PARALLEL	HY000
1756	ER_MTA_INCONSISTENT_DATA	HY000
1757	ER_FULLTEXT_NOT_SUPPORTED_WITH_PARTITIONING	HY000
1758	ER_DA_INVALID_CONDITION_NUMBER	35000
1759	ER_INSECURE_PLAIN_TEXT	HY000
1760	ER_INSECURE_CHANGE_SOURCE	HY000
1761	ER_FOREIGN_DUPLICATE_KEY_WITH_CHILD_INFO	23000
1762	ER_FOREIGN_DUPLICATE_KEY_WITHOUT_CHILD_INFO	23000
1763	ER_SQLTHREAD_WITH_SECURE_REPLICA	HY000
1764	ER_TABLE_HAS_NO_FT	HY000
1765	ER_VARIABLE_NOT_SETTABLE_IN_SF_OR_TRIGGER	HY000
1766	ER_VARIABLE_NOT_SETTABLE_IN_TRANSACTION	HY000
1769	ER_SET_STATEMENT_CANNOT_INVOKE_FUNCTION	HY000
1770	ER_GTID_NEXT_CANT_BE_AUTOMATIC_IF_GTID_NEXT_LIST_IS_NON_NULL	HY000
1772	ER_MALFORMED_GTID_SET_SPECIFICATION	HY000
1773	ER_MALFORMED_GTID_SET_ENCODING	HY000
1774	ER_MALFORMED_GTID_SPECIFICATION	HY000
1775	ER_GNO_EXHAUSTED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1776	ER_BAD_REPLICA_AUTO_POSITION	HY000
1777	ER_AUTO_POSITION_REQUIRES_GTID_MODE_NOT_OFF	HY000
1778	ER_CANT_DO_IMPLICIT_COMMIT_IN_TRX_WHEN_GTID_NEXT_IS_SET	HY000
1779	ER_GTID_MODE_ON_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON	HY000
1781	ER_CANT_SET_GTID_NEXT_TO_GTID_WHEN_GTID_MODE_IS_OFF	HY000
1782	ER_CANT_SET_GTID_NEXT_TO_ANONYMOUS_WHEN_GTID_MODE_IS_ON	HY000
1783	ER_CANT_SET_GTID_NEXT_LIST_TO_NON_NULL_WHEN_GTID_MODE_IS_OFF	HY000
1785	ER_GTID_UNSAFE_NON_TRANSACTIONAL_TABLE	HY000
1786	ER_GTID_UNSAFE_CREATE_SELECT	HY000
1788	ER_GTID_MODE_CAN_ONLY_CHANGE_ONE_STEP_AT_A_TIME	HY000
1789	ER_SOURCE_HAS_PURGED_REQUIRED_GTIDS	HY000
1790	ER_CANT_SET_GTID_NEXT_WHEN_OWNING_GTID	HY000
1791	ER_UNKNOWN_EXPLAIN_FORMAT	HY000
1792	ER_CANT_EXECUTE_IN_READ_ONLY_TRANSACTION	25006
1793	ER_TOO_LONG_TABLE_PARTITION_COMMENT	HY000
1794	ER_REPLICA_CONFIGURATION	HY000
1795	ER_INNODB_FT_LIMIT	HY000
1796	ER_INNODB_NO_FT_TEMP_TABLE	HY000
1797	ER_INNODB_FT_WRONG_DOCID_COLUMN	HY000
1798	ER_INNODB_FT_WRONG_DOCID_INDEX	HY000
1799	ER_INNODB_ONLINE_LOG_TOO_BIG	HY000
1800	ER_UNKNOWN_ALTER_ALGORITHM	HY000
1801	ER_UNKNOWN_ALTER_LOCK	HY000
1802	ER_MTA_CHANGE_SOURCE_CANT_RUN_WITH_GAPS	HY000
1803	ER_MTA_RECOVERY_FAILURE	HY000
1804	ER_MTA_RESET_WORKERS	HY000
1805	ER_COL_COUNT_DOESNT_MATCH_CORRUPTED_V2	HY000
1806	ER_REPLICA_SILENT_RETRY_TRANSACTION	HY000
1807	ER_DISCARD_FK_CHECKS_RUNNING	HY000
1808	ER_TABLE_SCHEMA_MISMATCH	HY000
1809	ER_TABLE_IN_SYSTEM_TABLESPACE	HY000
1810	ER_IO_READ_ERROR	HY000
1811	ER_IO_WRITE_ERROR	HY000
1812	ER_TABLESPACE_MISSING	HY000
1813	ER_TABLESPACE_EXISTS	HY000
1814	ER_TABLESPACE_DISCARDED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1815	ER_INTERNAL_ERROR	HY000
1816	ER_INNODB_IMPORT_ERROR	HY000
1817	ER_INNODB_INDEX_CORRUPT	HY000
1818	ER_INVALID_YEAR_COLUMN_LENGTH	HY000
1819	ER_NOT_VALID_PASSWORD	HY000
1820	ER_MUST_CHANGE_PASSWORD	HY000
1821	ER_FK_NO_INDEX_CHILD	HY000
1822	ER_FK_NO_INDEX_PARENT	HY000
1823	ER_FK_FAIL_ADD_SYSTEM	HY000
1824	ER_FK_CANNOT_OPEN_PARENT	HY000
1825	ER_FK_INCORRECT_OPTION	HY000
1826	ER_FK_DUP_NAME	HY000
1827	ER_PASSWORD_FORMAT	HY000
1828	ER_FK_COLUMN_CANNOT_DROP	HY000
1829	ER_FK_COLUMN_CANNOT_DROP_CHILD	HY000
1830	ER_FK_COLUMN_NOT_NULL	HY000
1831	ER_DUP_INDEX	HY000
1832	ER_FK_COLUMN_CANNOT_CHANGE	HY000
1833	ER_FK_COLUMN_CANNOT_CHANGE_CHILD	HY000
1835	ER_MALFORMED_PACKET	HY000
1836	ER_READ_ONLY_MODE	HY000
1837	ER_GTID_NEXT_TYPE_UNDEFINED_GTID	HY000
1838	ER_VARIABLE_NOT_SETTABLE_IN_SP	HY000
1840	ER_CANT_SET_GTID_PURGED_WHEN_GTID_EXECUTED_IS_NOT_EMPTY	HY000
1841	ER_CANT_SET_GTID_PURGED_WHEN_OWNED_GTIDS_IS_NOT_EMPTY	HY000
1842	ER_GTID_PURGED_WAS_CHANGED	HY000
1843	ER_GTID_EXECUTED_WAS_CHANGED	HY000
1844	ER_BINLOG_STMT_MODE_AND_NO_REPL_TABLES	HY000
1845	ER_ALTER_OPERATION_NOT_SUPPORTED	0A000
1846	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON	0A000
1847	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COPY	HY000
1848	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_PARTITION	HY000
1849	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_RENAME	HY000
1850	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COLUMN_TYPE	HY000
1851	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FK_CHECK	HY000
1853	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOPK	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
1854	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_AUTOINC	HY000
1855	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_HIDDEN_FTS	HY000
1856	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_CHANGE_FTS	HY000
1857	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_FTS	HY000
1859	ER_DUP_UNKNOWN_IN_INDEX	23000
1860	ER_IDENT_CAUSES_TOO_LONG_PATH	HY000
1861	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_NOT_NULL	HY000
1862	ER_MUST_CHANGE_PASSWORD_LOGIN	HY000
1863	ER_ROW_IN_WRONG_PARTITION	HY000
1864	ER_MTA_EVENT_BIGGER_PENDING_JOBS_SIZE_MAX	HY000
1866	ER_BINLOG_LOGICAL_CORRUPTION	HY000
1867	ER_WARN_PURGE_LOG_IN_USE	HY000
1868	ER_WARN_PURGE_LOG_IS_ACTIVE	HY000
1869	ER_AUTO_INCREMENT_CONFLICT	HY000
1870	WARN_ON_BLOCKHOLE_IN_RBR	HY000
1871	ER_REPLICA_CM_INIT_REPOSITORY	HY000
1872	ER_REPLICA_AM_INIT_REPOSITORY	HY000
1873	ER_ACCESS_DENIED_CHANGE_USER_ERROR	28000
1874	ER_INNODB_READ_ONLY	HY000
1875	ER_STOP_REPLICA_SQL_THREAD_TIMEOUT	HY000
1876	ER_STOP_REPLICA_IO_THREAD_TIMEOUT	HY000
1877	ER_TABLE_CORRUPT	HY000
1878	ER_TEMP_FILE_WRITE_FAILURE	HY000
1879	ER_INNODB_FT_AUX_NOT_HEX_ID	HY000
1880	ER_OLD_TEMPORALS_UPGRADED	HY000
1881	ER_INNODB_FORCED_RECOVERY	HY000
1882	ER_AES_INVALID_IV	HY000
1883	ER_PLUGIN_CANNOT_BE_UNINSTALLED	HY000
1884	ER_GTID_UNSAFE_BINLOG_SPLITTABLE_STATEMENT_AND_ASSIGNED_GTID	HY000
1885	ER_REPLICA_HAS_MORE_GTIDS_THAN_SOURCE	HY000
1886	ER_MISSING_KEY	HY000
1887	WARN_NAMED_PIPE_ACCESS_EVERYONE	HY000
3000	ER_FILE_CORRUPT	HY000
3001	ER_ERROR_ON_SOURCE	HY000
3003	ER_STORAGE_ENGINE_NOT_LOADED	HY000
3004	ER_GET_STACKED_DA_WITHOUT_ACTIVE_HANDLER	0Z002

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3005	ER_WARN_LEGACY_SYNTAX_CONVERTED	HY000
3006	ER_BINLOG_UNSAFE_FULLTEXT_PLUGIN	HY000
3007	ER_CANNOT_DISCARD_TEMPORARY_TABLE	HY000
3008	ER_FK_DEPTH_EXCEEDED	HY000
3009	ER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE_V2	HY000
3010	ER_WARN_TRIGGER_DOESNT_HAVE_CREATED	HY000
3011	ER_REFERENCED_TRG_DOES_NOT_EXIST	HY000
3012	ER_EXPLAIN_NOT_SUPPORTED	HY000
3013	ER_INVALID_FIELD_SIZE	HY000
3014	ER_MISSING_HA_CREATE_OPTION	HY000
3015	ER_ENGINE_OUT_OF_MEMORY	HY000
3016	ER_PASSWORD_EXPIRE_ANONYMOUS_USER	HY000
3017	ER_REPLICA_SQL_THREAD_MUST_STOP	HY000
3018	ER_NO_FT_MATERIALIZED_SUBQUERY	HY000
3019	ER_INNODB_UNDO_LOG_FULL	HY000
3020	ER_INVALID_ARGUMENT_FOR_LOGARITHM	2201E
3021	ER_REPLICA_CHANNEL_IO_THREAD_MUST_STOP	HY000
3022	ER_WARN_OPEN_TEMP_TABLES_MUST_BE_ZERO	HY000
3023	ER_WARN_ONLY_SOURCE_LOG_FILE_NO_POS	HY000
3024	ER_QUERY_TIMEOUT	HY000
3025	ER_NON_RO_SELECT_DISABLE_TIMER	HY000
3026	ER_DUP_LIST_ENTRY	HY000
3028	ER_AGGREGATE_ORDER_FOR_UNION	HY000
3029	ER_AGGREGATE_ORDER_NON_AGG_QUERY	HY000
3030	ER_REPLICA_WORKER_STOPPED_PREVIOUS_THD_ERROR	HY000
3031	ER_DONT_SUPPORT_REPLICA_PRESERVE_COMMIT_ORDER	HY000
3032	ER_SERVER_OFFLINE_MODE	HY000
3033	ER_GIS_DIFFERENT_SRIDS	HY000
3034	ER_GIS_UNSUPPORTED_ARGUMENT	HY000
3035	ER_GIS_UNKNOWN_ERROR	HY000
3036	ER_GIS_UNKNOWN_EXCEPTION	HY000
3037	ER_GIS_INVALID_DATA	22023
3038	ER_BOOST_GEOMETRY_EMPTY_INPUT_EXCEPTION	HY000
3039	ER_BOOST_GEOMETRY_CENTROID_EXCEPTION	HY000
3040	ER_BOOST_GEOMETRY_OVERLAY_INVALID_INPUT_EXCEPTION	HY000
3041	ER_BOOST_GEOMETRY_TURN_INFO_EXCEPTION	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3042	ER_BOOST_GEOMETRY_SELF_INTERSECTION_POINT_EXCEPTION	HY000
3043	ER_BOOST_GEOMETRY_UNKNOWN_EXCEPTION	HY000
3044	ER_STD_BAD_ALLOC_ERROR	HY000
3045	ER_STD_DOMAIN_ERROR	HY000
3046	ER_STD_LENGTH_ERROR	HY000
3047	ER_STD_INVALID_ARGUMENT	HY000
3048	ER_STD_OUT_OF_RANGE_ERROR	HY000
3049	ER_STD_OVERFLOW_ERROR	HY000
3050	ER_STD_RANGE_ERROR	HY000
3051	ER_STD_UNDERFLOW_ERROR	HY000
3052	ER_STD_LOGIC_ERROR	HY000
3053	ER_STD_RUNTIME_ERROR	HY000
3054	ER_STD_UNKNOWN_EXCEPTION	HY000
3055	ER_GIS_DATA_WRONG_ENDIANESS	HY000
3056	ER_CHANGE_SOURCE_PASSWORD_LENGTH	HY000
3057	ER_USER_LOCK_WRONG_NAME	42000
3058	ER_USER_LOCK_DEADLOCK	HY000
3059	ER_REPLACE_INACCESSIBLE_ROWS	HY000
3060	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_GIS	HY000
3061	ER_ILLEGAL_USER_VAR	42000
3062	ER_GTID_MODE_OFF	HY000
3064	ER_INCORRECT_TYPE	HY000
3065	ER_FIELD_IN_ORDER_NOT_SELECT	HY000
3066	ER_AGGREGATE_IN_ORDER_NOT_SELECT	HY000
3067	ER_INVALID_RPL_WILD_TABLE_FILTER_PATTERN	HY000
3068	ER_NET_OK_PACKET_TOO_LARGE	08S01
3069	ER_INVALID_JSON_DATA	HY000
3070	ER_INVALID_GEOJSON_MISSING_MEMBER	HY000
3071	ER_INVALID_GEOJSON_WRONG_TYPE	HY000
3072	ER_INVALID_GEOJSON_UNSPECIFIED	HY000
3073	ER_DIMENSION_UNSUPPORTED	HY000
3074	ER_REPLICA_CHANNEL_DOES_NOT_EXIST	HY000
3076	ER_REPLICA_CHANNEL_NAME_INVALID_OR_TOO_LONG	HY000
3077	ER_REPLICA_NEW_CHANNEL_WRONG_REPOSITORY	HY000
3079	ER_REPLICA_MULTIPLE_CHANNELS_CMD	HY000
3080	ER_REPLICA_MAX_CHANNELS_EXCEEDED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3081	ER_REPLICA_CHANNEL_MUST_STOP	HY000
3082	ER_REPLICA_CHANNEL_NOT_RUNNING	HY000
3083	ER_REPLICA_CHANNEL_WAS_RUNNING	HY000
3084	ER_REPLICA_CHANNEL_WAS_NOT_RUNNING	HY000
3085	ER_REPLICA_CHANNEL_SQL_THREAD_MUST_STOP	HY000
3086	ER_REPLICA_CHANNEL_SQL_SKIP_COUNTER	HY000
3087	ER_WRONG_FIELD_WITH_GROUP_V2	HY000
3088	ER_MIX_OF_GROUP_FUNC_AND_FIELDS_V2	HY000
3089	ER_WARN_DEPRECATED_SYSVAR_UPDATE	HY000
3090	ER_WARN_DEPRECATED_SQLMODE	HY000
3091	ER_CANNOT_LOG_PARTIAL_DROP_DATABASE_WITH_GTID	HY000
3092	ER_GROUP_REPLICATION_CONFIGURATION	HY000
3093	ER_GROUP_REPLICATION_RUNNING	HY000
3094	ER_GROUP_REPLICATION_APPLIER_INIT_ERROR	HY000
3095	ER_GROUP_REPLICATION_STOP_APPLIER_THREAD_TIMEOUT	HY000
3096	ER_GROUP_REPLICATION_COMMUNICATION_LAYER_SESSION_ERROR	HY000
3097	ER_GROUP_REPLICATION_COMMUNICATION_LAYER_JOIN_ERROR	HY000
3098	ER_BEFORE_DML_VALIDATION_ERROR	HY000
3099	ER_PREVENTS_VARIABLE_WITHOUT_RBR	HY000
3100	ER_RUN_HOOK_ERROR	HY000
3101	ER_TRANSACTION_ROLLBACK_DURING_COMMIT	40000
3102	ER_GENERATED_COLUMN_FUNCTION_IS_NOT_ALLOWED	HY000
3103	ER_UNSUPPORTED_ALTER_INPLACE_ON_VIRTUAL_COLUMN	HY000
3104	ER_WRONG_FK_OPTION_FOR_GENERATED_COLUMN	HY000
3105	ER_NON_DEFAULT_VALUE_FOR_GENERATED_COLUMN	HY000
3106	ER_UNSUPPORTED_ACTION_ON_GENERATED_COLUMN	HY000
3107	ER_GENERATED_COLUMN_NON_PRIOR	HY000
3108	ER_DEPENDENT_BY_GENERATED_COLUMN	HY000
3109	ER_GENERATED_COLUMN_REF_AUTO_INC	HY000
3110	ER_FEATURE_NOT_AVAILABLE	HY000
3111	ER_CANT_SET_GTID_MODE	HY000
3112	ER_CANT_USE_AUTO_POSITION_WITH_GTID_MODE_OFF	HY000
3116	ER_CANT_ENFORCE_GTID_CONSISTENCY_WITH_ONGOING_GTID_VIOLATING_TX	HY000
3117	ER_ENFORCE_GTID_CONSISTENCY_WARN_WITH_ONGOING_GTID_VIOLATING_TX	HY000
3118	ER_ACCOUNT_HAS_BEEN_LOCKED	HY000
3119	ER_WRONG_TABLESPACE_NAME	42000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3120	ER_TABLESPACE_IS_NOT_EMPTY	HY000
3121	ER_WRONG_FILE_NAME	HY000
3122	ER_BOOST_GEOMETRY_INCONSISTENT_TURNS_EXCEPTION	HY000
3123	ER_WARN_OPTIMIZER_HINT_SYNTAX_ERROR	HY000
3124	ER_WARN_BAD_MAX_EXECUTION_TIME	HY000
3125	ER_WARN_UNSUPPORTED_MAX_EXECUTION_TIME	HY000
3126	ER_WARN_CONFLICTING_HINT	HY000
3127	ER_WARN_UNKNOWN_QB_NAME	HY000
3128	ER_UNRESOLVED_HINT_NAME	HY000
3129	ER_WARN_ON_MODIFYING_GTID_EXECUTED_TABLE	HY000
3130	ER_PLUGGABLE_PROTOCOL_COMMAND_NOT_SUPPORTED	HY000
3131	ER_LOCKING_SERVICE_WRONG_NAME	42000
3132	ER_LOCKING_SERVICE_DEADLOCK	HY000
3133	ER_LOCKING_SERVICE_TIMEOUT	HY000
3134	ER_GIS_MAX_POINTS_IN_GEOMETRY_OVERFLOWED	HY000
3135	ER_SQL_MODE_MERGED	HY000
3136	ER_VTOKEN_PLUGIN_TOKEN_MISMATCH	HY000
3137	ER_VTOKEN_PLUGIN_TOKEN_NOT_FOUND	HY000
3138	ER_CANT_SET_VARIABLE_WHEN_OWNING_GTID	HY000
3139	ER_REPLICA_CHANNEL_OPERATION_NOT_ALLOWED	HY000
3140	ER_INVALID_JSON_TEXT	22032
3141	ER_INVALID_JSON_TEXT_IN_PARAM	22032
3142	ER_INVALID_JSON_BINARY_DATA	HY000
3143	ER_INVALID_JSON_PATH	42000
3144	ER_INVALID_JSON_CHARSET	22032
3145	ER_INVALID_JSON_CHARSET_IN_FUNCTION	22032
3146	ER_INVALID_TYPE_FOR_JSON	22032
3147	ER_INVALID_CAST_TO_JSON	22032
3148	ER_INVALID_JSON_PATH_CHARSET	42000
3149	ER_INVALID_JSON_PATH_WILDCARD	42000
3150	ER_JSON_VALUE_TOO_BIG	22032
3151	ER_JSON_KEY_TOO_BIG	22032
3152	ER_JSON_USED_AS_KEY	42000
3153	ER_JSON_VACUOUS_PATH	42000
3154	ER_JSON_BAD_ONE_OR_ALL_ARG	42000
3155	ER_NUMERIC_JSON_VALUE_OUT_OF_RANGE	22003

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3156	ER_INVALID_JSON_VALUE_FOR_CAST	22018
3157	ER_JSON_DOCUMENT_TOO_DEEP	22032
3158	ER_JSON_DOCUMENT_NULL_KEY	22032
3159	ER_SECURE_TRANSPORT_REQUIRED	HY000
3160	ER_NO_SECURE_TRANSPORTS_CONFIGURED	HY000
3161	ER_DISABLED_STORAGE_ENGINE	HY000
3162	ER_USER_DOES_NOT_EXIST	HY000
3163	ER_USER_ALREADY_EXISTS	HY000
3164	ER_AUDIT_API_ABORT	HY000
3165	ER_INVALID_JSON_PATH_ARRAY_CELL	42000
3166	ER_BUFPOOL_RESIZE_INPROGRESS	HY000
3167	ER_FEATURE_DISABLED_SEE_DOC	HY000
3168	ER_SERVER_ISNT_AVAILABLE	HY000
3169	ER_SESSION_WAS_KILLED	HY000
3170	ER_CAPACITY_EXCEEDED	HY000
3171	ER_CAPACITY_EXCEEDED_IN_RANGE_OPTIMIZER	HY000
3173	ER_CANT_WAIT_FOR_EXECUTED_GTID_SET_WHILE_OWNING_A_GTID	HY000
3174	ER_CANNOT_ADD_FOREIGN_BASE_COL_VIRTUAL	HY000
3175	ER_CANNOT_CREATE_VIRTUAL_INDEX_CONSTRAINT	HY000
3176	ER_ERROR_ON_MODIFYING_GTID_EXECUTED_TABLE	HY000
3177	ER_LOCK_REFUSED_BY_ENGINE	HY000
3178	ER_UNSUPPORTED_ALTER_ONLINE_ON_VIRTUAL_COLUMN	HY000
3179	ER_MASTER_KEY_ROTATION_NOT_SUPPORTED_BY_SE	HY000
3181	ER_MASTER_KEY_ROTATION_BINLOG_FAILED	HY000
3182	ER_MASTER_KEY_ROTATION_SE_UNAVAILABLE	HY000
3183	ER_TABLESPACE_CANNOT_ENCRYPT	HY000
3184	ER_INVALID_ENCRYPTION_OPTION	HY000
3185	ER_CANNOT_FIND_KEY_IN_KEYRING	HY000
3186	ER_CAPACITY_EXCEEDED_IN_PARSER	HY000
3187	ER_UNSUPPORTED_ALTER_ENCRYPTION_INPLACE	HY000
3188	ER_KEYRING_UDF_KEYRING_SERVICE_ERROR	HY000
3189	ER_USER_COLUMN_OLD_LENGTH	HY000
3190	ER_CANT_RESET_SOURCE	HY000
3191	ER_GROUP_REPLICATION_MAX_GROUP_SIZE	HY000
3192	ER_CANNOT_ADD_FOREIGN_BASE_COL_STORED	HY000
3193	ER_TABLE_REFERENCED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3197	ER_XA_RETRY	HY000
3198	ER_KEYRING_AWS_UDF_AWS_KMS_ERROR	HY000
3199	ER_BINLOG_UNSAFE_XA	HY000
3200	ER_UDF_ERROR	HY000
3201	ER_KEYRING_MIGRATION_FAILURE	HY000
3202	ER_KEYRING_ACCESS_DENIED_ERROR	42000
3203	ER_KEYRING_MIGRATION_STATUS	HY000
3218	ER_AUDIT_LOG_UDF_READ_INVALID_MAX_ARRAY_LENGTH_ARG_VALUE	HY000
3231	ER_WRITE_SET_EXCEEDS_LIMIT	HY000
3235	ER_AES_INVALID_KDF_NAME	HY000
3236	ER_AES_INVALID_KDF_ITERATIONS	HY000
3237	WARN_AES_KEY_SIZE	HY000
3238	ER_AES_INVALID_KDF_OPTION_SIZE	HY000
3500	ER_UNSUPPORT_COMPRESSED_TEMPORARY_TABLE	HY000
3501	ER_ACL_OPERATION_FAILED	HY000
3502	ER_UNSUPPORTED_INDEX_ALGORITHM	HY000
3503	ER_NO_SUCH_DB	42Y07
3504	ER_TOO_BIG_ENUM	HY000
3505	ER_TOO_LONG_SET_ENUM_VALUE	HY000
3506	ER_INVALID_DD_OBJECT	HY000
3507	ER_UPDATING_DD_TABLE	HY000
3508	ER_INVALID_DD_OBJECT_ID	HY000
3509	ER_INVALID_DD_OBJECT_NAME	HY000
3510	ER_TABLESPACE_MISSING_WITH_NAME	HY000
3511	ER_TOO_LONG_ROUTINE_COMMENT	HY000
3512	ER_SP_LOAD_FAILED	HY000
3513	ER_INVALID_BITWISE_OPERANDS_SIZE	HY000
3514	ER_INVALID_BITWISE_AGGREGATE_OPERANDS_SIZE	HY000
3515	ER_WARN_UNSUPPORTED_HINT	HY000
3516	ER_UNEXPECTED_GEOMETRY_TYPE	22S01
3517	ER_SRS_PARSE_ERROR	SR002
3518	ER_SRS_PROJ_PARAMETER_MISSING	SR003
3519	ER_WARN_SRS_NOT_FOUND	01000
3520	ER_SRS_NOT_CARTESIAN	22S00
3521	ER_SRS_NOT_CARTESIAN_UNDEFINED	SR001
3522	ER_PK_INDEX_CANT_BE_INVISIBLE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3523	ER_UNKNOWN_AUTHID	HY000
3524	ER_FAILED_ROLE_GRANT	HY000
3525	ER_OPEN_ROLE_TABLES	HY000
3526	ER_FAILED_DEFAULT_ROLES	HY000
3527	ER_COMPONENTS_NO_SCHEME	HY000
3528	ER_COMPONENTS_NO_SCHEME_SERVICE	HY000
3529	ER_COMPONENTS_CANT_LOAD	HY000
3530	ER_ROLE_NOT_GRANTED	HY000
3531	ER_FAILED_REVOKE_ROLE	HY000
3532	ER_RENAME_ROLE	HY000
3533	ER_COMPONENTS_CANT_ACQUIRE_SERVICE_IMPLEMENTATION	HY000
3534	ER_COMPONENTS_CANT_SATISFY_DEPENDENCY	HY000
3535	ER_COMPONENTS_LOAD_CANT_REGISTER_SERVICE_IMPLEMENTATION	HY000
3536	ER_COMPONENTS_LOAD_CANT_INITIALIZE	HY000
3537	ER_COMPONENTS_UNLOAD_NOT_LOADED	HY000
3538	ER_COMPONENTS_UNLOAD_CANT_DEINITIALIZE	HY000
3539	ER_COMPONENTS_CANT_RELEASE_SERVICE	HY000
3540	ER_COMPONENTS_UNLOAD_CANT_UNREGISTER_SERVICE	HY000
3541	ER_COMPONENTS_CANT_UNLOAD	HY000
3542	ER_WARN_UNLOAD_THE_NOT_PERSISTED	HY000
3543	ER_COMPONENT_TABLE_INCORRECT	HY000
3544	ER_COMPONENT_MANIPULATE_ROW_FAILED	HY000
3545	ER_COMPONENTS_UNLOAD_DUPLICATE_IN_GROUP	HY000
3546	ER_CANT_SET_GTID_PURGED_DUE_SETS_CONSTRAINTS	HY000
3547	ER_CANNOT_LOCK_USER_MANAGEMENT_CACHES	HY000
3548	ER_SRS_NOT_FOUND	SR001
3549	ER_VARIABLE_NOT_PERSISTED	HY000
3550	ER_IS_QUERY_INVALID_CLAUSE	HY000
3551	ER_UNABLE_TO_STORE_STATISTICS	HY000
3552	ER_NO_SYSTEM_SCHEMA_ACCESS	HY000
3553	ER_NO_SYSTEM_TABLESPACE_ACCESS	HY000
3554	ER_NO_SYSTEM_TABLE_ACCESS	HY000
3555	ER_NO_SYSTEM_TABLE_ACCESS_FOR_DICTIONARY_TABLE	HY000
3556	ER_NO_SYSTEM_TABLE_ACCESS_FOR_SYSTEM_TABLE	HY000
3557	ER_NO_SYSTEM_TABLE_ACCESS_FOR_TABLE	HY000
3558	ER_INVALID_OPTION_KEY	22023

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3559	ER_INVALID_OPTION_VALUE	22023
3560	ER_INVALID_OPTION_KEY_VALUE_PAIR	22023
3561	ER_INVALID_OPTION_START_CHARACTER	22023
3562	ER_INVALID_OPTION_END_CHARACTER	22023
3563	ER_INVALID_OPTION_CHARACTERS	22023
3564	ER_DUPLICATE_OPTION_KEY	22023
3565	ER_WARN_SRS_NOT_FOUND_AXIS_ORDER	01000
3566	ER_NO_ACCESS_TO_NATIVE_FCT	HY000
3567	ER_RESET_SOURCE_TO_VALUE_OUT_OF_RANGE	HY000
3568	ER_UNRESOLVED_TABLE_LOCK	HY000
3569	ER_DUPLICATE_TABLE_LOCK	HY000
3570	ER_BINLOG_UNSAFE_SKIP_LOCKED	HY000
3571	ER_BINLOG_UNSAFE_NOWAIT	HY000
3572	ER_LOCK_NOWAIT	HY000
3573	ER_CTE_RECURSIVE_REQUIRES_UNION	HY000
3574	ER_CTE_RECURSIVE_REQUIRES_NONRECURSIVE_FIRST	HY000
3575	ER_CTE_RECURSIVE_FORBIDS_AGGREGATION	HY000
3576	ER_CTE_RECURSIVE_FORBIDDEN_JOIN_ORDER	HY000
3577	ER_CTE_RECURSIVE_REQUIRES_SINGLE_REFERENCE	HY000
3578	ER_SWITCH_TMP_ENGINE	HY000
3579	ER_WINDOW_NO_SUCH_WINDOW	HY000
3580	ER_WINDOW_CIRCULARITY_IN_WINDOW_GRAPH	HY000
3581	ER_WINDOW_NO_CHILD_PARTITIONING	HY000
3582	ER_WINDOW_NO_INHERIT_FRAME	HY000
3583	ER_WINDOW_NO_REDEFINE_ORDER_BY	HY000
3584	ER_WINDOW_FRAME_START_ILLEGAL	HY000
3585	ER_WINDOW_FRAME_END_ILLEGAL	HY000
3586	ER_WINDOW_FRAME_ILLEGAL	HY000
3587	ER_WINDOW_RANGE_FRAME_ORDER_TYPE	HY000
3588	ER_WINDOW_RANGE_FRAME_TEMPORAL_TYPE	HY000
3589	ER_WINDOW_RANGE_FRAME_NUMERIC_TYPE	HY000
3590	ER_WINDOW_RANGE_BOUND_NOT_CONSTANT	HY000
3591	ER_WINDOW_DUPLICATE_NAME	HY000
3592	ER_WINDOW_ILLEGAL_ORDER_BY	HY000
3593	ER_WINDOW_INVALID_WINDOW_FUNC_USE	HY000
3594	ER_WINDOW_INVALID_WINDOW_FUNC_ALIAS_USE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3595	ER_WINDOW_NESTED_WINDOW_FUNC_USE_IN_WINDOW_SPEC	HY000
3596	ER_WINDOW_ROWS_INTERVAL_USE	HY000
3597	ER_WINDOW_NO_GROUP_ORDER_UNUSED	HY000
3598	ER_WINDOW_EXPLAIN_JSON	HY000
3599	ER_WINDOW_FUNCTION_IGNORES_FRAME	HY000
3600	ER_WL9236_NOW_UNUSED	HY000
3601	ER_INVALID_NO_OF_ARGS	HY000
3602	ER_FIELD_IN_GROUPING_NOT_GROUP_BY	HY000
3603	ER_TOO_LONG_TABLESPACE_COMMENT	HY000
3604	ER_ENGINE_CANT_DROP_TABLE	HY000
3605	ER_ENGINE_CANT_DROP_MISSING_TABLE	HY000
3606	ER_TABLESPACE_DUP_FILENAME	HY000
3607	ER_DB_DROP_RMDIR2	HY000
3608	ER_IMP_NO_FILES_MATCHED	HY000
3609	ER_IMP_SCHEMA_DOES_NOT_EXIST	HY000
3610	ER_IMP_TABLE_ALREADY_EXISTS	HY000
3611	ER_IMP_INCOMPATIBLE_MYSQLD_VERSION	HY000
3612	ER_IMP_INCOMPATIBLE_DD_VERSION	HY000
3613	ER_IMP_INCOMPATIBLE_SDI_VERSION	HY000
3614	ER_WARN_INVALID_HINT	HY000
3615	ER_VAR_DOES_NOT_EXIST	HY000
3616	ER_LONGITUDE_OUT_OF_RANGE	22S02
3617	ER_LATITUDE_OUT_OF_RANGE	22S03
3618	ER_NOT_IMPLEMENTED_FOR_GEOGRAPHIC_SRS	22S00
3619	ER_ILLEGAL_PRIVILEGE_LEVEL	HY000
3620	ER_NO_SYSTEM_VIEW_ACCESS	HY000
3621	ER_COMPONENT_FILTER_FLABBERGASTED	HY000
3622	ER_PART_EXPR_TOO_LONG	HY000
3623	ER_UDF_DROP_DYNAMICALLY_REGISTERED	HY000
3624	ER_UNABLE_TO_STORE_COLUMN_STATISTICS	HY000
3625	ER_UNABLE_TO_UPDATE_COLUMN_STATISTICS	HY000
3626	ER_UNABLE_TO_DROP_COLUMN_STATISTICS	HY000
3627	ER_UNABLE_TO_BUILD_HISTOGRAM	HY000
3628	ER_MANDATORY_ROLE	HY000
3629	ER_MISSING_TABLESPACE_FILE	HY000
3630	ER_PERSIST_ONLY_ACCESS_DENIED_ERROR	42000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3631	ER_CMD_NEED_SUPER	HY000
3632	ER_PATH_IN_DATADIR	HY000
3633	ER_CLONE_DDL_IN_PROGRESS	HY000
3634	ER_CLONE_TOO_MANY_CONCURRENT_CLONES	HY000
3635	ER_APPLIER_LOG_EVENT_VALIDATION_ERROR	HY000
3636	ER_CTE_MAX_RECURSION_DEPTH	HY000
3637	ER_NOT_HINT_UPDATABLE_VARIABLE	HY000
3638	ER_CREDENTIALS_CONTRADICT_TO_HISTORY	HY000
3639	ER_WARNING_PASSWORD_HISTORY_CLAUSES_VOID	HY000
3640	ER_CLIENT_DOES_NOT_SUPPORT	HY000
3641	ER_I_S_SKIPPED_TABLESPACE	HY000
3642	ER_TABLESPACE_ENGINE_MISMATCH	HY000
3643	ER_WRONG_SRID_FOR_COLUMN	HY000
3644	ER_CANNOT_ALTER_SRID_DUE_TO_INDEX	HY000
3645	ER_WARN_BINLOG_PARTIAL_UPDATES_DISABLED	HY000
3647	ER_WARN_BINLOG_PARTIAL_UPDATES_SUGGESTS_PARTIAL_IMAGES	HY000
3648	ER_COULD_NOT_APPLY_JSON_DIFF	HY000
3649	ER_CORRUPTED_JSON_DIFF	HY000
3650	ER_RESOURCE_GROUP_EXISTS	HY000
3651	ER_RESOURCE_GROUP_NOT_EXISTS	HY000
3652	ER_INVALID_VCPU_ID	HY000
3653	ER_INVALID_VCPU_RANGE	HY000
3654	ER_INVALID_THREAD_PRIORITY	HY000
3655	ER_DISALLOWED_OPERATION	HY000
3656	ER_RESOURCE_GROUP_BUSY	HY000
3657	ER_RESOURCE_GROUP_DISABLED	HY000
3658	ER_FEATURE_UNSUPPORTED	HY000
3659	ER_ATTRIBUTE_IGNORED	HY000
3660	ER_INVALID_THREAD_ID	HY000
3661	ER_RESOURCE_GROUP_BIND_FAILED	HY000
3662	ER_INVALID_USE_OF_FORCE_OPTION	HY000
3663	ER_GROUP_REPLICATION_COMMAND_FAILURE	HY000
3664	ER_SDI_OPERATION_FAILED	HY000
3665	ER_MISSING_JSON_TABLE_VALUE	22035
3666	ER_WRONG_JSON_TABLE_VALUE	2203F
3667	ER_TF_MUST_HAVE_ALIAS	42000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3668	ER_TF_FORBIDDEN_JOIN_TYPE	HY000
3669	ER_JT_VALUE_OUT_OF_RANGE	22003
3670	ER_JT_MAX_NESTED_PATH	42000
3671	ER_PASSWORD_EXPIRATION_NOT_SUPPORTED_BY_AUTH_METHOD	HY000
3672	ER_INVALID_GEOJSON_CRS_NOT_TOP_LEVEL	HY000
3673	ER_BAD_NULL_ERROR_NOT_IGNORED	23000
3674	WARN_USELESS_SPATIAL_INDEX	HY000
3675	ER_DISK_FULL_NOWAIT	HY000
3676	ER_PARSE_ERROR_IN_DIGEST_FN	HY000
3677	ER_UNDISCLOSED_PARSE_ERROR_IN_DIGEST_FN	HY000
3678	ER_SCHEMA_DIR_EXISTS	HY000
3679	ER_SCHEMA_DIR_MISSING	HY000
3680	ER_SCHEMA_DIR_CREATE_FAILED	HY000
3681	ER_SCHEMA_DIR_UNKNOWN	HY000
3682	ER_ONLY_IMPLEMENTED_FOR_SRID_0_AND_4326	22S00
3684	ER_REGEXP_BUFFER_OVERFLOW	HY000
3685	ER_REGEXP_ILLEGAL_ARGUMENT	HY000
3686	ER_REGEXP_INDEX_OUTOFBOUNDS_ERROR	HY000
3687	ER_REGEXP_INTERNAL_ERROR	HY000
3688	ER_REGEXP_RULE_SYNTAX	HY000
3689	ER_REGEXP_BAD_ESCAPE_SEQUENCE	HY000
3690	ER_REGEXP_UNIMPLEMENTED	HY000
3691	ER_REGEXP_MISMATCHED_PAREN	HY000
3692	ER_REGEXP_BAD_INTERVAL	HY000
3693	ER_REGEXP_MAX_LT_MIN	HY000
3694	ER_REGEXP_INVALID_BACK_REF	HY000
3695	ER_REGEXP_LOOK_BEHIND_LIMIT	HY000
3696	ER_REGEXP_MISSING_CLOSE_BRACKET	HY000
3697	ER_REGEXP_INVALID_RANGE	HY000
3698	ER_REGEXP_STACK_OVERFLOW	HY000
3699	ER_REGEXP_TIME_OUT	HY000
3700	ER_REGEXP_PATTERN_TOO_BIG	HY000
3701	ER_CANT_SET_ERROR_LOG_SERVICE	HY000
3702	ER_EMPTY_PIPELINE_FOR_ERROR_LOG_SERVICE	HY000
3703	ER_COMPONENT_FILTER_DIAGNOSTICS	HY000
3704	ER_NOT_IMPLEMENTED_FOR_CARTESIAN_SRS	22S00

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3705	ER_NOT_IMPLEMENTED_FOR_PROJECTED_SRS	22S00
3706	ER_NONPOSITIVE_RADIUS	22003
3707	ER_RESTART_SERVER_FAILED	HY000
3708	ER_SRS_MISSING_MANDATORY_ATTRIBUTE	SR006
3709	ER_SRS_MULTIPLE_ATTRIBUTE_DEFINITIONS	SR006
3710	ER_SRS_NAME_CANT_BE_EMPTY_OR_WHITESPACE	SR006
3711	ER_SRS_ORGANIZATION_CANT_BE_EMPTY_OR_WHITESPACE	SR006
3712	ER_SRS_ID_ALREADY_EXISTS	SR004
3713	ER_WARN_SRS_ID_ALREADY_EXISTS	01S00
3714	ER_CANT_MODIFY_SRID_0	SR000
3715	ER_WARN_RESERVED_SRID_RANGE	01S01
3716	ER_CANT_MODIFY_SRS_USED_BY_COLUMN	SR005
3717	ER_SRS_INVALID_CHARACTER_IN_ATTRIBUTE	SR006
3718	ER_SRS_ATTRIBUTE_STRING_TOO_LONG	SR006
3719	ER_DEPRECATED_UTF8_ALIAS	HY000
3720	ER_DEPRECATED_NATIONAL	HY000
3721	ER_INVALID_DEFAULT_UTF8MB4_COLLATION	HY000
3722	ER_UNABLE_TO_COLLECT_LOG_STATUS	HY000
3723	ER_RESERVED_TABLESPACE_NAME	HY000
3724	ER_UNABLE_TO_SET_OPTION	HY000
3725	ER_REPLICA_POSSIBLY_DIVERGED_AFTER_DDL	HY000
3726	ER_SRS_NOT_GEOGRAPHIC	22S00
3727	ER_POLYGON_TOO_LARGE	22023
3728	ER_SPATIAL_UNIQUE_INDEX	HY000
3729	ER_INDEX_TYPE_NOT_SUPPORTED_FOR_SPATIAL_INDEX	HY000
3730	ER_FK_CANNOT_DROP_PARENT	HY000
3731	ER_GEOMETRY_PARAM_LONGITUDE_OUT_OF_RANGE	22S02
3732	ER_GEOMETRY_PARAM_LATITUDE_OUT_OF_RANGE	22S03
3733	ER_FK_CANNOT_USE_VIRTUAL_COLUMN	HY000
3734	ER_FK_NO_COLUMN_PARENT	HY000
3735	ER_CANT_SET_ERROR_SUPPRESSION_LIST	HY000
3736	ER_SRS_GEOGCS_INVALID_AXES	SR002
3737	ER_SRS_INVALID_SEMI_MAJOR_AXIS	SR002
3738	ER_SRS_INVALID_INVERSE_FLATTENING	SR002
3739	ER_SRS_INVALID_ANGULAR_UNIT	SR002
3740	ER_SRS_INVALID_PRIME_MERIDIAN	SR002

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3741	ER_TRANSFORM_SOURCE_SRS_NOT_SUPPORTED	22S00
3742	ER_TRANSFORM_TARGET_SRS_NOT_SUPPORTED	22S00
3743	ER_TRANSFORM_SOURCE_SRS_MISSING_TOWGS84	22S00
3744	ER_TRANSFORM_TARGET_SRS_MISSING_TOWGS84	22S00
3745	ER_TEMP_TABLE_PREVENTS_SWITCH_SESSION_BINLOG_FORMAT	HY000
3746	ER_TEMP_TABLE_PREVENTS_SWITCH_GLOBAL_BINLOG_FORMAT	HY000
3747	ER_RUNNING_APPLIER_PREVENTS_SWITCH_GLOBAL_BINLOG_FORMAT	HY000
3748	ER_CLIENT_GTID_UNSAFE_CREATE_DROP_TEMP_TABLE_IN_TRX_IN_SBR	HY000
3750	ER_TABLE_WITHOUT_PK	HY000
3751	ER_WARN_DATA_TRUNCATED_FUNCTIONAL_INDEX	01000
3752	ER_WARN_DATA_OUT_OF_RANGE_FUNCTIONAL_INDEX	22003
3753	ER_FUNCTIONAL_INDEX_ON_JSON_OR_GEOMETRY_FUNCTION	42000
3754	ER_FUNCTIONAL_INDEX_REF_AUTO_INCREMENT	HY000
3755	ER_CANNOT_DROP_COLUMN_FUNCTIONAL_INDEX	HY000
3756	ER_FUNCTIONAL_INDEX_PRIMARY_KEY	HY000
3757	ER_FUNCTIONAL_INDEX_ON_LOB	HY000
3758	ER_FUNCTIONAL_INDEX_FUNCTION_IS_NOT_ALLOWED	HY000
3759	ER_FULLTEXT_FUNCTIONAL_INDEX	HY000
3760	ER_SPATIAL_FUNCTIONAL_INDEX	HY000
3761	ER_WRONG_KEY_COLUMN_FUNCTIONAL_INDEX	HY000
3762	ER_FUNCTIONAL_INDEX_ON_FIELD	HY000
3763	ER_GENERATED_COLUMN_NAMED_FUNCTION_IS_NOT_ALLOWED	HY000
3764	ER_GENERATED_COLUMN_ROW_VALUE	HY000
3765	ER_GENERATED_COLUMN_VARIABLES	HY000
3766	ER_DEPENDENT_BY_DEFAULT_GENERATED_VALUE	HY000
3767	ER_DEFAULT_VAL_GENERATED_NON_PRIOR	HY000
3768	ER_DEFAULT_VAL_GENERATED_REF_AUTO_INC	HY000
3769	ER_DEFAULT_VAL_GENERATED_FUNCTION_IS_NOT_ALLOWED	HY000
3770	ER_DEFAULT_VAL_GENERATED_NAMED_FUNCTION_IS_NOT_ALLOWED	HY000
3771	ER_DEFAULT_VAL_GENERATED_ROW_VALUE	HY000
3772	ER_DEFAULT_VAL_GENERATED_VARIABLES	HY000
3773	ER_DEFAULT_AS_VAL_GENERATED	HY000
3774	ER_UNSUPPORTED_ACTION_ON_DEFAULT_VAL_GENERATED	HY000
3775	ER_GTID_UNSAFE_ALTER_ADD_COL_WITH_DEFAULT_EXPRESSION	HY000
3776	ER_FK_CANNOT_CHANGE_ENGINE	HY000
3777	ER_WARN_DEPRECATED_USER_SET_EXPR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3778	ER_WARN_DEPRECATED_UTF8MB3_COLLATION	HY000
3779	ER_WARN_DEPRECATED_NESTED_COMMENT_SYNTAX	HY000
3780	ER_FK_INCOMPATIBLE_COLUMNS	HY000
3781	ER_GR_HOLD_WAIT_TIMEOUT	HY000
3782	ER_GR_HOLD_KILLED	HY000
3783	ER_GR_HOLD_MEMBER_STATUS_ERROR	HY000
3784	ER_RPL_ENCRYPTION_FAILED_TO_FETCH_KEY	HY000
3785	ER_RPL_ENCRYPTION_KEY_NOT_FOUND	HY000
3786	ER_RPL_ENCRYPTION_KEYRING_INVALID_KEY	HY000
3787	ER_RPL_ENCRYPTION_HEADER_ERROR	HY000
3788	ER_RPL_ENCRYPTION_FAILED_TO_ROTATE_LOGS	HY000
3789	ER_RPL_ENCRYPTION_KEY_EXISTS_UNEXPECTED	HY000
3790	ER_RPL_ENCRYPTION_FAILED_TO_GENERATE_KEY	HY000
3791	ER_RPL_ENCRYPTION_FAILED_TO_STORE_KEY	HY000
3792	ER_RPL_ENCRYPTION_FAILED_TO_REMOVE_KEY	HY000
3793	ER_RPL_ENCRYPTION_UNABLE_TO_CHANGE_OPTION	HY000
3794	ER_RPL_ENCRYPTION_MASTER_KEY_RECOVERY_FAILED	HY000
3795	ER_SLOW_LOG_MODE_IGNORED_WHEN_NOT_LOGGING_TO_FILE	HY000
3796	ER_GRP_TRX_CONSISTENCY_NOT_ALLOWED	HY000
3797	ER_GRP_TRX_CONSISTENCY_BEFORE	HY000
3798	ER_GRP_TRX_CONSISTENCY_AFTER_ON_TRX_BEGIN	HY000
3799	ER_GRP_TRX_CONSISTENCY_BEGIN_NOT_ALLOWED	HY000
3800	ER_FUNCTIONAL_INDEX_ROW_VALUE_IS_NOT_ALLOWED	HY000
3801	ER_RPL_ENCRYPTION_FAILED_TO_ENCRYPT	HY000
3802	ER_PAGE_TRACKING_NOT_STARTED	HY000
3803	ER_PAGE_TRACKING_RANGE_NOT_TRACKED	HY000
3804	ER_PAGE_TRACKING_CANNOT_PURGE	HY000
3805	ER_RPL_ENCRYPTION_CANNOT_ROTATE_BINLOG_MASTER_KEY	HY000
3806	ER_BINLOG_MASTER_KEY_RECOVERY_OUT_OF_COMBINATION	HY000
3807	ER_BINLOG_MASTER_KEY_ROTATION_FAIL_TO_OPERATE_KEY	HY000
3808	ER_BINLOG_MASTER_KEY_ROTATION_FAIL_TO_ROTATE_LOGS	HY000
3809	ER_BINLOG_MASTER_KEY_ROTATION_FAIL_TO_REENCRYPT_LOG	HY000
3810	ER_BINLOG_MASTER_KEY_ROTATION_FAIL_TO_CLEANUP_UNUSED_KEYS	HY000
3811	ER_BINLOG_MASTER_KEY_ROTATION_FAIL_TO_CLEANUP_AUX_KEY	HY000
3812	ER_NON_BOOLEAN_EXPR_FOR_CHECK_CONSTRAINT	HY000
3813	ER_COLUMN_CHECK_CONSTRAINT_REFERENCES_OTHER_COLUMN	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3814	ER_CHECK_CONSTRAINT_NAMED_FUNCTION_IS_NOT_ALLOWED	HY000
3815	ER_CHECK_CONSTRAINT_FUNCTION_IS_NOT_ALLOWED	HY000
3816	ER_CHECK_CONSTRAINT_VARIABLES	HY000
3817	ER_CHECK_CONSTRAINT_ROW_VALUE	HY000
3818	ER_CHECK_CONSTRAINT_REFERS_AUTO_INCREMENT_COLUMN	HY000
3819	ER_CHECK_CONSTRAINT_VIOLATED	HY000
3820	ER_CHECK_CONSTRAINT_REFERS_UNKNOWN_COLUMN	HY000
3821	ER_CHECK_CONSTRAINT_NOT_FOUND	HY000
3822	ER_CHECK_CONSTRAINT_DUP_NAME	HY000
3823	ER_CHECK_CONSTRAINT_CLAUSE_USING_FK_REFER_ACTION_COLUMN	HY000
3824	WARN_UNENCRYPTED_TABLE_IN_ENCRYPTED_DB	HY000
3825	ER_INVALID_ENCRYPTION_REQUEST	HY000
3826	ER_CANNOT_SET_TABLE_ENCRYPTION	HY000
3827	ER_CANNOT_SET_DATABASE_ENCRYPTION	HY000
3828	ER_CANNOT_SET_TABLESPACE_ENCRYPTION	HY000
3829	ER_TABLESPACE_CANNOT_BE_ENCRYPTED	HY000
3830	ER_TABLESPACE_CANNOT_BE_DECRYPTED	HY000
3831	ER_TABLESPACE_TYPE_UNKNOWN	HY000
3832	ER_TARGET_TABLESPACE_UNENCRYPTED	HY000
3833	ER_CANNOT_USE_ENCRYPTION_CLAUSE	HY000
3834	ER_INVALID_MULTIPLE_CLAUSES	HY000
3835	ER_UNSUPPORTED_USE_OF_GRANT_AS	HY000
3836	ER_UKNOWN_AUTH_ID_OR_ACCESS_DENIED_FOR_GRANT_AS	HY000
3837	ER_DEPENDENT_BY_FUNCTIONAL_INDEX	HY000
3838	ER_PLUGIN_NOT_EARLY	HY000
3839	ER_INNODB_REDO_LOG_ARCHIVE_START_SUBDIR_PATH	HY000
3840	ER_INNODB_REDO_LOG_ARCHIVE_START_TIMEOUT	HY000
3841	ER_INNODB_REDO_LOG_ARCHIVE_DIRS_INVALID	HY000
3842	ER_INNODB_REDO_LOG_ARCHIVE_LABEL_NOT_FOUND	HY000
3843	ER_INNODB_REDO_LOG_ARCHIVE_DIR_EMPTY	HY000
3844	ER_INNODB_REDO_LOG_ARCHIVE_NO_SUCH_DIR	HY000
3845	ER_INNODB_REDO_LOG_ARCHIVE_DIR_CLASH	HY000
3846	ER_INNODB_REDO_LOG_ARCHIVE_DIR_PERMISSIONS	HY000
3847	ER_INNODB_REDO_LOG_ARCHIVE_FILE_CREATE	HY000
3848	ER_INNODB_REDO_LOG_ARCHIVE_ACTIVE	HY000
3849	ER_INNODB_REDO_LOG_ARCHIVE_INACTIVE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3850	ER_INNODB_REDO_LOG_ARCHIVE_FAILED	HY000
3851	ER_INNODB_REDO_LOG_ARCHIVE_SESSION	HY000
3852	ER_STD_REGEX_ERROR	HY000
3853	ER_INVALID_JSON_TYPE	22032
3854	ER_CANNOT_CONVERT_STRING	HY000
3855	ER_DEPENDENT_BY_PARTITION_FUNC	HY000
3856	ER_WARN_DEPRECATED_FLOAT_AUTO_INCREMENT	HY000
3857	ER_RPL_CANT_STOP_REPLICA_WHILE_LOCKED_BACKUP	HY000
3858	ER_WARN_DEPRECATED_FLOAT_DIGITS	HY000
3859	ER_WARN_DEPRECATED_FLOAT_UNSIGNED	HY000
3860	ER_WARN_DEPRECATED_INTEGER_DISPLAY_WIDTH	HY000
3861	ER_WARN_DEPRECATED_ZEROFILL	HY000
3862	ER_CLONE_DONOR	HY000
3863	ER_CLONE_PROTOCOL	HY000
3864	ER_CLONE_DONOR_VERSION	HY000
3865	ER_CLONE_OS	HY000
3866	ER_CLONE_PLATFORM	HY000
3867	ER_CLONE_CHARSET	HY000
3868	ER_CLONE_CONFIG	HY000
3869	ER_CLONE_SYS_CONFIG	HY000
3870	ER_CLONE_PLUGIN_MATCH	HY000
3871	ER_CLONE_LOOPBACK	HY000
3872	ER_CLONE_ENCRYPTION	HY000
3873	ER_CLONE_DISK_SPACE	HY000
3874	ER_CLONE_IN_PROGRESS	HY000
3875	ER_CLONE_DISALLOWED	HY000
3876	ER_CANNOT_GRANT_ROLES_TO_ANONYMOUS_USER	HY000
3877	ER_SECONDARY_ENGINE_PLUGIN	HY000
3878	ER_SECOND_PASSWORD_CANNOT_BE_EMPTY	HY000
3879	ER_DB_ACCESS_DENIED	HY000
3880	ER_DA_AUTH_ID_WITH_SYSTEM_USER_PRIV_IN_MANDATORY_ROLES	HY000
3881	ER_DA_RPL_GTID_TABLE_CANNOT_OPEN	HY000
3882	ER_GEOMETRY_IN_UNKNOWN_LENGTH_UNIT	SU001
3883	ER_DA_PLUGIN_INSTALL_ERROR	HY000
3884	ER_NO_SESSION_TEMP	HY000
3885	ER_DA_UNKNOWN_ERROR_NUMBER	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3886	ER_COLUMN_CHANGE_SIZE	HY000
3887	ER_REGEXP_INVALID_CAPTURE_GROUP_NAME	HY000
3888	ER_DA_SSL_LIBRARY_ERROR	HY000
3889	ER_SECONDARY_ENGINE	HY000
3890	ER_SECONDARY_ENGINE_DDL	HY000
3891	ER_INCORRECT_CURRENT_PASSWORD	HY000
3892	ER_MISSING_CURRENT_PASSWORD	HY000
3893	ER_CURRENT_PASSWORD_NOT_REQUIRED	HY000
3894	ER_PASSWORD_CANNOT_BE_RETAINED_ON_PLUGIN_CHANGE	HY000
3895	ER_CURRENT_PASSWORD_CANNOT_BE_RETAINED	HY000
3896	ER_PARTIAL_REVOKES_EXIST	HY000
3897	ER_CANNOT_GRANT_SYSTEM_PRIV_TO_MANDATORY_ROLE	HY000
3898	ER_XA_REPLICATION_FILTERS	HY000
3899	ER_UNSUPPORTED_SQL_MODE	HY000
3900	ER_REGEXP_INVALID_FLAG	HY000
3901	ER_PARTIAL_REVOKE_AND_DB_GRANT_BOTH_EXISTS	HY000
3902	ER_UNIT_NOT_FOUND	SU001
3903	ER_INVALID_JSON_VALUE_FOR_FUNC_INDEX	22018
3904	ER_JSON_VALUE_OUT_OF_RANGE_FOR_FUNC_INDEX	22003
3905	ER_EXCEEDED_MV_KEYS_NUM	HY000
3906	ER_EXCEEDED_MV_KEYS_SPACE	HY000
3907	ER_FUNCTIONAL_INDEX_DATA_IS_TOO_LONG	22001
3908	ER_WRONG_MVI_VALUE	HY000
3909	ER_WARN_FUNC_INDEX_NOT_APPLICABLE	HY000
3910	ER_GRP_RPL_UDF_ERROR	HY000
3911	ER_UPDATE_GTID_PURGED_WITH_GR	HY000
3912	ER_GROUPING_ON_TIMESTAMP_IN_DST	HY000
3913	ER_TABLE_NAME_CAUSES_TOO_LONG_PATH	HY000
3914	ER_AUDIT_LOG_INSUFFICIENT_PRIVILEGE	HY000
3916	ER_DA_GRP_RPL_STARTED_AUTO_REJOIN	HY000
3917	ER_SYSVAR_CHANGE_DURING_QUERY	HY000
3918	ER_GLOBSTAT_CHANGE_DURING_QUERY	HY000
3919	ER_GRP_RPL_MESSAGE_SERVICE_INIT_FAILURE	HY000
3920	ER_CHANGE_SOURCE_WRONG_COMPRESSION_ALGORITHM_CLIENT	HY000
3921	ER_CHANGE_SOURCE_WRONG_COMPRESSION_LEVEL_CLIENT	HY000
3922	ER_WRONG_COMPRESSION_ALGORITHM_CLIENT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3923	ER_WRONG_COMPRESSION_LEVEL_CLIENT	HY000
3924	ER_CHANGE_SOURCE_WRONG_COMPRESSION_ALGORITHM_LIST_CLIENT	HY000
3925	ER_CLIENT_PRIVILEGE_CHECKS_USER_CANNOT_BE_ANONYMOUS	HY000
3926	ER_CLIENT_PRIVILEGE_CHECKS_USER_DOES_NOT_EXIST	HY000
3927	ER_CLIENT_PRIVILEGE_CHECKS_USER_CORRUPT	HY000
3928	ER_CLIENT_PRIVILEGE_CHECKS_USER_NEEDS_RPL_APPLIER_PRIV	HY000
3929	ER_WARN_DA_PRIVILEGE_NOT_REGISTERED	HY000
3930	ER_CLIENT_KEYRING_UDF_KEY_INVALID	HY000
3931	ER_CLIENT_KEYRING_UDF_KEY_TYPE_INVALID	HY000
3932	ER_CLIENT_KEYRING_UDF_KEY_TOO_LONG	HY000
3933	ER_CLIENT_KEYRING_UDF_KEY_TYPE_TOO_LONG	HY000
3934	ER_JSON_SCHEMA_VALIDATION_ERROR_WITH_DETAILED_REPORT	HY000
3935	ER_DA_UDF_INVALID_CHARSET_SPECIFIED	HY000
3936	ER_DA_UDF_INVALID_CHARSET	HY000
3937	ER_DA_UDF_INVALID_COLLATION	HY000
3938	ER_DA_UDF_INVALID_EXTENSION_ARGUMENT_TYPE	HY000
3939	ER_MULTIPLE_CONSTRAINTS_WITH_SAME_NAME	HY000
3940	ER_CONSTRAINT_NOT_FOUND	HY000
3941	ER_ALTER_CONSTRAINT_ENFORCEMENT_NOT_SUPPORTED	HY000
3942	ER_TABLE_VALUE_CONSTRUCTOR_MUST_HAVE_COLUMNS	HY000
3943	ER_TABLE_VALUE_CONSTRUCTOR_CANNOT_HAVE_DEFAULT	HY000
3944	ER_CLIENT_QUERY_FAILURE_INVALID_NON_ROW_FORMAT	HY000
3945	ER_REQUIRE_ROW_FORMAT_INVALID_VALUE	HY000
3946	ER_FAILED_TO_DETERMINE_IF_ROLE_IS_MANDATORY	HY000
3947	ER_FAILED_TO_FETCH_MANDATORY_ROLE_LIST	HY000
3948	ER_CLIENT_LOCAL_FILES_DISABLED	42000
3949	ER_IMP_INCOMPATIBLE_CFG_VERSION	HY000
3950	ER_DA_OOM	HY000
3951	ER_DA_UDF_INVALID_ARGUMENT_TO_SET_CHARSET	HY000
3952	ER_DA_UDF_INVALID_RETURN_TYPE_TO_SET_CHARSET	HY000
3953	ER_MULTIPLE_INTO_CLAUSES	HY000
3954	ER_MISPLACED_INTO	HY000
3955	ER_USER_ACCESS_DENIED_FOR_USER_ACCOUNT_BLOCKED_BY_PASSWORD_LOCK	HY000
3956	ER_WARN_DEPRECATED_YEAR_UNSIGNED	HY000
3957	ER_CLONE_NETWORK_PACKET	HY000
3958	ER_SDI_OPERATION_FAILED_MISSING_RECORD	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3959	ER_DEPENDENT_BY_CHECK_CONSTRAINT	HY000
3960	ER_GRP_OPERATION_NOT_ALLOWED_GR_MUST_STOP	HY000
3961	ER_WARN_DEPRECATED_JSON_TABLE_ON_ERROR_ON_EMPTY	HY000
3962	ER_WARN_DEPRECATED_INNER_INTO	HY000
3963	ER_WARN_DEPRECATED_VALUES_FUNCTION_ALWAYS_NULL	HY000
3964	ER_WARN_DEPRECATED_SQL_CALC_FOUND_ROWS	HY000
3965	ER_WARN_DEPRECATED_FOUND_ROWS	HY000
3966	ER_MISSING_JSON_VALUE	22035
3967	ER_MULTIPLE_JSON_VALUES	22034
3968	ER_HOSTNAME_TOO_LONG	HY000
3970	ER_GROUP_REPLICATION_USER_EMPTY_MSG	HY000
3971	ER_GROUP_REPLICATION_USER_MANDATORY_MSG	HY000
3972	ER_GROUP_REPLICATION_PASSWORD_LENGTH	HY000
3973	ER_SUBQUERY_TRANSFORM_REJECTED	HY000
3974	ER_DA_GRP_RPL_RECOVERY_ENDPOINT_FORMAT	HY000
3975	ER_DA_GRP_RPL_RECOVERY_ENDPOINT_INVALID	HY000
3976	ER_WRONG_VALUE_FOR_VAR_PLUS_ACTIONABLE_PART	HY000
3977	ER_STATEMENT_NOT_ALLOWED_AFTER_START_TRANSACTION	HY000
3978	ER_FOREIGN_KEY_WITH_ATOMIC_CREATE_SELECT	HY000
3979	ER_NOT_ALLOWED_WITH_START_TRANSACTION	HY000
3980	ER_INVALID_JSON_ATTRIBUTE	HY000
3981	ER_ENGINE_ATTRIBUTE_NOT_SUPPORTED	HY000
3982	ER_INVALID_USER_ATTRIBUTE_JSON	HY000
3983	ER_INNODB_REDO_DISABLED	HY000
3984	ER_INNODB_REDO_ARCHIVING_ENABLED	HY000
3985	ER_MDL_OUT_OF_RESOURCES	HY000
3986	ER_IMPLICIT_COMPARISON_FOR_JSON	HY000
3987	ER_FUNCTION_DOES_NOT_SUPPORT_CHARACTER_SET	HY000
3988	ER_IMPOSSIBLE_STRING_CONVERSION	HY000
3989	ER_SCHEMA_READ_ONLY	HY000
3990	ER_RPL_ASYNC_RECONNECT_GTID_MODE_OFF	HY000
3991	ER_RPL_ASYNC_RECONNECT_AUTO_POSITION_OFF	HY000
3992	ER_DISABLE_GTID_MODE_REQUIRES_ASYNC_RECONNECT_OFF	HY000
3993	ER_DISABLE_AUTO_POSITION_REQUIRES_ASYNC_RECONNECT_OFF	HY000
3994	ER_INVALID_PARAMETER_USE	HY000
3995	ER_CHARACTER_SET_MISMATCH	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
3996	ER_WARN_VAR_VALUE_CHANGE_NOT_SUPPORTED	HY000
3997	ER_INVALID_TIME_ZONE_INTERVAL	HY000
3998	ER_INVALID_CAST	HY000
3999	ER_HYPERGRAPH_NOT_SUPPORTED_YET	42000
4000	ER_WARN_HYPERGRAPH_EXPERIMENTAL	HY000
4001	ER_DA_NO_ERROR_LOG_PARSER_CONFIGURED	HY000
4002	ER_DA_ERROR_LOG_TABLE_DISABLED	HY000
4003	ER_DA_ERROR_LOG_MULTIPLE_FILTERS	HY000
4004	ER_DA_CANT_OPEN_ERROR_LOG	HY000
4005	ER_USER_REFERENCED_AS_DEFINER	HY000
4006	ER_CANNOT_USER_REFERENCED_AS_DEFINER	HY000
4007	ER_REGEX_NUMBER_TOO_BIG	HY000
4008	ER_SPVAR_NONINTEGER_TYPE	HY000
4009	WARN_UNSUPPORTED_ACL_TABLES_READ	HY000
4010	ER_BINLOG_UNSAFE_ACL_TABLE_READ_IN_DML_DDL	HY000
4011	ER_STOP_REPLICA_MONITOR_IO_THREAD_TIMEOUT	HY000
4012	ER_STARTING_REPLICA_MONITOR_IO_THREAD	HY000
4013	ER_CANT_USE_ANONYMOUS_TO_GTID_WITH_GTID_MODE_NOT_ON	HY000
4014	ER_CANT_COMBINE_ANONYMOUS_TO_GTID_AND_AUTOPOSITION	HY000
4015	ER_ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS_REQUIRES_GTID_MODE_ON	HY000
4016	ER_SQL_REPLICA_SKIP_COUNTER_USED_WITH_GTID_MODE_ON	HY000
4017	ER_USING_ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS_AS_LOCAL_OR_UUID	HY000
4019	ER_CANT_SET_SQL_AFTER_OR_BEFORE_GTIDS_WITH_ANONYMOUS_TO_GTID	HY000
4020	ER_ANONYMOUS_TO_GTID_UUID_SAME_AS_GROUP_NAME	HY000
4021	ER_CANT_USE_SAME_UUID_AS_GROUP_NAME	HY000
4022	ER_GRP_RPL_RECOVERY_CHANNEL_STILL_RUNNING	HY000
4023	ER_INNODB_INVALID_AUTOEXTEND_SIZE_VALUE	HY000
4024	ER_INNODB_INCOMPATIBLE_WITH_TABLESPACE	HY000
4025	ER_INNODB_AUTOEXTEND_SIZE_OUT_OF_RANGE	HY000
4026	ER_CANNOT_USE_AUTOEXTEND_SIZE_CLAUSE	HY000
4027	ER_ROLE_GRANTED_TO_ITSELF	HY000
4028	ER_TABLE_MUST_HAVE_A_VISIBLE_COLUMN	HY000
4029	ER_INNODB_COMPRESSION_FAILURE	HY000
4030	ER_WARN_ASYNC_CONN_FAILOVER_NETWORK_NAMESPACE	HY000
4031	ER_CLIENT_INTERACTION_TIMEOUT	HY000
4032	ER_INVALID_CAST_TO_GEOMETRY	22S01

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
4033	ER_INVALID_CAST_POLYGON_RING_DIRECTION	22S04
4034	ER_GIS_DIFFERENT_SRIDS_AGGREGATION	22S05
4035	ER_RELOAD_KEYRING_FAILURE	HY000
4036	ER_SDI_GET_KEYS_INVALID_TABLESPACE	HY000
4037	ER_CHANGE_RPL_SRC_WRONG_COMPRESSION_ALGORITHM_SIZE	HY000
4039	ER_CANT_USE_SAME_UUID_AS_VIEW_CHANGE_UUID	HY000
4040	ER_ANONYMOUS_TO_GTID_UUID_SAME_AS_VIEW_CHANGE_UUID	HY000
4041	ER_GRP_RPL_VIEW_CHANGE_UUID_FAIL_GET_VARIABLE	HY000
4042	ER_WARN_ADUIT_LOG_MAX_SIZE_AND_PRUNE_SECONDS	HY000
4043	ER_WARN_ADUIT_LOG_MAX_SIZE_CLOSE_TO_ROTATE_ON_SIZE	HY000
4044	ER_KERBEROS_CREATE_USER	HY000
4045	ER_INSTALL_PLUGIN_CONFLICT_CLIENT	HY000
4046	ER_DA_ERROR_LOG_COMPONENT_FLUSH_FAILED	HY000
4047	ER_WARN_SQL_AFTER_MTS_GAPS_GAP_NOT_CALCULATED	HY000
4048	ER_INVALID_ASSIGNMENT_TARGET	42000
4049	ER_OPERATION_NOT_ALLOWED_ON_GR_SECONDARY	HY000
4050	ER_GRP_RPL_FAILOVER_CHANNEL_STATUS_PROPAGATION	HY000
4051	ER_WARN_AUDIT_LOG_FORMAT_UNIX_TIMESTAMP_ONLY_WHEN_JSON	HY000
4052	ER_INVALID_MFA_PLUGIN_SPECIFIED	HY000
4053	ER_IDENTIFIED_BY_UNSUPPORTED	HY000
4054	ER_INVALID_PLUGIN_FOR_REGISTRATION	HY000
4055	ER_PLUGIN_REQUIRES_REGISTRATION	HY000
4056	ER_MFA_METHOD_EXISTS	HY000
4057	ER_MFA_METHOD_NOT_EXISTS	HY000
4058	ER_AUTHENTICATION_POLICY_MISMATCH	HY000
4059	ER_PLUGIN_REGISTRATION_DONE	HY000
4060	ER_INVALID_USER_FOR_REGISTRATION	HY000
4061	ER_USER_REGISTRATION_FAILED	HY000
4062	ER_MFA_METHODS_INVALID_ORDER	HY000
4063	ER_MFA_METHODS_IDENTICAL	HY000
4064	ER_INVALID_MFA_OPERATIONS_FOR_PASSWORDLESS_USER	HY000
4065	ER_CHANGE_REPLICATION_SOURCE_NO_OPTIONS_FOR_GTID_ONLY	HY000
4066	ER_CHANGE_REP_SOURCE_CANT_DISABLE_REQ_ROW_FORMAT_WITH_GTID_ONLY	HY000
4067	ER_CHANGE_REP_SOURCE_CANT_DISABLE_AUTO_POSITION_WITH_GTID_ONLY	HY000
4068	ER_CHANGE_REP_SOURCE_CANT_DISABLE_GTID_ONLY_WITHOUT_POSITIONS	HY000
4069	ER_CHANGE_REP_SOURCE_CANT_DISABLE_AUTO_POS_WITHOUT_POSITIONS	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
4070	ER_CHANGE_REP_SOURCE_GR_CHANNEL_WITH_GTID_MODE_NOT_ON	HY000
4071	ER_CANT_USE_GTID_ONLY_WITH_GTID_MODE_NOT_ON	HY000
4072	ER_WARN_C_DISABLE_GTID_ONLY_WITH_SOURCE_AUTO_POS_INVALID_POS	HY000
4073	ER_DA_SSL_FIPS_MODE_ERROR	HY000
4074	ER_VALUE_OUT_OF_RANGE	HY000
4075	ER_FULLTEXT_WITH_ROLLUP	HY000
4076	ER_REGEXP_MISSING_RESOURCE	HY000
4077	ER_WARN_REGEXP_USING_DEFAULT	HY000
4078	ER_REGEXP_MISSING_FILE	HY000
4079	ER_WARN_DEPRECATED_COLLATION	HY000
4080	ER_CONCURRENT_PROCEDURE_USAGE	HY000
4081	ER_DA_GLOBAL_CONN_LIMIT	HY000
4082	ER_DA_CONN_LIMIT	HY000
4083	ER_ALTER_OPERATION_NOT_SUPPORTED_REASON_COLUMN_TYPE_INSTANT	HY000
4084	ER_WARN_SF_UDF_NAME_COLLISION	HY000
4085	ER_CANNOT_PURGE_BINLOG_WITH_BACKUP_LOCK	HY000
4086	ER_TOO_MANY_WINDOWS	HY000
4087	ER_MYSQLBACKUP_CLIENT_MSG	HY000
4088	ER_COMMENT_CONTAINS_INVALID_STRING	HY000
4089	ER_DEFINITION_CONTAINS_INVALID_STRING	HY000
4090	ER_CANT_EXECUTE_COMMAND_WITH_ASSIGNED_GTID_NEXT	HY000
4091	ER_XA_TEMP_TABLE	HY000
4092	ER_INNODB_MAX_ROW_VERSION	HY000
4094	ER_OPERATION_NOT_ALLOWED_WHILE_PRIMARY_CHANGE_IS_RUNNING	HY000
4095	ER_WARN_DEPRECATED_DATETIME_DELIMITER	HY000
4096	ER_WARN_DEPRECATED_SUPERFLUOUS_DELIMITER	HY000
4097	ER_CANNOT_PERSIST_SENSITIVE_VARIABLES	HY000
4098	ER_WARN_CANNOT_SECURELY_PERSIST_SENSITIVE_VARIABLES	HY000
4099	ER_WARN_TRG_ALREADY_EXISTS	HY000
4100	ER_IF_NOT_EXISTS_UNSUPPORTED_TRG_EXISTS_ON_DIFFERENT_TABLE	HY000
4101	ER_IF_NOT_EXISTS_UNSUPPORTED_UDF_NATIVE_FCT_NAME_COLLISION	HY000
4102	ER_SET_PASSWORD_AUTH_PLUGIN_ERROR	HY000
4105	ER_SRS_INVALID_LATITUDE_OF_ORIGIN	SR002
4106	ER_SRS_INVALID_LONGITUDE_OF_ORIGIN	SR002
4107	ER_SRS_UNUSED_PROJ_PARAMETER_PRESENT	SR002
4108	ER_GIPK_COLUMN_EXISTS	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
4109	ER_GIPK_FAILED_AUTOINC_COLUMN_EXISTS	HY000
4110	ER_GIPK_COLUMN_ALTER_NOT_ALLOWED	HY000
4111	ER_DROP_PK_COLUMN_TO_DROP_GIPK	HY000
4112	ER_CREATE_SELECT_WITH_GIPK_DISALLOWED_IN_SBR	HY000
4114	ER_CTE_RECURSIVE_NOT_UNION	HY000
4115	ER_COMMAND_BACKEND_FAILED_TO_FETCH_SECURITY_CTX	HY000
4116	ER_COMMAND_SERVICE_BACKEND_FAILED	HY000
4117	ER_CLIENT_FILE_PRIVILEGE_FOR_REPLICATION_CHECKS	HY000
4118	ER_GROUP_REPLICATION_FORCE_MEMBERS_COMMAND_FAILURE	HY000
4119	ER_WARN_DEPRECATED_IDENT	HY000
4120	ER_INTERSECT_ALL_MAX_DUPLICATES_EXCEEDED	HY000
4121	ER_TP_QUERY_THRS_PER_GRP_EXCEEDS_TXN_THR_LIMIT	HY000
4122	ER_BAD_TIMESTAMP_FORMAT	HY000
4123	ER_SHAPE_PRIDICTION_UDF	HY000
4124	ER_SRS_INVALID_HEIGHT	SR002
4125	ER_SRS_INVALID_SCALING	SR002
4126	ER_SRS_INVALID_ZONE_WIDTH	SR002
4127	ER_SRS_INVALID_LATITUDE_POLAR_STERE_VAR_A	SR002
4128	ER_WARN_DEPRECATED_CLIENT_NO_SCHEMA_OPTION	HY000
4129	ER_TABLE_NOT_EMPTY	HY000
4130	ER_TABLE_NO_PRIMARY_KEY	HY000
4131	ER_TABLE_IN_SHARED_TABLESPACE	HY000
4132	ER_INDEX_OTHER_THAN_PK	HY000
4133	ER_LOAD_BULK_DATA_UNSORTED	HY000
4134	ER_BULK_EXECUTOR_ERROR	HY000
4135	ER_BULK_READER_LIBCURL_INIT_FAILED	HY000
4136	ER_BULK_READER_LIBCURL_ERROR	HY000
4137	ER_BULK_READER_SERVER_ERROR	HY000
4138	ER_BULK_READER_COMMUNICATION_ERROR	HY000
4139	ER_BULK_LOAD_DATA_FAILED	HY000
4140	ER_BULK_LOADER_COLUMN_TOO_BIG_FOR_LEFTOVER_BUFFER	HY000
4141	ER_BULK_LOADER_COMPONENT_ERROR	HY000
4142	ER_BULK_LOADER_FILE_CONTAINS_LESS_LINES_THAN_IGNORE_CLAUSE	HY000
4143	ER_BULK_PARSER_MISSING_ENCLOSED_BY	HY000
4144	ER_BULK_PARSER_ROW_BUFFER_MAX_TOTAL_COLS_EXCEEDED	HY000
4145	ER_BULK_PARSER_COPY_BUFFER_SIZE_EXCEEDED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
4146	ER_BULK_PARSER_UNEXPECTED_END_OF_INPUT	HY000
4147	ER_BULK_PARSER_UNEXPECTED_ROW_TERMINATOR	HY000
4148	ER_BULK_PARSER_UNEXPECTED_CHAR_AFTER_ENDING_ENCLOSED_BY	HY000
4149	ER_BULK_PARSER_UNEXPECTED_CHAR_AFTER_NULL_ESCAPE	HY000
4150	ER_BULK_PARSER_UNEXPECTED_CHAR_AFTER_COLUMN_TERMINATOR	HY000
4151	ER_BULK_PARSER_INCOMPLETE_ESCAPE_SEQUENCE	HY000
4152	ER_LOAD_BULK_DATA_FAILED	HY000
4153	ER_LOAD_BULK_DATA_WRONG_VALUE_FOR_FIELD	HY000
4154	ER_LOAD_BULK_DATA_WARN_NULL_TO_NOTNULL	HY000
4155	ER_REQUIRE_TABLE_PRIMARY_KEY_CHECK_GENERATE_WITH_GR	HY000
4156	ER_CANT_CHANGE_SYS_VAR_IN_READ_ONLY_MODE	HY000
4157	ER_INNODB_INSTANT_ADD_DROP_NOT_SUPPORTED_MAX_SIZE	HY000
4158	ER_INNODB_INSTANT_ADD_NOT_SUPPORTED_MAX_FIELDS	HY000
4159	ER_CANT_SET_PERSISTED	HY000
4160	ER_INSTALL_COMPONENT_SET_NULL_VALUE	HY000
4161	ER_INSTALL_COMPONENT_SET_UNUSED_VALUE	HY000
4162	ER_WARN_DEPRECATED_USER_DEFINED_COLLATIONS	HY000
4163	ER_USER_LOCK_OVERLONG_NAME	42000
4164	ER_WARN_NO_SPACE_VERSION_COMMENT	HY000
4165	ER_VALIDATE_PASSWORD_INSUFFICIENT_CHANGED_CHARACTERS	HY000
4166	ER_WARN_DEPRECATED_WITH_NOTE	HY000
5000	ER_X_BAD_MESSAGE	HY000
5001	ER_X_CAPABILITIES_PREPARE_FAILED	HY000
5002	ER_X_CAPABILITY_NOT_FOUND	HY000
5003	ER_X_INVALID_PROTOCOL_DATA	HY000
5004	ER_X_BAD_CONNECTION_SESSION_ATTRIBUTE_VALUE_LENGTH	HY000
5005	ER_X_BAD_CONNECTION_SESSION_ATTRIBUTE_KEY_LENGTH	HY000
5006	ER_X_BAD_CONNECTION_SESSION_ATTRIBUTE_EMPTY_KEY	HY000
5007	ER_X_BAD_CONNECTION_SESSION_ATTRIBUTE_LENGTH	HY000
5008	ER_X_BAD_CONNECTION_SESSION_ATTRIBUTE_TYPE	HY000
5009	ER_X_CAPABILITY_SET_NOT_ALLOWED	HY000
5010	ER_X_SERVICE_ERROR	HY000
5011	ER_X_SESSION	HY000
5012	ER_X_INVALID_ARGUMENT	HY000
5013	ER_X_MISSING_ARGUMENT	HY000
5014	ER_X_BAD_INSERT_DATA	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
5015	ER_X_CMD_NUM_ARGUMENTS	HY000
5016	ER_X_CMD_ARGUMENT_TYPE	HY000
5017	ER_X_CMD_ARGUMENT_VALUE	HY000
5018	ER_X_BAD_UPSERT_DATA	HY000
5019	ER_X_DUPLICATED_CAPABILITIES	HY000
5020	ER_X_CMD_ARGUMENT_OBJECT_EMPTY	HY000
5021	ER_X_CMD_INVALID_ARGUMENT	HY000
5050	ER_X_BAD_UPDATE_DATA	HY000
5051	ER_X_BAD_TYPE_OF_UPDATE	HY000
5052	ER_X_BAD_COLUMN_TO_UPDATE	HY000
5053	ER_X_BAD_MEMBER_TO_UPDATE	HY000
5110	ER_X_BAD_STATEMENT_ID	HY000
5111	ER_X_BAD_CURSOR_ID	HY000
5112	ER_X_BAD_SCHEMA	HY000
5113	ER_X_BAD_TABLE	HY000
5114	ER_X_BAD_PROJECTION	HY000
5115	ER_X_DOC_ID_MISSING	HY000
5116	ER_X_DUPLICATE_ENTRY	HY000
5117	ER_X_DOC_REQUIRED_FIELD_MISSING	HY000
5120	ER_X_PROJ_BAD_KEY_NAME	HY000
5121	ER_X_BAD_DOC_PATH	HY000
5122	ER_X_CURSOR_EXISTS	HY000
5123	ER_X_CURSOR_REACHED_EOF	HY000
5131	ER_X_PREPARED_STATMENT_CAN_HAVE_ONE_CURSOR	HY000
5133	ER_X_PREPARED_EXECUTE_ARGUMENT_NOT_SUPPORTED	HY000
5134	ER_X_PREPARED_EXECUTE_ARGUMENT_CONSISTENCY	HY000
5150	ER_X_EXPR_BAD_OPERATOR	HY000
5151	ER_X_EXPR_BAD_NUM_ARGS	HY000
5152	ER_X_EXPR_MISSING_ARG	HY000
5153	ER_X_EXPR_BAD_TYPE_VALUE	HY000
5154	ER_X_EXPR_BAD_VALUE	HY000
5156	ER_X_INVALID_COLLECTION	HY000
5157	ER_X_INVALID_ADMIN_COMMAND	HY000
5158	ER_X_EXPECT_NOT_OPEN	HY000
5159	ER_X_EXPECT_NO_ERROR_FAILED	HY000
5160	ER_X_EXPECT_BAD_CONDITION	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
5161	ER_X_EXPECT_BAD_CONDITION_VALUE	HY000
5162	ER_X_INVALID_NAMESPACE	HY000
5163	ER_X_BAD_NOTICE	HY000
5164	ER_X_CANNOT_DISABLE_NOTICE	HY000
5165	ER_X_BAD_CONFIGURATION	HY000
5167	ER_X_MYSQLX_ACCOUNT_MISSING_PERMISSIONS	HY000
5168	ER_X_EXPECT_FIELD_EXISTS_FAILED	HY000
5169	ER_X_BAD_LOCKING	HY000
5170	ER_X_FRAME_COMPRESSION_DISABLED	HY000
5171	ER_X_DECOMPRESSION_FAILED	HY000
5174	ER_X_BAD_COMPRESSED_FRAME	HY000
5175	ER_X_CAPABILITY_COMPRESSION_INVALID_ALGORITHM	HY000
5176	ER_X_CAPABILITY_COMPRESSION_INVALID_SERVER_STYLE	HY000
5177	ER_X_CAPABILITY_COMPRESSION_INVALID_CLIENT_STYLE	HY000
5178	ER_X_CAPABILITY_COMPRESSION_INVALID_OPTION	HY000
5179	ER_X_CAPABILITY_COMPRESSION_MISSING_REQUIRED_FIELDS	HY000
5180	ER_X_DOCUMENT_DOESNT_MATCH_EXPECTED_SCHEMA	HY000
5181	ER_X_COLLECTION_OPTION_DOESNT_EXISTS	HY000
5182	ER_X_INVALID_VALIDATION_SCHEMA	HY000
6000	ER_LANGUAGE_COMPONENT	HY000
6001	ER_LANGUAGE_COMPONENT_NOT_AVAILABLE	HY000
6002	ER_LANGUAGE_COMPONENT_UNSUPPORTED_LANGUAGE	HY000
6003	ER_LANGUAGE_COMPONENT_CANNOT_UNINSTALL	HY000
6004	ER_SP_NO_ALTER_LANGUAGE	HY000
6005	ER_EXPLAIN_INTO_ANALYZE_NOT_SUPPORTED	0A000
6006	ER_EXPLAIN_INTO_IMPLICIT_FORMAT_NOT_SUPPORTED	0A000
6007	ER_EXPLAIN_INTO_FORMAT_NOT_SUPPORTED	0A000
6008	ER_NULL_CANT_BE_PERSISTED_FOR_READONLY	HY000
6009	ER_EXPLAIN_INTO_FOR_CONNECTION_NOT_SUPPORTED	0A000
6010	ER_INNODB_IMPORT_WRONG_DROPPED_ENUM_LENGTH	HY000
6011	ER_INNODB_IMPORT_WRONG_NUMBER_OF_INDEXES_ZERO	HY000
6012	ER_INNODB_IMPORT_WRONG_NUMBER_OF_INDEXES_TOO_HIGH	HY000
6013	ER_INNODB_IMPORT_DROP_COL_METADATA_MISMATCH	HY000
6014	ER_INNODB_IMPORT_ENUM_NULL_TERMINATOR_MISSING	HY000
6015	ER_SIMULATED_INJECTION_ERROR	HY000
6016	ER_WARN_DEPRECATED_DYNAMIC_PRIV_IN_GRANT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
6017	ER_BULK_MULTI_READER_OPEN_FILE_FAILED	HY000
6018	ER_BULK_MULTI_READER_READ_FILE_FAILED	HY000
6019	ER_BULK_MERGE_INVALID_CHUNK	HY000
6020	ER_BULK_MERGE_NOT_ALL_CHUNKS_CONSUMED	HY000
6021	ER_BULK_WRITER_LIBCURL_INIT_FAILED	HY000
6022	ER_BULK_WRITER_LIBCURL_ERROR	HY000
6023	ER_BULK_SORTING_LOADER_WRITE	HY000
6024	ER_BULK_SORTING_LOADER_WAIT	HY000
6025	ER_BULK_READER_OPEN_FILE_FAILED	HY000
6026	ER_BULK_LOAD_TABLE_HAS_INSTANT_COLS	HY000
6027	ER_BULK_LOAD_RESOURCE	HY000
6028	ER_BULK_LOAD_SECONDARY_ENGINE	HY000
6029	ER_BULK_READER_ERROR	HY000
6030	ER_BULK_READER_FILE_DOESNT_EXIST	HY000
6031	ER_BULK_READER_COULDNT_RESOLVE_HOST	HY000
6032	ER_START_REPLICA_CHANNEL_INVALID_CONFIGURATION	HY000
6033	ER_CANNOT_EXECUTE_IN_PRIMARY	HY000
6034	ER_TOO_MANY_GROUP_BY_MODIFIER_BRANCHES	HY000
6035	ER_WARN_DEPRECATED_ENGINE_SYNTAX_NO_REPLACEMENT	HY000
6036	ER_QUALIFY_WITHOUT_WINDOW_FUNCTION	HY000
6037	ER_SUPPORTED_ONLY_WITH_HYPERGRAPH	HY000
6038	ER_SPECIFIC_ACCESS_DENIED	HY000
6039	ER_CANT_SET_GTID_NEXT_TO_AUTOMATIC_TAGGED_WHEN_GTID_MODE_IS_OFF	HY000
6040	ER_GTID_NEXT_TAG_GTID_MODE_OFF	HY000
6041	ER_LH_COL_NOT_NULLABLE	HY000
6042	ER_LH_WARN_COL_MISSING_NOT_NULLABLE	HY000
6043	ER_LH_COL_IS_EMPTY	HY000
6044	ER_LH_COL_IS_EMPTY_WARN	HY000
6045	ER_LH_BAD_VALUE	HY000
6046	ER_LH_DECIMAL_UNKNOWN_ERR	HY000
6047	ER_LH_DECIMAL_OOM_ERR	HY000
6048	ER_LH_WARN_DECIMAL_ROUNDING	HY000
6049	ER_LH_DECIMAL_PRECISION_EXCEEDS_SCHEMA	HY000
6050	ER_LH_EXCEEDS_MIN	HY000
6051	ER_LH_EXCEEDS_MAX	HY000
6052	ER_LH_WARN_EXCEEDS_MIN_TRUNCATING	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
6053	ER_LH_WARN_EXCEEDS_MAX_TRUNCATING	HY000
6054	ER_LH_REAL_IS_NAN	HY000
6055	ER_LH_OUT_OF_RANGE	HY000
6056	ER_LH_DATETIME_FORMAT	HY000
6057	ER_LH_WARN_TRUNCATED	HY000
6058	ER_LH_CANNOT_CONVERT_STRING	HY000
6059	ER_LH_RESOURCE_PRINCIPAL_ERR	HY000
6060	ER_LH_AWS_AUTH_ERR	HY000
6061	ER_LH_CSV_PARSING_ERR	HY000
6062	ER_LH_COLUMN_MISMATCH_ERR	HY000
6063	ER_LH_COLUMN_MAX_ERR	HY000
6064	ER_LH_CHARSET_UNSUPPORTED	HY000
6065	ER_LH_PARQUET_DECIMAL_CONVERSION_ERR	HY000
6066	ER_LH_STRING_TOO_LONG	HY000
6067	ER_LH_RESOURCE_PRINCIPAL_BUCKET_ERR	HY000
6068	ER_LH_NO_FILES_FOUND	HY000
6069	ER_LH_EMPTY_FILE	HY000
6070	ER_LH_DUPLICATE_FILE	HY000
6071	ER_LH_AVRO_SCHEMA_DEPTH_EXCEEDS_MAX	HY000
6072	ER_LH_AVRO_HEADER_MISMATCH	HY000
6073	ER_LH_AVRO_ENUM_CANNOT_CONVERT_CHARSET	HY000
6074	ER_LH_AVRO_ENUM_MISMATCH	HY000
6075	ER_LH_AVRO_TYPE_CANNOT_CONVERT	HY000
6076	ER_LH_AVRO_FILE_ENDS_UNEXPECTEDLY	HY000
6077	ER_LH_AVRO_FILE_DATA_CORRUPT	HY000
6078	ER_LH_AVRO_INVALID_UNION	HY000
6079	ER_LH_AVRO_INVALID_BLOCK_SIZE	HY000
6080	ER_LH_AVRO_INVALID_BLOCK_RECORD_COUNT	HY000
6081	ER_LH_FORMAT_HEADER_NO_MAGIC_BYTES	HY000
6082	ER_LH_AVRO_HEADER_METADATA_ERR	HY000
6083	ER_LH_AVRO_HEADER_NO_SCHEMA	HY000
6084	ER_LH_AVRO_NO_CODEC_IN_HEADER	HY000
6085	ER_LH_AVRO_INVALID_NAME_IN_SCHEMA	HY000
6086	ER_LH_AVRO_DECODING_ERR	HY000
6087	ER_LH_PARQUET_NON_UTF8_FILE_ENC	HY000
6088	ER_LH_PARQUET_SCHEMA_MISMATCH	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
6089	ER_LH_PARQUET_ROW_GROUP_SIZE_EXCEEDS_MAX	HY000
6090	ER_LH_PARQUET_CANNOT_LOCATE_OFFSET	HY000
6091	ER_LH_PARQUET_TYPE_CANNOT_CONVERT	HY000
6092	ER_LH_PARQUET_CANNOT_LOCATE_SCHEMA	HY000
6093	ER_LH_INFER_SCHEMA_MISMATCH	HY000
6094	ER_LH_OOM	HY000
6095	ER_LH_WARN_INFER_SKIPPED_LINES	HY000
6096	ER_LH_WARN_INFER_SKIPPED_FILES	HY000
6097	ER_LH_INFER_FILE_HAS_NO_DATA	HY000
6098	ER_LH_INFER_NO_DATA	HY000
6099	ER_LH_INFER_NO_FILES	HY000
6100	ER_LH_WARN_INFER_USE_DEFAULT_COL_NAMES	HY000
6101	ER_LH_PARQUET_CANNOT_READ_HEADER	HY000
6102	ER_LH_INFER_WARN_GOT_EXCEPTION	HY000
6103	ER_LH_AVRO_CANNOT_PARSE_HEADER	HY000
6104	ER_LH_PARQUET_CANT_OPEN_FILE	HY000
6105	ER_LH_TOO_LARGE_VALUE_ERR	HY000
6106	ER_LH_TOO_LARGE_ROW_ERR	HY000
6107	ER_TABLESAMPLE_PERCENTAGE	2202H
6108	ER_TABLESAMPLE_ONLY_ON_BASE_TABLES	HY000
6110	ER_RESULT_SIZE_LIMIT_EXCEEDED	HY000
6111	ER_LANGUAGE_COMPONENT_INTERNAL	HY000
6112	ER_LANGUAGE_COMPONENT_CONCURRENCY_LIMIT	HY000
6113	ER_LANGUAGE_COMPONENT_RUNTIME	HY000
6114	ER_LANGUAGE_COMPONENT_TIMEZONE	HY000
6115	ER_LANGUAGE_COMPONENT_KEYWORD	HY000
6116	ER_LANGUAGE_COMPONENT_SET_SYSTEM_VARIABLE	HY000
6117	ER_LANGUAGE_COMPONENT_UNSUPPORTED_TYPE	HY000
6118	ER_LANGUAGE_COMPONENT_CONVERSION	HY000
6119	ER_WARN_SP_STATEMENT_PARTIALLY_EXECUTED	HY000
6120	ER_STMT_EXECUTION_NOT_ALLOWED_WITHIN_SP_OR_TRG_OR_UDF	HY000
6121	ER_LH_JSON_PARSING	HY000
6122	ER_ENGINE_CANNOT_BE_DEFAULT	HY000
6123	ER_PARTITION_PREFIX_KEY_NOT_SUPPORTED	HY000
6124	ER_WARN_DEPRECATED_NON_STANDARD_KEY	HY000
6125	ER_FK_NO_UNIQUE_INDEX_PARENT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
6126	ER_ACCESS_DENIED_NO_PROXY_GRANT	HY000
6127	ER_ACCESS_DENIED_NO_PROXY	HY000
6128	ER_LH_USER_DATA_ACCESS_FAILED	HY000
6129	ER_BULK_READER_ZSTD_ERROR	HY000
6130	ER_BULK_PARSER_ERROR	HY000
6131	ER_LH_INVALID_JSON_FILE_FORMAT_SCHEMA	HY000
6132	ER_LH_INFER_JSON_INVALID_SCHEMA	HY000
6133	ER_LH_JSON_FILE_FORMAT_WARN_INFER_SCHEMA	HY000
6134	ER_NON_SCALAR_USED_AS_KEY	HY000
6135	ER_INCOMPATIBLE_TYPE_AGG	HY000
6136	ER_DATA_INCOMPATIBLE_WITH_VECTOR	HY000
6137	ER_EXCEEDS_VECTOR_MAX_DIMENSIONS	HY000
6138	ER_TO_VECTOR_CONVERSION	HY000
6139	ER_EXTERNAL_UNSUPPORTED_INDEX_ALGORITHM	HY000
10000	ER_PARSER_TRACE	XX999
10001	ER_BOOTSTRAP_CANT_THREAD	HY000
10002	ER_TRIGGER_INVALID_VALUE	HY000
10003	ER_OPT_WRONG_TREE	HY000
10004	ER_DD_FAILSAFE	HY000
10005	ER_DD_NO_WRITES_NO_REPOPULATION	HY000
10006	ER_DD_VERSION_FOUND	HY000
10007	ER_DD_VERSION_INSTALLED	HY000
10008	ER_DD_VERSION_UNSUPPORTED	HY000
10010	ER_LOG_SYSLOG_CANNOT_OPEN	HY000
10011	ER_LOG_SLOW_CANNOT_OPEN	HY000
10012	ER_LOG_GENERAL_CANNOT_OPEN	HY000
10013	ER_LOG_CANNOT_WRITE	HY000
10014	ER_RPL_ZOMBIE_ENCOUNTERED	HY000
10015	ER_RPL_GTID_TABLE_CANNOT_OPEN	HY000
10016	ER_SYSTEM_SCHEMA_NOT_FOUND	HY000
10017	ER_DD_INIT_UPGRADE_FAILED	HY000
10018	ER_VIEW_UNKNOWN_CHARSET_OR_COLLATION	HY000
10019	ER_DD_VIEW_CANT_ALLOC_CHARSET	HY000
10020	ER_DD_INIT_FAILED	HY000
10021	ER_DD_UPDATING_PLUGIN_MD_FAILED	HY000
10023	ER_DD_VIEW_CANT_CREATE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10024	ER_DD_METADATA_NOT_FOUND	HY000
10025	ER_DD_CACHE_NOT_EMPTY_AT_SHUTDOWN	HY000
10026	ER_DD_OBJECT_REMAINS	HY000
10027	ER_DD_OBJECT_REMAINS_IN_RELEASER	HY000
10028	ER_DD_OBJECT_RELEASER_REMAINS	HY000
10029	ER_DD_CANT_GET_OBJECT_KEY	HY000
10030	ER_DD_CANT_CREATE_OBJECT_KEY	HY000
10031	ER_CANT_CREATE_HANDLE_MGR_THREAD	HY000
10032	ER_RPL_REPO_HAS_GAPS	HY000
10033	ER_INVALID_VALUE_FOR_ENFORCE_GTID_CONSISTENCY	HY000
10034	ER_CHANGED_ENFORCE_GTID_CONSISTENCY	HY000
10035	ER_CHANGED_GTID_MODE	HY000
10036	ER_DISABLED_STORAGE_ENGINE_AS_DEFAULT	HY000
10037	ER_DEBUG_SYNC_HIT	HY000
10038	ER_DEBUG_SYNC_EXECUTED	HY000
10039	ER_DEBUG_SYNC_THREAD_MAX	HY000
10040	ER_DEBUG_SYNC_OOM	HY000
10041	ER_CANT_INIT_TC_LOG	HY000
10042	ER_EVENT_CANT_INIT_QUEUE	HY000
10043	ER_EVENT_PURGING_QUEUE	HY000
10044	ER_EVENT_LAST_EXECUTION	HY000
10045	ER_EVENT_MESSAGE_STACK	HY000
10046	ER_EVENT_EXECUTION_FAILED	HY000
10047	ER_CANT_INIT_SCHEDULER_THREAD	HY000
10048	ER_SCHEDULER_STOPPED	HY000
10049	ER_CANT_CREATE_SCHEDULER_THREAD	HY000
10050	ER_SCHEDULER_WAITING	HY000
10051	ER_SCHEDULER_STARTED	HY000
10052	ER_SCHEDULER_STOPPING_FAILED_TO_GET_EVENT	HY000
10053	ER_SCHEDULER_STOPPING_FAILED_TO_CREATE_WORKER	HY000
10054	ER_SCHEDULER_KILLING	HY000
10055	ER_UNABLE_TO_RESOLVE_IP	HY000
10056	ER_UNABLE_TO_RESOLVE_HOSTNAME	HY000
10057	ER_HOSTNAME_RESEMBLES_IPV4	HY000
10058	ER_HOSTNAME_DOESNT_RESOLVE_TO	HY000
10059	ER_ADDRESSES_FOR_HOSTNAME_HEADER	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10060	ER_ADDRESSES_FOR_HOSTNAME_LIST_ITEM	HY000
10061	ER_TRG_WITHOUT_DEFINER	HY000
10062	ER_TRG_NO_CLIENT_CHARSET	HY000
10063	ER_PARSING_VIEW	HY000
10064	ER_COMPONENTS_INFRASTRUCTURE_BOOTSTRAP	HY000
10065	ER_COMPONENTS_INFRASTRUCTURE_SHUTDOWN	HY000
10066	ER_COMPONENTS_PERSIST_LOADER_BOOTSTRAP	HY000
10067	ER_DEPART_WITH_GRACE	HY000
10068	ER_CA_SELF_SIGNED	HY000
10069	ER_SSL_LIBRARY_ERROR	HY000
10070	ER_NO_THD_NO_UUID	HY000
10071	ER_UUID_SALT	HY000
10072	ER_UUID_IS	HY000
10073	ER_UUID_INVALID	HY000
10074	ER_UUID_SCRUB	HY000
10075	ER_CREATING_NEW_UUID	HY000
10076	ER_CANT_CREATE_UUID	HY000
10077	ER_UNKNOWN_UNSUPPORTED_STORAGE_ENGINE	HY000
10078	ER_SECURE_AUTH_VALUE_UNSUPPORTED	HY000
10079	ER_INVALID_INSTRUMENT	HY000
10080	ER_INNODB_MANDATORY	HY000
10083	ER_VERBOSE_REQUIRES_HELP	HY000
10084	ER_POINTLESS_WITHOUT_SLOWLOG	HY000
10085	ER_WASTEFUL_NET_BUFFER_SIZE	HY000
10086	ER_DEPRECATED_TIMESTAMP_IMPLICIT_DEFAULTS	HY000
10087	ER_FT_BOOL_SYNTAX_INVALID	HY000
10088	ER_CREDENTIALLESS_AUTO_USER_BAD	HY000
10089	ER_CONNECTION_HANDLING_OOM	HY000
10090	ER_THREAD_HANDLING_OOM	HY000
10091	ER_CANT_CREATE_TEST_FILE	HY000
10092	ER_CANT_CREATE_PID_FILE	HY000
10093	ER_CANT_REMOVE_PID_FILE	HY000
10094	ER_CANT_CREATE_SHUTDOWN_THREAD	HY000
10095	ER_SEC_FILE_PRIV_CANT_ACCESS_DIR	HY000
10096	ER_SEC_FILE_PRIV_IGNORED	HY000
10097	ER_SEC_FILE_PRIV_EMPTY	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10098	ER_SEC_FILE_PRIV_NULL	HY000
10099	ER_SEC_FILE_PRIV_DIRECTORY_INSECURE	HY000
10100	ER_SEC_FILE_PRIV_CANT_STAT	HY000
10101	ER_SEC_FILE_PRIV_DIRECTORY_PERMISSIONS	HY000
10102	ER_SEC_FILE_PRIV_ARGUMENT_TOO_LONG	HY000
10103	ER_CANT_CREATE_NAMED_PIPES_THREAD	HY000
10104	ER_CANT_CREATE_TCPIP_THREAD	HY000
10105	ER_CANT_CREATE_SHM_THREAD	HY000
10106	ER_CANT_CREATE_INTERRUPT_THREAD	HY000
10107	ER_WRITABLE_CONFIG_REMOVED	HY000
10108	ER_CORE_VALUES	HY000
10109	ER_WRONG_DATETIME_SPEC	HY000
10110	ER_RPL_BINLOG_FILTERS_OOM	HY000
10111	ER_KEYCACHE_OOM	HY000
10112	ER_CONFIRMING_THE_FUTURE	HY000
10113	ER_BACK_IN_TIME	HY000
10114	ER_FUTURE_DATE	HY000
10115	ER_UNSUPPORTED_DATE	HY000
10116	ER_STARTING_AS	HY000
10117	ER_SHUTTING_DOWN_REPLICA_THREADS	HY000
10118	ER_DISCONNECTING_REMAINING_CLIENTS	HY000
10119	ER_ABORTING	HY000
10120	ER_BINLOG_END	HY000
10121	ER_CALL_ME_LOCALHOST	HY000
10122	ER_USER_REQUIRES_ROOT	HY000
10123	ER_REALLY_RUN_AS_ROOT	HY000
10124	ER_USER_WHAT_USER	HY000
10125	ER_TRANSPORTS_WHAT_TRANSPORTS	HY000
10126	ER_FAIL_SETGID	HY000
10127	ER_FAIL_SETUID	HY000
10128	ER_FAIL_SETREGID	HY000
10129	ER_FAIL_SETREUID	HY000
10130	ER_FAIL_CHROOT	HY000
10131	ER_WIN_LISTEN_BUT_HOW	HY000
10132	ER_NOT_RIGHT_NOW	HY000
10133	ER_FIXING_CLIENT_CHARSET	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10134	ER_OOM	HY000
10135	ER_FAILED_TO_LOCK_MEM	HY000
10136	ER_MYINIT_FAILED	HY000
10137	ER_BEG_INITFILE	HY000
10138	ER_END_INITFILE	HY000
10139	ER_CHANGED_MAX_OPEN_FILES	HY000
10140	ER_CANT_INCREASE_MAX_OPEN_FILES	HY000
10141	ER_CHANGED_MAX_CONNECTIONS	HY000
10142	ER_CHANGED_TABLE_OPEN_CACHE	HY000
10143	ER_THE_USER_ABIDES	HY000
10144	ER_RPL_CANT_ADD_DO_TABLE	HY000
10145	ER_RPL_CANT_ADD_IGNORE_TABLE	HY000
10146	ER_TRACK_VARIABLES_BOGUS	HY000
10147	ER_EXCESS_ARGUMENTS	HY000
10148	ER_VERBOSE_HINT	HY000
10149	ER_CANT_READ_ERRMSGS	HY000
10150	ER_CANT_INIT_DBS	HY000
10151	ER_LOG_OUTPUT_CONTRADICTORY	HY000
10152	ER_NO_CSV_NO_LOG_TABLES	HY000
10153	ER_RPL_REWRITEDB_MISSING_ARROW	HY000
10154	ER_RPL_REWRITEDB_EMPTY_FROM	HY000
10155	ER_RPL_REWRITEDB_EMPTY_TO	HY000
10156	ER_LOG_FILES_GIVEN_LOG_OUTPUT_IS_TABLE	HY000
10157	ER_LOG_FILE_INVALID	HY000
10158	ER_LOWER_CASE_TABLE_NAMES_CS_DD_ON_CI_FS_UNSUPPORTED	HY000
10159	ER_LOWER_CASE_TABLE_NAMES_USING_2	HY000
10160	ER_LOWER_CASE_TABLE_NAMES_USING_0	HY000
10161	ER_NEED_LOG_BIN	HY000
10162	ER_NEED_FILE_INSTEAD_OF_DIR	HY000
10163	ER_LOG_BIN_BETTER_WITH_NAME	HY000
10164	ER_BINLOG_NEEDS_SERVERID	HY000
10165	ER_RPL_CANT_MAKE_PATHS	HY000
10166	ER_CANT_INITIALIZE_GTID	HY000
10167	ER_CANT_INITIALIZE_EARLY_PLUGINS	HY000
10168	ER_CANT_INITIALIZE_BUILTIN_PLUGINS	HY000
10169	ER_CANT_INITIALIZE_DYNAMIC_PLUGINS	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10170	ER_PERFSCHEMA_INIT_FAILED	HY000
10171	ER_STACKSIZE_UNEXPECTED	HY000
10173	ER_CANT_STAT_DATADIR	HY000
10174	ER_CANT_CHOWN_DATADIR	HY000
10175	ER_CANT_SET_UP_PERSISTED_VALUES	HY000
10176	ER_CANT_SAVE_GTIDS	HY000
10178	ER_CANT_JOIN_SHUTDOWN_THREAD	HY000
10179	ER_CANT_HASH_DO_AND_IGNORE_RULES	HY000
10180	ER_CANT_OPEN_CA	HY000
10181	ER_CANT_ACCESS_CAPATH	HY000
10182	ER_SSL_TRYING_DATADIR_DEFAULTS	HY000
10183	ER_AUTO_OPTIONS_FAILED	HY000
10184	ER_CANT_INIT_TIMER	HY000
10185	ER_SERVERID_TOO_LARGE	HY000
10186	ER_DEFAULT_SE_UNAVAILABLE	HY000
10187	ER_CANT_OPEN_ERROR_LOG	HY000
10188	ER_INVALID_ERROR_LOG_NAME	HY000
10189	ER_RPL_INFINITY_DENIED	HY000
10190	ER_RPL_INFINITY_IGNORED	HY000
10192	ER_TABLE_CHECK_INTACT	HY000
10193	ER_DD_TABLESPACE_NOT_FOUND	HY000
10194	ER_DD_TRG_CONNECTION_COLLATION_MISSING	HY000
10195	ER_DD_TRG_DB_COLLATION_MISSING	HY000
10196	ER_DD_TRG_DEFINER_OOM	HY000
10197	ER_DD_TRG_FILE_UNREADABLE	HY000
10198	ER_TRG_CANT_PARSE	HY000
10199	ER_DD_TRG_CANT_ADD	HY000
10200	ER_DD_CANT_RESOLVE_VIEW	HY000
10201	ER_DD_VIEW_WITHOUT_DEFINER	HY000
10202	ER_PLUGIN_INIT_FAILED	HY000
10203	ER_RPL_TRX_DELEGATES_INIT_FAILED	HY000
10204	ER_RPL_BINLOG_STORAGE_DELEGATES_INIT_FAILED	HY000
10205	ER_RPL_BINLOG_TRANSMIT_DELEGATES_INIT_FAILED	HY000
10206	ER_RPL_BINLOG_RELAY_DELEGATES_INIT_FAILED	HY000
10207	ER_RPL_PLUGIN_FUNCTION_FAILED	HY000
10208	ER_SQL_HA_READ_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10209	ER_SR_BOGUS_VALUE	HY000
10210	ER_SR_INVALID_CONTEXT	HY000
10211	ER_READING_TABLE_FAILED	HY000
10212	ER_DES_FILE_WRONG_KEY	HY000
10214	ER_JSON_PARSE_ERROR	HY000
10215	ER_CONFIG_OPTION_WITHOUT_GROUP	HY000
10216	ER_VALGRIND_DO_QUICK_LEAK_CHECK	HY000
10217	ER_VALGRIND_COUNT_LEAKS	HY000
10218	ER_LOAD_DATA_INFILE_FAILED_IN_UNEXPECTED_WAY	HY000
10219	ER_UNKNOWN_ERROR_NUMBER	HY000
10220	ER_UDF_CANT_ALLOC_FOR_STRUCTURES	HY000
10221	ER_UDF_CANT_ALLOC_FOR_FUNCTION	HY000
10222	ER_UDF_INVALID_ROW_IN_FUNCTION_TABLE	HY000
10223	ER_UDF_CANT_OPEN_FUNCTION_TABLE	HY000
10224	ER_XA_RECOVER_FOUND_TRX_IN_SE	HY000
10225	ER_XA_RECOVER_FOUND_XA_TRX	HY000
10229	ER_XA_STARTING_RECOVERY	HY000
10230	ER_XA_NO_MULTI_2PC_HEURISTIC_RECOVER	HY000
10231	ER_XA_RECOVER_EXPLANATION	HY000
10232	ER_XA_RECOVERY_DONE	HY000
10233	ER_TRX_GTID_COLLECT_REJECT	HY000
10234	ER_SQL_AUTHOR_DEFAULT_ROLES_FAIL	HY000
10235	ER_SQL_USER_TABLE_CREATE_WARNING	HY000
10236	ER_SQL_USER_TABLE_ALTER_WARNING	HY000
10237	ER_ROW_IN_WRONG_PARTITION_PLEASE_REPAIR	HY000
10238	ER_MYISAM_CRASHED_ERROR_IN_THREAD	HY000
10239	ER_MYISAM_CRASHED_ERROR_IN	HY000
10240	ER_TOO_MANY_STORAGE_ENGINES	HY000
10241	ER_SE_TYPECODE_CONFLICT	HY000
10242	ER_TRX_WRITE_SET_OOM	HY000
10243	ER_HANDLERTON_OOM	HY000
10244	ER_CONN_SHM_LISTENER	HY000
10245	ER_CONN_SHM_CANT_CREATE_SERVICE	HY000
10246	ER_CONN_SHM_CANT_CREATE_CONNECTION	HY000
10247	ER_CONN_PIP_CANT_CREATE_EVENT	HY000
10248	ER_CONN_PIP_CANT_CREATE_PIPE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10249	ER_CONN_PER_THREAD_NO_THREAD	HY000
10250	ER_CONN_TCP_NO_SOCKET	HY000
10251	ER_CONN_TCP_CREATED	HY000
10252	ER_CONN_TCP_ADDRESS	HY000
10253	ER_CONN_TCP_IPV6_AVAILABLE	HY000
10254	ER_CONN_TCP_IPV6_UNAVAILABLE	HY000
10255	ER_CONN_TCP_ERROR_WITH_STRERROR	HY000
10256	ER_CONN_TCP_CANT_RESOLVE_HOSTNAME	HY000
10257	ER_CONN_TCP_IS_THERE_ANOTHER_USING_PORT	HY000
10258	ER_CONN_UNIX_IS_THERE_ANOTHER_USING_SOCKET	HY000
10259	ER_CONN_UNIX_PID_CLAIMED_SOCKET_FILE	HY000
10260	ER_CONN_TCP_CANT_RESET_V6ONLY	HY000
10261	ER_CONN_TCP_BIND_RETRY	HY000
10262	ER_CONN_TCP_BIND_FAIL	HY000
10263	ER_CONN_TCP_IP_NOT_LOGGED	HY000
10264	ER_CONN_TCP_RESOLVE_INFO	HY000
10265	ER_CONN_TCP_START_FAIL	HY000
10266	ER_CONN_TCP_LISTEN_FAIL	HY000
10267	ER_CONN_UNIX_PATH_TOO_LONG	HY000
10268	ER_CONN_UNIX_LOCK_FILE_FAIL	HY000
10269	ER_CONN_UNIX_NO_FD	HY000
10270	ER_CONN_UNIX_NO_BIND_NO_START	HY000
10271	ER_CONN_UNIX_LISTEN_FAILED	HY000
10272	ER_CONN_UNIX_LOCK_FILE_GIVING_UP	HY000
10273	ER_CONN_UNIX_LOCK_FILE_CANT_CREATE	HY000
10274	ER_CONN_UNIX_LOCK_FILE_CANT_OPEN	HY000
10275	ER_CONN_UNIX_LOCK_FILE_CANT_READ	HY000
10276	ER_CONN_UNIX_LOCK_FILE_EMPTY	HY000
10277	ER_CONN_UNIX_LOCK_FILE_PIDLESS	HY000
10278	ER_CONN_UNIX_LOCK_FILE_CANT_WRITE	HY000
10279	ER_CONN_UNIX_LOCK_FILE_CANT_DELETE	HY000
10280	ER_CONN_UNIX_LOCK_FILE_CANT_SYNC	HY000
10281	ER_CONN_UNIX_LOCK_FILE_CANT_CLOSE	HY000
10282	ER_CONN_SOCKET_SELECT_FAILED	HY000
10283	ER_CONN_SOCKET_ACCEPT_FAILED	HY000
10284	ER_AUTH_RSA_CANT_FIND	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10285	ER_AUTH_RSA_CANT_PARSE	HY000
10286	ER_AUTH_RSA_CANT_READ	HY000
10287	ER_AUTH_RSA_FILES_NOT_FOUND	HY000
10288	ER_CONN_ATTR_TRUNCATED	HY000
10289	ER_X509_CIPHERS_MISMATCH	HY000
10290	ER_X509_ISSUER_MISMATCH	HY000
10291	ER_X509_SUBJECT_MISMATCH	HY000
10292	ER_AUTH_CANT_ACTIVATE_ROLE	HY000
10293	ER_X509_NEEDS_RSA_PRIVKEY	HY000
10294	ER_X509_CANT_WRITE_KEY	HY000
10295	ER_X509_CANT_CHMOD_KEY	HY000
10296	ER_X509_CANT_READ_CA_KEY	HY000
10297	ER_X509_CANT_READ_CA_CERT	HY000
10298	ER_X509_CANT_CREATE_CERT	HY000
10299	ER_X509_CANT_WRITE_CERT	HY000
10300	ER_AUTH_CANT_CREATE_RSA_PAIR	HY000
10301	ER_AUTH_CANT_WRITE_PRIVKEY	HY000
10302	ER_AUTH_CANT_WRITE_PUBKEY	HY000
10303	ER_AUTH_SSL_CONF_PREVENTS_CERT_GENERATION	HY000
10304	ER_AUTH_USING_EXISTING_CERTS	HY000
10305	ER_AUTH_CERTS_SAVED_TO_DATADIR	HY000
10306	ER_AUTH_CERT_GENERATION_DISABLED	HY000
10307	ER_AUTH_RSA_CONF_PREVENTS_KEY_GENERATION	HY000
10308	ER_AUTH_KEY_GENERATION_SKIPPED_PAIR_PRESENT	HY000
10309	ER_AUTH_KEYS_SAVED_TO_DATADIR	HY000
10310	ER_AUTH_KEY_GENERATION_DISABLED	HY000
10311	ER_AUTHCACHE_PROXIES_PRIV_SKIPPED_NEEDS_RESOLVE	HY000
10312	ER_AUTHCACHE_PLUGIN_MISSING	HY000
10313	ER_AUTHCACHE_PLUGIN_CONFIG	HY000
10315	ER_AUTHCACHE_USER_SKIPPED_NEEDS_RESOLVE	HY000
10316	ER_AUTHCACHE_USER_TABLE_DODGY	HY000
10317	ER_AUTHCACHE_USER_IGNORED_DEPRECATED_PASSWORD	HY000
10318	ER_AUTHCACHE_USER_IGNORED_NEEDS_PLUGIN	HY000
10319	ER_AUTHCACHE_USER_IGNORED_INVALID_PASSWORD	HY000
10320	ER_AUTHCACHE_EXPIRED_PASSWORD_UNSUPPORTED	HY000
10321	ER_NO_SUPER_WITHOUT_USER_PLUGIN	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10322	ER_AUTHCACHE_DB_IGNORED_EMPTY_NAME	HY000
10323	ER_AUTHCACHE_DB_SKIPPED_NEEDS_RESOLVE	HY000
10324	ER_AUTHCACHE_DB_ENTRY_LOWERCASED_REVOKE_WILL_FAIL	HY000
10325	ER_AUTHCACHE_TABLE_PROXIES_PRIV_MISSING	HY000
10326	ER_AUTHCACHE_CANT_OPEN_AND_LOCK_PRIVILEGE_TABLES	HY000
10327	ER_AUTHCACHE_CANT_INIT_GRANT_SUBSYSTEM	HY000
10328	ER_AUTHCACHE_PROCS_PRIV_SKIPPED_NEEDS_RESOLVE	HY000
10329	ER_AUTHCACHE_PROCS_PRIV_ENTRY_IGNORED_BAD_ROUTINE_TYPE	HY000
10330	ER_AUTHCACHE_TABLES_PRIV_SKIPPED_NEEDS_RESOLVE	HY000
10331	ER_USER_NOT_IN_EXTRA_USERS_BINLOG_POSSIBLY_INCOMPLETE	HY000
10332	ER_DD_SCHEMA_NOT_FOUND	HY000
10333	ER_DD_TABLE_NOT_FOUND	HY000
10334	ER_DD_SE_INIT_FAILED	HY000
10335	ER_DD_ABORTING_PARTIAL_UPGRADE	HY000
10336	ER_DD_FRM_EXISTS_FOR_TABLE	HY000
10337	ER_DD_CREATED_FOR_UPGRADE	HY000
10338	ER_ERRMSG_CANT_FIND_FILE	HY000
10340	ER_ERRMSG_MISSING_IN_FILE	HY000
10341	ER_ERRMSG_OOM	HY000
10342	ER_ERRMSG_CANT_READ	HY000
10343	ER_TABLE_INCOMPATIBLE_DECIMAL_FIELD	HY000
10344	ER_TABLE_INCOMPATIBLE_YEAR_FIELD	HY000
10345	ER_INVALID_CHARSET_AND_DEFAULT_IS_MB	HY000
10346	ER_TABLE_WRONG_KEY_DEFINITION	HY000
10347	ER_CANT_OPEN_FRM_FILE	HY000
10348	ER_CANT_READ_FRM_FILE	HY000
10349	ER_TABLE_CREATED_WITH_DIFFERENT_VERSION	HY000
10350	ER_VIEW_UNPARSABLE	HY000
10351	ER_FILE_TYPE_UNKNOWN	HY000
10352	ER_INVALID_INFO_IN_FRM	HY000
10353	ER_CANT_OPEN_AND_LOCK_PRIVILEGE_TABLES	HY000
10354	ER_AUDIT_PLUGIN_DOES_NOT_SUPPORT_AUDIT_AUTH_EVENTS	HY000
10355	ER_AUDIT_PLUGIN_HAS_INVALID_DATA	HY000
10356	ER_TZ_OOM_INITIALIZING_TIME_ZONES	HY000
10357	ER_TZ_CANT_OPEN_AND_LOCK_TIME_ZONE_TABLE	HY000
10358	ER_TZ_OOM_LOADING_LEAP_SECOND_TABLE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10359	ER_TZ_TOO_MANY_LEAPS_IN_LEAP_SECOND_TABLE	HY000
10360	ER_TZ_ERROR_LOADING_LEAP_SECOND_TABLE	HY000
10361	ER_TZ_UNKNOWN_OR_ILLEGAL_DEFAULT_TIME_ZONE	HY000
10362	ER_TZ_CANT_FIND_DESCRIPTION_FOR_TIME_ZONE	HY000
10363	ER_TZ_CANT_FIND_DESCRIPTION_FOR_TIME_ZONE_ID	HY000
10364	ER_TZ_TRANSITION_TYPE_TABLE_TYPE_TOO_LARGE	HY000
10365	ER_TZ_TRANSITION_TYPE_TABLE_ABBREVIATIONS_EXCEED_SPACE	HY000
10366	ER_TZ_TRANSITION_TYPE_TABLE_LOAD_ERROR	HY000
10367	ER_TZ_TRANSITION_TABLE_TOO_MANY_TRANSITIONS	HY000
10368	ER_TZ_TRANSITION_TABLE_BAD_TRANSITION_TYPE	HY000
10369	ER_TZ_TRANSITION_TABLE_LOAD_ERROR	HY000
10370	ER_TZ_NO_TRANSITION_TYPES_IN_TIME_ZONE	HY000
10371	ER_TZ_OOM_LOADING_TIME_ZONE_DESCRIPTION	HY000
10372	ER_TZ_CANT_BUILD_MKTIME_MAP	HY000
10373	ER_TZ_OOM_WHILE_LOADING_TIME_ZONE	HY000
10374	ER_TZ_OOM_WHILE_SETTING_TIME_ZONE	HY000
10375	ER_REPLICA_SQL_THREAD_STOPPED_UNTIL_CONDITION_BAD	HY000
10376	ER_REPLICA_SQL_THREAD_STOPPED_UNTIL_POSITION_REACHED	HY000
10377	ER_REPLICA_SQL_THREAD_STOPPED_BEFORE_GTIDS_ALREADY_APPLIED	HY000
10378	ER_REPLICA_SQL_THREAD_STOPPED_BEFORE_GTIDS_REACHED	HY000
10379	ER_REPLICA_SQL_THREAD_STOPPED_AFTER_GTIDS_REACHED	HY000
10380	ER_REPLICA_SQL_THREAD_STOPPED_GAP_TRX_PROCESSED	HY000
10381	ER_GROUP_REPLICATION_PLUGIN_NOT_INSTALLED	HY000
10382	ER_GTID_ALREADY_ADDED_BY_USER	HY000
10383	ER_FAILED_TO_DELETE_FROM_GTID_EXECUTED_TABLE	HY000
10384	ER_FAILED_TO_COMPRESS_GTID_EXECUTED_TABLE	HY000
10385	ER_FAILED_TO_COMPRESS_GTID_EXECUTED_TABLE_OOM	HY000
10386	ER_FAILED_TO_INIT_THREAD_ATTR_FOR_GTID_TABLE_COMPRESSION	HY000
10387	ER_FAILED_TO_CREATE_GTID_TABLE_COMPRESSION_THREAD	HY000
10388	ER_FAILED_TO_JOIN_GTID_TABLE_COMPRESSION_THREAD	HY000
10389	ER_NPIPE_FAILED_TO_INIT_SECURITY_DESCRIPTOR	HY000
10390	ER_NPIPE_FAILED_TO_SET_SECURITY_DESCRIPTOR	HY000
10391	ER_NPIPE_PIPE_ALREADY_IN_USE	HY000
10405	ER_RPL_CANT_OPEN_INFO_TABLE	HY000
10406	ER_RPL_CANT_SCAN_INFO_TABLE	HY000
10407	ER_RPL_CORRUPTED_INFO_TABLE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10408	ER_RPL_CORRUPTED_KEYS_IN_INFO_TABLE	HY000
10409	ER_RPL_WORKER_ID_IS	HY000
10410	ER_RPL_INCONSISTENT_TIMESTAMPS_IN_TRX	HY000
10411	ER_RPL_INCONSISTENT_SEQUENCE_NO_IN_TRX	HY000
10413	ER_RPL_CHANNELS_REQUIRE_NON_ZERO_SERVER_ID	HY000
10415	ER_RPL_ERROR_CREATING_CONNECTION_METADATA	HY000
10418	ER_RPL_ERROR_CREATING_APPLIER_METADATA	HY000
10420	ER_RPL_FAILED_TO_DELETE_FROM_REPLICA_WORKERS_INFO_REPOSITORY	HY000
10421	ER_RPL_FAILED_TO_RESET_STATE_IN_REPLICA_INFO_REPOSITORY	HY000
10423	ER_RPL_REPLICA_GENERIC_MESSAGE	HY000
10424	ER_RPL_REPLICA_COULD_NOT_CREATE_CHANNEL_LIST	HY000
10425	ER_RPL_MULTISOURCE_REQUIRES_TABLE_TYPE_REPOSITORIES	HY000
10426	ER_RPL_REPLICA_FAILED_TO_INIT_A_CONNECTION_METADATA_STRUCTURE	HY000
10427	ER_RPL_REPLICA_FAILED_TO_INIT_CONNECTION_METADATA_STRUCTURE	HY000
10428	ER_RPL_REPLICA_FAILED_TO_CREATE_CHANNEL_FROM_CONNECTION_METADATA	HY000
10429	ER_RPL_FAILED_TO_CREATE_NEW_INFO_FILE	HY000
10430	ER_RPL_FAILED_TO_CREATE_CACHE_FOR_INFO_FILE	HY000
10431	ER_RPL_FAILED_TO_OPEN_INFO_FILE	HY000
10432	ER_RPL_GTID_MEMORY_FINALLY_AVAILABLE	HY000
10433	ER_SERVER_COST_UNKNOWN_COST_CONSTANT	HY000
10434	ER_SERVER_COST_INVALID_COST_CONSTANT	HY000
10435	ER_ENGINE_COST_UNKNOWN_COST_CONSTANT	HY000
10436	ER_ENGINE_COST_UNKNOWN_STORAGE_ENGINE	HY000
10437	ER_ENGINE_COST_INVALID_DEVICE_TYPE_FOR_SE	HY000
10438	ER_ENGINE_COST_INVALID_CONST_CONSTANT_FOR_SE_AND_DEVICE	HY000
10439	ER_SERVER_COST_FAILED_TO_READ	HY000
10440	ER_ENGINE_COST_FAILED_TO_READ	HY000
10441	ER_FAILED_TO_OPEN_COST_CONSTANT_TABLES	HY000
10442	ER_RPL_UNSUPPORTED_UNIGNORABLE_EVENT_IN_STREAM	HY000
10443	ER_RPL_GTID_LOG_EVENT_IN_STREAM	HY000
10444	ER_RPL_UNEXPECTED_BEGIN_IN_STREAM	HY000
10445	ER_RPL_UNEXPECTED_COMMIT_ROLLBACK_OR_XID_LOG_EVENT_IN_STREAM	HY000
10446	ER_RPL_UNEXPECTED_XA_ROLLBACK_IN_STREAM	HY000
10447	ER_EVENT_EXECUTION_FAILED_CANT_AUTHENTICATE_USER	HY000
10448	ER_EVENT_EXECUTION_FAILED_USER_LOST_EVEN_PRIVILEGE	HY000
10449	ER_EVENT_ERROR_DURING_COMPILATION	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10450	ER_EVENT_DROPPING	HY000
10452	ER_RPL_INCOMPATIBLE_DECIMAL_IN_RBR	HY000
10453	ER_INIT_ROOT_WITHOUT_PASSWORD	HY000
10454	ER_INIT_GENERATING_TEMP_PASSWORD_FOR_ROOT	HY000
10455	ER_INIT_CANT_OPEN_BOOTSTRAP_FILE	HY000
10456	ER_INIT_BOOTSTRAP_COMPLETE	HY000
10457	ER_INIT_DATADIR_NOT_EMPTY_WONT_INITIALIZE	HY000
10458	ER_INIT_DATADIR_EXISTS_WONT_INITIALIZE	HY000
10459	ER_INIT_DATADIR_EXISTS_AND_PATH_TOO_LONG_WONT_INITIALIZE	HY000
10460	ER_INIT_DATADIR_EXISTS_AND_NOT_WRITABLE_WONT_INITIALIZE	HY000
10461	ER_INIT_CREATING_DD	HY000
10462	ER_RPL_BINLOG_STARTING_DUMP	HY000
10463	ER_RPL_BINLOG_SOURCE_SENDS_HEARTBEAT	HY000
10464	ER_RPL_BINLOG_SKIPPING_REMAINING_HEARTBEAT_INFO	HY000
10465	ER_RPL_BINLOG_SOURCE_USES_CHECKSUM_AND_REPLICA_CANT	HY000
10467	ER_KILLING_THREAD	HY000
10468	ER_DETACHING_SESSION_LEFT_BY_PLUGIN	HY000
10469	ER_CANT_DETACH_SESSION_LEFT_BY_PLUGIN	HY000
10470	ER_DETACHED_SESSIONS_LEFT_BY_PLUGIN	HY000
10471	ER_FAILED_TO_DECREMENT_NUMBER_OF_THREADS	HY000
10472	ER_PLUGIN_DID_NOT_DEINITIALIZE_THREADS	HY000
10473	ER_KILLED_THREADS_OF_PLUGIN	HY000
10517	ER_DBUG_CHECK_SHARES_OPEN	HY000
10518	ER_DBUG_CHECK_SHARES_INFO	HY000
10519	ER_DBUG_CHECK_SHARES_DROPPED	HY000
10520	ER_INVALID_OR_OLD_TABLE_OR_DB_NAME	HY000
10521	ER_TC_RECOVERING_AFTER_CRASH_USING	HY000
10522	ER_TC_CANT_AUTO_RECOVER_WITH_TC_HEURISTIC_RECOVER	HY000
10523	ER_TC_BAD_MAGIC_IN_TC_LOG	HY000
10524	ER_TC_NEED_N_SE_SUPPORTING_2PC_FOR_RECOVERY	HY000
10525	ER_TC_RECOVERY_FAILED_THESE_ARE_YOUR_OPTIONS	HY000
10526	ER_TC_HEURISTIC_RECOVERY_MODE	HY000
10527	ER_TC_HEURISTIC_RECOVERY_FAILED	HY000
10528	ER_TC_RESTART_WITHOUT_TC_HEURISTIC_RECOVER	HY000
10529	ER_RPL_REPLICA_FAILED_TO_CREATE_OR_RECOVER_INFO_REPOSITORIES	HY000
10530	ER_RPL_REPLICA_AUTO_POSITION_IS_1_AND_GTID_MODE_IS_OFF	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10531	ER_RPL_REPLICA_CANT_START_REPLICA_FOR_CHANNEL	HY000
10532	ER_RPL_REPLICA_CANT_STOP_REPLICA_FOR_CHANNEL	HY000
10533	ER_RPL_RECOVERY_NO_ROTATE_EVENT_FROM_SOURCE	HY000
10534	ER_RPL_RECOVERY_ERROR_READ_RELAY_LOG	HY000
10536	ER_RPL_RECOVERY_SKIPPED_GROUP_REPLICATION_CHANNEL	HY000
10537	ER_RPL_RECOVERY_ERROR	HY000
10538	ER_RPL_RECOVERY_IO_ERROR_READING_RELAY_LOG_INDEX	HY000
10539	ER_RPL_RECOVERY_FILE_SOURCE_POS_INFO	HY000
10540	ER_RPL_RECOVERY_REPLICATE_SAME_SERVER_ID_REQUIRES_POSITION	HY000
10541	ER_RPL_MTA_RECOVERY_STARTING_COORDINATOR	HY000
10542	ER_RPL_MTA_RECOVERY_FAILED_TO_START_COORDINATOR	HY000
10543	ER_RPL_MTA_AUTOMATIC_RECOVERY_FAILED	HY000
10544	ER_RPL_MTA_RECOVERY_CANT_OPEN_RELAY_LOG	HY000
10545	ER_RPL_MTA_RECOVERY_SUCCESSFUL	HY000
10546	ER_RPL_SERVER_ID_MISSING	HY000
10547	ER_RPL_CANT_CREATE_REPLICA_THREAD	HY000
10548	ER_RPL_REPLICA_IO_THREAD_WAS_KILLED	HY000
10550	ER_RPL_REPLICA_USES_CHECKSUM_AND_SOURCE_PRE_50	HY000
10551	ER_RPL_REPLICA_SECONDS_BEHIND_SOURCE_DUBIOUS	HY000
10552	ER_RPL_REPLICA_CANT_FLUSH_CONNECTION_METADATA_REPOS	HY000
10553	ER_RPL_REPLICA_REPORT_HOST_TOO_LONG	HY000
10554	ER_RPL_REPLICA_REPORT_USER_TOO_LONG	HY000
10555	ER_RPL_REPLICA_REPORT_PASSWORD_TOO_LONG	HY000
10556	ER_RPL_REPLICA_ERROR_RETRYING	HY000
10557	ER_RPL_REPLICA_ERROR_READING_FROM_SERVER	HY000
10558	ER_RPL_REPLICA_DUMP_THREAD_KILLED_BY_SOURCE	HY000
10559	ER_RPL_MTA_STATISTICS	HY000
10560	ER_RPL_MTA_RECOVERY_COMPLETE	HY000
10561	ER_RPL_REPLICA_CANT_INIT_RELAY_LOG_POSITION	HY000
10563	ER_RPL_REPLICA_IO_THREAD_KILLED	HY000
10564	ER_RPL_REPLICA_IO_THREAD_CANT_REGISTER_ON_SOURCE	HY000
10565	ER_RPL_REPLICA_FORCING_TO_RECONNECT_IO_THREAD	HY000
10566	ER_RPL_REPLICA_ERROR_REQUESTING_BINLOG_DUMP	HY000
10567	ER_RPL_LOG_ENTRY_EXCEEDS_REPLICA_MAX_ALLOWED_PACKET	HY000
10568	ER_RPL_REPLICA_STOPPING_AS_SOURCE_OOM	HY000
10569	ER_RPL_REPLICA_IO_THREAD_ABORTED_WAITING_FOR_RELAY_LOG_SPACE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10570	ER_RPL_REPLICA_IO_THREAD_EXITING	HY000
10571	ER_RPL_REPLICA_CANT_INITIALIZE_REPLICA_WORKER	HY000
10572	ER_RPL_MTA_GROUP_RECOVERY_APPLIER_METADATA_FOR_WORKER	HY000
10573	ER_RPL_ERROR_LOOKING_FOR_LOG	HY000
10574	ER_RPL_MTA_GROUP_RECOVERY_APPLIER_METADATA	HY000
10575	ER_RPL_CANT_FIND_FOLLOWUP_FILE	HY000
10576	ER_RPL_MTA_CHECKPOINT_PERIOD_DIFFERS_FROM_CNT	HY000
10577	ER_RPL_REPLICA_WORKER_THREAD_CREATION_FAILED	HY000
10578	ER_RPL_REPLICA_WORKER_THREAD_CREATION_FAILED_WITH_ERRNO	HY000
10579	ER_RPL_REPLICA_FAILED_TO_INIT_PARTITIONS_HASH	HY000
10581	ER_RPL_REPLICA_SQL_THREAD_STARTING	HY000
10582	ER_RPL_REPLICA_SKIP_COUNTER_EXECUTED	HY000
10583	ER_RPL_REPLICA_ADDITIONAL_ERROR_INFO_FROM_DA	HY000
10584	ER_RPL_REPLICA_ERROR_INFO_FROM_DA	HY000
10585	ER_RPL_REPLICA_ERROR_LOADING_USER_DEFINED_LIBRARY	HY000
10586	ER_RPL_REPLICA_ERROR_RUNNING_QUERY	HY000
10587	ER_RPL_REPLICA_SQL_THREAD_EXITING	HY000
10588	ER_RPL_REPLICA_READ_INVALID_EVENT_FROM_SOURCE	HY000
10589	ER_RPL_REPLICA_QUEUE_EVENT_FAILED_INVALID_CONFIGURATION	HY000
10590	ER_RPL_REPLICA_IO_THREAD_DETECTED_UNEXPECTED_EVENT_SEQUENCE	HY000
10591	ER_RPL_REPLICA_CANT_USE_CHARSET	HY000
10592	ER_RPL_REPLICA_CONNECTED_TO_SOURCE_REPLICATION_RESUMED	HY000
10593	ER_RPL_REPLICA_NEXT_LOG_IS_ACTIVE	HY000
10594	ER_RPL_REPLICA_NEXT_LOG_IS_INACTIVE	HY000
10595	ER_RPL_REPLICA_SQL_THREAD_IO_ERROR_READING_EVENT	HY000
10596	ER_RPL_REPLICA_ERROR_READING_RELAY_LOG_EVENTS	HY000
10597	ER_REPLICA_CHANGE_SOURCE_TO_EXECUTED	HY000
10598	ER_RPL_REPLICA_NEW_C_M_NEEDS_REPOS_TYPE_OTHER_THAN_FILE	HY000
10599	ER_RPL_FAILED_TO_STAT_LOG_IN_INDEX	HY000
10600	ER_RPL_LOG_NOT_FOUND_WHILE_COUNTING_RELAY_LOG_SPACE	HY000
10601	ER_REPLICA_CANT_USE_TEMPDIR	HY000
10602	ER_RPL_RELAY_LOG_NEEDS_FILE_NOT_DIRECTORY	HY000
10603	ER_RPL_RELAY_LOG_INDEX_NEEDS_FILE_NOT_DIRECTORY	HY000
10604	ER_RPL_PLEASE_USE_OPTION_RELAY_LOG	HY000
10605	ER_RPL_OPEN_INDEX_FILE_FAILED	HY000
10606	ER_RPL_CANT_INITIALIZE_GTID_SETS_IN_AM_INIT_INFO	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10607	ER_RPL_CANT_OPEN_LOG_IN_AM_INIT_INFO	HY000
10608	ER_RPL_ERROR_WRITING_RELAY_LOG_CONFIGURATION	HY000
10709	ER_TREE_CORRUPT_PARENT_SHOULD_POINT_AT_PARENT	HY000
10710	ER_TREE_CORRUPT_ROOT_SHOULD_BE_BLACK	HY000
10711	ER_TREE_CORRUPT_2_CONSECUTIVE_REDS	HY000
10712	ER_TREE_CORRUPT_RIGHT_IS_LEFT	HY000
10713	ER_TREE_CORRUPT_INCORRECT_BLACK_COUNT	HY000
10714	ER_WRONG_COUNT_FOR_ORIGIN	HY000
10715	ER_WRONG_COUNT_FOR_KEY	HY000
10716	ER_WRONG_COUNT_OF_ELEMENTS	HY000
10717	ER_RPL_ERROR_READING_REPLICA_WORKER_CONFIGURATION	HY000
10719	ER_RPL_FAILED_TO_OPEN_RELAY_LOG	HY000
10720	ER_RPL_WORKER_CANT_READ_RELAY_LOG	HY000
10721	ER_RPL_WORKER_CANT_FIND_NEXT_RELAY_LOG	HY000
10722	ER_RPL_MTA_REPLICA_COORDINATOR_HAS_WAITED	HY000
10723	ER_BINLOG_FAILED_TO_WRITE_DROP_FOR_TEMP_TABLES	HY000
10724	ER_BINLOG_OOM_WRITING_DELETE_WHILE_OPENING_HEAP_TABLE	HY000
10725	ER_FAILED_TO_REPAIR_TABLE	HY000
10726	ER_FAILED_TO_REMOVE_TEMP_TABLE	HY000
10727	ER_SYSTEM_TABLE_NOT_TRANSACTIONAL	HY000
10728	ER_RPL_ERROR_WRITING_SOURCE_CONFIGURATION	HY000
10729	ER_RPL_ERROR_READING_SOURCE_CONFIGURATION	HY000
10730	ER_RPL_SSL_INFO_IN_CONNECTION_METADATA_IGNORED	HY000
10731	ER_PLUGIN_FAILED_DEINITIALIZATION	HY000
10732	ER_PLUGIN_HAS_NONZERO_REFCOUNT_AFTER_DEINITIALIZATION	HY000
10733	ER_PLUGIN_SHUTTING_DOWN_PLUGIN	HY000
10734	ER_PLUGIN_REGISTRATION_FAILED	HY000
10735	ER_PLUGIN_CANT_OPEN_PLUGIN_TABLE	HY000
10736	ER_PLUGIN_CANT_LOAD	HY000
10737	ER_PLUGIN_LOAD_PARAMETER_TOO_LONG	HY000
10738	ER_PLUGIN_FORCING_SHUTDOWN	HY000
10739	ER_PLUGIN_HAS_NONZERO_REFCOUNT_AFTER_SHUTDOWN	HY000
10740	ER_PLUGIN_UNKNOWN_VARIABLE_TYPE	HY000
10741	ER_PLUGIN_VARIABLE_SET_READ_ONLY	HY000
10742	ER_PLUGIN_VARIABLE_MISSING_NAME	HY000
10743	ER_PLUGIN_VARIABLE_NOT_ALLOCATED_THREAD_LOCAL	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10744	ER_PLUGIN_OOM	HY000
10745	ER_PLUGIN_BAD_OPTIONS	HY000
10746	ER_PLUGIN_PARSING_OPTIONS_FAILED	HY000
10747	ER_PLUGIN_DISABLED	HY000
10748	ER_PLUGIN_HAS_CONFLICTING_SYSTEM_VARIABLES	HY000
10749	ER_PLUGIN_CANT_SET_PERSISTENT_OPTIONS	HY000
10750	ER_MY_NET_WRITE_FAILED_FALLING_BACK_ON_STDERR	HY000
10751	ER_RETRYING_REPAIR_WITHOUT_QUICK	HY000
10752	ER_RETRYING_REPAIR_WITH_KEYCACHE	HY000
10753	ER_FOUND_ROWS_WHILE_REPAIRING	HY000
10754	ER_ERROR_DURING_OPTIMIZE_TABLE	HY000
10755	ER_ERROR_ENABLING_KEYS	HY000
10756	ER_CHECKING_TABLE	HY000
10757	ER_RECOVERING_TABLE	HY000
10758	ER_CANT_CREATE_TABLE_SHARE_FROM_FRM	HY000
10759	ER_CANT_LOCK_TABLE	HY000
10760	ER_CANT_ALLOC_TABLE_OBJECT	HY000
10761	ER_CANT_CREATE_HANDLER_OBJECT_FOR_TABLE	HY000
10762	ER_CANT_SET_HANDLER_REFERENCE_FOR_TABLE	HY000
10763	ER_CANT_LOCK_TABLESPACE	HY000
10764	ER_CANT_UPGRADE_GENERATED_COLUMNS_TO_DD	HY000
10765	ER_DD_ERROR_CREATING_ENTRY	HY000
10766	ER_DD_CANT_FETCH_TABLE_DATA	HY000
10767	ER_DD_CANT_FIX_SE_DATA	HY000
10768	ER_DD_CANT_CREATE_SP	HY000
10769	ER_CANT_OPEN_DB_OPT_USING_DEFAULT_CHARSET	HY000
10770	ER_CANT_CREATE_CACHE_FOR_DB_OPT	HY000
10771	ER_CANT_IDENTIFY_CHARSET_USING_DEFAULT	HY000
10772	ER_DB_OPT_NOT_FOUND_USING_DEFAULT_CHARSET	HY000
10773	ER_EVENT_CANT_GET_TIMEZONE_FROM_FIELD	HY000
10774	ER_EVENT_CANT_FIND_TIMEZONE	HY000
10775	ER_EVENT_CANT_GET_CHARSET	HY000
10776	ER_EVENT_CANT_GET_COLLATION	HY000
10777	ER_EVENT_CANT_OPEN_TABLE_MYSQL_EVENT	HY000
10778	ER_CANT_PARSE_STORED_ROUTINE_BODY	HY000
10779	ER_CANT_OPEN_TABLE_MYSQL_PROC	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10780	ER_CANT_READ_TABLE_MYSQL_PROC	HY000
10781	ER_FILE_EXISTS_DURING_UPGRADE	HY000
10782	ER_CANT_OPEN_DATADIR_AFTER_UPGRADE_FAILURE	HY000
10783	ER_CANT_SET_PATH_FOR	HY000
10784	ER_CANT_OPEN_DIR	HY000
10801	ER_EVENT_ERROR_CREATING_QUERY_TO_WRITE_TO_BINLOG	HY000
10802	ER_EVENT_SCHEDULER_ERROR_LOADING_FROM_DB	HY000
10803	ER_EVENT_SCHEDULER_ERROR_GETTING_EVENT_OBJECT	HY000
10804	ER_EVENT_SCHEDULER_GOT_BAD_DATA_FROM_TABLE	HY000
10805	ER_EVENT_CANT_GET_LOCK_FOR_DROPPING_EVENT	HY000
10806	ER_EVENT_UNABLE_TO_DROP_EVENT	HY000
10808	ER_BINLOG_CANT_RESIZE_CACHE	HY000
10809	ER_BINLOG_FILE_BEING_READ_NOT_PURGED	HY000
10810	ER_BINLOG_IO_ERROR_READING_HEADER	HY000
10813	ER_BINLOG_FILE_EXTENSION_NUMBER_EXHAUSTED	HY000
10814	ER_BINLOG_FILE_NAME_TOO_LONG	HY000
10815	ER_BINLOG_FILE_EXTENSION_NUMBER_RUNNING_LOW	HY000
10816	ER_BINLOG_CANT_OPEN_FOR_LOGGING	HY000
10817	ER_BINLOG_FAILED_TO_SYNC_INDEX_FILE	HY000
10818	ER_BINLOG_ERROR_READING_GTIDS_FROM_RELAY_LOG	HY000
10819	ER_BINLOG_EVENTS_READ_FROM_APPLIER_METADATA	HY000
10820	ER_BINLOG_ERROR_READING_GTIDS_FROM_BINARY_LOG	HY000
10821	ER_BINLOG_EVENTS_READ_FROM_BINLOG_INFO	HY000
10822	ER_BINLOG_CANT_GENERATE_NEW_FILE_NAME	HY000
10823	ER_BINLOG_FAILED_TO_SYNC_INDEX_FILE_IN_OPEN	HY000
10824	ER_BINLOG_CANT_USE_FOR_LOGGING	HY000
10825	ER_BINLOG_FAILED_TO_CLOSE_INDEX_FILE_WHILE_REBUILDING	HY000
10826	ER_BINLOG_FAILED_TO_DELETE_INDEX_FILE_WHILE_REBUILDING	HY000
10827	ER_BINLOG_FAILED_TO_RENAME_INDEX_FILE_WHILE_REBUILDING	HY000
10828	ER_BINLOG_FAILED_TO_OPEN_INDEX_FILE_AFTER_REBUILDING	HY000
10829	ER_BINLOG_CANT_APPEND_LOG_TO_TMP_INDEX	HY000
10830	ER_BINLOG_CANT_LOCATE_OLD_BINLOG_OR_RELAY_LOG_FILES	HY000
10831	ER_BINLOG_CANT_DELETE_FILE	HY000
10832	ER_BINLOG_CANT_SET_TMP_INDEX_NAME	HY000
10833	ER_BINLOG_FAILED_TO_OPEN_TEMPORARY_INDEX_FILE	HY000
10835	ER_BINLOG_CANT_OPEN_TMP_INDEX	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10836	ER_BINLOG_CANT_COPY_INDEX_TO_TMP	HY000
10837	ER_BINLOG_CANT_CLOSE_TMP_INDEX	HY000
10838	ER_BINLOG_CANT_MOVE_TMP_TO_INDEX	HY000
10839	ER_BINLOG_PURGE_LOGS_CALLED_WITH_FILE_NOT_IN_INDEX	HY000
10840	ER_BINLOG_PURGE_LOGS_CANT_SYNC_INDEX_FILE	HY000
10841	ER_BINLOG_PURGE_LOGS_CANT_COPY_TO_REGISTER_FILE	HY000
10842	ER_BINLOG_PURGE_LOGS_CANT_FLUSH_REGISTER_FILE	HY000
10843	ER_BINLOG_PURGE_LOGS_CANT_UPDATE_INDEX_FILE	HY000
10844	ER_BINLOG_PURGE_LOGS_FAILED_TO_PURGE_LOG	HY000
10845	ER_BINLOG_FAILED_TO_SET_PURGE_INDEX_FILE_NAME	HY000
10846	ER_BINLOG_FAILED_TO_OPEN_REGISTER_FILE	HY000
10847	ER_BINLOG_FAILED_TO_REINIT_REGISTER_FILE	HY000
10848	ER_BINLOG_FAILED_TO_READ_REGISTER_FILE	HY000
10849	ER_CANT_STAT_FILE	HY000
10850	ER_BINLOG_CANT_DELETE_LOG_FILE_DOES_INDEX_MATCH_FILES	HY000
10851	ER_BINLOG_CANT_DELETE_FILE_AND_READ_BINLOG_INDEX	HY000
10852	ER_BINLOG_FAILED_TO_DELETE_LOG_FILE	HY000
10853	ER_BINLOG_LOGGING_INCIDENT_TO_STOP_REPLICAS	HY000
10854	ER_BINLOG_CANT_FIND_LOG_IN_INDEX	HY000
10855	ER_BINLOG_RECOVERING_AFTER_CRASH_USING	HY000
10856	ER_BINLOG_CANT_OPEN_CRASHED_BINLOG	HY000
10857	ER_BINLOG_CANT_TRIM_CRASHED_BINLOG	HY000
10858	ER_BINLOG_CRASHED_BINLOG_TRIMMED	HY000
10859	ER_BINLOG_CANT_CLEAR_IN_USE_FLAG_FOR_CRASHED_BINLOG	HY000
10860	ER_BINLOG_FAILED_TO_RUN_AFTER_SYNC_HOOK	HY000
10861	ER_TURNING_LOGGING_OFF_FOR_THE_DURATION	HY000
10862	ER_BINLOG_FAILED_TO_RUN_AFTER_FLUSH_HOOK	HY000
10864	ER_BINLOG_WARNING_SUPPRESSED	HY000
10865	ER_NDB_LOG_ENTRY	HY000
10866	ER_NDB_LOG_ENTRY_WITH_PREFIX	HY000
10868	ER_INNODB_UNKNOWN_COLLATION	HY000
10869	ER_INNODB_INVALID_LOG_GROUP_HOME_DIR	HY000
10870	ER_INNODB_INVALID_INNODB_UNDO_DIRECTORY	HY000
10871	ER_INNODB_ILLEGAL_COLON_IN_POOL	HY000
10872	ER_INNODB_INVALID_PAGE_SIZE	HY000
10873	ER_INNODB_DIRTY_WATER_MARK_NOT_LOW	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10874	ER_INNODB_IO_CAPACITY_EXCEEDS_MAX	HY000
10875	ER_INNODB_FILES_SAME	HY000
10876	ER_INNODB_UNREGISTERED_TRX_ACTIVE	HY000
10877	ER_INNODB_CLOSING_CONNECTION_ROLLS_BACK	HY000
10878	ER_INNODB_TRX_XLATION_TABLE_OOM	HY000
10879	ER_INNODB_CANT_FIND_INDEX_IN_INNODB_DD	HY000
10880	ER_INNODB_INDEX_COLUMN_INFO_UNLIKE_MYSQLS	HY000
10882	ER_INNODB_CANT_BUILD_INDEX_XLATION_TABLE_FOR	HY000
10883	ER_INNODB_PK_NOT_IN_MYSQL	HY000
10884	ER_INNODB_PK_ONLY_IN_MYSQL	HY000
10885	ER_INNODB_CLUSTERED_INDEX_PRIVATE	HY000
10887	ER_ERRMSG_REPLACEMENT_DODGY	HY000
10888	ER_ERRMSG_REPLACEMENTS_FAILED	HY000
10889	ER_NPIPE_CANT_CREATE	HY000
10890	ER_PARTITION_MOVE_CREATED_DUPLICATE_ROW_PLEASE_FIX	HY000
10891	ER_AUDIT_CANT_ABORT_COMMAND	HY000
10892	ER_AUDIT_CANT_ABORT_EVENT	HY000
10893	ER_AUDIT_WARNING	HY000
10897	ER_RPL_REPLICA_INSECURE_CHANGE_SOURCE	HY000
10899	ER_RPL_REPLICA_INCORRECT_CHANNEL	HY000
10900	ER_FAILED_TO_FIND_DL_ENTRY	HY000
10901	ER_FAILED_TO_OPEN_SHARED_LIBRARY	HY000
10902	ER_THREAD_PRIORITY_IGNORED	HY000
10903	ER_BINLOG_CACHE_SIZE_TOO_LARGE	HY000
10904	ER_BINLOG_STMT_CACHE_SIZE_TOO_LARGE	HY000
10905	ER_FAILED_TO_GENERATE_UNIQUE_LOGFILE	HY000
10906	ER_FAILED_TO_READ_FILE	HY000
10907	ER_FAILED_TO_WRITE_TO_FILE	HY000
10908	ER_BINLOG_UNSAFE_MESSAGE_AND_STATEMENT	HY000
10909	ER_FORCE_CLOSE_THREAD	HY000
10910	ER_SERVER_SHUTDOWN_COMPLETE	HY000
10911	ER_RPL_CANT_HAVE_SAME_BASENAME	HY000
10912	ER_RPL_GTID_MODE_REQUIRES_ENFORCE_GTID_CONSISTENCY_ON	HY000
10913	ER_WARN_NO_SERVERID_SPECIFIED	HY000
10914	ER_ABORTING_USER_CONNECTION	HY000
10915	ER_SQL_MODE_MERGED_WITH_STRICT_MODE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10916	ER_GTID_PURGED_WAS_UPDATED	HY000
10917	ER_GTID_EXECUTED_WAS_UPDATED	HY000
10918	ER_DEPRECATE_MSG_WITH_REPLACEMENT	HY000
10919	ER_TRG_CREATION_CTX_NOT_SET	HY000
10920	ER_FILE_HAS_OLD_FORMAT	HY000
10921	ER_VIEW_CREATION_CTX_NOT_SET	HY000
10923	ER_TABLE_UPGRADE_REQUIRED	HY000
10924	ER_GET_ERRNO_FROM_STORAGE_ENGINE	HY000
10925	ER_ACCESS_DENIED_ERROR_WITHOUT_PASSWORD	HY000
10926	ER_ACCESS_DENIED_ERROR_WITH_PASSWORD	HY000
10927	ER_ACCESS_DENIED_FOR_USER_ACCOUNT_LOCKED	HY000
10929	ER_SYSTEM_TABLES_NOT_SUPPORTED_BY_STORAGE_ENGINE	HY000
10931	ER_SERVER_STARTUP_MSG	HY000
10932	ER_FAILED_TO_FIND_LOCALE_NAME	HY000
10933	ER_FAILED_TO_FIND_COLLATION_NAME	HY000
10934	ER_SERVER_OUT_OF_RESOURCES	HY000
10935	ER_SERVER_OUTOFMEMORY	HY000
10936	ER_INVALID_COLLATION_FOR_CHARSET	HY000
10937	ER_CANT_START_ERROR_LOG_SERVICE	HY000
10938	ER_CREATING_NEW_UUID_FIRST_START	HY000
10939	ER_FAILED_TO_GET_ABSOLUTE_PATH	HY000
10940	ER_PERFSCHEMA_COMPONENTS_INFRASTRUCTURE_BOOTSTRAP	HY000
10941	ER_PERFSCHEMA_COMPONENTS_INFRASTRUCTURE_SHUTDOWN	HY000
10942	ER_DUP_FD_OPEN_FAILED	HY000
10943	ER_SYSTEM_VIEW_INIT_FAILED	HY000
10944	ER_RESOURCE_GROUP_POST_INIT_FAILED	HY000
10945	ER_RESOURCE_GROUP_SUBSYSTEM_INIT_FAILED	HY000
10946	ER_FAILED_START_MYSQLD_DAEMON	HY000
10947	ER_CANNOT_CHANGE_TO_ROOT_DIR	HY000
10948	ER_PERSISTENT_PRIVILEGES_BOOTSTRAP	HY000
10949	ER_BASEDIR_SET_TO	HY000
10950	ER_RPL_FILTER_ADD_WILD_DO_TABLE_FAILED	HY000
10951	ER_RPL_FILTER_ADD_WILD_IGNORE_TABLE_FAILED	HY000
10952	ER_PRIVILEGE_SYSTEM_INIT_FAILED	HY000
10953	ER_CANNOT_SET_LOG_ERROR_SERVICES	HY000
10954	ER_PERFSCHEMA_TABLES_INIT_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10958	ER_BINLOG_FILE_OPEN_FAILED	HY000
10959	ER_BINLOG_EVENT_WRITE_TO_STMT_CACHE_FAILED	HY000
10960	ER_REPLICA_RELAY_LOG_TRUNCATE_INFO	HY000
10961	ER_REPLICA_RELAY_LOG_PURGE_FAILED	HY000
10962	ER_RPL_REPLICA_FILTER_CREATE_FAILED	HY000
10963	ER_RPL_REPLICA_GLOBAL_FILTERS_COPY_FAILED	HY000
10964	ER_RPL_REPLICA_RESET_FILTER_OPTIONS	HY000
10965	ER_MISSING_GRANT_SYSTEM_TABLE	HY000
10966	ER_MISSING_ACL_SYSTEM_TABLE	HY000
10967	ER_ANONYMOUS_AUTH_ID_NOT_ALLOWED_IN_MANDATORY_ROLES	HY000
10968	ER_UNKNOWN_AUTH_ID_IN_MANDATORY_ROLE	HY000
10969	ER_WRITE_ROW_TO_PARTITION_FAILED	HY000
10970	ER_RESOURCE_GROUP_METADATA_UPDATE_SKIPPED	HY000
10971	ER_FAILED_TO_PERSIST_RESOURCE_GROUP_METADATA	HY000
10972	ER_FAILED_TO_DESERIALIZE_RESOURCE_GROUP	HY000
10973	ER_FAILED_TO_UPDATE_RESOURCE_GROUP	HY000
10974	ER_RESOURCE_GROUP_VALIDATION_FAILED	HY000
10975	ER_FAILED_TO_ALLOCATE_MEMORY_FOR_RESOURCE_GROUP	HY000
10976	ER_FAILED_TO_ALLOCATE_MEMORY_FOR_RESOURCE_GROUP_HASH	HY000
10977	ER_FAILED_TO_ADD_RESOURCE_GROUP_TO_MAP	HY000
10978	ER_RESOURCE_GROUP_IS_DISABLED	HY000
10979	ER_FAILED_TO_APPLY_RESOURCE_GROUP_CONTROLLER	HY000
10980	ER_FAILED_TO_ACQUIRE_LOCK_ON_RESOURCE_GROUP	HY000
10981	ER_PFS_NOTIFICATION_FUNCTION_REGISTER_FAILED	HY000
10982	ER_RES_GRP_SET_THR_AFFINITY_FAILED	HY000
10983	ER_RES_GRP_SET_THR_AFFINITY_TO_CPUS_FAILED	HY000
10984	ER_RES_GRP_THD_UNBIND_FROM_CPU_FAILED	HY000
10985	ER_RES_GRP_SET_THREAD_PRIORITY_FAILED	HY000
10986	ER_RES_GRP_FAILED_TO_DETERMINE_NICE_CAPABILITY	HY000
10987	ER_RES_GRP_FAILED_TO_GET_THREAD_HANDLE	HY000
10988	ER_RES_GRP_GET_THREAD_PRIO_NOT_SUPPORTED	HY000
10989	ER_RES_GRP_FAILED_DETERMINE_CPU_COUNT	HY000
10990	ER_RES_GRP_FEATURE_NOT_AVAILABLE	HY000
10991	ER_RES_GRP_INVALID_THREAD_PRIORITY	HY000
10992	ER_RES_GRP_SOLARIS_PROCESSOR_BIND_TO_CPUID_FAILED	HY000
10993	ER_RES_GRP_SOLARIS_PROCESSOR_BIND_TO_THREAD_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
10994	ER_RES_GRP_SOLARIS_PROCESSOR_AFFINITY_FAILED	HY000
10995	ER_DD_UPGRADE_RENAME_IDX_STATS_FILE_FAILED	HY000
10996	ER_DD_UPGRADE_DD_OPEN_FAILED	HY000
10997	ER_DD_UPGRADE_FAILED_TO_FETCH_TABLESPACES	HY000
10998	ER_DD_UPGRADE_FAILED_TO_ACQUIRE_TABLESPACE	HY000
10999	ER_DD_UPGRADE_FAILED_TO_RESOLVE_TABLESPACE_ENGINE	HY000
11000	ER_FAILED_TO_CREATE_SDI_FOR_TABLESPACE	HY000
11001	ER_FAILED_TO_STORE_SDI_FOR_TABLESPACE	HY000
11002	ER_DD_UPGRADE_FAILED_TO_FETCH_TABLES	HY000
11003	ER_DD_UPGRADE_DD_POPULATED	HY000
11004	ER_DD_UPGRADE_INFO_FILE_OPEN_FAILED	HY000
11005	ER_DD_UPGRADE_INFO_FILE_CLOSE_FAILED	HY000
11006	ER_DD_UPGRADE_TABLESPACE_MIGRATION_FAILED	HY000
11007	ER_DD_UPGRADE_FAILED_TO_CREATE_TABLE_STATS	HY000
11008	ER_DD_UPGRADE_TABLE_STATS_MIGRATE_COMPLETED	HY000
11009	ER_DD_UPGRADE_FAILED_TO_CREATE_INDEX_STATS	HY000
11010	ER_DD_UPGRADE_INDEX_STATS_MIGRATE_COMPLETED	HY000
11011	ER_DD_UPGRADE_FAILED_FIND_VALID_DATA_DIR	HY000
11012	ER_DD_UPGRADE_START	HY000
11013	ER_DD_UPGRADE_FAILED_INIT_DD_SE	HY000
11014	ER_DD_UPGRADE_FOUND_PARTIALLY_UPGRADED_DD_ABORT	HY000
11015	ER_DD_UPGRADE_FOUND_PARTIALLY_UPGRADED_DD_CONTINUE	HY000
11016	ER_DD_UPGRADE_SE_LOGS_FAILED	HY000
11017	ER_DD_UPGRADE_SDI_INFO_UPDATE_FAILED	HY000
11018	ER_SKIP_UPDATING_METADATA_IN_SE_RO_MODE	HY000
11019	ER_CREATED_SYSTEM_WITH_VERSION	HY000
11020	ER_UNKNOWN_ERROR_DETECTED_IN_SE	HY000
11021	ER_READ_LOG_EVENT_FAILED	HY000
11022	ER_ROW_DATA_TOO_BIG_TO_WRITE_IN_BINLOG	HY000
11023	ER_FAILED_TO_CONSTRUCT_DROP_EVENT_QUERY	HY000
11024	ER_FAILED_TO_BINLOG_DROP_EVENT	HY000
11025	ER_FAILED_TO_START_REPLICA_THREAD	HY000
11026	ER_RPL_IO_THREAD_KILLED	HY000
11027	ER_REPLICA_RECONNECT_FAILED	HY000
11028	ER_REPLICA_KILLED_AFTER_RECONNECT	HY000
11029	ER_REPLICA_NOT_STARTED_ON_SOME_CHANNELS	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11030	ER_FAILED_TO_ADD_RPL_FILTER	HY000
11031	ER_PER_CHANNEL_RPL_FILTER_CONF_FOR_GRP_RPL	HY000
11032	ER_RPL_FILTERS_NOT_ATTACHED_TO_CHANNEL	HY000
11033	ER_FAILED_TO_BUILD_DO_AND_IGNORE_TABLE_HASHES	HY000
11034	ER_CLONE_PLUGIN_NOT_LOADED_TRACE	HY000
11035	ER_CLONE_HANDLER_EXIST_TRACE	HY000
11036	ER_CLONE_CREATE_HANDLER_FAIL_TRACE	HY000
11037	ER_CYCLE_TIMER_IS_NOT_AVAILABLE	HY000
11038	ER_NANOSECOND_TIMER_IS_NOT_AVAILABLE	HY000
11039	ER_MICROSECOND_TIMER_IS_NOT_AVAILABLE	HY000
11040	ER_PFS_MALLOC_ARRAY_OVERFLOW	HY000
11041	ER_PFS_MALLOC_ARRAY_OOM	HY000
11042	ER_INNODB_FAILED_TO_FIND_IDX_WITH_KEY_NO	HY000
11043	ER_INNODB_FAILED_TO_FIND_IDX	HY000
11044	ER_INNODB_FAILED_TO_FIND_IDX_FROM_DICT_CACHE	HY000
11045	ER_INNODB_ACTIVE_INDEX_CHANGE_FAILED	HY000
11046	ER_INNODB_DIFF_IN_REF_LEN	HY000
11047	ER_WRONG_TYPE_FOR_COLUMN_PREFIX_IDX_FLD	HY000
11048	ER_INNODB_CANNOT_CREATE_TABLE	HY000
11049	ER_INNODB_INTERNAL_INDEX	HY000
11050	ER_INNODB_IDX_CNT_MORE_THAN_DEFINED_IN_MYSQL	HY000
11051	ER_INNODB_IDX_CNT_FEWER_THAN_DEFINED_IN_MYSQL	HY000
11052	ER_INNODB_IDX_COLUMN_CNT_DIFF	HY000
11053	ER_INNODB_USE_MONITOR_GROUP_NAME	HY000
11054	ER_INNODB_MONITOR_DEFAULT_VALUE_NOT_DEFINED	HY000
11055	ER_INNODB_MONITOR_IS_ENABLED	HY000
11056	ER_INNODB_INVALID_MONITOR_COUNTER_NAME	HY000
11057	ER_WIN_LOAD_LIBRARY_FAILED	HY000
11058	ER_PARTITION_HANDLER_ADMIN_MSG	HY000
11059	ER_RPL_AM_INIT_INFO_MSG	HY000
11060	ER_DD_UPGRADE_TABLE_INTACT_ERROR	HY000
11061	ER_SERVER_INIT_COMPILED_IN_COMMANDS	HY000
11062	ER_MYISAM_CHECK_METHOD_ERROR	HY000
11063	ER_MYISAM_CRASHED_ERROR	HY000
11064	ER_WAITPID_FAILED	HY000
11065	ER_FAILED_TO_FIND_MYSQLD_STATUS	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11066	ER_INNODB_ERROR_LOGGER_MSG	HY000
11067	ER_INNODB_ERROR_LOGGER_FATAL_MSG	HY000
11068	ER_DEPRECATED_SYNTAX_WITH_REPLACEMENT	HY000
11069	ER_DEPRECATED_SYNTAX_NO_REPLACEMENT	HY000
11070	ER_DEPRECATE_MSG_NO_REPLACEMENT	HY000
11071	ER_LOG_PRINTF_MSG	HY000
11072	ER_BINLOG_LOGGING_NOT_POSSIBLE	HY000
11073	ER_FAILED_TO_SET_PERSISTED_OPTIONS	HY000
11074	ER_COMPONENTS_FAILED_TO_ACQUIRE_SERVICE_IMPLEMENTATION	HY000
11075	ER_RES_GRP_INVALID_VCPU_RANGE	HY000
11076	ER_RES_GRP_INVALID_VCPU_ID	HY000
11077	ER_ERROR_DURING_FLUSH_LOG_COMMIT_PHASE	HY000
11078	ER_DROP_DATABASE_FAILED_RMDIR_MANUALLY	HY000
11080	ER_BINLOG_MALFORMED_OR_OLD_RELAY_LOG	HY000
11081	ER_DD_UPGRADE_VIEW_COLUMN_NAME_TOO_LONG	HY000
11082	ER_TABLE_NEEDS_DUMP_UPGRADE	HY000
11083	ER_DD_UPGRADE_FAILED_TO_UPDATE_VER_NO_IN_TABLESPACE	HY000
11084	ER_KEYRING_MIGRATION_FAILED	HY000
11085	ER_KEYRING_MIGRATION_SUCCESSFUL	HY000
11086	ER_RESTART_RECEIVED_INFO	HY000
11087	ER_LCTN_CHANGED	HY000
11088	ER_DD_INITIALIZE	HY000
11089	ER_DD_RESTART	HY000
11090	ER_DD_UPGRADE	HY000
11091	ER_DD_UPGRADE_OFF	HY000
11092	ER_DD_UPGRADE_VERSION_NOT_SUPPORTED	HY000
11093	ER_DD_UPGRADE_SCHEMA_UNAVAILABLE	HY000
11094	ER_DD_MINOR_DOWNGRADE	HY000
11095	ER_DD_MINOR_DOWNGRADE_VERSION_NOT_SUPPORTED	HY000
11096	ER_DD_NO_VERSION_FOUND	HY000
11097	ER_THREAD_POOL_NOT_SUPPORTED_ON_PLATFORM	HY000
11098	ER_THREAD_POOL_SIZE_TOO_LOW	HY000
11099	ER_THREAD_POOL_SIZE_TOO_HIGH	HY000
11100	ER_THREAD_POOL_ALGORITHM_INVALID	HY000
11101	ER_THREAD_POOL_INVALID_STALL_LIMIT	HY000
11102	ER_THREAD_POOL_INVALID_PRIO_KICKUP_TIMER	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11103	ER_THREAD_POOL_MAX_UNUSED_THREADS_INVALID	HY000
11104	ER_THREAD_POOL_CON_HANDLER_INIT_FAILED	HY000
11105	ER_THREAD_POOL_INIT_FAILED	HY000
11107	ER_THREAD_POOL_CANNOT_SET_THREAD_SPECIFIC_DATA	HY000
11108	ER_THREAD_POOL_FAILED_TO_CREATE_CONNECT_HANDLER_THD	HY000
11109	ER_THREAD_POOL_FAILED_TO_CREATE_THD_AND_AUTH_CONN	HY000
11110	ER_THREAD_POOL_FAILED_PROCESS_CONNECT_EVENT	HY000
11111	ER_THREAD_POOL_FAILED_TO_CREATE_POOL	HY000
11112	ER_THREAD_POOL_RATE_LIMITED_ERROR_MSGS	HY000
11113	ER_TRHEAD_POOL_LOW_LEVEL_INIT_FAILED	HY000
11114	ER_THREAD_POOL_LOW_LEVEL_REARM_FAILED	HY000
11115	ER_THREAD_POOL_BUFFER_TOO_SMALL	HY000
11116	ER_MECAB_NOT_SUPPORTED	HY000
11117	ER_MECAB_NOT_VERIFIED	HY000
11118	ER_MECAB_CREATING_MODEL	HY000
11119	ER_MECAB_FAILED_TO_CREATE_MODEL	HY000
11120	ER_MECAB_FAILED_TO_CREATE_TRIGGER	HY000
11121	ER_MECAB_UNSUPPORTED_CHARSET	HY000
11122	ER_MECAB_CHARSET_LOADED	HY000
11123	ER_MECAB_PARSE_FAILED	HY000
11124	ER_MECAB_OOM_WHILE_PARSING_TEXT	HY000
11125	ER_MECAB_CREATE_LATTICE_FAILED	HY000
11126	ER_SEMISYNC_TRACE_ENTER_FUNC	HY000
11127	ER_SEMISYNC_TRACE_EXIT_WITH_INT_EXIT_CODE	HY000
11128	ER_SEMISYNC_TRACE_EXIT_WITH_BOOL_EXIT_CODE	HY000
11129	ER_SEMISYNC_TRACE_EXIT	HY000
11130	ER_SEMISYNC_RPL_INIT_FOR_TRX	HY000
11131	ER_SEMISYNC_FAILED_TO_ALLOCATE_TRX_NODE	HY000
11132	ER_SEMISYNC_BINLOG_WRITE_OUT_OF_ORDER	HY000
11133	ER_SEMISYNC_INSERT_LOG_INFO_IN_ENTRY	HY000
11134	ER_SEMISYNC_PROBE_LOG_INFO_IN_ENTRY	HY000
11135	ER_SEMISYNC_CLEARED_ALL_ACTIVE_TRANSACTION_NODES	HY000
11136	ER_SEMISYNC_CLEARED_ACTIVE_TRANSACTION_TILL_POS	HY000
11137	ER_SEMISYNC_REPLY_MAGIC_NO_ERROR	HY000
11138	ER_SEMISYNC_REPLY_PKT_LENGTH_TOO_SMALL	HY000
11139	ER_SEMISYNC_REPLY_BINLOG_FILE_TOO_LARGE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11140	ER_SEMISYNC_SERVER_REPLY	HY000
11141	ER_SEMISYNC_FUNCTION_CALLED_TWICE	HY000
11142	ER_SEMISYNC_RPL_ENABLED_ON_SOURCE	HY000
11143	ER_SEMISYNC_SOURCE_OOM	HY000
11144	ER_SEMISYNC_DISABLED_ON_SOURCE	HY000
11145	ER_SEMISYNC_FORCED_SHUTDOWN	HY000
11146	ER_SEMISYNC_SOURCE_GOT_REPLY_AT_POS	HY000
11147	ER_SEMISYNC_SOURCE_SIGNAL_ALL_WAITING_THREADS	HY000
11148	ER_SEMISYNC_SOURCE_TRX_WAIT_POS	HY000
11149	ER_SEMISYNC_BINLOG_REPLY_IS_AHEAD	HY000
11150	ER_SEMISYNC_MOVE_BACK_WAIT_POS	HY000
11151	ER_SEMISYNC_INIT_WAIT_POS	HY000
11152	ER_SEMISYNC_WAIT_TIME_FOR_BINLOG_SENT	HY000
11153	ER_SEMISYNC_WAIT_FOR_BINLOG_TIMEDOUT	HY000
11154	ER_SEMISYNC_WAIT_TIME_ASSESSMENT_FOR_COMMIT_TRX_FAILED	HY000
11155	ER_SEMISYNC_RPL_SWITCHED_OFF	HY000
11156	ER_SEMISYNC_RPL_SWITCHED_ON	HY000
11157	ER_SEMISYNC_NO_SPACE_IN_THE_PKT	HY000
11158	ER_SEMISYNC_SYNC_HEADER_UPDATE_INFO	HY000
11159	ER_SEMISYNC_FAILED_TO_INSERT_TRX_NODE	HY000
11160	ER_SEMISYNC_TRX_SKIPPED_AT_POS	HY000
11161	ER_SEMISYNC_SOURCE_FAILED_ON_NET_FLUSH	HY000
11162	ER_SEMISYNC_RECEIVED_ACK_IS_SMALLER	HY000
11163	ER_SEMISYNC_ADD_ACK_TO_SLOT	HY000
11164	ER_SEMISYNC_UPDATE_EXISTING_REPLICA_ACK	HY000
11165	ER_SEMISYNC_FAILED_TO_START_ACK_RECEIVER_THD	HY000
11166	ER_SEMISYNC_STARTING_ACK_RECEIVER_THD	HY000
11167	ER_SEMISYNC_FAILED_TO_WAIT_ON_DUMP_SOCKET	HY000
11168	ER_SEMISYNC_STOPPING_ACK_RECEIVER_THREAD	HY000
11169	ER_SEMISYNC_FAILED_REGISTER_REPLICA_TO_RECEIVER	HY000
11170	ER_SEMISYNC_START_BINLOG_DUMP_TO_REPLICA	HY000
11171	ER_SEMISYNC_STOP_BINLOG_DUMP_TO_REPLICA	HY000
11172	ER_SEMISYNC_UNREGISTER_TRX_OBSERVER_FAILED	HY000
11173	ER_SEMISYNC_UNREGISTER_BINLOG_STORAGE_OBSERVER_FAILED	HY000
11174	ER_SEMISYNC_UNREGISTER_BINLOG_TRANSMIT_OBSERVER_FAILED	HY000
11175	ER_SEMISYNC_UNREGISTERED_REPLICATOR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11176	ER_SEMISYNC_SOCKET_FD_TOO_LARGE	HY000
11177	ER_SEMISYNC_REPLICA_REPLY	HY000
11178	ER_SEMISYNC_MISSING_MAGIC_NO_FOR_SEMISYNC_PKT	HY000
11179	ER_SEMISYNC_REPLICA_START	HY000
11180	ER_SEMISYNC_REPLICA_REPLY_WITH_BINLOG_INFO	HY000
11181	ER_SEMISYNC_REPLICA_NET_FLUSH_REPLY_FAILED	HY000
11182	ER_SEMISYNC_REPLICA_SEND_REPLY_FAILED	HY000
11183	ER_SEMISYNC_EXECUTION_FAILED_ON_SOURCE	HY000
11184	ER_SEMISYNC_NOT_SUPPORTED_BY_SOURCE	HY000
11185	ER_SEMISYNC_REPLICA_SET_FAILED	HY000
11186	ER_SEMISYNC_FAILED_TO_STOP_ACK_RECEIVER_THD	HY000
11187	ER_FIREWALL_FAILED_TO_READ_FIREWALL_TABLES	HY000
11188	ER_FIREWALL_FAILED_TO_REG_DYNAMIC_PRIVILEGES	HY000
11189	ER_FIREWALL_RECORDING_STMT_WAS_TRUNCATED	HY000
11190	ER_FIREWALL_RECORDING_STMT_WITHOUT_TEXT	HY000
11191	ER_FIREWALL_SUSPICIOUS_STMT	HY000
11192	ER_FIREWALL_ACCESS_DENIED	HY000
11193	ER_FIREWALL_SKIPPED_UNKNOWN_USER_MODE	HY000
11194	ER_FIREWALL_RELOADING_CACHE	HY000
11195	ER_FIREWALL_RESET_FOR_USER	HY000
11196	ER_FIREWALL_STATUS_FLUSHED	HY000
11197	ER_KEYRING_LOGGER_ERROR_MSG	HY000
11198	ER_AUDIT_LOG_FILTER_IS_NOT_INSTALLED	HY000
11199	ER_AUDIT_LOG_SWITCHING_TO_INCLUDE_LIST	HY000
11200	ER_AUDIT_LOG_CANNOT_SET_LOG_POLICY_WITH_OTHER_POLICIES	HY000
11201	ER_AUDIT_LOG_ONLY_INCLUDE_LIST_USED	HY000
11202	ER_AUDIT_LOG_INDEX_MAP_CANNOT_ACCESS_DIR	HY000
11203	ER_AUDIT_LOG_WRITER_RENAME_FILE_FAILED	HY000
11204	ER_AUDIT_LOG_WRITER_DEST_FILE_ALREADY_EXISTS	HY000
11205	ER_AUDIT_LOG_WRITER_RENAME_FILE_FAILED_REMOVE_FILE_MANUALLY	HY000
11206	ER_AUDIT_LOG_WRITER_INCOMPLETE_FILE_RENAMED	HY000
11207	ER_AUDIT_LOG_WRITER_FAILED_TO_WRITE_TO_FILE	HY000
11208	ER_AUDIT_LOG_EC_WRITER_FAILED_TO_INIT_ENCRYPTION	HY000
11209	ER_AUDIT_LOG_EC_WRITER_FAILED_TO_INIT_COMPRESSION	HY000
11210	ER_AUDIT_LOG_EC_WRITER_FAILED_TO_CREATE_FILE	HY000
11211	ER_AUDIT_LOG_RENAME_LOG_FILE_BEFORE_FLUSH	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11212	ER_AUDIT_LOG_FILTER_RESULT_MSG	HY000
11213	ER_AUDIT_LOG_JSON_READER_FAILED_TO_PARSE	HY000
11214	ER_AUDIT_LOG_JSON_READER_BUF_TOO_SMALL	HY000
11215	ER_AUDIT_LOG_JSON_READER_FAILED_TO_OPEN_FILE	HY000
11216	ER_AUDIT_LOG_JSON_READER_FILE_PARSING_ERROR	HY000
11219	ER_AUDIT_LOG_FILTER_FAILED_TO_STORE_TABLE_FLDS	HY000
11220	ER_AUDIT_LOG_FILTER_FAILED_TO_UPDATE_TABLE	HY000
11221	ER_AUDIT_LOG_FILTER_FAILED_TO_INSERT_INTO_TABLE	HY000
11222	ER_AUDIT_LOG_FILTER_FAILED_TO_DELETE_FROM_TABLE	HY000
11223	ER_AUDIT_LOG_FILTER_FAILED_TO_INIT_TABLE_FOR_READ	HY000
11224	ER_AUDIT_LOG_FILTER_FAILED_TO_READ_TABLE	HY000
11225	ER_AUDIT_LOG_FILTER_FAILED_TO_CLOSE_TABLE_AFTER_READING	HY000
11226	ER_AUDIT_LOG_FILTER_USER_AND_HOST_CANNOT_BE_EMPTY	HY000
11227	ER_AUDIT_LOG_FILTER_FLD_FILTERNAME_CANNOT_BE_EMPTY	HY000
11228	ER_VALIDATE_PWD_DICT_FILE_NOT_SPECIFIED	HY000
11229	ER_VALIDATE_PWD_DICT_FILE_NOT_LOADED	HY000
11230	ER_VALIDATE_PWD_DICT_FILE_TOO_BIG	HY000
11231	ER_VALIDATE_PWD_FAILED_TO_READ_DICT_FILE	HY000
11232	ER_VALIDATE_PWD_FAILED_TO_GET_FLD_FROM_SECURITY_CTX	HY000
11233	ER_VALIDATE_PWD_FAILED_TO_GET_SECURITY_CTX	HY000
11234	ER_VALIDATE_PWD_LENGTH_CHANGED	HY000
11235	ER_REWRITER_QUERY_ERROR_MSG	HY000
11236	ER_REWRITER_QUERY_FAILED	HY000
11237	ER_XPLUGIN_STARTUP_FAILED	HY000
11240	ER_XPLUGIN_USING_SSL_CONF_FROM_SERVER	HY000
11241	ER_XPLUGIN_USING_SSL_CONF_FROM_MYSQLX	HY000
11242	ER_XPLUGIN_FAILED_TO_USE_SSL_CONF	HY000
11243	ER_XPLUGIN_USING_SSL_FOR_TLS_CONNECTION	HY000
11244	ER_XPLUGIN_REFERENCE_TO_SECURE_CONN_WITH_XPLUGIN	HY000
11245	ER_XPLUGIN_ERROR_MSG	HY000
11246	ER_SHA_PWD_FAILED_TO_PARSE_AUTH_STRING	HY000
11247	ER_SHA_PWD_FAILED_TO_GENERATE_MULTI_ROUND_HASH	HY000
11248	ER_SHA_PWD_AUTH_REQUIRES_RSA_OR_SSL	HY000
11249	ER_SHA_PWD_RSA_KEY_TOO_LONG	HY000
11250	ER_PLUGIN_COMMON_FAILED_TO_OPEN_FILTER_TABLES	HY000
11251	ER_PLUGIN_COMMON_FAILED_TO_OPEN_TABLE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11252	ER_AUTH_LDAP_ERROR_LOGGER_ERROR_MSG	HY000
11253	ER_CONN_CONTROL_ERROR_MSG	HY000
11254	ER_GRP_RPL_ERROR_MSG	HY000
11255	ER_SHA_PWD_SALT_FOR_USER_CORRUPT	HY000
11256	ER_SYS_VAR_COMPONENT_OOM	HY000
11257	ER_SYS_VAR_COMPONENT_VARIABLE_SET_READ_ONLY	HY000
11258	ER_SYS_VAR_COMPONENT_UNKNOWN_VARIABLE_TYPE	HY000
11259	ER_SYS_VAR_COMPONENT_FAILED_TO_PARSE_VARIABLE_OPTIONS	HY000
11260	ER_SYS_VAR_COMPONENT_FAILED_TO_MAKE_VARIABLE_PERSISTENT	HY000
11261	ER_COMPONENT_FILTER_CONFUSED	HY000
11262	ER_STOP_REPLICA_IO_THREAD_DISK_SPACE	HY000
11263	ER_LOG_FILE_CANNOT_OPEN	HY000
11268	ER_PERSIST_OPTION_STATUS	HY000
11269	ER_NOT_IMPLEMENTED_GET_TABLESPACE_STATISTICS	HY000
11272	ER_SSL_FIPS_MODE_ERROR	HY000
11273	ER_CONN_INIT_CONNECT_IGNORED	HY000
11275	ER_REWRITER_OOM	HY000
11276	ER_REWRITER_TABLE_MALFORMED_ERROR	HY000
11277	ER_REWRITER_LOAD_FAILED	HY000
11278	ER_REWRITER_READ_FAILED	HY000
11279	ER_CONN_CONTROL_EVENT_COORDINATOR_INIT_FAILED	HY000
11280	ER_CONN_CONTROL_STAT_CONN_DELAY_TRIGGERED_UPDATE_FAILED	HY000
11281	ER_CONN_CONTROL_STAT_CONN_DELAY_TRIGGERED_RESET_FAILED	HY000
11282	ER_CONN_CONTROL_INVALID_CONN_DELAY_TYPE	HY000
11283	ER_CONN_CONTROL_DELAY_ACTION_INIT_FAILED	HY000
11284	ER_CONN_CONTROL_FAILED_TO_SET_CONN_DELAY	HY000
11285	ER_CONN_CONTROL_FAILED_TO_UPDATE_CONN_DELAY_HASH	HY000
11286	ER_XPLUGIN_FORCE_STOP_CLIENT	HY000
11287	ER_XPLUGIN_MAX_AUTH_ATTEMPTS_REACHED	HY000
11288	ER_XPLUGIN_BUFFER_PAGE_ALLOC_FAILED	HY000
11289	ER_XPLUGIN_DETECTED_HANGING_CLIENTS	HY000
11290	ER_XPLUGIN_FAILED_TO_ACCEPT_CLIENT	HY000
11291	ER_XPLUGIN_FAILED_TO_SCHEDULE_CLIENT	HY000
11292	ER_XPLUGIN_FAILED_TO_PREPARE_IO_INTERFACES	HY000
11293	ER_XPLUGIN_SRV_SESSION_INIT_THREAD_FAILED	HY000
11294	ER_XPLUGIN_UNABLE_TO_USE_USER_SESSION_ACCOUNT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11295	ER_XPLUGIN_REFERENCE_TO_USER_ACCOUNT_DOC_SECTION	HY000
11296	ER_XPLUGIN_UNEXPECTED_EXCEPTION_DISPATCHING_CMD	HY000
11297	ER_XPLUGIN_EXCEPTION_IN_TASK_SCHEDULER	HY000
11298	ER_XPLUGIN_TASK_SCHEDULING_FAILED	HY000
11299	ER_XPLUGIN_EXCEPTION_IN_EVENT_LOOP	HY000
11300	ER_XPLUGIN_LISTENER_SETUP_FAILED	HY000
11301	ER_XPLUING_NET_STARTUP_FAILED	HY000
11302	ER_XPLUGIN_FAILED_AT_SSL_CONF	HY000
11305	ER_XPLUGIN_FAILED_TO_CREATE_SESSION_FOR_CONN	HY000
11306	ER_XPLUGIN_FAILED_TO_INITIALIZE_SESSION	HY000
11307	ER_XPLUGIN_MESSAGE_TOO_LONG	HY000
11308	ER_XPLUGIN_UNINITIALIZED_MESSAGE	HY000
11309	ER_XPLUGIN_FAILED_TO_SET_MIN_NUMBER_OF_WORKERS	HY000
11310	ER_XPLUGIN_UNABLE_TO_ACCEPT_CONNECTION	HY000
11311	ER_XPLUGIN_ALL_IO_INTERFACES_DISABLED	HY000
11314	ER_XPLUGIN_ERROR_READING_SOCKET	HY000
11315	ER_XPLUGIN_PEER_DISCONNECTED_WHILE_READING_MSG_BODY	HY000
11316	ER_XPLUGIN_READ_FAILED_CLOSING_CONNECTION	HY000
11322	ER_XPLUGIN_LISTENER_SYS_VARIABLE_ERROR	HY000
11323	ER_XPLUGIN_LISTENER_STATUS_MSG	HY000
11324	ER_XPLUGIN_RETRYING_BIND_ON_PORT	HY000
11327	ER_XPLUGIN_EXISTING_USER_ACCOUNT_WITH_INCOMPLETE_GRANTS	HY000
11332	ER_XPLUGIN_IPv6_AVAILABLE	HY000
11334	ER_XPLUGIN_CLIENT_KILL_MSG	HY000
11335	ER_XPLUGIN_FAILED_TO_GET_SECURITY_CTX	HY000
11337	ER_XPLUGIN_FAILED_TO_CLOSE_SQL_SESSION	HY000
11338	ER_XPLUGIN_FAILED_TO_EXECUTE_ADMIN_CMD	HY000
11339	ER_XPLUGIN_EMPTY_ADMIN_CMD	HY000
11340	ER_XPLUGIN_FAILED_TO_GET_SYS_VAR	HY000
11341	ER_XPLUGIN_FAILED_TO_GET_CREATION_STMT	HY000
11342	ER_XPLUGIN_FAILED_TO_GET_ENGINE_INFO	HY000
11345	ER_XPLUGIN_FAILED_TO_SET_SO_REUSEADDR_FLAG	HY000
11346	ER_XPLUGIN_FAILED_TO_OPEN_INTERNAL_SESSION	HY000
11347	ER_XPLUGIN_FAILED_TO_SWITCH_CONTEXT	HY000
11348	ER_XPLUGIN_FAILED_TO_UNREGISTER_UDF	HY000
11351	ER_XPLUGIN_FAILED_TO_RESET_IPV6_V6ONLY_FLAG	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11352	ER_KEYRING_INVALID_KEY_TYPE	HY000
11353	ER_KEYRING_INVALID_KEY_LENGTH	HY000
11358	ER_KEYRING_CHECK_KEY_FAILED_DUE_TO_INVALID_KEY	HY000
11359	ER_KEYRING_CHECK_KEY_FAILED_DUE_TO_EMPTY_KEY_ID	HY000
11360	ER_KEYRING_OPERATION_FAILED_DUE_TO_INTERNAL_ERROR	HY000
11361	ER_KEYRING_INCORRECT_FILE	HY000
11362	ER_KEYRING_FOUND_MALFORMED_BACKUP_FILE	HY000
11363	ER_KEYRING_FAILED_TO_RESTORE_FROM_BACKUP_FILE	HY000
11364	ER_KEYRING_FAILED_TO_FLUSH_KEYRING_TO_FILE	HY000
11365	ER_KEYRING_FAILED_TO_GET_FILE_STAT	HY000
11366	ER_KEYRING_FAILED_TO_REMOVE_FILE	HY000
11367	ER_KEYRING_FAILED_TO_TRUNCATE_FILE	HY000
11368	ER_KEYRING_UNKNOWN_ERROR	HY000
11370	ER_KEYRING_FILE_IO_ERROR	HY000
11371	ER_KEYRING_FAILED_TO_LOAD_KEYRING_CONTENT	HY000
11372	ER_KEYRING_FAILED_TO_FLUSH_KEYS_TO_KEYRING	HY000
11373	ER_KEYRING_FAILED_TO_FLUSH_KEYS_TO_KEYRING_BACKUP	HY000
11374	ER_KEYRING_KEY_FETCH_FAILED_DUE_TO_EMPTY_KEY_ID	HY000
11375	ER_KEYRING_FAILED_TO_REMOVE_KEY_DUE_TO_EMPTY_ID	HY000
11376	ER_KEYRING_OKV_INCORRECT_KEY_VAULT_CONFIGURED	HY000
11377	ER_KEYRING_OKV_INIT_FAILED_DUE_TO_INCORRECT_CONF	HY000
11378	ER_KEYRING_OKV_INIT_FAILED_DUE_TO_INTERNAL_ERROR	HY000
11379	ER_KEYRING_OKV_INVALID_KEY_TYPE	HY000
11380	ER_KEYRING_OKV_INVALID_KEY_LENGTH_FOR_CIPHER	HY000
11381	ER_KEYRING_OKV_FAILED_TO_GENERATE_KEY_DUE_TO_INTERNAL_ERROR	HY000
11382	ER_KEYRING_OKV_FAILED_TO_FIND_SERVER_ENTRY	HY000
11383	ER_KEYRING_OKV_FAILED_TO_FIND_STANDBY_SERVER_ENTRY	HY000
11384	ER_KEYRING_OKV_FAILED_TO_PARSE_CONF_FILE	HY000
11385	ER_KEYRING_OKV_FAILED_TO_LOAD_KEY_UID	HY000
11386	ER_KEYRING_OKV_FAILED_TO_INIT_SSL_LAYER	HY000
11387	ER_KEYRING_OKV_FAILED_TO_INIT_CLIENT	HY000
11388	ER_KEYRING_OKV_CONNECTION_TO_SERVER_FAILED	HY000
11389	ER_KEYRING_OKV_FAILED_TO_REMOVE_KEY	HY000
11390	ER_KEYRING_OKV_FAILED_TO_ADD_ATTRIBUTE	HY000
11391	ER_KEYRING_OKV_FAILED_TO_GENERATE_KEY	HY000
11392	ER_KEYRING_OKV_FAILED_TO_STORE_KEY	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11393	ER_KEYRING_OKV_FAILED_TO_ACTIVATE_KEYS	HY000
11394	ER_KEYRING_OKV_FAILED_TO_FETCH_KEY	HY000
11395	ER_KEYRING_OKV_FAILED_TO_STORE_OR_GENERATE_KEY	HY000
11396	ER_KEYRING_OKV_FAILED_TO_RETRIEVE_KEY_SIGNATURE	HY000
11397	ER_KEYRING_OKV_FAILED_TO_RETRIEVE_KEY	HY000
11398	ER_KEYRING_OKV_FAILED_TO_LOAD_SSL_TRUST_STORE	HY000
11399	ER_KEYRING_OKV_FAILED_TO_SET_CERTIFICATE_FILE	HY000
11400	ER_KEYRING_OKV_FAILED_TO_SET_KEY_FILE	HY000
11401	ER_KEYRING_OKV_KEY_MISMATCH	HY000
11415	ER_KEYRING_AWS_FAILED_TO_SET_CMK_ID	HY000
11416	ER_KEYRING_AWS_FAILED_TO_SET_REGION	HY000
11417	ER_KEYRING_AWS_FAILED_TO_OPEN_CONF_FILE	HY000
11418	ER_KEYRING_AWS_FAILED_TO_ACCESS_KEY_ID_FROM_CONF_FILE	HY000
11419	ER_KEYRING_AWS_FAILED_TO_ACCESS_KEY_FROM_CONF_FILE	HY000
11420	ER_KEYRING_AWS_INVALID_CONF_FILE_PATH	HY000
11421	ER_KEYRING_AWS_INVALID_DATA_FILE_PATH	HY000
11422	ER_KEYRING_AWS_FAILED_TO_ACCESS_OR_CREATE_KEYRING_DIR	HY000
11423	ER_KEYRING_AWS_FAILED_TO_ACCESS_OR_CREATE_KEYRING_DATA_FILE	HY000
11424	ER_KEYRING_AWS_FAILED_TO_INIT_DUE_TO_INTERNAL_ERROR	HY000
11425	ER_KEYRING_AWS_FAILED_TO_ACCESS_DATA_FILE	HY000
11426	ER_KEYRING_AWS_CMK_ID_NOT_SET	HY000
11427	ER_KEYRING_AWS_FAILED_TO_GET_KMS_CREDENTIAL_FROM_CONF_FILE	HY000
11428	ER_KEYRING_AWS_INIT_FAILURE	HY000
11429	ER_KEYRING_AWS_FAILED_TO_INIT_DUE_TO_PLUGIN_INTERNAL_ERROR	HY000
11430	ER_KEYRING_AWS_INVALID_KEY_LENGTH_FOR_CIPHER	HY000
11431	ER_KEYRING_AWS_FAILED_TO_GENERATE_KEY_DUE_TO_INTERNAL_ERROR	HY000
11432	ER_KEYRING_AWS_INCORRECT_FILE	HY000
11433	ER_KEYRING_AWS_FOUND_MALFORMED_BACKUP_FILE	HY000
11434	ER_KEYRING_AWS_FAILED_TO_RESTORE_FROM_BACKUP_FILE	HY000
11435	ER_KEYRING_AWS_FAILED_TO_FLUSH_KEYRING_TO_FILE	HY000
11436	ER_KEYRING_AWS_INCORRECT_REGION	HY000
11437	ER_KEYRING_AWS_FAILED_TO_CONNECT_KMS	HY000
11438	ER_KEYRING_AWS_FAILED_TO_GENERATE_NEW_KEY	HY000
11439	ER_KEYRING_AWS_FAILED_TO_ENCRYPT_KEY	HY000
11440	ER_KEYRING_AWS_FAILED_TO_RE_ENCRYPT_KEY	HY000
11441	ER_KEYRING_AWS_FAILED_TO_DECRYPT_KEY	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11442	ER_KEYRING_AWS_FAILED_TO_ROTATE_CMK	HY000
11443	ER_GRP_RPL_GTID_ALREADY_USED	HY000
11444	ER_GRP_RPL_APPLIER_THD_KILLED	HY000
11445	ER_GRP_RPL_EVENT_HANDLING_ERROR	HY000
11446	ER_GRP_RPL_ERROR_GTID_EXECUTION_INFO	HY000
11448	ER_GRP_RPL_CREATE_APPLIER_CACHE_ERROR	HY000
11449	ER_GRP_RPL_UNBLOCK_WAITING_THD	HY000
11450	ER_GRP_RPL_APPLIER_PIPELINE_NOT_DISPOSED	HY000
11451	ER_GRP_RPL_APPLIER_THD_EXECUTION_ABORTED	HY000
11452	ER_GRP_RPL_APPLIER_EXECUTION_FATAL_ERROR	HY000
11453	ER_GRP_RPL_ERROR_STOPPING_CHANNELS	HY000
11454	ER_GRP_RPL_ERROR_SENDING_SINGLE_PRIMARY_MSSG	HY000
11457	ER_GRP_RPL_BROADCAST_COMMIT_TRANS_MSSG_FAILED	HY000
11458	ER_GRP_RPL_GROUP_NAME_PARSE_ERROR	HY000
11459	ER_GRP_RPL_ADD_GRPSID_TO_GRPGTIDSID_MAP_ERROR	HY000
11460	ER_GRP_RPL_UPDATE_GRPGTID_EXECUTED_ERROR	HY000
11461	ER_GRP_RPL_DONOR_TRANS_INFO_ERROR	HY000
11462	ER_GRP_RPL_SERVER_CONN_ERROR	HY000
11463	ER_GRP_RPL_ERROR_FETCHING_GTID_EXECUTED_SET	HY000
11464	ER_GRP_RPL_ADD_GTID_TO_GRPGTID_EXECUTED_ERROR	HY000
11465	ER_GRP_RPL_ERROR_FETCHING_GTID_SET	HY000
11466	ER_GRP_RPL_ADD_RETRIEVED_SET_TO_GRP_GTID_EXECUTED_ERROR	HY000
11467	ER_GRP_RPL_CERTIFICATION_INITIALIZATION_FAILURE	HY000
11468	ER_GRP_RPL_UPDATE_LAST_CONFLICT_FREE_TRANS_ERROR	HY000
11469	ER_GRP_RPL_UPDATE_TRANS_SNAPSHOT_REF_VER_ERROR	HY000
11472	ER_GRP_RPL_CANT_GENERATE_GTID	HY000
11473	ER_GRP_RPL_INVALID_GTID_SET	HY000
11474	ER_GRP_RPL_UPDATE_GTID_SET_ERROR	HY000
11475	ER_GRP_RPL_RECEIVED_SET_MISSING_GTIDS	HY000
11477	ER_GRP_RPL_NULL_PACKET	HY000
11478	ER_GRP_RPL_CANT_READ_GTID	HY000
11479	ER_GRP_RPL_PROCESS_GTID_SET_ERROR	HY000
11480	ER_GRP_RPL_PROCESS_INTERSECTION_GTID_SET_ERROR	HY000
11481	ER_GRP_RPL_SET_STABLE_TRANS_ERROR	HY000
11482	ER_GRP_RPL_CANT_READ_GRP_GTID_EXTRACTED	HY000
11483	ER_GRP_RPL_CANT_READ_WRITE_SET_ITEM	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11484	ER_GRP_RPL_INIT_CERTIFICATION_INFO_FAILURE	HY000
11485	ER_GRP_RPL_CONFLICT_DETECTION_DISABLED	HY000
11486	ER_GRP_RPL_MSG_DISCARDED	HY000
11487	ER_GRP_RPL_MISSING_GRP_RPL_APPLIER	HY000
11488	ER_GRP_RPL_CERTIFIER_MSSG_PROCESS_ERROR	HY000
11489	ER_GRP_RPL_SRV_NOT_ONLINE	HY000
11490	ER_GRP_RPL_SRV_ONLINE	HY000
11491	ER_GRP_RPL_DISABLE_SRV_READ_MODE_RESTRICTED	HY000
11492	ER_GRP_RPL_MEM_ONLINE	HY000
11493	ER_GRP_RPL_MEM_UNREACHABLE	HY000
11494	ER_GRP_RPL_MEM_REACHABLE	HY000
11495	ER_GRP_RPL_SRV_BLOCKED	HY000
11496	ER_GRP_RPL_SRV_BLOCKED_FOR_SECS	HY000
11497	ER_GRP_RPL_CHANGE_GRP_MEM_NOT_PROCESSED	HY000
11498	ER_GRP_RPL_MEMBER_CONTACT_RESTORED	HY000
11499	ER_GRP_RPL_MEMBER_REMOVED	HY000
11500	ER_GRP_RPL_PRIMARY_MEMBER_LEFT_GRP	HY000
11501	ER_GRP_RPL_MEMBER_ADDED	HY000
11502	ER_GRP_RPL_MEMBER_EXIT_PLUGIN_ERROR	HY000
11503	ER_GRP_RPL_MEMBER_CHANGE	HY000
11504	ER_GRP_RPL_MEMBER_LEFT_GRP	HY000
11505	ER_GRP_RPL_MEMBER_EXPELLED	HY000
11506	ER_GRP_RPL_SESSION_OPEN_FAILED	HY000
11507	ER_GRP_RPL_NEW_PRIMARY_ELECTED	HY000
11508	ER_GRP_RPL_DISABLE_READ_ONLY_FAILED	HY000
11509	ER_GRP_RPL_ENABLE_READ_ONLY_FAILED	HY000
11510	ER_GRP_RPL_SRV_PRIMARY_MEM	HY000
11511	ER_GRP_RPL_SRV_SECONDARY_MEM	HY000
11512	ER_GRP_RPL_NO_SUITABLE_PRIMARY_MEM	HY000
11513	ER_GRP_RPL_SUPER_READ_ONLY_ACTIVATE_ERROR	HY000
11514	ER_GRP_RPL_EXCEEDS_AUTO_INC_VALUE	HY000
11515	ER_GRP_RPL_DATA_NOT_PROVIDED_BY_MEM	HY000
11516	ER_GRP_RPL_MEMBER_ALREADY_EXISTS	HY000
11518	ER_GRP_RPL_GTID_EXECUTED_EXTRACT_ERROR	HY000
11519	ER_GRP_RPL_GTID_SET_EXTRACT_ERROR	HY000
11520	ER_GRP_RPL_START_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11521	ER_GRP_RPL_MEMBER_VER_INCOMPATIBLE	HY000
11522	ER_GRP_RPL_TRANS_NOT_PRESENT_IN_GRP	HY000
11523	ER_GRP_RPL_TRANS_GREATER_THAN_GRP	HY000
11524	ER_GRP_RPL_MEMBER_VERSION_LOWER_THAN_GRP	HY000
11525	ER_GRP_RPL_LOCAL_GTID_SETS_PROCESS_ERROR	HY000
11526	ER_GRP_RPL_MEMBER_TRANS_GREATER_THAN_GRP	HY000
11527	ER_GRP_RPL_BLOCK_SIZE_DIFF_FROM_GRP	HY000
11528	ER_GRP_RPL_TRANS_WRITE_SET_EXTRACT_DIFF_FROM_GRP	HY000
11529	ER_GRP_RPL_MEMBER_CFG_INCOMPATIBLE_WITH_GRP_CFG	HY000
11530	ER_GRP_RPL_MEMBER_STOP_RPL_CHANNELS_ERROR	HY000
11531	ER_GRP_RPL_PURGE_APPLIER_LOGS	HY000
11532	ER_GRP_RPL_RESET_APPLIER_MODULE_LOGS_ERROR	HY000
11533	ER_GRP_RPL_APPLIER_THD_SETUP_ERROR	HY000
11534	ER_GRP_RPL_APPLIER_THD_START_ERROR	HY000
11535	ER_GRP_RPL_APPLIER_THD_STOP_ERROR	HY000
11536	ER_GRP_RPL_FETCH_TRANS_DATA_FAILED	HY000
11537	ER_GRP_RPL_REPLICA_IO_THD_PRIMARY_UNKNOWN	HY000
11538	ER_GRP_RPL_SALVE_IO_THD_ON_SECONDARY_MEMBER	HY000
11539	ER_GRP_RPL_REPLICA_SQL_THD_PRIMARY_UNKNOWN	HY000
11540	ER_GRP_RPL_REPLICA_SQL_THD_ON_SECONDARY_MEMBER	HY000
11541	ER_GRP_RPL_NEEDS_INNODB_TABLE	HY000
11542	ER_GRP_RPL_PRIMARY_KEY_NOT_DEFINED	HY000
11543	ER_GRP_RPL_FK_WITH_CASCADE_UNSUPPORTED	HY000
11544	ER_GRP_RPL_AUTO_INC_RESET	HY000
11545	ER_GRP_RPL_AUTO_INC_OFFSET_RESET	HY000
11546	ER_GRP_RPL_AUTO_INC_SET	HY000
11547	ER_GRP_RPL_AUTO_INC_OFFSET_SET	HY000
11548	ER_GRP_RPL_FETCH_TRANS_CONTEXT_FAILED	HY000
11549	ER_GRP_RPL_FETCH_FORMAT_DESC_LOG_EVENT_FAILED	HY000
11550	ER_GRP_RPL_FETCH_TRANS_CONTEXT_LOG_EVENT_FAILED	HY000
11551	ER_GRP_RPL_FETCH_SNAPSHOT_VERSION_FAILED	HY000
11552	ER_GRP_RPL_FETCH_GTID_LOG_EVENT_FAILED	HY000
11553	ER_GRP_RPL_UPDATE_SERV_CERTIFICATE_FAILED	HY000
11554	ER_GRP_RPL_ADD_GTID_INFO_WITH_LOCAL_GTID_FAILED	HY000
11555	ER_GRP_RPL_ADD_GTID_INFO_WITHOUT_LOCAL_GTID_FAILED	HY000
11556	ER_GRP_RPL_NOTIFY_CERTIFICATION_OUTCOME_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11557	ER_GRP_RPL_ADD_GTID_INFO_WITH_REMOTE_GTID_FAILED	HY000
11558	ER_GRP_RPL_ADD_GTID_INFO_WITHOUT_REMOTE_GTID_FAILED	HY000
11559	ER_GRP_RPL_FETCH_VIEW_CHANGE_LOG_EVENT_FAILED	HY000
11562	ER_GRP_RPL_FETCH_LOG_EVENT_FAILED	HY000
11563	ER_GRP_RPL_START_GRP_RPL_FAILED	HY000
11564	ER_GRP_RPL_CONN_INTERNAL_PLUGIN_FAIL	HY000
11565	ER_GRP_RPL_SUPER_READ_ON	HY000
11566	ER_GRP_RPL_SUPER_READ_OFF	HY000
11567	ER_GRP_RPL_KILLED_SESSION_ID	HY000
11568	ER_GRP_RPL_KILLED_FAILED_ID	HY000
11569	ER_GRP_RPL_INTERNAL_QUERY	HY000
11570	ER_GRP_RPL_COPY_FROM_EMPTY_STRING	HY000
11571	ER_GRP_RPL_QUERY_FAIL	HY000
11572	ER_GRP_RPL_CREATE_SESSION_UNABLE	HY000
11573	ER_GRP_RPL_MEMBER_NOT_FOUND	HY000
11574	ER_GRP_RPL_MAXIMUM_CONNECTION_RETRIES_REACHED	HY000
11575	ER_GRP_RPL_ALL_DONORS_LEFT_ABORT_RECOVERY	HY000
11576	ER_GRP_RPL_ESTABLISH_RECOVERY_WITH_DONOR	HY000
11577	ER_GRP_RPL_ESTABLISH_RECOVERY_WITH_ANOTHER_DONOR	HY000
11578	ER_GRP_RPL_NO_VALID_DONOR	HY000
11579	ER_GRP_RPL_CONFIG_RECOVERY	HY000
11580	ER_GRP_RPL_ESTABLISHING_CONN_GRP_REC_DONOR	HY000
11581	ER_GRP_RPL_CREATE_GRP_RPL_REC_CHANNEL	HY000
11582	ER_GRP_RPL_DONOR_SERVER_CONN	HY000
11583	ER_GRP_RPL_CHECK_STATUS_TABLE	HY000
11584	ER_GRP_RPL_STARTING_GRP_REC	HY000
11585	ER_GRP_RPL_DONOR_CONN_TERMINATION	HY000
11586	ER_GRP_RPL_STOPPING_GRP_REC	HY000
11587	ER_GRP_RPL_PURGE_REC	HY000
11588	ER_GRP_RPL_UNABLE_TO_KILL_CONN_REC_DONOR_APPLIER	HY000
11589	ER_GRP_RPL_UNABLE_TO_KILL_CONN_REC_DONOR_FAILOVER	HY000
11590	ER_GRP_RPL_FAILED_TO_NOTIFY_GRP_MEMBERSHIP_EVENT	HY000
11591	ER_GRP_RPL_FAILED_TO_BROADCAST_GRP_MEMBERSHIP_NOTIFICATION	HY000
11592	ER_GRP_RPL_FAILED_TO_BROADCAST_MEMBER_STATUS_NOTIFICATION	HY000
11593	ER_GRP_RPL_OOM_FAILED_TO_GENERATE_IDENTIFICATION_HASH	HY000
11594	ER_GRP_RPL_WRITE_IDENT_HASH_BASE64_ENCODING_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11595	ER_GRP_RPL_INVALID_BINLOG_FORMAT	HY000
11598	ER_GRP_RPL_UNSUPPORTED_TRANS_ISOLATION	HY000
11599	ER_GRP_RPL_CANNOT_EXECUTE_TRANS_WHILE_STOPPING	HY000
11600	ER_GRP_RPL_CANNOT_EXECUTE_TRANS_WHILE_RECOVERING	HY000
11601	ER_GRP_RPL_CANNOT_EXECUTE_TRANS_IN_ERROR_STATE	HY000
11602	ER_GRP_RPL_CANNOT_EXECUTE_TRANS_IN_OFFLINE_MODE	HY000
11603	ER_GRP_RPL_MULTIPLE_CACHE_TYPE_NOT_SUPPORTED_FOR_SESSION	HY000
11604	ER_GRP_RPL_FAILED_TO_REINIT_BINLOG_CACHE_FOR_READ	HY000
11605	ER_GRP_RPL_FAILED_TO_CREATE_TRANS_CONTEXT	HY000
11606	ER_GRP_RPL_FAILED_TO_EXTRACT_TRANS_WRITE_SET	HY000
11607	ER_GRP_RPL_FAILED_TO_GATHER_TRANS_WRITE_SET	HY000
11608	ER_GRP_RPL_TRANS_SIZE_EXCEEDS_LIMIT	HY000
11611	ER_GRP_RPL_WRITE_TO_TRANSACTION_MESSAGE_FAILED	HY000
11612	ER_GRP_RPL_FAILED_TO_REGISTER_TRANS_OUTCOME_NOTIFICTION	HY000
11613	ER_GRP_RPL_MSG_TOO_LONG_BROADCASTING_TRANS_FAILED	HY000
11614	ER_GRP_RPL_BROADCASTING_TRANS_TO_GRP_FAILED	HY000
11615	ER_GRP_RPL_ERROR_WHILE_WAITING_FOR_CONFLICT_DETECTION	HY000
11620	ER_GRP_RPL_FATAL_REC_PROCESS	HY000
11622	ER_GRP_RPL_UNABLE_TO_EVALUATE_APPLIER_STATUS	HY000
11623	ER_GRP_RPL_ONLY_ONE_SERVER_ALIVE	HY000
11624	ER_GRP_RPL_CERTIFICATION_REC_PROCESS	HY000
11625	ER_GRP_RPL_UNABLE_TO_ENSURE_EXECUTION_REC	HY000
11626	ER_GRP_RPL_WHILE_SENDING_MSG_REC	HY000
11628	ER_GRP_RPL_READ_UNABLE_FOR_READ_ONLY_SUPER_READ_ONLY	HY000
11629	ER_GRP_RPL_UNABLE_TO_RESET_SERVER_READ_MODE	HY000
11630	ER_GRP_RPL_UNABLE_TO_CERTIFY_PLUGIN_TRANS	HY000
11631	ER_GRP_RPL_UNBLOCK_CERTIFIED_TRANS	HY000
11633	ER_GRP_RPL_FAILED_TO_START_WITH_INVALID_SERVER_ID	HY000
11634	ER_GRP_RPL_FORCE_MEMBERS_MUST_BE_EMPTY	HY000
11635	ER_GRP_RPL_PLUGIN_STRUCT_INIT_NOT_POSSIBLE_ON_SERVER_START	HY000
11636	ER_GRP_RPL_FAILED_TO_ENABLE_SUPER_READ_ONLY_MODE	HY000
11637	ER_GRP_RPL_FAILED_TO_INIT_COMMUNICATION_ENGINE	HY000
11638	ER_GRP_RPL_FAILED_TO_START_ON_SECONDARY_WITH_ASYNC_CHANNELS	HY000
11639	ER_GRP_RPL_FAILED_TO_START_COMMUNICATION_ENGINE	HY000
11640	ER_GRP_RPL_TIMEOUT_ON_VIEW_AFTER_JOINING_GRP	HY000
11641	ER_GRP_RPL_FAILED_TO_CALL_GRP_COMMUNICATION_INTERFACE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11642	ER_GRP_RPL_MEMBER_SERVER_UUID_IS_INCOMPATIBLE_WITH_GRP	HY000
11643	ER_GRP_RPL_MEMBER_CONF_INFO	HY000
11644	ER_GRP_RPL_FAILED_TO_CONFIRM_IF_SERVER_LEFT_GRP	HY000
11645	ER_GRP_RPL_SERVER_IS_ALREADY_LEAVING	HY000
11646	ER_GRP_RPL_SERVER_ALREADY_LEFT	HY000
11647	ER_GRP_RPL_WAITING_FOR_VIEW_UPDATE	HY000
11648	ER_GRP_RPL_TIMEOUT_RECEIVING_VIEW_CHANGE_ON_SHUTDOWN	HY000
11649	ER_GRP_RPL_REQUESTING_NON_MEMBER_SERVER_TO_LEAVE	HY000
11650	ER_GRP_RPL_IS_STOPPING	HY000
11651	ER_GRP_RPL_IS_STOPPED	HY000
11652	ER_GRP_RPL_FAILED_TO_ENABLE_READ_ONLY_MODE_ON_SHUTDOWN	HY000
11653	ER_GRP_RPL_RECOVERY_MODULE_TERMINATION_TIMED_OUT_ON_SHUTDOWN	HY000
11654	ER_GRP_RPL_APPLIER_TERMINATION_TIMED_OUT_ON_SHUTDOWN	HY000
11655	ER_GRP_RPL_FAILED_TO_SHUTDOWN_REGISTRY_MODULE	HY000
11656	ER_GRP_RPL_FAILED_TO_INIT_HANDLER	HY000
11657	ER_GRP_RPL_FAILED_TO_REGISTER_SERVER_STATE_OBSERVER	HY000
11658	ER_GRP_RPL_FAILED_TO_REGISTER_TRANS_STATE_OBSERVER	HY000
11659	ER_GRP_RPL_FAILED_TO_REGISTER_BINLOG_STATE_OBSERVER	HY000
11660	ER_GRP_RPL_FAILED_TO_START_ON_BOOT	HY000
11661	ER_GRP_RPL_FAILED_TO_STOP_ON_PLUGIN_UNINSTALL	HY000
11662	ER_GRP_RPL_FAILED_TO_UNREGISTER_SERVER_STATE_OBSERVER	HY000
11663	ER_GRP_RPL_FAILED_TO_UNREGISTER_TRANS_STATE_OBSERVER	HY000
11664	ER_GRP_RPL_FAILED_TO_UNREGISTER_BINLOG_STATE_OBSERVER	HY000
11665	ER_GRP_RPL_ALL_OBSERVERS_UNREGISTERED	HY000
11666	ER_GRP_RPL_FAILED_TO_PARSE_THE_GRP_NAME	HY000
11667	ER_GRP_RPL_FAILED_TO_GENERATE_SIDNO_FOR_GRP	HY000
11668	ER_GRP_RPL_APPLIER_NOT_STARTED_DUE_TO_RUNNING_PREV_SHUTDOWN	HY000
11669	ER_GRP_RPL_FAILED_TO_INIT_APPLIER_MODULE	HY000
11670	ER_GRP_RPL_APPLIER_INITIALIZED	HY000
11671	ER_GRP_RPL_COMMUNICATION_SSL_CONF_INFO	HY000
11672	ER_GRP_RPL_ABORTS_AS_SSL_NOT_SUPPORTED_BY_MYSQLD	HY000
11673	ER_GRP_RPL_SSL_DISABLED	HY000
11674	ER_GRP_RPL_UNABLE_TO_INIT_COMMUNICATION_ENGINE	HY000
11675	ER_GRP_RPL_BINLOG_DISABLED	HY000
11676	ER_GRP_RPL_GTID_MODE_OFF	HY000
11677	ER_GRP_RPL_LOG_REPLICA_UPDATES_NOT_SET	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11679	ER_GRP_RPL_APPLIER_METADATA_REPO_MUST_BE_TABLE	HY000
11680	ER_GRP_RPL_CONNECTION_METADATA_REPO_MUST_BE_TABLE	HY000
11681	ER_GRP_RPL_INCORRECT_TYPE_SET_FOR_PARALLEL_APPLIER	HY000
11682	ER_GRP_RPL_REPLICA_PRESERVE_COMMIT_ORDER_NOT_SET	HY000
11683	ER_GRP_RPL_SINGLE_PRIM_MODE_NOT_ALLOWED_WITH_UPDATE_EVERYWHERE	HY000
11684	ER_GRP_RPL_MODULE_TERMINATE_ERROR	HY000
11685	ER_GRP_RPL_GRP_NAME_OPTION_MANDATORY	HY000
11686	ER_GRP_RPL_GRP_NAME_IS_TOO_LONG	HY000
11687	ER_GRP_RPL_GRP_NAME_IS_NOT_VALID_UUID	HY000
11688	ER_GRP_RPL_FLOW_CTRL_MIN_QUOTA_GREATER_THAN_MAX_QUOTA	HY000
11689	ER_GRP_RPL_FLOW_CTRL_MIN_RECOVERY_QUOTA_GREATER_THAN_MAX_QUOTA	HY000
11690	ER_GRP_RPL_FLOW_CTRL_MAX_QUOTA_SMALLER_THAN_MIN_QUOTAS	HY000
11691	ER_GRP_RPL_INVALID_SSL_RECOVERY_STRING	HY000
11694	ER_GRP_RPL_GRP_COMMUNICATION_INIT_WITH_CONF	HY000
11695	ER_GRP_RPL_UNKNOWN_GRP_RPL_APPLIER_PIPELINE_REQUESTED	HY000
11696	ER_GRP_RPL_FAILED_TO_BOOTSTRAP_EVENT_HANDLING_INFRASTRUCTURE	HY000
11697	ER_GRP_RPL_APPLIER_HANDLER_NOT_INITIALIZED	HY000
11698	ER_GRP_RPL_APPLIER_HANDLER_IS_IN_USE	HY000
11699	ER_GRP_RPL_APPLIER_HANDLER_ROLE_IS_IN_USE	HY000
11700	ER_GRP_RPL_FAILED_TO_INIT_APPLIER_HANDLER	HY000
11701	ER_GRP_RPL_SQL_SERVICE_FAILED_TO_INIT_SESSION_THREAD	HY000
11702	ER_GRP_RPL_SQL_SERVICE_COMM_SESSION_NOT_INITIALIZED	HY000
11703	ER_GRP_RPL_SQL_SERVICE_SERVER_SESSION_KILLED	HY000
11704	ER_GRP_RPL_SQL_SERVICE_FAILED_TO_RUN_SQL_QUERY	HY000
11705	ER_GRP_RPL_SQL_SERVICE_SERVER_INTERNAL_FAILURE	HY000
11706	ER_GRP_RPL_SQL_SERVICE_RETRIES_EXCEEDED_ON_SESSION_STATE	HY000
11707	ER_GRP_RPL_SQL_SERVICE_FAILED_TO_FETCH_SECURITY_CTX	HY000
11708	ER_GRP_RPL_SQL_SERVICE_SERVER_ACCESS_DENIED_FOR_USER	HY000
11709	ER_GRP_RPL_SQL_SERVICE_MAX_CONN_ERROR_FROM_SERVER	HY000
11710	ER_GRP_RPL_SQL_SERVICE_SERVER_ERROR_ON_CONN	HY000
11711	ER_GRP_RPL_UNREACHABLE_MAJORITY_TIMEOUT_FOR_MEMBER	HY000
11712	ER_GRP_RPL_SERVER_SET_TO_READ_ONLY_DUE_TO_ERRORS	HY000
11713	ER_GRP_RPL_GMS_LISTENER_FAILED_TO_LOG_NOTIFICATION	HY000
11714	ER_GRP_RPL_GRP_COMMUNICATION_ENG_INIT_FAILED	HY000
11715	ER_GRP_RPL_SET_GRP_COMMUNICATION_ENG_LOGGER_FAILED	HY000
11716	ER_GRP_RPL_DEBUG_OPTIONS	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11717	ER_GRP_RPL_INVALID_DEBUG_OPTIONS	HY000
11718	ER_GRP_RPL_EXIT_GRP_GCS_ERROR	HY000
11719	ER_GRP_RPL_GRP_MEMBER_OFFLINE	HY000
11720	ER_GRP_RPL_GCS_INTERFACE_ERROR	HY000
11721	ER_GRP_RPL_FORCE_MEMBER_VALUE_SET_ERROR	HY000
11722	ER_GRP_RPL_FORCE_MEMBER_VALUE_SET	HY000
11723	ER_GRP_RPL_FORCE_MEMBER_VALUE_TIME_OUT	HY000
11724	ER_GRP_RPL_BROADCAST_COMMIT_MSSG_TOO_BIG	HY000
11725	ER_GRP_RPL_SEND_STATS_ERROR	HY000
11726	ER_GRP_RPL_MEMBER_STATS_INFO	HY000
11727	ER_GRP_RPL_FLOW_CONTROL_STATS	HY000
11728	ER_GRP_RPL_UNABLE_TO_CONVERT_PACKET_TO_EVENT	HY000
11729	ER_GRP_RPL_PIPELINE_CREATE_FAILED	HY000
11730	ER_GRP_RPL_PIPELINE_REINIT_FAILED_WRITE	HY000
11731	ER_GRP_RPL_UNABLE_TO_CONVERT_EVENT_TO_PACKET	HY000
11732	ER_GRP_RPL_PIPELINE_FLUSH_FAIL	HY000
11733	ER_GRP_RPL_PIPELINE_REINIT_FAILED_READ	HY000
11735	ER_GRP_RPL_GCS_GR_ERROR_MSG	HY000
11736	ER_GRP_RPL_REPLICA_IO_THREAD_UNBLOCKED	HY000
11737	ER_GRP_RPL_REPLICA_IO_THREAD_ERROR_OUT	HY000
11738	ER_GRP_RPL_REPLICA_APPLIER_THREAD_UNBLOCKED	HY000
11739	ER_GRP_RPL_REPLICA_APPLIER_THREAD_ERROR_OUT	HY000
11740	ER_LDAP_AUTH_FAILED_TO_CREATE_OR_GET_CONNECTION	HY000
11741	ER_LDAP_AUTH_DEINIT_FAILED	HY000
11742	ER_LDAP_AUTH_SKIPPING_USER_GROUP_SEARCH	HY000
11743	ER_LDAP_AUTH_POOL_DISABLE_MAX_SIZE_ZERO	HY000
11744	ER_LDAP_AUTH_FAILED_TO_CREATE_LDAP_OBJECT_CREATOR	HY000
11745	ER_LDAP_AUTH_FAILED_TO_CREATE_LDAP_OBJECT	HY000
11746	ER_LDAP_AUTH_TLS_CONF	HY000
11747	ER_LDAP_AUTH_TLS_CONNECTION	HY000
11748	ER_LDAP_AUTH_CONN_POOL_NOT_CREATED	HY000
11749	ER_LDAP_AUTH_CONN_POOL_INITIALIZING	HY000
11750	ER_LDAP_AUTH_CONN_POOL_DEINITIALIZING	HY000
11751	ER_LDAP_AUTH_ZERO_MAX_POOL_SIZE_UNCHANGED	HY000
11752	ER_LDAP_AUTH_POOL_REINITIALIZING	HY000
11753	ER_LDAP_AUTH_FAILED_TO_WRITE_PACKET	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11754	ER_LDAP_AUTH_SETTING_USERNAME	HY000
11755	ER_LDAP_AUTH_USER_AUTH_DATA	HY000
11757	ER_LDAP_AUTH_USER_GROUP_SEARCH_INFO	HY000
11758	ER_LDAP_AUTH_GRP_SEARCH_SPECIAL_HDL	HY000
11759	ER_LDAP_AUTH_GRP_IS_FULL_DN	HY000
11760	ER_LDAP_AUTH_USER_NOT_FOUND_IN_ANY_GRP	HY000
11761	ER_LDAP_AUTH_USER_FOUND_IN_MANY_GRPS	HY000
11762	ER_LDAP_AUTH_USER_HAS_MULTIPLE_GRP_NAMES	HY000
11763	ER_LDAP_AUTH_SEARCHED_USER_GRP_NAME	HY000
11764	ER_LDAP_AUTH_OBJECT_CREATE_TIMESTAMP	HY000
11765	ER_LDAP_AUTH_CERTIFICATE_NAME	HY000
11766	ER_LDAP_AUTH_FAILED_TO_POOL_DEINIT	HY000
11767	ER_LDAP_AUTH_FAILED_TO_INITIALIZE_POOL_IN_RECONSTRUCTING	HY000
11768	ER_LDAP_AUTH_FAILED_TO_INITIALIZE_POOL_IN_INIT_STATE	HY000
11769	ER_LDAP_AUTH_FAILED_TO_INITIALIZE_POOL_IN_DEINIT_STATE	HY000
11770	ER_LDAP_AUTH_FAILED_TO_DEINITIALIZE_POOL_IN_RECONSTRUCT_STATE	HY000
11771	ER_LDAP_AUTH_FAILED_TO_DEINITIALIZE_NOT_READY_POOL	HY000
11772	ER_LDAP_AUTH_FAILED_TO_GET_CONNECTION_AS_PLUGIN_NOT_READY	HY000
11773	ER_LDAP_AUTH_CONNECTION_POOL_INIT_FAILED	HY000
11774	ER_LDAP_AUTH_MAX_ALLOWED_CONNECTION_LIMIT_HIT	HY000
11775	ER_LDAP_AUTH_MAX_POOL_SIZE_SET_FAILED	HY000
11776	ER_LDAP_AUTH_PLUGIN_FAILED_TO_READ_PACKET	HY000
11777	ER_LDAP_AUTH_CREATING_LDAP_CONNECTION	HY000
11778	ER_LDAP_AUTH_GETTING_CONNECTION_FROM_POOL	HY000
11779	ER_LDAP_AUTH_RETURNING_CONNECTION_TO_POOL	HY000
11780	ER_LDAP_AUTH_SEARCH_USER_GROUP_ATTR_NOT_FOUND	HY000
11781	ER_LDAP_AUTH_LDAP_INFO_NULL	HY000
11782	ER_LDAP_AUTH_FREEING_CONNECTION	HY000
11783	ER_LDAP_AUTH_CONNECTION_PUSHED_TO_POOL	HY000
11784	ER_LDAP_AUTH_CONNECTION_CREATOR_ENTER	HY000
11785	ER_LDAP_AUTH_STARTING_TLS	HY000
11786	ER_LDAP_AUTH_CONNECTION_GET_LDAP_INFO_NULL	HY000
11787	ER_LDAP_AUTH_DELETING_CONNECTION_KEY	HY000
11788	ER_LDAP_AUTH_POOLED_CONNECTION_KEY	HY000
11789	ER_LDAP_AUTH_CREATE_CONNECTION_KEY	HY000
11790	ER_LDAP_AUTH_COMMUNICATION_HOST_INFO	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11791	ER_LDAP_AUTH_METHOD_TO_CLIENT	HY000
11792	ER_LDAP_AUTH_SASL_REQUEST_FROM_CLIENT	HY000
11793	ER_LDAP_AUTH_SASL_PROCESS_SASL	HY000
11794	ER_LDAP_AUTH_SASL_BIND_SUCCESS_INFO	HY000
11795	ER_LDAP_AUTH_STARTED_FOR_USER	HY000
11796	ER_LDAP_AUTH_DISTINGUISHED_NAME	HY000
11797	ER_LDAP_AUTH_INIT_FAILED	HY000
11798	ER_LDAP_AUTH_OR_GROUP_RETRIEVAL_FAILED	HY000
11799	ER_LDAP_AUTH_USER_GROUP_SEARCH_FAILED	HY000
11800	ER_LDAP_AUTH_USER_BIND_FAILED	HY000
11801	ER_LDAP_AUTH_POOL_GET_FAILED_TO_CREATE_CONNECTION	HY000
11802	ER_LDAP_AUTH_FAILED_TO_CREATE_LDAP_CONNECTION	HY000
11803	ER_LDAP_AUTH_FAILED_TO_ESTABLISH_TLS_CONNECTION	HY000
11804	ER_LDAP_AUTH_FAILED_TO_SEARCH_DN	HY000
11805	ER_LDAP_AUTH_CONNECTION_POOL_REINIT_ENTER	HY000
11806	ER_SYSTEMD_NOTIFY_PATH_TOO_LONG	HY000
11807	ER_SYSTEMD_NOTIFY_CONNECT_FAILED	HY000
11808	ER_SYSTEMD_NOTIFY_WRITE_FAILED	HY000
11809	ER_FOUND_MISSING_GTIDS	HY000
11810	ER_PID_FILE_PRIV_DIRECTORY_INSECURE	HY000
11811	ER_CANT_CHECK_PID_PATH	HY000
11812	ER_VALIDATE_PWD_STATUS_VAR_REGISTRATION_FAILED	HY000
11813	ER_VALIDATE_PWD_STATUS_VAR_UNREGISTRATION_FAILED	HY000
11814	ER_VALIDATE_PWD_DICT_FILE_OPEN_FAILED	HY000
11815	ER_VALIDATE_PWD_COULD_BE_NULL	HY000
11816	ER_VALIDATE_PWD_STRING_CONV_TO_LOWERCASE_FAILED	HY000
11817	ER_VALIDATE_PWD_STRING_CONV_TO_BUFFER_FAILED	HY000
11818	ER_VALIDATE_PWD_STRING_HANDLER_MEM_ALLOCATION_FAILED	HY000
11819	ER_VALIDATE_PWD_STRONG_POLICY_DICT_FILE_UNSPECIFIED	HY000
11820	ER_VALIDATE_PWD_CONVERT_TO_BUFFER_FAILED	HY000
11821	ER_VALIDATE_PWD_VARIABLE_REGISTRATION_FAILED	HY000
11822	ER_VALIDATE_PWD_VARIABLE_UNREGISTRATION_FAILED	HY000
11823	ER_KEYRING_MIGRATION_EXTRA_OPTIONS	HY000
11825	ER_IB_MSG_0	HY000
11826	ER_IB_MSG_1	HY000
11827	ER_IB_MSG_2	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11828	ER_IB_MSG_3	HY000
11829	ER_IB_MSG_4	HY000
11830	ER_IB_MSG_5	HY000
11831	ER_IB_MSG_6	HY000
11832	ER_IB_MSG_7	HY000
11833	ER_IB_MSG_8	HY000
11834	ER_IB_MSG_9	HY000
11835	ER_IB_MSG_10	HY000
11836	ER_IB_MSG_11	HY000
11837	ER_IB_MSG_12	HY000
11838	ER_IB_MSG_13	HY000
11839	ER_IB_MSG_14	HY000
11840	ER_IB_MSG_15	HY000
11841	ER_IB_MSG_16	HY000
11842	ER_IB_MSG_17	HY000
11843	ER_IB_MSG_18	HY000
11844	ER_IB_MSG_19	HY000
11845	ER_IB_MSG_20	HY000
11846	ER_IB_MSG_21	HY000
11847	ER_IB_MSG_22	HY000
11848	ER_IB_MSG_23	HY000
11849	ER_IB_MSG_24	HY000
11850	ER_IB_MSG_25	HY000
11851	ER_IB_MSG_26	HY000
11852	ER_IB_MSG_27	HY000
11853	ER_IB_MSG_28	HY000
11854	ER_IB_MSG_29	HY000
11855	ER_IB_MSG_30	HY000
11857	ER_IB_MSG_32	HY000
11858	ER_IB_MSG_33	HY000
11859	ER_IB_MSG_34	HY000
11860	ER_IB_MSG_35	HY000
11861	ER_IB_MSG_36	HY000
11862	ER_IB_MSG_37	HY000
11863	ER_IB_MSG_38	HY000
11864	ER_IB_MSG_39	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11865	ER_IB_MSG_40	HY000
11866	ER_IB_MSG_41	HY000
11867	ER_IB_MSG_42	HY000
11868	ER_IB_MSG_43	HY000
11869	ER_IB_MSG_44	HY000
11870	ER_IB_MSG_45	HY000
11871	ER_IB_MSG_46	HY000
11872	ER_IB_MSG_47	HY000
11873	ER_IB_MSG_48	HY000
11874	ER_IB_MSG_49	HY000
11875	ER_IB_MSG_50	HY000
11876	ER_IB_MSG_51	HY000
11877	ER_IB_MSG_52	HY000
11878	ER_IB_MSG_53	HY000
11879	ER_IB_MSG_54	HY000
11880	ER_IB_MSG_55	HY000
11881	ER_IB_MSG_56	HY000
11882	ER_IB_MSG_57	HY000
11883	ER_IB_MSG_58	HY000
11884	ER_IB_MSG_59	HY000
11885	ER_IB_MSG_60	HY000
11886	ER_IB_MSG_61	HY000
11887	ER_IB_MSG_62	HY000
11888	ER_IB_MSG_63	HY000
11889	ER_IB_MSG_64	HY000
11890	ER_IB_MSG_65	HY000
11891	ER_IB_MSG_66	HY000
11892	ER_IB_MSG_67	HY000
11893	ER_IB_MSG_68	HY000
11894	ER_IB_MSG_69	HY000
11895	ER_IB_MSG_70	HY000
11896	ER_IB_MSG_71	HY000
11897	ER_IB_MSG_72	HY000
11898	ER_IB_MSG_73	HY000
11899	ER_IB_MSG_74	HY000
11900	ER_IB_MSG_75	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11901	ER_IB_MSG_76	HY000
11902	ER_IB_MSG_77	HY000
11903	ER_IB_MSG_78	HY000
11904	ER_IB_MSG_79	HY000
11905	ER_IB_MSG_80	HY000
11906	ER_IB_MSG_81	HY000
11907	ER_IB_MSG_82	HY000
11908	ER_IB_MSG_83	HY000
11909	ER_IB_MSG_84	HY000
11910	ER_IB_MSG_85	HY000
11911	ER_IB_MSG_86	HY000
11920	ER_IB_MSG_95	HY000
11921	ER_IB_MSG_96	HY000
11922	ER_IB_MSG_97	HY000
11923	ER_IB_MSG_98	HY000
11924	ER_IB_MSG_99	HY000
11925	ER_IB_MSG_100	HY000
11926	ER_IB_MSG_101	HY000
11927	ER_IB_MSG_102	HY000
11928	ER_IB_MSG_103	HY000
11929	ER_IB_MSG_104	HY000
11930	ER_IB_MSG_105	HY000
11931	ER_IB_MSG_106	HY000
11932	ER_IB_MSG_107	HY000
11933	ER_IB_MSG_108	HY000
11934	ER_IB_MSG_109	HY000
11935	ER_IB_MSG_110	HY000
11936	ER_IB_MSG_111	HY000
11937	ER_IB_MSG_112	HY000
11944	ER_IB_MSG_119	HY000
11945	ER_IB_MSG_120	HY000
11946	ER_IB_MSG_121	HY000
11947	ER_IB_MSG_122	HY000
11948	ER_IB_MSG_123	HY000
11949	ER_IB_MSG_124	HY000
11950	ER_IB_MSG_125	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11951	ER_IB_MSG_126	HY000
11952	ER_IB_MSG_127	HY000
11953	ER_IB_MSG_128	HY000
11954	ER_IB_MSG_129	HY000
11955	ER_IB_MSG_130	HY000
11956	ER_IB_MSG_131	HY000
11957	ER_IB_MSG_132	HY000
11958	ER_IB_MSG_133	HY000
11959	ER_IB_MSG_134	HY000
11960	ER_IB_MSG_135	HY000
11961	ER_IB_MSG_136	HY000
11962	ER_IB_MSG_137	HY000
11963	ER_IB_MSG_138	HY000
11964	ER_IB_MSG_139	HY000
11965	ER_IB_MSG_140	HY000
11966	ER_IB_MSG_141	HY000
11967	ER_IB_MSG_142	HY000
11968	ER_IB_MSG_143	HY000
11969	ER_IB_MSG_144	HY000
11970	ER_IB_MSG_145	HY000
11971	ER_IB_MSG_146	HY000
11972	ER_IB_MSG_147	HY000
11973	ER_IB_MSG_148	HY000
11974	ER_IB_CLONE_INTERNAL	HY000
11975	ER_IB_CLONE_TIMEOUT	HY000
11976	ER_IB_CLONE_STATUS_FILE	HY000
11977	ER_IB_CLONE_SQL	HY000
11978	ER_IB_CLONE_VALIDATE	HY000
11979	ER_IB_CLONE_PUNCH_HOLE	HY000
11980	ER_IB_CLONE_GTID_PERSIST	HY000
11981	ER_IB_MSG_156	HY000
11982	ER_IB_MSG_157	HY000
11983	ER_IB_MSG_158	HY000
11984	ER_IB_MSG_159	HY000
11985	ER_IB_MSG_160	HY000
11986	ER_IB_MSG_161	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
11987	ER_IB_MSG_162	HY000
11988	ER_IB_MSG_163	HY000
11989	ER_IB_MSG_164	HY000
11990	ER_IB_MSG_165	HY000
11991	ER_IB_MSG_166	HY000
11992	ER_IB_MSG_167	HY000
11993	ER_IB_MSG_168	HY000
11994	ER_IB_MSG_169	HY000
11995	ER_IB_MSG_170	HY000
11996	ER_IB_MSG_171	HY000
11997	ER_IB_MSG_172	HY000
11998	ER_IB_MSG_173	HY000
11999	ER_IB_MSG_174	HY000
12000	ER_IB_MSG_175	HY000
12001	ER_IB_MSG_176	HY000
12002	ER_IB_MSG_177	HY000
12003	ER_IB_MSG_178	HY000
12004	ER_IB_MSG_179	HY000
12005	ER_IB_MSG_180	HY000
12006	ER_IB_LONG_AHI_DISABLE_WAIT	HY000
12007	ER_IB_MSG_182	HY000
12008	ER_IB_MSG_183	HY000
12009	ER_IB_MSG_184	HY000
12012	ER_IB_MSG_187	HY000
12013	ER_IB_MSG_188	HY000
12014	ER_IB_MSG_189	HY000
12015	ER_IB_MSG_190	HY000
12016	ER_IB_MSG_191	HY000
12017	ER_IB_MSG_192	HY000
12018	ER_IB_MSG_193	HY000
12019	ER_IB_MSG_194	HY000
12020	ER_IB_MSG_195	HY000
12021	ER_IB_MSG_196	HY000
12022	ER_IB_MSG_197	HY000
12023	ER_IB_MSG_198	HY000
12024	ER_IB_MSG_199	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12025	ER_IB_MSG_200	HY000
12026	ER_IB_MSG_201	HY000
12027	ER_IB_MSG_202	HY000
12028	ER_IB_MSG_203	HY000
12029	ER_IB_MSG_204	HY000
12030	ER_IB_MSG_205	HY000
12031	ER_IB_MSG_206	HY000
12032	ER_IB_MSG_207	HY000
12033	ER_IB_MSG_208	HY000
12034	ER_IB_MSG_209	HY000
12035	ER_IB_MSG_210	HY000
12036	ER_IB_MSG_211	HY000
12037	ER_IB_MSG_212	HY000
12038	ER_IB_MSG_213	HY000
12039	ER_IB_MSG_214	HY000
12040	ER_IB_MSG_215	HY000
12041	ER_IB_MSG_216	HY000
12042	ER_IB_MSG_217	HY000
12043	ER_IB_MSG_218	HY000
12044	ER_IB_MSG_219	HY000
12045	ER_IB_MSG_220	HY000
12046	ER_IB_MSG_221	HY000
12047	ER_IB_MSG_222	HY000
12048	ER_IB_MSG_223	HY000
12049	ER_IB_MSG_224	HY000
12050	ER_IB_MSG_225	HY000
12051	ER_IB_MSG_226	HY000
12054	ER_IB_MSG_229	HY000
12055	ER_IB_MSG_230	HY000
12056	ER_IB_MSG_231	HY000
12057	ER_IB_MSG_232	HY000
12058	ER_IB_MSG_233	HY000
12059	ER_IB_MSG_234	HY000
12060	ER_IB_MSG_235	HY000
12061	ER_IB_MSG_236	HY000
12062	ER_IB_MSG_237	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12063	ER_IB_MSG_238	HY000
12064	ER_IB_MSG_239	HY000
12065	ER_IB_MSG_240	HY000
12066	ER_IB_MSG_241	HY000
12067	ER_IB_MSG_242	HY000
12068	ER_IB_MSG_243	HY000
12069	ER_IB_MSG_244	HY000
12070	ER_IB_MSG_245	HY000
12071	ER_IB_MSG_246	HY000
12072	ER_IB_MSG_247	HY000
12073	ER_IB_MSG_248	HY000
12074	ER_IB_MSG_249	HY000
12075	ER_IB_MSG_250	HY000
12076	ER_IB_MSG_251	HY000
12077	ER_IB_MSG_252	HY000
12078	ER_IB_MSG_253	HY000
12079	ER_IB_MSG_254	HY000
12080	ER_IB_MSG_255	HY000
12081	ER_IB_MSG_256	HY000
12082	ER_IB_MSG_257	HY000
12083	ER_IB_MSG_258	HY000
12084	ER_IB_MSG_259	HY000
12085	ER_IB_MSG_260	HY000
12086	ER_IB_MSG_261	HY000
12087	ER_IB_MSG_262	HY000
12088	ER_IB_MSG_263	HY000
12089	ER_IB_MSG_264	HY000
12090	ER_IB_MSG_265	HY000
12091	ER_IB_MSG_266	HY000
12092	ER_IB_MSG_267	HY000
12093	ER_IB_MSG_268	HY000
12094	ER_IB_MSG_269	HY000
12095	ER_IB_MSG_270	HY000
12096	ER_IB_MSG_271	HY000
12097	ER_IB_MSG_272	HY000
12098	ER_IB_MSG_273	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12103	ER_IB_MSG_278	HY000
12105	ER_IB_MSG_280	HY000
12106	ER_IB_MSG_281	HY000
12107	ER_IB_MSG_282	HY000
12108	ER_IB_MSG_283	HY000
12109	ER_IB_MSG_284	HY000
12110	ER_IB_MSG_285	HY000
12111	ER_IB_WARN_ACCESSING_NONEXISTINC_SPACE	HY000
12112	ER_IB_MSG_287	HY000
12113	ER_IB_MSG_288	HY000
12114	ER_IB_MSG_289	HY000
12116	ER_IB_MSG_291	HY000
12117	ER_IB_MSG_292	HY000
12118	ER_IB_MSG_293	HY000
12119	ER_IB_MSG_294	HY000
12120	ER_IB_MSG_295	HY000
12121	ER_IB_MSG_296	HY000
12122	ER_IB_MSG_297	HY000
12123	ER_IB_MSG_298	HY000
12124	ER_IB_MSG_299	HY000
12125	ER_IB_MSG_300	HY000
12126	ER_IB_MSG_301	HY000
12127	ER_IB_MSG_UNEXPECTED_FILE_EXISTS	HY000
12128	ER_IB_MSG_303	HY000
12129	ER_IB_MSG_304	HY000
12130	ER_IB_MSG_305	HY000
12131	ER_IB_MSG_306	HY000
12132	ER_IB_MSG_307	HY000
12133	ER_IB_MSG_308	HY000
12134	ER_IB_MSG_309	HY000
12135	ER_IB_MSG_310	HY000
12136	ER_IB_MSG_311	HY000
12137	ER_IB_MSG_312	HY000
12138	ER_IB_MSG_313	HY000
12139	ER_IB_MSG_314	HY000
12140	ER_IB_MSG_315	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12141	ER_IB_MSG_316	HY000
12142	ER_IB_MSG_317	HY000
12143	ER_IB_MSG_318	HY000
12144	ER_IB_MSG_319	HY000
12145	ER_IB_MSG_320	HY000
12146	ER_IB_MSG_321	HY000
12147	ER_IB_MSG_322	HY000
12148	ER_IB_MSG_323	HY000
12149	ER_IB_MSG_324	HY000
12150	ER_IB_MSG_325	HY000
12151	ER_IB_MSG_326	HY000
12153	ER_IB_MSG_328	HY000
12154	ER_IB_MSG_329	HY000
12155	ER_IB_MSG_330	HY000
12156	ER_IB_MSG_331	HY000
12157	ER_IB_MSG_332	HY000
12158	ER_IB_MSG_333	HY000
12159	ER_IB_MSG_334	HY000
12160	ER_IB_MSG_335	HY000
12161	ER_IB_MSG_336	HY000
12162	ER_IB_MSG_337	HY000
12163	ER_IB_MSG_338	HY000
12164	ER_IB_MSG_339	HY000
12165	ER_IB_MSG_340	HY000
12166	ER_IB_MSG_341	HY000
12167	ER_IB_MSG_342	HY000
12168	ER_IB_MSG_343	HY000
12169	ER_IB_MSG_344	HY000
12170	ER_IB_MSG_345	HY000
12171	ER_IB_MSG_346	HY000
12172	ER_IB_MSG_347	HY000
12173	ER_IB_MSG_348	HY000
12174	ER_IB_MSG_349	HY000
12175	ER_IB_MSG_350	HY000
12177	ER_IB_MSG_UNPROTECTED_LOCATION_ALLOWED	HY000
12179	ER_IB_MSG_354	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12180	ER_IB_MSG_355	HY000
12181	ER_IB_MSG_356	HY000
12182	ER_IB_MSG_357	HY000
12183	ER_IB_MSG_358	HY000
12184	ER_IB_MSG_359	HY000
12185	ER_IB_MSG_360	HY000
12186	ER_IB_MSG_361	HY000
12187	ER_IB_MSG_362	HY000
12189	ER_IB_MSG_364	HY000
12190	ER_IB_MSG_365	HY000
12191	ER_IB_MSG_IGNORE_SCAN_PATH	HY000
12192	ER_IB_MSG_367	HY000
12193	ER_IB_MSG_368	HY000
12194	ER_IB_MSG_369	HY000
12195	ER_IB_MSG_370	HY000
12196	ER_IB_MSG_371	HY000
12197	ER_IB_MSG_372	HY000
12198	ER_IB_MSG_373	HY000
12199	ER_IB_MSG_374	HY000
12200	ER_IB_MSG_375	HY000
12201	ER_IB_MSG_376	HY000
12202	ER_IB_MSG_377	HY000
12203	ER_IB_MSG_378	HY000
12204	ER_IB_MSG_379	HY000
12205	ER_IB_MSG_380	HY000
12206	ER_IB_MSG_381	HY000
12207	ER_IB_MSG_382	HY000
12208	ER_IB_MSG_383	HY000
12209	ER_IB_MSG_384	HY000
12210	ER_IB_MSG_385	HY000
12211	ER_IB_MSG_386	HY000
12212	ER_IB_MSG_387	HY000
12213	ER_IB_MSG_GENERAL_TABLESPACE_UNDER_DATADIR	HY000
12214	ER_IB_MSG_IMPLICIT_TABLESPACE_IN_DATADIR	HY000
12215	ER_IB_MSG_390	HY000
12216	ER_IB_MSG_391	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12217	ER_IB_MSG_392	HY000
12218	ER_IB_MSG_393	HY000
12219	ER_IB_MSG_394	HY000
12220	ER_IB_MSG_395	HY000
12221	ER_IB_MSG_396	HY000
12222	ER_IB_MSG_397	HY000
12223	ER_IB_MSG_398	HY000
12224	ER_IB_MSG_399	HY000
12226	ER_IB_MSG_401	HY000
12227	ER_IB_MSG_402	HY000
12228	ER_IB_MSG_403	HY000
12229	ER_IB_MSG_404	HY000
12230	ER_IB_MSG_405	HY000
12231	ER_IB_MSG_406	HY000
12232	ER_IB_MSG_407	HY000
12233	ER_IB_MSG_408	HY000
12234	ER_IB_MSG_409	HY000
12235	ER_IB_MSG_410	HY000
12236	ER_IB_MSG_411	HY000
12237	ER_IB_MSG_412	HY000
12238	ER_IB_MSG_413	HY000
12239	ER_IB_MSG_414	HY000
12240	ER_IB_MSG_415	HY000
12241	ER_IB_MSG_416	HY000
12242	ER_IB_MSG_417	HY000
12243	ER_IB_MSG_418	HY000
12244	ER_IB_MSG_419	HY000
12245	ER_IB_MSG_420	HY000
12246	ER_IB_MSG_421	HY000
12247	ER_IB_MSG_422	HY000
12248	ER_IB_MSG_423	HY000
12249	ER_IB_MSG_424	HY000
12250	ER_IB_MSG_425	HY000
12251	ER_IB_MSG_426	HY000
12252	ER_IB_MSG_427	HY000
12253	ER_IB_MSG_428	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12254	ER_IB_MSG_429	HY000
12255	ER_IB_MSG_430	HY000
12256	ER_IB_MSG_431	HY000
12257	ER_IB_MSG_432	HY000
12258	ER_IB_MSG_433	HY000
12259	ER_IB_MSG_434	HY000
12260	ER_IB_MSG_435	HY000
12261	ER_IB_MSG_436	HY000
12262	ER_IB_MSG_437	HY000
12263	ER_IB_MSG_438	HY000
12264	ER_IB_MSG_439	HY000
12265	ER_IB_MSG_440	HY000
12266	ER_IB_MSG_441	HY000
12267	ER_IB_MSG_442	HY000
12268	ER_IB_MSG_443	HY000
12270	ER_IB_MSG_445	HY000
12271	ER_IB_MSG_446	HY000
12272	ER_IB_MSG_447	HY000
12273	ER_IB_MSG_448	HY000
12274	ER_IB_MSG_449	HY000
12275	ER_IB_MSG_450	HY000
12276	ER_IB_MSG_451	HY000
12277	ER_IB_MSG_452	HY000
12278	ER_IB_MSG_453	HY000
12279	ER_IB_MSG_454	HY000
12280	ER_IB_MSG_455	HY000
12281	ER_IB_MSG_456	HY000
12282	ER_IB_MSG_457	HY000
12283	ER_IB_MSG_458	HY000
12284	ER_IB_MSG_459	HY000
12285	ER_IB_MSG_460	HY000
12286	ER_IB_MSG_461	HY000
12287	ER_IB_MSG_462	HY000
12288	ER_IB_MSG_463	HY000
12289	ER_IB_MSG_464	HY000
12290	ER_IB_MSG_465	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12291	ER_IB_MSG_466	HY000
12292	ER_IB_MSG_467	HY000
12293	ER_IB_MSG_468	HY000
12294	ER_IB_MSG_469	HY000
12295	ER_IB_MSG_470	HY000
12296	ER_IB_MSG_471	HY000
12297	ER_IB_MSG_472	HY000
12298	ER_IB_MSG_473	HY000
12299	ER_IB_MSG_474	HY000
12300	ER_IB_MSG_475	HY000
12301	ER_IB_MSG_476	HY000
12302	ER_IB_MSG_477	HY000
12303	ER_IB_MSG_478	HY000
12304	ER_IB_MSG_479	HY000
12305	ER_IB_MSG_480	HY000
12306	ER_IB_MSG_481	HY000
12307	ER_IB_MSG_482	HY000
12308	ER_IB_MSG_483	HY000
12309	ER_IB_MSG_484	HY000
12310	ER_IB_MSG_485	HY000
12311	ER_IB_MSG_486	HY000
12312	ER_IB_MSG_487	HY000
12313	ER_IB_MSG_488	HY000
12314	ER_IB_MSG_489	HY000
12315	ER_IB_MSG_490	HY000
12316	ER_IB_MSG_491	HY000
12317	ER_IB_MSG_492	HY000
12318	ER_IB_MSG_493	HY000
12319	ER_IB_MSG_494	HY000
12320	ER_IB_MSG_495	HY000
12321	ER_IB_MSG_496	HY000
12322	ER_IB_MSG_497	HY000
12323	ER_IB_MSG_498	HY000
12324	ER_IB_MSG_499	HY000
12325	ER_IB_MSG_500	HY000
12326	ER_IB_MSG_501	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12327	ER_IB_MSG_502	HY000
12328	ER_IB_MSG_503	HY000
12329	ER_IB_MSG_504	HY000
12330	ER_IB_MSG_505	HY000
12331	ER_IB_MSG_506	HY000
12332	ER_IB_MSG_507	HY000
12333	ER_IB_MSG_508	HY000
12334	ER_IB_MSG_509	HY000
12335	ER_IB_MSG_510	HY000
12336	ER_IB_MSG_511	HY000
12337	ER_IB_MSG_512	HY000
12338	ER_IB_MSG_513	HY000
12339	ER_IB_MSG_514	HY000
12340	ER_IB_MSG_515	HY000
12341	ER_IB_MSG_516	HY000
12342	ER_IB_MSG_517	HY000
12343	ER_IB_MSG_518	HY000
12344	ER_IB_MSG_519	HY000
12345	ER_IB_MSG_520	HY000
12346	ER_IB_MSG_521	HY000
12347	ER_IB_MSG_522	HY000
12348	ER_IB_MSG_523	HY000
12349	ER_IB_MSG_524	HY000
12350	ER_IB_MSG_525	HY000
12351	ER_IB_MSG_526	HY000
12352	ER_IB_MSG_527	HY000
12355	ER_IB_MSG_530	HY000
12356	ER_IB_MSG_531	HY000
12357	ER_IB_MSG_532	HY000
12358	ER_IB_MSG_533	HY000
12359	ER_IB_MSG_534	HY000
12362	ER_IB_MSG_537	HY000
12363	ER_IB_MSG_538	HY000
12364	ER_IB_MSG_539	HY000
12365	ER_IB_MSG_540	HY000
12366	ER_IB_MSG_541	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12368	ER_IB_MSG_543	HY000
12369	ER_IB_MSG_544	HY000
12370	ER_IB_MSG_545	HY000
12371	ER_IB_MSG_546	HY000
12372	ER_IB_MSG_547	HY000
12373	ER_IB_MSG_548	HY000
12374	ER_IB_MSG_549	HY000
12375	ER_IB_MSG_550	HY000
12376	ER_IB_MSG_551	HY000
12377	ER_IB_MSG_552	HY000
12378	ER_IB_MSG_553	HY000
12379	ER_IB_MSG_554	HY000
12380	ER_IB_MSG_555	HY000
12381	ER_IB_MSG_556	HY000
12382	ER_IB_MSG_557	HY000
12383	ER_IB_MSG_558	HY000
12384	ER_IB_MSG_559	HY000
12385	ER_IB_MSG_560	HY000
12386	ER_IB_MSG_561	HY000
12387	ER_IB_MSG_562	HY000
12388	ER_IB_MSG_563	HY000
12389	ER_IB_MSG_564	HY000
12390	ER_IB_MSG_INVALID_LOCATION_FOR_TABLE	HY000
12391	ER_IB_MSG_566	HY000
12392	ER_IB_MSG_567	HY000
12393	ER_IB_MSG_568	HY000
12394	ER_IB_MSG_569	HY000
12395	ER_IB_MSG_570	HY000
12396	ER_IB_MSG_571	HY000
12398	ER_IB_MSG_573	HY000
12399	ER_IB_MSG_574	HY000
12403	ER_IB_MSG_578	HY000
12404	ER_IB_MSG_579	HY000
12405	ER_IB_MSG_580	HY000
12406	ER_IB_MSG_581	HY000
12407	ER_IB_MSG_582	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12408	ER_IB_MSG_583	HY000
12409	ER_IB_MSG_584	HY000
12410	ER_IB_MSG_585	HY000
12411	ER_IB_MSG_586	HY000
12412	ER_IB_MSG_587	HY000
12413	ER_IB_MSG_588	HY000
12414	ER_IB_MSG_589	HY000
12415	ER_IB_MSG_590	HY000
12416	ER_IB_MSG_591	HY000
12417	ER_IB_MSG_592	HY000
12418	ER_IB_MSG_593	HY000
12419	ER_IB_MSG_594	HY000
12420	ER_IB_MSG_595	HY000
12421	ER_IB_MSG_596	HY000
12422	ER_IB_MSG_597	HY000
12423	ER_IB_MSG_598	HY000
12424	ER_IB_MSG_599	HY000
12425	ER_IB_MSG_600	HY000
12426	ER_IB_MSG_601	HY000
12427	ER_IB_MSG_602	HY000
12428	ER_IB_MSG_603	HY000
12429	ER_IB_MSG_604	HY000
12430	ER_IB_MSG_605	HY000
12431	ER_IB_MSG_606	HY000
12432	ER_IB_MSG_607	HY000
12433	ER_IB_MSG_608	HY000
12434	ER_IB_MSG_609	HY000
12436	ER_IB_MSG_611	HY000
12437	ER_IB_MSG_612	HY000
12438	ER_IB_MSG_613	HY000
12439	ER_IB_MSG_614	HY000
12440	ER_IB_MSG_615	HY000
12441	ER_IB_MSG_616	HY000
12442	ER_IB_MSG_617	HY000
12444	ER_IB_MSG_619	HY000
12445	ER_IB_MSG_IBUF_CURSOR_RESTORATION_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12447	ER_IB_MSG_IBUF_FAILED_TO_RESTORE_POSITION	HY000
12448	ER_IB_MSG_623	HY000
12449	ER_IB_MSG_624	HY000
12451	ER_IB_MSG_626	HY000
12452	ER_IB_MSG_627	HY000
12453	ER_IB_MSG_628	HY000
12454	ER_IB_MSG_629	HY000
12455	ER_IB_MSG_630	HY000
12456	ER_IB_MSG_631	HY000
12457	ER_IB_MSG_632	HY000
12458	ER_IB_MSG_633	HY000
12459	ER_IB_MSG_634	HY000
12460	ER_IB_MSG_635	HY000
12461	ER_IB_MSG_636	HY000
12462	ER_IB_MSG_637	HY000
12463	ER_IB_MSG_638	HY000
12464	ER_IB_MSG_639	HY000
12467	ER_IB_MSG_642	HY000
12468	ER_IB_MSG_643	HY000
12469	ER_IB_MSG_644	HY000
12470	ER_IB_MSG_645	HY000
12471	ER_IB_MSG_646	HY000
12472	ER_IB_MSG_647	HY000
12473	ER_IB_MSG_648	HY000
12474	ER_IB_MSG_649	HY000
12475	ER_IB_MSG_650	HY000
12476	ER_IB_MSG_651	HY000
12477	ER_IB_MSG_652	HY000
12478	ER_IB_MSG_DDL_LOG_DELETE_BY_ID_OK	HY000
12479	ER_IB_MSG_654	HY000
12480	ER_IB_MSG_655	HY000
12481	ER_IB_MSG_656	HY000
12482	ER_IB_MSG_657	HY000
12483	ER_IB_MSG_658	HY000
12484	ER_IB_MSG_659	HY000
12485	ER_IB_MSG_660	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12486	ER_IB_MSG_661	HY000
12487	ER_IB_MSG_662	HY000
12488	ER_IB_MSG_663	HY000
12519	ER_IB_MSG_694	HY000
12520	ER_IB_MSG_695	HY000
12521	ER_IB_MSG_696	HY000
12522	ER_IB_MSG_697	HY000
12523	ER_IB_MSG_LOG_CORRUPT	HY000
12524	ER_IB_MSG_699	HY000
12525	ER_IB_MSG_LOG_FORMAT_OLD_AND_LOG_CORRUPTED	HY000
12526	ER_IB_MSG_LOG_FORMAT_OLD_AND_NO_CLEAN_SHUTDOWN	HY000
12529	ER_IB_MSG_LOG_FORMAT_BEFORE_8_0_30	HY000
12530	ER_IB_MSG_LOG_FILE_FORMAT_UNKNOWN	HY000
12531	ER_IB_MSG_RECOVERY_CHECKPOINT_NOT_FOUND	HY000
12532	ER_IB_MSG_707	HY000
12533	ER_IB_MSG_708	HY000
12534	ER_IB_MSG_709	HY000
12535	ER_IB_MSG_710	HY000
12536	ER_IB_MSG_711	HY000
12537	ER_IB_MSG_712	HY000
12538	ER_IB_MSG_713	HY000
12539	ER_IB_MSG_714	HY000
12540	ER_IB_MSG_715	HY000
12541	ER_IB_MSG_716	HY000
12543	ER_IB_MSG_718	HY000
12544	ER_IB_MSG_719	HY000
12545	ER_IB_MSG_720	HY000
12546	ER_IB_MSG_RECOVERY_SKIPPED_IN_READ_ONLY_MODE	HY000
12547	ER_IB_MSG_722	HY000
12548	ER_IB_MSG_723	HY000
12549	ER_IB_MSG_724	HY000
12550	ER_IB_MSG_725	HY000
12551	ER_IB_MSG_726	HY000
12552	ER_IB_MSG_727	HY000
12553	ER_IB_MSG_728	HY000
12554	ER_IB_MSG_LOG_FILES_CREATED_BY_MEB_AND_READ_ONLY_MODE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12555	ER_IB_MSG_LOG_FILES_CREATED_BY_MEB	HY000
12556	ER_IB_MSG_LOG_FILES_CREATED_BY_CLONE	HY000
12557	ER_IB_MSG_LOG_FORMAT_OLD	HY000
12559	ER_IB_MSG_RECOVERY_CHECKPOINT_FROM_BEFORE_CLEAN_SHUTDOWN	HY000
12560	ER_IB_MSG_RECOVERY_IS_NEEDED	HY000
12561	ER_IB_MSG_RECOVERY_IN_READ_ONLY	HY000
12562	ER_IB_MSG_737	HY000
12563	ER_IB_MSG_738	HY000
12564	ER_IB_MSG_739	HY000
12565	ER_IB_MSG_740	HY000
12566	ER_IB_MSG_741	HY000
12567	ER_IB_MSG_742	HY000
12568	ER_IB_MSG_743	HY000
12569	ER_IB_MSG_744	HY000
12570	ER_IB_MSG_745	HY000
12571	ER_IB_MSG_746	HY000
12572	ER_IB_MSG_747	HY000
12573	ER_IB_MSG_748	HY000
12574	ER_IB_MSG_749	HY000
12575	ER_IB_MSG_750	HY000
12576	ER_IB_MSG_751	HY000
12577	ER_IB_MSG_752	HY000
12578	ER_IB_MSG_753	HY000
12579	ER_IB_MSG_754	HY000
12580	ER_IB_MSG_755	HY000
12581	ER_IB_MSG_756	HY000
12582	ER_IB_MSG_757	HY000
12583	ER_IB_MSG_758	HY000
12584	ER_IB_MSG_759	HY000
12585	ER_IB_MSG_760	HY000
12586	ER_IB_MSG_761	HY000
12587	ER_IB_MSG_762	HY000
12588	ER_IB_MSG_763	HY000
12589	ER_IB_MSG_764	HY000
12590	ER_IB_MSG_765	HY000
12591	ER_IB_MSG_766	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12592	ER_IB_MSG_767	HY000
12593	ER_IB_MSG_768	HY000
12594	ER_IB_MSG_769	HY000
12595	ER_IB_MSG_770	HY000
12596	ER_IB_MSG_771	HY000
12597	ER_IB_MSG_772	HY000
12598	ER_IB_MSG_773	HY000
12599	ER_IB_MSG_774	HY000
12600	ER_IB_MSG_775	HY000
12601	ER_IB_MSG_776	HY000
12602	ER_IB_MSG_777	HY000
12603	ER_IB_MSG_778	HY000
12604	ER_IB_MSG_779	HY000
12605	ER_IB_MSG_780	HY000
12606	ER_IB_MSG_781	HY000
12607	ER_IB_MSG_782	HY000
12608	ER_IB_MSG_783	HY000
12609	ER_IB_MSG_784	HY000
12610	ER_IB_MSG_785	HY000
12611	ER_IB_MSG_786	HY000
12612	ER_IB_MSG_787	HY000
12613	ER_IB_MSG_788	HY000
12614	ER_IB_MSG_789	HY000
12615	ER_IB_MSG_790	HY000
12616	ER_IB_MSG_791	HY000
12617	ER_IB_MSG_792	HY000
12618	ER_IB_MSG_793	HY000
12619	ER_IB_MSG_794	HY000
12620	ER_IB_MSG_795	HY000
12621	ER_IB_MSG_796	HY000
12622	ER_IB_MSG_797	HY000
12623	ER_IB_MSG_798	HY000
12624	ER_IB_MSG_799	HY000
12625	ER_IB_MSG_800	HY000
12626	ER_IB_MSG_801	HY000
12627	ER_IB_MSG_802	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12628	ER_IB_MSG_803	HY000
12629	ER_IB_MSG_804	HY000
12630	ER_IB_MSG_805	HY000
12631	ER_IB_MSG_806	HY000
12632	ER_IB_MSG_807	HY000
12633	ER_IB_MSG_808	HY000
12634	ER_IB_MSG_809	HY000
12635	ER_IB_MSG_810	HY000
12636	ER_IB_MSG_811	HY000
12637	ER_IB_MSG_812	HY000
12638	ER_IB_MSG_813	HY000
12639	ER_IB_MSG_814	HY000
12640	ER_IB_MSG_815	HY000
12641	ER_IB_MSG_816	HY000
12642	ER_IB_MSG_817	HY000
12643	ER_IB_MSG_818	HY000
12644	ER_IB_MSG_819	HY000
12645	ER_IB_MSG_820	HY000
12646	ER_IB_MSG_821	HY000
12647	ER_IB_MSG_822	HY000
12648	ER_IB_MSG_823	HY000
12649	ER_IB_MSG_824	HY000
12650	ER_IB_MSG_825	HY000
12651	ER_IB_MSG_826	HY000
12652	ER_IB_MSG_827	HY000
12653	ER_IB_MSG_828	HY000
12654	ER_IB_MSG_829	HY000
12655	ER_IB_MSG_830	HY000
12656	ER_IB_MSG_831	HY000
12657	ER_IB_MSG_832	HY000
12658	ER_IB_MSG_833	HY000
12659	ER_IB_MSG_834	HY000
12660	ER_IB_MSG_835	HY000
12661	ER_IB_MSG_836	HY000
12662	ER_IB_MSG_837	HY000
12663	ER_IB_MSG_838	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12664	ER_IB_MSG_839	HY000
12665	ER_IB_MSG_840	HY000
12666	ER_IB_MSG_841	HY000
12667	ER_IB_MSG_842	HY000
12668	ER_IB_MSG_CANT_ENCRYPT_REDO_LOG_DATA	HY000
12669	ER_IB_MSG_844	HY000
12670	ER_IB_MSG_845	HY000
12671	ER_IB_MSG_CANT_DECRYPT_REDO_LOG	HY000
12672	ER_IB_MSG_847	HY000
12673	ER_IB_MSG_848	HY000
12674	ER_IB_MSG_849	HY000
12675	ER_IB_MSG_850	HY000
12676	ER_IB_MSG_851	HY000
12677	ER_IB_MSG_852	HY000
12678	ER_IB_MSG_853	HY000
12679	ER_IB_MSG_854	HY000
12680	ER_IB_MSG_855	HY000
12681	ER_IB_MSG_856	HY000
12682	ER_IB_MSG_857	HY000
12683	ER_IB_MSG_858	HY000
12684	ER_IB_MSG_859	HY000
12685	ER_IB_MSG_860	HY000
12686	ER_IB_MSG_861	HY000
12687	ER_IB_MSG_862	HY000
12688	ER_IB_MSG_863	HY000
12689	ER_IB_MSG_864	HY000
12690	ER_IB_MSG_865	HY000
12691	ER_IB_MSG_866	HY000
12692	ER_IB_MSG_867	HY000
12693	ER_IB_MSG_868	HY000
12694	ER_IB_MSG_869	HY000
12695	ER_IB_MSG_870	HY000
12696	ER_IB_MSG_871	HY000
12697	ER_IB_MSG_872	HY000
12698	ER_IB_MSG_873	HY000
12699	ER_IB_MSG_874	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12700	ER_IB_MSG_875	HY000
12701	ER_IB_MSG_876	HY000
12702	ER_IB_MSG_877	HY000
12703	ER_IB_MSG_878	HY000
12704	ER_IB_MSG_879	HY000
12705	ER_IB_MSG_880	HY000
12706	ER_IB_MSG_881	HY000
12707	ER_IB_MSG_882	HY000
12708	ER_IB_MSG_883	HY000
12709	ER_IB_MSG_884	HY000
12710	ER_IB_MSG_885	HY000
12711	ER_IB_MSG_886	HY000
12712	ER_IB_MSG_887	HY000
12713	ER_IB_MSG_888	HY000
12714	ER_IB_MSG_889	HY000
12715	ER_IB_MSG_890	HY000
12716	ER_IB_MSG_891	HY000
12717	ER_IB_MSG_892	HY000
12718	ER_IB_MSG_893	HY000
12719	ER_IB_MSG_894	HY000
12720	ER_IB_MSG_895	HY000
12721	ER_IB_MSG_896	HY000
12722	ER_IB_MSG_897	HY000
12723	ER_IB_MSG_898	HY000
12724	ER_IB_MSG_899	HY000
12725	ER_IB_MSG_900	HY000
12726	ER_IB_MSG_901	HY000
12727	ER_IB_MSG_902	HY000
12728	ER_IB_MSG_903	HY000
12729	ER_IB_MSG_904	HY000
12730	ER_IB_MSG_905	HY000
12731	ER_IB_MSG_906	HY000
12732	ER_IB_MSG_907	HY000
12733	ER_IB_MSG_908	HY000
12734	ER_IB_MSG_909	HY000
12735	ER_IB_MSG_910	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12736	ER_IB_MSG_911	HY000
12737	ER_IB_MSG_912	HY000
12738	ER_IB_MSG_913	HY000
12739	ER_IB_MSG_914	HY000
12740	ER_IB_MSG_915	HY000
12741	ER_IB_MSG_916	HY000
12742	ER_IB_MSG_917	HY000
12743	ER_IB_MSG_918	HY000
12744	ER_IB_MSG_919	HY000
12745	ER_IB_MSG_920	HY000
12746	ER_IB_MSG_921	HY000
12747	ER_IB_MSG_922	HY000
12748	ER_IB_MSG_923	HY000
12749	ER_IB_MSG_924	HY000
12750	ER_IB_MSG_925	HY000
12751	ER_IB_MSG_926	HY000
12752	ER_IB_MSG_927	HY000
12753	ER_IB_MSG_928	HY000
12754	ER_IB_MSG_929	HY000
12755	ER_IB_MSG_930	HY000
12756	ER_IB_MSG_931	HY000
12757	ER_IB_MSG_932	HY000
12758	ER_IB_MSG_933	HY000
12759	ER_IB_MSG_934	HY000
12760	ER_IB_MSG_935	HY000
12761	ER_IB_MSG_936	HY000
12762	ER_IB_MSG_937	HY000
12763	ER_IB_MSG_938	HY000
12764	ER_IB_MSG_939	HY000
12765	ER_IB_MSG_940	HY000
12766	ER_IB_MSG_941	HY000
12767	ER_IB_MSG_942	HY000
12768	ER_IB_MSG_943	HY000
12769	ER_IB_MSG_944	HY000
12770	ER_IB_MSG_945	HY000
12771	ER_IB_MSG_946	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12772	ER_IB_IMPORT_INDEX_METADATA_READ_FAILED	HY000
12773	ER_IB_MSG_948	HY000
12774	ER_IB_MSG_949	HY000
12775	ER_IB_MSG_950	HY000
12776	ER_IB_MSG_951	HY000
12777	ER_IB_MSG_952	HY000
12778	ER_IB_MSG_953	HY000
12779	ER_IB_MSG_954	HY000
12780	ER_IB_MSG_955	HY000
12781	ER_IB_MSG_956	HY000
12782	ER_IB_MSG_957	HY000
12783	ER_IB_MSG_958	HY000
12784	ER_IB_MSG_959	HY000
12785	ER_IB_MSG_960	HY000
12786	ER_IB_MSG_961	HY000
12787	ER_IB_MSG_962	HY000
12788	ER_IB_MSG_963	HY000
12789	ER_IB_MSG_964	HY000
12790	ER_IB_MSG_965	HY000
12791	ER_IB_MSG_966	HY000
12792	ER_IB_MSG_967	HY000
12793	ER_IB_MSG_968	HY000
12794	ER_IB_MSG_969	HY000
12795	ER_IB_MSG_970	HY000
12796	ER_IB_MSG_971	HY000
12797	ER_IB_MSG_972	HY000
12798	ER_IB_MSG_973	HY000
12799	ER_IB_MSG_974	HY000
12800	ER_IB_MSG_975	HY000
12801	ER_IB_MSG_976	HY000
12802	ER_IB_MSG_977	HY000
12803	ER_IB_MSG_978	HY000
12804	ER_IB_MSG_979	HY000
12805	ER_IB_MSG_980	HY000
12806	ER_IB_MSG_981	HY000
12807	ER_IB_MSG_982	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12808	ER_IB_MSG_983	HY000
12809	ER_IB_MSG_984	HY000
12810	ER_IB_MSG_985	HY000
12811	ER_IB_MSG_986	HY000
12812	ER_IB_MSG_987	HY000
12813	ER_IB_MSG_988	HY000
12814	ER_IB_MSG_989	HY000
12815	ER_IB_MSG_990	HY000
12816	ER_IB_MSG_991	HY000
12817	ER_IB_MSG_992	HY000
12818	ER_IB_MSG_993	HY000
12819	ER_IB_MSG_994	HY000
12820	ER_IB_MSG_995	HY000
12821	ER_IB_MSG_996	HY000
12822	ER_IB_MSG_997	HY000
12823	ER_IB_MSG_998	HY000
12824	ER_IB_MSG_999	HY000
12825	ER_IB_MSG_1000	HY000
12826	ER_IB_MSG_1001	HY000
12827	ER_IB_MSG_1002	HY000
12828	ER_IB_MSG_1003	HY000
12829	ER_IB_MSG_1004	HY000
12830	ER_IB_MSG_1005	HY000
12831	ER_IB_MSG_1006	HY000
12832	ER_IB_MSG_1007	HY000
12833	ER_IB_MSG_1008	HY000
12834	ER_IB_MSG_1009	HY000
12835	ER_IB_MSG_1010	HY000
12836	ER_IB_MSG_1011	HY000
12837	ER_IB_MSG_1012	HY000
12838	ER_IB_MSG_1013	HY000
12839	ER_IB_IMPORT_START_CFG_NAME	HY000
12840	ER_IB_MSG_1015	HY000
12841	ER_IB_MSG_1016	HY000
12842	ER_IB_MSG_1017	HY000
12843	ER_IB_MSG_1018	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12844	ER_IB_MSG_1019	HY000
12845	ER_IB_MSG_1020	HY000
12846	ER_IB_MSG_1021	HY000
12847	ER_IB_MSG_1022	HY000
12848	ER_IB_MSG_1023	HY000
12849	ER_IB_MSG_1024	HY000
12850	ER_IB_MSG_1025	HY000
12851	ER_IB_MSG_1026	HY000
12852	ER_IB_MSG_1027	HY000
12853	ER_IB_MSG_1028	HY000
12854	ER_IB_MSG_1029	HY000
12855	ER_IB_MSG_1030	HY000
12856	ER_IB_MSG_1031	HY000
12857	ER_IB_MSG_1032	HY000
12858	ER_IB_MSG_1033	HY000
12859	ER_IB_MSG_1034	HY000
12860	ER_IB_MSG_1035	HY000
12861	ER_IB_MSG_1036	HY000
12862	ER_IB_MSG_1037	HY000
12863	ER_IB_MSG_1038	HY000
12864	ER_IB_MSG_1039	HY000
12865	ER_IB_MSG_1040	HY000
12866	ER_IB_MSG_1041	HY000
12867	ER_IB_MSG_1042	HY000
12868	ER_IB_MSG_1043	HY000
12869	ER_IB_MSG_1044	HY000
12870	ER_IB_MSG_1045	HY000
12871	ER_IB_MSG_1046	HY000
12872	ER_IB_MSG_1047	HY000
12873	ER_IB_MSG_1048	HY000
12874	ER_IB_MSG_1049	HY000
12876	ER_IB_MSG_1051	HY000
12877	ER_IB_MSG_1052	HY000
12878	ER_IB_MSG_1053	HY000
12879	ER_IB_MSG_1054	HY000
12880	ER_IB_MSG_1055	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12881	ER_IB_MSG_1056	HY000
12882	ER_IB_MSG_1057	HY000
12883	ER_IB_MSG_1058	HY000
12884	ER_IB_MSG_1059	HY000
12885	ER_IB_MSG_1060	HY000
12886	ER_IB_MSG_LOG_FILE_OS_CREATE_FAILED	HY000
12887	ER_IB_MSG_FILE_RESIZE	HY000
12888	ER_IB_MSG_LOG_FILE_RESIZE_FAILED	HY000
12889	ER_IB_MSG_LOG_FILES_CREATE_AND_READ_ONLY_MODE	HY000
12890	ER_IB_MSG_1065	HY000
12891	ER_IB_MSG_LOG_FILE_PREPARE_ON_CREATE_FAILED	HY000
12893	ER_IB_MSG_LOG_FILES_INITIALIZED	HY000
12894	ER_IB_MSG_LOG_FILE_OPEN_FAILED	HY000
12895	ER_IB_MSG_1070	HY000
12896	ER_IB_MSG_1071	HY000
12897	ER_IB_MSG_1072	HY000
12898	ER_IB_MSG_1073	HY000
12899	ER_IB_MSG_1074	HY000
12900	ER_IB_MSG_1075	HY000
12901	ER_IB_MSG_1076	HY000
12902	ER_IB_MSG_1077	HY000
12903	ER_IB_MSG_1078	HY000
12904	ER_IB_MSG_1079	HY000
12905	ER_IB_MSG_1080	HY000
12906	ER_IB_MSG_1081	HY000
12907	ER_IB_MSG_1082	HY000
12908	ER_IB_MSG_1083	HY000
12909	ER_IB_MSG_CANNOT_OPEN_57_UNDO	HY000
12910	ER_IB_MSG_1085	HY000
12911	ER_IB_MSG_1086	HY000
12912	ER_IB_MSG_1087	HY000
12913	ER_IB_MSG_1088	HY000
12914	ER_IB_MSG_1089	HY000
12915	ER_IB_MSG_1090	HY000
12916	ER_IB_MSG_1091	HY000
12917	ER_IB_MSG_1092	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12918	ER_IB_MSG_1093	HY000
12919	ER_IB_MSG_1094	HY000
12920	ER_IB_MSG_1095	HY000
12921	ER_IB_MSG_1096	HY000
12922	ER_IB_MSG_1097	HY000
12923	ER_IB_MSG_1098	HY000
12924	ER_IB_MSG_1099	HY000
12925	ER_IB_MSG_1100	HY000
12926	ER_IB_MSG_1101	HY000
12927	ER_IB_MSG_1102	HY000
12928	ER_IB_MSG_1103	HY000
12929	ER_IB_MSG_1104	HY000
12930	ER_IB_MSG_1105	HY000
12931	ER_IB_MSG_BUF_PENDING_IO	HY000
12932	ER_IB_MSG_1107	HY000
12933	ER_IB_MSG_1108	HY000
12934	ER_IB_MSG_1109	HY000
12935	ER_IB_MSG_1110	HY000
12936	ER_IB_MSG_1111	HY000
12937	ER_IB_MSG_1112	HY000
12938	ER_IB_MSG_1113	HY000
12939	ER_IB_MSG_1114	HY000
12940	ER_IB_MSG_1115	HY000
12941	ER_IB_MSG_1116	HY000
12942	ER_IB_MSG_1117	HY000
12944	ER_IB_MSG_1119	HY000
12945	ER_IB_MSG_1120	HY000
12946	ER_IB_MSG_1121	HY000
12947	ER_IB_MSG_1122	HY000
12948	ER_IB_MSG_1123	HY000
12949	ER_IB_MSG_1124	HY000
12950	ER_IB_MSG_1125	HY000
12951	ER_IB_MSG_1126	HY000
12952	ER_IB_MSG_1127	HY000
12953	ER_IB_MSG_1128	HY000
12954	ER_IB_MSG_1129	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12955	ER_IB_MSG_1130	HY000
12956	ER_IB_MSG_1131	HY000
12957	ER_IB_MSG_1132	HY000
12958	ER_IB_MSG_1133	HY000
12959	ER_IB_MSG_1134	HY000
12960	ER_IB_MSG_DATA_DIRECTORY_NOT_INITIALIZED_OR_CORRUPTED	HY000
12961	ER_IB_MSG_LOG_FILES_INVALID_SET	HY000
12962	ER_IB_MSG_LOG_FILE_SIZE_INVALID	HY000
12963	ER_IB_MSG_LOG_FILES_DIFFERENT_SIZES	HY000
12964	ER_IB_MSG_1139	HY000
12965	ER_IB_MSG_RECOVERY_CORRUPT	HY000
12967	ER_IB_MSG_1142	HY000
12968	ER_IB_MSG_LOG_FILES_REWRITING	HY000
12969	ER_IB_MSG_1144	HY000
12970	ER_IB_MSG_1145	HY000
12971	ER_IB_MSG_1146	HY000
12972	ER_IB_MSG_1147	HY000
12973	ER_IB_MSG_1148	HY000
12974	ER_IB_MSG_1149	HY000
12975	ER_IB_MSG_1150	HY000
12976	ER_IB_MSG_1151	HY000
12977	ER_IB_MSG_1152	HY000
12979	ER_IB_MSG_1154	HY000
12980	ER_IB_MSG_1155	HY000
12981	ER_IB_MSG_1156	HY000
12982	ER_IB_MSG_1157	HY000
12983	ER_IB_MSG_1158	HY000
12984	ER_IB_MSG_1159	HY000
12985	ER_IB_MSG_1160	HY000
12986	ER_IB_MSG_1161	HY000
12987	ER_IB_MSG_1162	HY000
12988	ER_IB_MSG_1163	HY000
12989	ER_IB_MSG_1164	HY000
12990	ER_IB_MSG_1165	HY000
12991	ER_IB_MSG_UNDO_TRUNCATE_FAIL_TO_READ_LOG_FILE	HY000
12992	ER_IB_MSG_UNDO_MARKED_FOR_TRUNCATE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
12994	ER_IB_MSG_UNDO_TRUNCATE_START	HY000
12996	ER_IB_MSG_UNDO_TRUNCATE_DELAY_BY_LOG_CREATE	HY000
12998	ER_IB_MSG_UNDO_TRUNCATE_DELAY_BY_FAILURE	HY000
13000	ER_IB_MSG_UNDO_TRUNCATE_COMPLETE	HY000
13002	ER_IB_MSG_1177	HY000
13003	ER_IB_MSG_1178	HY000
13004	ER_IB_MSG_1179	HY000
13005	ER_IB_MSG_1180	HY000
13006	ER_IB_MSG_1181	HY000
13007	ER_IB_MSG_1182	HY000
13008	ER_IB_MSG_1183	HY000
13009	ER_IB_ERR_ACCESSING_OUT_OF_BOUND_FIELD_IN_INDEX	HY000
13010	ER_IB_MSG_1185	HY000
13011	ER_IB_MSG_1186	HY000
13012	ER_IB_MSG_1187	HY000
13013	ER_IB_MSG_1188	HY000
13014	ER_IB_MSG_1189	HY000
13015	ER_IB_MSG_TRX_RECOVERY_ROLLBACK_COMPLETED	HY000
13016	ER_IB_MSG_1191	HY000
13017	ER_IB_MSG_1192	HY000
13018	ER_IB_MSG_1193	HY000
13019	ER_IB_MSG_1194	HY000
13020	ER_IB_MSG_1195	HY000
13021	ER_IB_MSG_1196	HY000
13022	ER_IB_MSG_1197	HY000
13023	ER_IB_MSG_1198	HY000
13024	ER_IB_MSG_1199	HY000
13025	ER_IB_MSG_1200	HY000
13026	ER_IB_MSG_1201	HY000
13027	ER_IB_MSG_1202	HY000
13028	ER_IB_MSG_1203	HY000
13029	ER_IB_MSG_1204	HY000
13030	ER_IB_MSG_1205	HY000
13031	ER_IB_MSG_1206	HY000
13032	ER_IB_MSG_1207	HY000
13033	ER_IB_MSG_1208	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13034	ER_IB_MSG_1209	HY000
13035	ER_IB_MSG_1210	HY000
13036	ER_IB_MSG_1211	HY000
13037	ER_IB_MSG_1212	HY000
13038	ER_IB_MSG_1213	HY000
13039	ER_IB_MSG_1214	HY000
13040	ER_IB_MSG_1215	HY000
13041	ER_IB_MSG_LOG_FILES_RESIZE_ON_START	HY000
13042	ER_IB_MSG_1217	HY000
13043	ER_IB_MSG_1218	HY000
13044	ER_IB_MSG_1219	HY000
13045	ER_IB_MSG_1220	HY000
13046	ER_IB_MSG_1221	HY000
13047	ER_IB_MSG_1222	HY000
13048	ER_IB_MSG_1223	HY000
13049	ER_IB_MSG_1224	HY000
13050	ER_IB_MSG_1225	HY000
13051	ER_IB_MSG_1226	HY000
13052	ER_IB_MSG_1227	HY000
13053	ER_IB_MSG_1228	HY000
13054	ER_IB_MSG_1229	HY000
13056	ER_IB_MSG_1231	HY000
13058	ER_IB_MSG_1233	HY000
13059	ER_IB_MSG_LOG_WRITER_OUT_OF_SPACE	HY000
13060	ER_IB_MSG_1235	HY000
13061	ER_IB_MSG_LOG_WRITER_ABORTS_LOG_ARCHIVER	HY000
13062	ER_IB_MSG_LOG_WRITER_WAITING_FOR_ARCHIVER	HY000
13063	ER_IB_MSG_1238	HY000
13064	ER_IB_MSG_1239	HY000
13066	ER_IB_MSG_1241	HY000
13067	ER_IB_MSG_LOG_FILES_CANNOT_ENCRYPT_IN_READ_ONLY	HY000
13068	ER_IB_MSG_LOG_FILES_ENCRYPTION_INIT_FAILED	HY000
13070	ER_IB_MSG_1245	HY000
13071	ER_IB_MSG_1246	HY000
13072	ER_IB_MSG_1247	HY000
13073	ER_IB_MSG_1248	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13074	ER_IB_MSG_1249	HY000
13075	ER_IB_MSG_1250	HY000
13076	ER_IB_MSG_1251	HY000
13077	ER_IB_MSG_BUF_PENDING_IO_ON_SHUTDOWN	HY000
13078	ER_IB_MSG_1253	HY000
13080	ER_IB_MSG_1255	HY000
13081	ER_IB_MSG_1256	HY000
13082	ER_IB_MSG_1257	HY000
13083	ER_IB_MSG_1258	HY000
13084	ER_IB_MSG_1259	HY000
13085	ER_IB_MSG_1260	HY000
13086	ER_IB_MSG_1261	HY000
13087	ER_IB_MSG_1262	HY000
13088	ER_IB_MSG_1263	HY000
13089	ER_IB_MSG_LOG_FILE_HEADER_INVALID_CHECKSUM	HY000
13090	ER_IB_MSG_LOG_FORMAT_BEFORE_5_7_9	HY000
13091	ER_IB_MSG_1266	HY000
13092	ER_IB_MSG_LOG_PARAMS_CONCURRENCY_MARGIN_UNSAFE	HY000
13093	ER_IB_MSG_1268	HY000
13094	ER_IB_MSG_1269	HY000
13095	ER_IB_MSG_THREAD_CONCURRENCY_CHANGED	HY000
13096	ER_RPL_REPLICA_SQL_THREAD_STOP_CMD_EXEC_TIMEOUT	HY000
13097	ER_RPL_REPLICA_IO_THREAD_STOP_CMD_EXEC_TIMEOUT	HY000
13098	ER_RPL_GTID_UNSAFE_STMT_ON_NON_TRANS_TABLE	HY000
13099	ER_RPL_GTID_UNSAFE_STMT_CREATE_SELECT	HY000
13101	ER_BINLOG_ROW_VALUE_OPTION_IGNORED	HY000
13103	ER_BINLOG_ROW_VALUE_OPTION_USED_ONLY_FOR_AFTER_IMAGES	HY000
13104	ER_CONNECTION_ABORTED	HY000
13105	ER_NORMAL_SERVER_SHUTDOWN	HY000
13106	ER_KEYRING_MIGRATE_FAILED	HY000
13107	ER_GRP_RPL_LOWER_CASE_TABLE_NAMES_DIFF_FROM_GRP	HY000
13108	ER_OOM_SAVE_GTIDS	HY000
13109	ER_LCTN_NOT_FOUND	HY000
13111	ER_COMPONENT_FILTER_WRONG_VALUE	HY000
13112	ER_XPLUGIN_FAILED_TO_STOP_SERVICES	HY000
13113	ER_INCONSISTENT_ERROR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13114	ER_SERVER_SOURCE_FATAL_ERROR_READING_BINLOG	HY000
13115	ER_NETWORK_READ_EVENT_CHECKSUM_FAILURE	HY000
13116	ER_REPLICA_CREATE_EVENT_FAILURE	HY000
13117	ER_REPLICA_FATAL_ERROR	HY000
13118	ER_REPLICA_HEARTBEAT_FAILURE	HY000
13119	ER_REPLICA_INCIDENT	HY000
13120	ER_REPLICA_SOURCE_COM_FAILURE	HY000
13121	ER_REPLICA_RELAY_LOG_READ_FAILURE	HY000
13122	ER_REPLICA_RELAY_LOG_WRITE_FAILURE	HY000
13123	ER_SERVER_REPLICA_CM_INIT_REPOSITORY	HY000
13124	ER_SERVER_REPLICA_AM_INIT_REPOSITORY	HY000
13125	ER_SERVER_NET_PACKET_TOO_LARGE	HY000
13126	ER_SERVER_NO_SYSTEM_TABLE_ACCESS	HY000
13128	ER_SERVER_UNKNOWN_SYSTEM_VARIABLE	HY000
13129	ER_SERVER_NO_SESSION_TO_SEND_TO	HY000
13130	ER_SERVER_NEW_ABORTING_CONNECTION	08S01
13131	ER_SERVER_OUT_OF_SORTMEMORY	HY000
13132	ER_SERVER_RECORD_FILE_FULL	HY000
13133	ER_SERVER_DISK_FULL_NOWAIT	HY000
13134	ER_SERVER_HANDLER_ERROR	HY000
13135	ER_SERVER_NOT_FORM_FILE	HY000
13136	ER_SERVER_CANT_OPEN_FILE	HY000
13137	ER_SERVER_FILE_NOT_FOUND	HY000
13138	ER_SERVER_FILE_USED	HY000
13139	ER_SERVER_CANNOT_LOAD_FROM_TABLE_V2	HY000
13140	ER_ERROR_INFO_FROM_DA	HY000
13141	ER_SERVER_TABLE_CHECK_FAILED	HY000
13142	ER_SERVER_COL_COUNT_DOESNT_MATCH_PLEASE_UPDATE_V2	HY000
13143	ER_SERVER_COL_COUNT_DOESNT_MATCH_CORRUPTED_V2	HY000
13144	ER_SERVER_ACL_TABLE_ERROR	HY000
13145	ER_SERVER_REPLICA_INIT_QUERY_FAILED	HY000
13146	ER_SERVER_REPLICA_CONVERSION_FAILED	HY000
13147	ER_SERVER_REPLICA_IGNORED_TABLE	HY000
13148	ER_CANT_REPLICATE_ANONYMOUS_WITH_AUTO_POSITION	HY000
13149	ER_CANT_REPLICATE_ANONYMOUS_WITH_GTID_MODE_ON	HY000
13150	ER_CANT_REPLICATE_GTID_WITH_GTID_MODE_OFF	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13151	ER_SERVER_TEST_MESSAGE	HY000
13152	ER_AUDIT_LOG_JSON_FILTER_PARSING_ERROR	HY000
13153	ER_AUDIT_LOG_JSON_FILTERING_NOT_ENABLED	HY000
13154	ER_PLUGIN_FAILED_TO_OPEN_TABLES	HY000
13155	ER_PLUGIN_FAILED_TO_OPEN_TABLE	HY000
13156	ER_AUDIT_LOG_JSON_FILTER_NAME_CANNOT_BE_EMPTY	HY000
13157	ER_AUDIT_LOG_USER_NAME_INVALID_CHARACTER	HY000
13158	ER_AUDIT_LOG_UDF_INSUFFICIENT_PRIVILEGE	HY000
13159	ER_AUDIT_LOG_NO_KEYRING_PLUGIN_INSTALLED	HY000
13160	ER_AUDIT_LOG_HOST_NAME_INVALID_CHARACTER	HY000
13161	ER_AUDIT_LOG_ENCRYPTION_PASSWORD_HAS_NOT_BEEN_SET	HY000
13162	ER_AUDIT_LOG_COULD_NOT_CREATE_AES_KEY	HY000
13163	ER_AUDIT_LOG_ENCRYPTION_PASSWORD_CANNOT_BE_FETCHED	HY000
13164	ER_COULD_NOT_REINITIALIZE_AUDIT_LOG_FILTERS	HY000
13165	ER_AUDIT_LOG_JSON_USER_NAME_CANNOT_BE_EMPTY	HY000
13166	ER_AUDIT_LOG_USER_FIRST_CHARACTER_MUST_BE_ALPHANUMERIC	HY000
13167	ER_AUDIT_LOG_JSON_FILTER_DOES_NOT_EXIST	HY000
13169	ER_STARTING_INIT	HY000
13170	ER_ENDING_INIT	HY000
13171	ER_IB_MSG_1272	HY000
13172	ER_SERVER_SHUTDOWN_INFO	HY000
13173	ER_GRP_RPL_PLUGIN_ABORT	HY000
13177	ER_AUDIT_LOG_TABLE_DEFINITION_NOT_UPDATED	HY000
13178	ER_DD_INITIALIZE_SQL_ERROR	HY000
13179	ER_NO_PATH_FOR_SHARED_LIBRARY	HY000
13180	ER_UDF_ALREADY_EXISTS	HY000
13181	ER_SET_EVENT_FAILED	HY000
13182	ER_FAILED_TO_ALLOCATE_SSL_BIO	HY000
13183	ER_IB_MSG_1273	HY000
13184	ER_PID_FILEPATH_LOCATIONS_INACCESSIBLE	HY000
13185	ER_UNKNOWN_VARIABLE_IN_PERSISTED_CONFIG_FILE	HY000
13186	ER_FAILED_TO_HANDLE_DEFAULTS_FILE	HY000
13187	ER_DUPLICATE_SYS_VAR	HY000
13188	ER_FAILED_TO_INIT_SYS_VAR	HY000
13189	ER_SYS_VAR_NOT_FOUND	HY000
13190	ER_IB_MSG_1274	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13191	ER_IB_MSG_1275	HY000
13193	ER_IB_MSG_WAIT_FOR_ENCRYPT_THREAD	HY000
13194	ER_IB_MSG_1277	HY000
13195	ER_IB_MSG_NO_ENCRYPT_PROGRESS_FOUND	HY000
13196	ER_IB_MSG_RESUME_OP_FOR_SPACE	HY000
13197	ER_IB_MSG_1280	HY000
13198	ER_IB_MSG_1281	HY000
13199	ER_IB_MSG_1282	HY000
13200	ER_IB_MSG_1283	HY000
13201	ER_IB_MSG_1284	HY000
13202	ER_CANT_SET_ERROR_SUPPRESSION_LIST_FROM_COMMAND_LINE	HY000
13203	ER_INVALID_VALUE_OF_BIND_ADDRESSES	HY000
13204	ER_RELAY_LOG_SPACE_LIMIT_DISABLED	HY000
13205	ER_GRP_RPL_ERROR_GTID_SET_EXTRACTION	HY000
13206	ER_GRP_RPL_MISSING_GRP_RPL_ACTION_COORDINATOR	HY000
13207	ER_GRP_RPL_JOIN_WHEN_GROUP_ACTION_RUNNING	HY000
13208	ER_GRP_RPL_JOINER_EXIT_WHEN_GROUP_ACTION_RUNNING	HY000
13209	ER_GRP_RPL_CHANNEL_THREAD_WHEN_GROUP_ACTION_RUNNING	HY000
13210	ER_GRP_RPL_APPOINTED_PRIMARY_NOT_PRESENT	HY000
13211	ER_GRP_RPL_ERROR_ON_MESSAGE_SENDING	HY000
13212	ER_GRP_RPL_CONFIGURATION_ACTION_ERROR	HY000
13213	ER_GRP_RPL_CONFIGURATION_ACTION_LOCAL_TERMINATION	HY000
13214	ER_GRP_RPL_CONFIGURATION_ACTION_START	HY000
13215	ER_GRP_RPL_CONFIGURATION_ACTION_END	HY000
13216	ER_GRP_RPL_CONFIGURATION_ACTION_KILLED_ERROR	HY000
13217	ER_GRP_RPL_PRIMARY_ELECTION_PROCESS_ERROR	HY000
13218	ER_GRP_RPL_PRIMARY_ELECTION_STOP_ERROR	HY000
13219	ER_GRP_RPL_NO_STAGE_SERVICE	HY000
13220	ER_GRP_RPL_UDF_REGISTER_ERROR	HY000
13221	ER_GRP_RPL_UDF_UNREGISTER_ERROR	HY000
13222	ER_GRP_RPL_UDF_REGISTER_SERVICE_ERROR	HY000
13223	ER_GRP_RPL_SERVER_UDF_ERROR	HY000
13227	ER_SERVER_WRONG_VALUE_FOR_VAR	HY000
13228	ER_COULD_NOT_CREATE_WINDOWS_REGISTRY_KEY	HY000
13229	ER_SERVER_GTID_UNSAFE_CREATE_DROP_TEMP_TABLE_IN_TRX_IN_SBR	HY000
13233	ER_XPLUGIN_FAILED_TO_SWITCH_SECURITY_CTX	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13234	ER_RPL_GTID_UNSAFE_ALTER_ADD_COL_WITH_DEFAULT_EXPRESSION	HY000
13235	ER_UPGRADE_PARSE_ERROR	HY000
13236	ER_DATA_DIRECTORY_UNUSABLE	HY000
13237	ER_LDAP_AUTH_USER_GROUP_SEARCH_ROOT_BIND	HY000
13238	ER_PLUGIN_INSTALL_ERROR	HY000
13239	ER_PLUGIN_UNINSTALL_ERROR	HY000
13240	ER_SHARED_TABLESPACE_USED_BY_PARTITIONED_TABLE	HY000
13241	ER_UNKNOWN_TABLESPACE_TYPE	HY000
13242	ER_WARN_DEPRECATED_UTF8_ALIAS_OPTION	HY000
13243	ER_WARN_DEPRECATED_UTF8MB3_CHARSET_OPTION	HY000
13244	ER_WARN_DEPRECATED_UTF8MB3_COLLATION_OPTION	HY000
13245	ER_SSL_MEMORY_INSTRUMENTATION_INIT_FAILED	HY000
13246	ER_IB_MSG_MADV_DONTDUMP_UNSUPPORTED	HY000
13247	ER_IB_MSG_MADVISE_FAILED	HY000
13249	ER_WARN_REMOVED_SQL_MODE	HY000
13250	ER_IB_MSG_FAILED_TO_ALLOCATE_WAIT	HY000
13252	ER_IB_MSG_USING_UNDO_SPACE	HY000
13253	ER_IB_MSG_FAIL_TO_SAVE_SPACE_STATE	HY000
13254	ER_IB_MSG_MAX_UNDO_SPACES_REACHED	HY000
13255	ER_IB_MSG_ERROR_OPENING_NEW_UNDO_SPACE	HY000
13256	ER_IB_MSG_FAILED_SDI_Z_BUF_ERROR	HY000
13257	ER_IB_MSG_FAILED_SDI_Z_MEM_ERROR	HY000
13258	ER_IB_MSG_SDI_Z_STREAM_ERROR	HY000
13259	ER_IB_MSG_SDI_Z_UNKNOWN_ERROR	HY000
13260	ER_IB_MSG_FOUND_WRONG_UNDO_SPACE	HY000
13261	ER_IB_MSG_NOT_END_WITH_IBU	HY000
13266	ER_IB_MSG_FAILED_TO_FINISH_TRUNCATE	HY000
13267	ER_IB_MSG_DEPRECATED_INNODB_UNDO_TABLESPACES	HY000
13268	ER_IB_MSG_WRONG_TABLESPACE_DIR	HY000
13269	ER_IB_MSG_LOCK_FREE_HASH_USAGE_STATS	HY000
13270	ER_CLONE_DONOR_TRACE	HY000
13271	ER_CLONE_PROTOCOL_TRACE	HY000
13272	ER_CLONE_CLIENT_TRACE	HY000
13273	ER_CLONE_SERVER_TRACE	HY000
13274	ER_THREAD_POOL_PFS_TABLES_INIT_FAILED	HY000
13275	ER_THREAD_POOL_PFS_TABLES_ADD_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13276	ER_CANT_SET_DATA_DIR	HY000
13277	ER_INNODB_INVALID_INNODB_UNDO_DIRECTORY_LOCATION	HY000
13278	ER_SERVER_RPL_ENCRYPTION_FAILED_TO_FETCH_KEY	HY000
13279	ER_SERVER_RPL_ENCRYPTION_KEY_NOT_FOUND	HY000
13280	ER_SERVER_RPL_ENCRYPTION_KEYRING_INVALID_KEY	HY000
13281	ER_SERVER_RPL_ENCRYPTION_HEADER_ERROR	HY000
13282	ER_SERVER_RPL_ENCRYPTION_FAILED_TO_ROTATE_LOGS	HY000
13283	ER_SERVER_RPL_ENCRYPTION_KEY_EXISTS_UNEXPECTED	HY000
13284	ER_SERVER_RPL_ENCRYPTION_FAILED_TO_GENERATE_KEY	HY000
13285	ER_SERVER_RPL_ENCRYPTION_FAILED_TO_STORE_KEY	HY000
13286	ER_SERVER_RPL_ENCRYPTION_FAILED_TO_REMOVE_KEY	HY000
13287	ER_SERVER_RPL_ENCRYPTION_MASTER_KEY_RECOVERY_FAILED	HY000
13288	ER_SERVER_RPL_ENCRYPTION_UNABLE_TO_INITIALIZE	HY000
13289	ER_SERVER_RPL_ENCRYPTION_UNABLE_TO_ROTATE_MASTER_KEY_AT_STARTUP	HY000
13290	ER_SERVER_RPL_ENCRYPTION_IGNORE_ROTATE_MASTER_KEY_AT_STARTUP	HY000
13291	ER_INVALID_ADMIN_ADDRESS	HY000
13292	ER_SERVER_STARTUP_ADMIN_INTERFACE	HY000
13293	ER_CANT_CREATE_ADMIN_THREAD	HY000
13294	ER_WARNING_RETAIN_CURRENT_PASSWORD_CLAUSE_VOID	HY000
13295	ER_WARNING_DISCARD_OLD_PASSWORD_CLAUSE_VOID	HY000
13299	ER_WARNING_AUTHCACHE_INVALID_USER_ATTRIBUTES	HY000
13300	ER_MYSQL_NATIVE_PASSWORD_SECOND_PASSWORD_USED_INFORMATION	HY000
13301	ER_SHA256_PASSWORD_SECOND_PASSWORD_USED_INFORMATION	HY000
13302	ER_CACHING_SHA2_PASSWORD_SECOND_PASSWORD_USED_INFORMATION	HY000
13303	ER_GRP_RPL_SEND_TRX_PREPARED_MESSAGE_FAILED	HY000
13304	ER_GRP_RPL_RELEASE_COMMIT_AFTER_GROUP_PREPARE_FAILED	HY000
13305	ER_GRP_RPL_TRX_ALREADY_EXISTS_ON_TCM_ON_AFTER_CERTIFICATION	HY000
13306	ER_GRP_RPL_FAILED_TO_INSERT_TRX_ON_TCM_ON_AFTER_CERTIFICATION	HY000
13307	ER_GRP_RPL_REGISTER_TRX_TO_WAIT_FOR_GROUP_PREPARE_FAILED	HY000
13308	ER_GRP_RPL_TRX_WAIT_FOR_GROUP_PREPARE_FAILED	HY000
13309	ER_GRP_RPL_TRX_DOES_NOT_EXIST_ON_TCM_ON_HANDLE_REMOTE_PREPARE	HY000
13310	ER_GRP_RPL_RELEASE_BEGIN_TRX_AFTER_DEPENDENCIES_COMMIT_FAILED	HY000
13311	ER_GRP_RPL_REGISTER_TRX_TO_WAIT_FOR_DEPENDENCIES_FAILED	HY000
13312	ER_GRP_RPL_WAIT_FOR_DEPENDENCIES_FAILED	HY000
13313	ER_GRP_RPL_REGISTER_TRX_TO_WAIT_FOR_SYNC_BEFORE_EXECUTION_FAILED	HY000
13314	ER_GRP_RPL_SEND_TRX_SYNC_BEFORE_EXECUTION_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13315	ER_GRP_RPL_TRX_WAIT_FOR_SYNC_BEFORE_EXECUTION_FAILED	HY000
13316	ER_GRP_RPL_RELEASE_BEGIN_TRX_AFTER_WAIT_FOR_SYNC_BEFORE_EXEC	HY000
13317	ER_GRP_RPL_TRX_WAIT_FOR_GROUP_GTID_EXECUTED	HY000
13320	ER_WARN_PROPERTY_STRING_PARSE_FAILED	HY000
13321	ER_INVALID_PROPERTY_KEY	HY000
13322	ER_GRP_RPL_GTID_SET_EXTRACT_ERROR_DURING_RECOVERY	HY000
13323	ER_SERVER_RPL_ENCRYPTION_FAILED_TO_ENCRYPT	HY000
13324	ER_CANNOT_GET_SERVER_VERSION_FROM_TABLESPACE_HEADER	HY000
13325	ER_CANNOT_SET_SERVER_VERSION_IN_TABLESPACE_HEADER	HY000
13326	ER_SERVER_UPGRADE_VERSION_NOT_SUPPORTED	HY000
13327	ER_SERVER_UPGRADE_FROM_VERSION	HY000
13328	ER_GRP_RPL_ERROR_ON_CERT_DB_INSTALL	HY000
13329	ER_GRP_RPL_FORCE_MEMBERS_WHEN_LEAVING	HY000
13330	ER_TRG_WRONG_ORDER	HY000
13332	ER_LDAP_AUTH_GRP_SEARCH_NOT_SPECIAL_HDL	HY000
13333	ER_LDAP_AUTH_GRP_USER_OBJECT_HAS_GROUP_INFO	HY000
13334	ER_LDAP_AUTH_GRP_INFO_FOUND_IN_MANY_OBJECTS	HY000
13335	ER_LDAP_AUTH_GRP_INCORRECT_ATTRIBUTE	HY000
13336	ER_LDAP_AUTH_GRP_NULL_ATTRIBUTE_VALUE	HY000
13337	ER_LDAP_AUTH_GRP_DN_PARSING_FAILED	HY000
13338	ER_LDAP_AUTH_GRP_OBJECT_HAS_USER_INFO	HY000
13339	ER_LDAP_AUTH_LDAPS	HY000
13340	ER_LDAP_MAPPING_GET_USER_PROXY	HY000
13341	ER_LDAP_MAPPING_USER_DONT_BELONG_GROUP	HY000
13342	ER_LDAP_MAPPING_INFO	HY000
13343	ER_LDAP_MAPPING_EMPTY_MAPPING	HY000
13344	ER_LDAP_MAPPING_PROCESS_MAPPING	HY000
13345	ER_LDAP_MAPPING_CHECK_DELIMI_QUOTE	HY000
13346	ER_LDAP_MAPPING_PROCESS_DELIMITER	HY000
13347	ER_LDAP_MAPPING_PROCESS_DELIMITER_EQUAL_NOT_FOUND	HY000
13348	ER_LDAP_MAPPING_PROCESS_DELIMITER_TRY_COMMA	HY000
13349	ER_LDAP_MAPPING_PROCESS_DELIMITER_COMMA_NOT_FOUND	HY000
13350	ER_LDAP_MAPPING_NO_SEPEARATOR_END_OF_GROUP	HY000
13351	ER_LDAP_MAPPING_GETTING_NEXT_MAPPING	HY000
13352	ER_LDAP_MAPPING_PARSING_CURRENT_STATE	HY000
13353	ER_LDAP_MAPPING_PARSING_MAPPING_INFO	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13354	ER_LDAP_MAPPING_PARSING_ERROR	HY000
13355	ER_LDAP_MAPPING_TRIMMING_SPACES	HY000
13356	ER_LDAP_MAPPING_IS_QUOTE	HY000
13357	ER_LDAP_MAPPING_NON_DESIRED_STATE	HY000
13358	ER_INVALID_NAMED_PIPE_FULL_ACCESS_GROUP	HY000
13359	ER_PREPARE_FOR_SECONDARY_ENGINE	HY000
13360	ER_SERVER_WARN_DEPRECATED	HY000
13361	ER_AUTH_ID_WITH_SYSTEM_USER_PRIV_IN_MANDATORY_ROLES	HY000
13362	ER_SERVER_BINLOG_MASTER_KEY_RECOVERY_OUT_OF_COMBINATION	HY000
13363	ER_SERVER_BINLOG_MASTER_KEY_ROTATION_FAIL_TO_CLEANUP_AUX_KEY	HY000
13368	ER_TURNING_ON_PARTIAL_REVOKES	HY000
13369	ER_WARN_PARTIAL_REVOKE_AND_DB_GRANT	HY000
13370	ER_WARN_INCORRECT_PRIVILEGE_FOR_DB_RESTRICTIONS	HY000
13371	ER_WARN_INVALID_DB_RESTRICTIONS	HY000
13372	ER_GRP_RPL_INVALID_COMMUNICATION_PROTOCOL	HY000
13373	ER_GRP_RPL_STARTED_AUTO_REJOIN	HY000
13374	ER_GRP_RPL_TIMEOUT_RECEIVED_VC_ON_REJOIN	HY000
13375	ER_GRP_RPL_FINISHED_AUTO_REJOIN	HY000
13376	ER_GRP_RPL_DEFAULT_TABLE_ENCRYPTION_DIFF_FROM_GRP	HY000
13377	ER_SERVER_UPGRADE_OFF	HY000
13378	ER_SERVER_UPGRADE_SKIP	HY000
13379	ER_SERVER_UPGRADE_PENDING	HY000
13380	ER_SERVER_UPGRADE_FAILED	HY000
13381	ER_SERVER_UPGRADE_STATUS	HY000
13382	ER_SERVER_UPGRADE_REPAIR_REQUIRED	HY000
13383	ER_SERVER_UPGRADE_REPAIR_STATUS	HY000
13384	ER_SERVER_UPGRADE_INFO_FILE	HY000
13385	ER_SERVER_UPGRADE_SYS_SCHEMA	HY000
13386	ER_SERVER_UPGRADE_MYSQL_TABLES	HY000
13387	ER_SERVER_UPGRADE_SYSTEM_TABLES	HY000
13388	ER_SERVER_UPGRADE_EMPTY_SYS	HY000
13389	ER_SERVER_UPGRADE_NO_SYS_VERSION	HY000
13390	ER_SERVER_UPGRADE_SYS_VERSION_EMPTY	HY000
13391	ER_SERVER_UPGRADE_SYS_SCHEMA_OUTDATED	HY000
13392	ER_SERVER_UPGRADE_SYS_SCHEMA_UP_TO_DATE	HY000
13393	ER_SERVER_UPGRADE_SYS_SCHEMA_OBJECT_COUNT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13394	ER_SERVER_UPGRADE_CHECKING_DB	HY000
13395	ER_IB_MSG_DDL_LOG_DELETE_BY_ID_TMCT	HY000
13396	ER_IB_MSG_POST_RECOVER_DDL_LOG_RECOVER	HY000
13397	ER_IB_MSG_POST_RECOVER_POST_TS_ENCRYPT	HY000
13398	ER_IB_MSG_DDL_LOG_FAIL_POST_DDL	HY000
13399	ER_SERVER_BINLOG_UNSAFE_SYSTEM_FUNCTION	HY000
13400	ER_SERVER_UPGRADE_HELP_TABLE_STATUS	HY000
13404	ER_BINLOG_UNABLE_TO_ROTATE_GTID_TABLE_READONLY	HY000
13405	ER_NETWORK_NAMESPACES_NOT_SUPPORTED	HY000
13406	ER_UNKNOWN_NETWORK_NAMESPACE	HY000
13407	ER_NETWORK_NAMESPACE_NOT_ALLOWED_FOR_WILDCARD_ADDRESS	HY000
13408	ER_SETNS_FAILED	HY000
13409	ER_WILDCARD_NOT_ALLOWED_FOR_MULTIADDRESS_BIND	HY000
13410	ER_NETWORK_NAMESPACE_FILE_PATH_TOO_LONG	HY000
13411	ER_IB_MSG_TOO_LONG_PATH	HY000
13412	ER_IB_RECV_FIRST_REC_GROUP_INVALID	HY000
13413	ER_DD_UPGRADE_COMPLETED	HY000
13415	ER_PERSIST_OPTION_USER_TRUNCATED	HY000
13416	ER_PERSIST_OPTION_HOST_TRUNCATED	HY000
13417	ER_NET_WAIT_ERROR	HY000
13418	ER_IB_MSG_1285	HY000
13419	ER_IB_MSG_CLOCK_MONOTONIC_UNSUPPORTED	HY000
13420	ER_IB_MSG_CLOCK_GETTIME_FAILED	HY000
13421	ER_PLUGIN_NOT_EARLY_DUP	HY000
13422	ER_PLUGIN_NO_INSTALL_DUP	HY000
13425	ER_BINLOG_UNSAFE_DEFAULT_EXPRESSION_IN_SUBSTATEMENT	HY000
13426	ER_GRP_RPL_MEMBER_VER_READ_COMPATIBLE	HY000
13427	ER_LOCK_ORDER_INIT_FAILED	HY000
13428	ER_AUDIT_LOG_KEYRING_ID_TIMESTAMP_VALUE_IS_INVALID	HY000
13429	ER_AUDIT_LOG_FILE_NAME_TIMESTAMP_VALUE_IS_MISSING_OR_INVALID	HY000
13430	ER_AUDIT_LOG_FILE_NAME_DOES_NOT_HAVE_REQUIRED_FORMAT	HY000
13431	ER_AUDIT_LOG_FILE_NAME_KEYRING_ID_VALUE_IS_MISSING	HY000
13432	ER_AUDIT_LOG_FILE_HAS_BEEN_SUCCESSFULLY_PROCESSED	HY000
13433	ER_AUDIT_LOG_COULD_NOT_OPEN_FILE_FOR_READING	HY000
13434	ER_AUDIT_LOG_INVALID_FILE_CONTENT	HY000
13435	ER_AUDIT_LOG_CANNOT_READ_PASSWORD	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13436	ER_AUDIT_LOG_CANNOT_STORE_PASSWORD	HY000
13437	ER_AUDIT_LOG_CANNOT_REMOVE_PASSWORD	HY000
13438	ER_AUDIT_LOG_PASSWORD_HAS_BEEN_COPIED	HY000
13447	ER_LDAP_EMPTY_USERDN_PASSWORD	HY000
13449	ER_ACL_WRONG_OR_MISSING_ACL_TABLES_LOG	HY000
13450	ER_LOCK_ORDER_FAILED_WRITE_FILE	HY000
13451	ER_LOCK_ORDER_FAILED_READ_FILE	HY000
13452	ER_LOCK_ORDER_MESSAGE	HY000
13453	ER_LOCK_ORDER_DEPENDENCIES_SYNTAX	HY000
13454	ER_LOCK_ORDER_SCANNER_SYNTAX	HY000
13455	ER_DATA_DIRECTORY_UNUSABLE_DELETABLE	HY000
13456	ER_IB_MSG_BTREE_LEVEL_LIMIT_EXCEEDED	HY000
13457	ER_IB_CLONE_START_STOP	HY000
13458	ER_IB_CLONE_OPERATION	HY000
13459	ER_IB_CLONE_RESTART	HY000
13460	ER_IB_CLONE_USER_DATA	HY000
13461	ER_IB_CLONE_NON_INNODB_TABLE	HY000
13462	ER_CLONE_SHUTDOWN_TRACE	HY000
13463	ER_GRP_RPL_GTID_PURGED_EXTRACT_ERROR	HY000
13464	ER_GRP_RPL_CLONE_PROCESS_PREPARE_ERROR	HY000
13465	ER_GRP_RPL_CLONE_PROCESS_EXEC_ERROR	HY000
13466	ER_GRP_RPL_RECOVERY_EVAL_ERROR	HY000
13467	ER_GRP_RPL_NO_POSSIBLE_RECOVERY	HY000
13468	ER_GRP_RPL_CANT_KILL_THREAD	HY000
13469	ER_GRP_RPL_RECOVERY_STRAT_CLONE_THRESHOLD	HY000
13470	ER_GRP_RPL_RECOVERY_STRAT_CLONE_PURGED	HY000
13471	ER_GRP_RPL_RECOVERY_STRAT_CHOICE	HY000
13472	ER_GRP_RPL_RECOVERY_STRAT_FALLBACK	HY000
13473	ER_GRP_RPL_RECOVERY_STRAT_NO_FALLBACK	HY000
13474	ER_GRP_RPL_REPLICA_THREAD_ERROR_ON_CLONE	HY000
13475	ER_UNKNOWN_TABLE_IN_UPGRADE	HY000
13476	ER_IDENT_CAUSES_TOO_LONG_PATH_IN_UPGRADE	HY000
13477	ER_XA_CANT_CREATE_MDL_BACKUP	HY000
13478	ER_AUDIT_LOG_SUPER_PRIVILEGE_REQUIRED	HY000
13479	ER_AUDIT_LOG_UDF_INVALID_ARGUMENT_TYPE	HY000
13480	ER_AUDIT_LOG_UDF_INVALID_ARGUMENT_COUNT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13481	ER_AUDIT_LOG_HAS_NOT_BEEN_INSTALLED	HY000
13482	ER_AUDIT_LOG_UDF_READ_INVALID_MAX_ARRAY_LENGTH_ARG_TYPE	HY000
13483	ER_LOG_CANNOT_WRITE_EXTENDED	HY000
13485	ER_KEYRING_AWS_INCORRECT_PROXY	HY000
13486	ER_GRP_RPL_SERVER_SET_TO_OFFLINE_MODE_DUE_TO_ERRORS	HY000
13487	ER_GRP_RPL_MESSAGE_SERVICE_FATAL_ERROR	HY000
13488	ER_WARN_WRONG_COMPRESSION_ALGORITHM_LOG	HY000
13489	ER_WARN_WRONG_COMPRESSION_LEVEL_LOG	HY000
13490	ER_PROTOCOL_COMPRESSION_RESET_LOG	HY000
13491	ER_XPLUGIN_COMPRESSION_ERROR	HY000
13492	ER_MYSQLBACKUP_MSG	HY000
13493	ER_WARN_UNKNOWN_KEYRING_AWS_REGION	HY000
13494	ER_WARN_LOG_PRIVILEGE_CHECKS_USER_DOES_NOT_EXIST	HY000
13495	ER_WARN_LOG_PRIVILEGE_CHECKS_USER_CORRUPT	HY000
13496	ER_WARN_LOG_PRIVILEGE_CHECKS_USER_NEEDS_RPL_APPLIER_PRIV	HY000
13497	ER_OBSOLETE_FILE_PRIVILEGE_FOR_REPLICATION_CHECKS	HY000
13498	ER_RPL_REPLICA_SQL_THREAD_STARTING_WITH_PRIVILEGE_CHECKS	HY000
13499	ER_AUDIT_LOG_CANNOT_GENERATE_PASSWORD	HY000
13500	ER_INIT_FAILED_TO_GENERATE_ROOT_PASSWORD	HY000
13501	ER_PLUGIN_LOAD_OPTIONS_IGNORED	HY000
13502	ER_WARN_AUTH_ID_WITH_SYSTEM_USER_PRIV_IN_MANDATORY_ROLES	HY000
13503	ER_IB_MSG_SKIP_HIDDEN_DIR	HY000
13504	ER_WARN_RPL_RECOVERY_NO_ROTATE_EVENT_FROM_SOURCE_EOF	HY000
13505	ER_IB_LOB_ROLLBACK_INDEX_LEN	HY000
13506	ER_CANT_PROCESS_EXPRESSION_FOR_GENERATED_COLUMN_TO_DD	HY000
13507	ER_RPL_REPLICA_QUEUE_EVENT_FAILED_INVALID_NON_ROW_FORMAT	HY000
13508	ER_OBSOLETE_REQUIRE_ROW_FORMAT_VIOLATION	HY000
13509	ER_LOG_PRIV_CHECKS_REQUIRE_ROW_FORMAT_NOT_SET	HY000
13510	ER_RPL_REPLICA_SQL_THREAD_DETECTED_UNEXPECTED_EVENT_SEQUENCE	HY000
13511	ER_IB_MSG_UPGRADE_PARTITION_FILE	HY000
13512	ER_IB_MSG_DOWNGRADE_PARTITION_FILE	HY000
13513	ER_IB_MSG_UPGRADE_PARTITION_FILE_IMPORT	HY000
13514	ER_IB_WARN_OPEN_PARTITION_FILE	HY000
13515	ER_IB_MSG_FIL_STATE_MOVED_CORRECTED	HY000
13516	ER_IB_MSG_FIL_STATE_MOVED_CHANGED_PATH	HY000
13517	ER_IB_MSG_FIL_STATE_MOVED_CHANGED_NAME	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13518	ER_IB_MSG_FIL_STATE_MOVED_TOO_MANY	HY000
13519	ER_GR_ELECTED_PRIMARY_GTID_INFORMATION	HY000
13520	ER_SCHEMA_NAME_IN_UPPER_CASE_NOT_ALLOWED	HY000
13521	ER_TABLE_NAME_IN_UPPER_CASE_NOT_ALLOWED	HY000
13522	ER_SCHEMA_NAME_IN_UPPER_CASE_NOT_ALLOWED_FOR_FK	HY000
13523	ER_TABLE_NAME_IN_UPPER_CASE_NOT_ALLOWED_FOR_FK	HY000
13524	ER_IB_MSG_DICT_PARTITION_NOT_FOUND	HY000
13525	ER_ACCESS_DENIED_FOR_USER_ACCOUNT_BLOCKED_BY_PASSWORD_LOCK	HY000
13526	ER_INNODB_OUT_OF_RESOURCES	HY000
13528	ER_MIGRATE_TABLE_TO_DD_OOM	HY000
13529	ER_RPL_RELAY_LOG_RECOVERY_INFO_AFTER_CLONE	HY000
13530	ER_IB_MSG_57_UNDO_SPACE_DELETE_FAIL	HY000
13531	ER_IB_MSG_DBLWR_1285	HY000
13532	ER_IB_MSG_DBLWR_1286	HY000
13533	ER_IB_MSG_DBLWR_1287	HY000
13534	ER_IB_MSG_DBLWR_1288	HY000
13535	ER_IB_MSG_DBLWR_1290	HY000
13536	ER_IB_MSG_BAD_DBLWR_FILE_NAME	HY000
13538	ER_IB_MSG_DBLWR_1293	HY000
13539	ER_IB_MSG_DBLWR_1294	HY000
13540	ER_IB_MSG_DBLWR_1295	HY000
13541	ER_IB_MSG_DBLWR_1296	HY000
13542	ER_IB_MSG_DBLWR_1297	HY000
13543	ER_IB_MSG_DBLWR_1298	HY000
13544	ER_IB_MSG_DBLWR_1300	HY000
13545	ER_IB_MSG_DBLWR_1301	HY000
13546	ER_IB_MSG_DBLWR_1304	HY000
13547	ER_IB_MSG_DBLWR_1305	HY000
13548	ER_IB_MSG_DBLWR_1306	HY000
13549	ER_IB_MSG_DBLWR_1307	HY000
13550	ER_IB_MSG_DBLWR_1308	HY000
13551	ER_IB_MSG_DBLWR_1309	HY000
13552	ER_IB_MSG_DBLWR_1310	HY000
13553	ER_IB_MSG_DBLWR_1311	HY000
13554	ER_IB_MSG_DBLWR_1312	HY000
13555	ER_IB_MSG_DBLWR_1313	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13556	ER_IB_MSG_DBLWR_1314	HY000
13557	ER_IB_MSG_DBLWR_1315	HY000
13558	ER_IB_MSG_DBLWR_1316	HY000
13559	ER_IB_MSG_DBLWR_1317	HY000
13560	ER_IB_MSG_DBLWR_1318	HY000
13561	ER_IB_MSG_DBLWR_LOAD_WRONG_SIZE	HY000
13562	ER_IB_MSG_DBLWR_1320	HY000
13563	ER_IB_MSG_DBLWR_1321	HY000
13564	ER_IB_MSG_DBLWR_OPEN_OR_CREATE_WRONG_SIZE	HY000
13565	ER_IB_MSG_DBLWR_1323	HY000
13566	ER_IB_MSG_DBLWR_1324	HY000
13567	ER_IB_MSG_DBLWR_1325	HY000
13568	ER_IB_MSG_DBLWR_1326	HY000
13569	ER_IB_MSG_DBLWR_1327	HY000
13570	ER_IB_MSG_GTID_FLUSH_AT_SHUTDOWN	HY000
13571	ER_IB_MSG_57_STAT_SPACE_DELETE_FAIL	HY000
13572	ER_NDBINFO_UPGRADING_SCHEMA	HY000
13573	ER_NDBINFO_NOT_UPGRADING_SCHEMA	HY000
13574	ER_NDBINFO_UPGRADING_SCHEMA_FAIL	HY000
13576	ER_IB_MSG_INNODB_START_INITIALIZE	HY000
13577	ER_IB_MSG_INNODB_END_INITIALIZE	HY000
13578	ER_IB_MSG_PAGE_ARCH_NO_RESET_POINTS	HY000
13579	ER_IB_WRN_PAGE_ARCH_FLUSH_DATA	HY000
13580	ER_IB_ERR_PAGE_ARCH_INVALID_DOUBLE_WRITE_BUF	HY000
13581	ER_IB_ERR_PAGE_ARCH_RECOVERY_FAILED	HY000
13582	ER_IB_ERR_PAGE_ARCH_INVALID_FORMAT	HY000
13583	ER_INVALID_XPLUGIN_SOCKET_SAME_AS_SERVER	HY000
13584	ER_INNODB_UNABLE_TO_ACQUIRE_DD_OBJECT	HY000
13586	ER_IB_MSG_UNDO_TRUNCATE_TOO_OFTEN	HY000
13587	ER_GRP_RPL_IS_STARTING	HY000
13588	ER_IB_MSG_INVALID_LOCATION_FOR_TABLESPACE	HY000
13589	ER_IB_MSG_INVALID_LOCATION_WRONG_DB	HY000
13590	ER_IB_MSG_CANNOT_FIND_DD_UNDO_SPACE	HY000
13591	ER_GRP_RPL_RECOVERY_ENDPOINT_FORMAT	HY000
13592	ER_GRP_RPL_RECOVERY_ENDPOINT_INVALID	HY000
13593	ER_GRP_RPL_RECOVERY_ENDPOINT_INVALID_DONOR_ENDPOINT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13594	ER_GRP_RPL_RECOVERY_ENDPOINT_INTERFACES_IPS	HY000
13595	ER_WARN_TLS_CHANNEL_INITIALIZATION_ERROR	HY000
13596	ER_XPLUGIN_FAILED_TO_VALIDATE_ADDRESS	HY000
13597	ER_XPLUGIN_FAILED_TO_BIND_INTERFACE_ADDRESS	HY000
13598	ER_IB_ERR_RECOVERY_REDO_DISABLED	HY000
13599	ER_IB_WRN_FAST_SHUTDOWN_REDO_DISABLED	HY000
13600	ER_IB_WRN_REDO_DISABLED	HY000
13601	ER_IB_WRN_REDO_ENABLED	HY000
13602	ER_TLS_CONFIGURED_FOR_CHANNEL	HY000
13603	ER_TLS_CONFIGURATION_REUSED	HY000
13604	ER_IB_TABLESPACE_PATH_VALIDATION_SKIPPED	HY000
13605	ER_IB_CANNOT_UPGRADE_WITH_DISCARDED_TABLESPACES	HY000
13606	ER_USERNAME_TRUNKATED	HY000
13607	ER_HOSTNAME_TRUNKATED	HY000
13608	ER_IB_MSG_TRX_RECOVERY_ROLLBACK_NOT_COMPLETED	HY000
13609	ER_AUTHCACHE_ROLE_EDGES_IGNORED_EMPTY_NAME	HY000
13610	ER_AUTHCACHE_ROLE_EDGES_UNKNOWN_AUTHORIZATION_ID	HY000
13611	ER_AUTHCACHE_DEFAULT_ROLES_IGNORED_EMPTY_NAME	HY000
13612	ER_AUTHCACHE_DEFAULT_ROLES_UNKNOWN_AUTHORIZATION_ID	HY000
13613	ER_IB_ERR_DDL_LOG_INSERT_FAILURE	HY000
13614	ER_IB_LOCK_VALIDATE_LATCH_ORDER_VIOLATION	HY000
13615	ER_IB_RELOCK_LATCH_ORDER_VIOLATION	HY000
13621	ER_IB_MSG_1357	HY000
13622	ER_IB_MSG_1358	HY000
13623	ER_IB_MSG_1359	HY000
13624	ER_IB_FAILED_TO_DELETE_TABLESPACE_FILE	HY000
13625	ER_IB_UNABLE_TO_EXPAND_TEMPORARY_TABLESPACE_POOL	HY000
13626	ER_IB_TMP_TABLESPACE_CANNOT_CREATE_DIRECTORY	HY000
13627	ER_IB_MSG_SCANNING_TEMP_TABLESPACE_DIR	HY000
13628	ER_IB_ERR_TEMP_TABLESPACE_DIR_DOESNT_EXIST	HY000
13629	ER_IB_ERR_TEMP_TABLESPACE_DIR_EMPTY	HY000
13630	ER_IB_ERR_TEMP_TABLESPACE_DIR_CONTAINS_SEMICOLON	HY000
13631	ER_IB_ERR_TEMP_TABLESPACE_DIR_SUBDIR_OF_DATADIR	HY000
13632	ER_IB_ERR_SCHED_SETAFFNINITY_FAILED	HY000
13633	ER_IB_ERR_UNKNOWN_PAGE_FETCH_MODE	HY000
13634	ER_IB_ERR_LOG_PARSING_BUFFER_OVERFLOW	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13635	ER_IB_ERR_NOT_ENOUGH_MEMORY_FOR_PARSE_BUFFER	HY000
13636	ER_IB_MSG_1372	HY000
13637	ER_IB_MSG_1373	HY000
13638	ER_IB_MSG_1374	HY000
13639	ER_IB_MSG_1375	HY000
13640	ER_IB_ERR_ZLIB_UNCOMPRESS_FAILED	HY000
13641	ER_IB_ERR_ZLIB_BUF_ERROR	HY000
13642	ER_IB_ERR_ZLIB_MEM_ERROR	HY000
13643	ER_IB_ERR_ZLIB_DATA_ERROR	HY000
13644	ER_IB_ERR_ZLIB_UNKNOWN_ERROR	HY000
13645	ER_IB_MSG_1381	HY000
13646	ER_IB_ERR_INDEX_RECORDS_WRONG_ORDER	HY000
13647	ER_IB_ERR_INDEX_DUPLICATE_KEY	HY000
13648	ER_IB_ERR_FOUND_N_DUPLICATE_KEYS	HY000
13649	ER_IB_ERR_FOUND_N_RECORDS_WRONG_ORDER	HY000
13650	ER_IB_ERR_PARALLEL_READ_OOM	HY000
13651	ER_IB_MSG_UNDO_MARKED_ACTIVE	HY000
13652	ER_IB_MSG_UNDO_ALTERED_ACTIVE	HY000
13653	ER_IB_MSG_UNDO_ALTERED_INACTIVE	HY000
13654	ER_IB_MSG_UNDO_MARKED_EMPTY	HY000
13655	ER_IB_MSG_UNDO_TRUNCATE_DELAY_BY_CLONE	HY000
13656	ER_IB_MSG_UNDO_TRUNCATE_DELAY_BY_MDL	HY000
13657	ER_IB_MSG_INJECT_CRASH	HY000
13658	ER_IB_MSG_INJECT_FAILURE	HY000
13659	ER_GRP_RPL_TIMEOUT_RECEIVED_VC_LEAVE_ON_REJOIN	HY000
13660	ER_RPL_ASYNC_RECONNECT_FAIL_NO_SOURCE	HY000
13661	ER_UDF_REGISTER_SERVICE_ERROR	HY000
13662	ER_UDF_REGISTER_ERROR	HY000
13663	ER_UDF_UNREGISTER_ERROR	HY000
13664	ER_EMPTY_PRIVILEGE_NAME_IGNORED	HY000
13665	ER_IB_MSG_INCORRECT_SIZE	HY000
13666	ER_TMPDIR_PATH_TOO_LONG	HY000
13667	ER_ERROR_LOG_DESTINATION_NOT_A_FILE	HY000
13668	ER_NO_ERROR_LOG_PARSER_CONFIGURED	HY000
13669	ER_UPGRADE_NONEXISTENT_SCHEMA	HY000
13670	ER_IB_MSG_CREATED_UNDO_SPACE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13671	ER_IB_MSG_DROPPED_UNDO_SPACE	HY000
13672	ER_IB_MSG_MASTER_KEY_ROTATED	HY000
13673	ER_IB_DBLWR_DECOMPRESS_FAILED	HY000
13674	ER_IB_DBLWR_DECRYPT_FAILED	HY000
13675	ER_IB_DBLWR_KEY_MISSING	HY000
13676	ER_INNODB_IO_WRITE_ERROR_RETRYING	HY000
13677	ER_INNODB_IO_WRITE_FAILED	HY000
13678	ER_LOG_COMPONENT_CANNOT_INIT	HY000
13679	ER_RPL_ASYNC_CHANNEL_CANT_CONNECT	HY000
13680	ER_RPL_ASYNC_SENDER_ADDED	HY000
13681	ER_RPL_ASYNC_SENDER_REMOVED	HY000
13682	ER_RPL_ASYNC_CHANNEL_STOPPED_QUORUM_LOST	HY000
13683	ER_RPL_ASYNC_CHANNEL_CANT_CONNECT_NO_QUORUM	HY000
13685	ER_RPL_REPLICA_MONITOR_IO_THREAD_EXITING	HY000
13686	ER_RPL_ASYNC_MANAGED_NAME_REMOVED	HY000
13687	ER_RPL_ASYNC_MANAGED_NAME_ADDED	HY000
13688	ER_RPL_ASYNC_READ_FAILOVER_TABLE	HY000
13689	ER_RPL_REPLICA_MONITOR_IO_THREAD_RECONNECT_CHANNEL	HY000
13690	ER_REPLICA_ANON_TO_GTID_IS_LOCAL_OR_UUID_AND_GTID_MODE_NOT_ON	HY000
13691	ER_REPLICA_ANONYMOUS_TO_GTID_UUID_SAME_AS_GROUP_NAME	HY000
13692	ER_GRP_RPL_GRP_NAME_IS_SAME_AS_ANONYMOUS_TO_GTID_UUID	HY000
13693	ER_WARN_GTID_THRESHOLD_BREACH	HY000
13694	ER_HEALTH_INFO	HY000
13695	ER_HEALTH_WARNING	HY000
13696	ER_HEALTH_ERROR	HY000
13697	ER_HEALTH_WARNING_DISK_USAGE_LEVEL_1	HY000
13698	ER_HEALTH_WARNING_DISK_USAGE_LEVEL_2	HY000
13699	ER_HEALTH_WARNING_DISK_USAGE_LEVEL_3	HY000
13700	ER_IB_INNODB_TBSP_OUT_OF_SPACE	HY000
13701	ER_GRP_RPL_APPLIER_CHANNEL_STILL_RUNNING	HY000
13702	ER_RPL_ASYNC_RECONNECT_GTID_MODE_OFF_CHANNEL	HY000
13703	ER_FIREWALL_SERVICES_NOT_ACQUIRED	HY000
13704	ER_FIREWALL_UDF_REGISTER_FAILED	HY000
13705	ER_FIREWALL_PFS_TABLE_REGISTER_FAILED	HY000
13706	ER_IB_MSG_STATS_SAMPLING_TOO_LARGE	HY000
13707	ER_AUDIT_LOG_FILE_PRUNE_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13708	ER_AUDIT_LOG_FILE_AUTO_PRUNED	HY000
13709	ER_COMPONENTS_INFRASTRUCTURE_MANIFEST_INIT	HY000
13710	ER_COMPONENTS_INFRASTRUCTURE_MANIFEST_DEINIT	HY000
13711	ER_WARN_COMPONENTS_INFRASTRUCTURE_MANIFEST_NOT_RO	HY000
13712	ER_WARN_NO_KEYRING_COMPONENT_SERVICE_FOUND	HY000
13713	ER_NOTE_KEYRING_COMPONENT_INITIALIZED	HY000
13714	ER_KEYRING_COMPONENT_NOT_INITIALIZED	HY000
13715	ER_KEYRING_COMPONENT_EXCEPTION	HY000
13716	ER_KEYRING_COMPONENT_MEMORY_ALLOCATION_ERROR	HY000
13717	ER_NOTE_KEYRING_COMPONENT_AES_INVALID_MODE_BLOCK_SIZE	HY000
13718	ER_NOTE_KEYRING_COMPONENT_AES_DATA_IDENTIFIER_EMPTY	HY000
13719	ER_NOTE_KEYRING_COMPONENT_AES_INVALID_KEY	HY000
13720	ER_NOTE_KEYRING_COMPONENT_AES_OPERATION_ERROR	HY000
13721	ER_NOTE_KEYRING_COMPONENT_READ_DATA_NOT_FOUND	HY000
13722	ER_NOTE_KEYRING_COMPONENT_WRITE_MAXIMUM_DATA_LENGTH	HY000
13723	ER_NOTE_KEYRING_COMPONENT_STORE_FAILED	HY000
13724	ER_NOTE_KEYRING_COMPONENT_REMOVE_FAILED	HY000
13725	ER_NOTE_KEYRING_COMPONENT_GENERATE_FAILED	HY000
13726	ER_NOTE_KEYRING_COMPONENT_KEYS_METADATA_ITERATOR_FETCH_FAILED	HY000
13727	ER_NOTE_KEYRING_COMPONENT_METADATA_ITERATOR_INVALID_OUT_PARAM	HY000
13728	ER_IB_WRN_FAILED_TO_ACQUIRE_SERVICE	HY000
13729	ER_IB_WRN_OLD_GEOMETRY_TYPE	HY000
13730	ER_NET_WAIT_ERROR2	HY000
13731	ER_GRP_RPL_MEMBER_ACTION_TRIGGERED	HY000
13732	ER_GRP_RPL_MEMBER_ACTION_FAILURE_IGNORE	HY000
13733	ER_GRP_RPL_MEMBER_ACTION_FAILURE	HY000
13734	ER_GRP_RPL_MEMBER_ACTION_PARSE_ON_RECEIVE	HY000
13735	ER_GRP_RPL_MEMBER_ACTION_UPDATE_ACTIONS	HY000
13736	ER_GRP_RPL_MEMBER_ACTION_GET_EXCHANGEABLE_DATA	HY000
13737	ER_GRP_RPL_MEMBER_ACTION_DEFAULT_CONFIGURATION	HY000
13738	ER_GRP_RPL_MEMBER_ACTION_UNABLE_TO_SET_DEFAULT_CONFIGURATION	HY000
13739	ER_GRP_RPL_MEMBER_ACTION_PARSE_ON_MEMBER_JOIN	HY000
13740	ER_GRP_RPL_MEMBER_ACTION_UPDATE_ACTIONS_ON_MEMBER_JOIN	HY000
13741	ER_GRP_RPL_MEMBER_ACTION_INVALID_ACTIONS_ON_MEMBER_JOIN	HY000
13742	ER_GRP_RPL_MEMBER_ACTION_ENABLED	HY000
13743	ER_GRP_RPL_MEMBER_ACTION_DISABLED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13744	ER_GRP_RPL_MEMBER_ACTIONS_RESET	HY000
13747	ER_FIREWALL_DEPRECATED_USER_PROFILE	HY000
13748	ER_GRP_RPL_VIEW_CHANGE_UUID_INVALID	HY000
13749	ER_GRP_RPL_VIEW_CHANGE_UUID_SAME_AS_GROUP_NAME	HY000
13750	ER_GRP_RPL_GROUP_NAME_SAME_AS_VIEW_CHANGE_UUID	HY000
13751	ER_GRP_RPL_VIEW_CHANGE_UUID_IS_SAME_AS_ANONYMOUS_TO_GTID_UUID	HY000
13752	ER_GRP_RPL_GRP_VIEW_CHANGE_UUID_IS_INCOMPATIBLE_WITH_SERVER_UUID	HY000
13753	ER_GRP_RPL_VIEW_CHANGE_UUID_DIFF_FROM_GRP	HY000
13754	ER_WARN_REPLICA_ANONYMOUS_TO_GTID_UUID_SAME_AS_VIEW_CHANGE_UUID	HY000
13755	ER_GRP_RPL_FAILED_TO_PARSE_THE_VIEW_CHANGE_UUID	HY000
13756	ER_GRP_RPL_FAILED_TO_GENERATE_SIDNO_FOR_VIEW_CHANGE_UUID	HY000
13757	ER_GRP_RPL_VIEW_CHANGE_UUID_PARSE_ERROR	HY000
13758	ER_GRP_RPL_UPDATE_GRPGTID_VIEW_CHANGE_UUID_EXECUTED_ERROR	HY000
13759	ER_GRP_RPL_ADD_VIEW_CHANGE_UUID_TO_GRP_SID_MAP_ERROR	HY000
13760	ER_GRP_RPL_DONOR_VIEW_CHANGE_UUID_TRANS_INFO_ERROR	HY000
13761	ER_WARN_GRP_RPL_VIEW_CHANGE_UUID_FAIL_GET_VARIABLE	HY000
13762	ER_WARN_ADUIT_LOG_MAX_SIZE_AND_PRUNE_SECONDS_LOG	HY000
13763	ER_WARN_ADUIT_LOG_MAX_SIZE_CLOSE_TO_ROTATE_ON_SIZE_LOG	HY000
13764	ER_PLUGIN_INVALID_TABLE_DEFINITION	HY000
13765	ER_AUTH_KERBEROS_LOGGER_GENERIC_MSG	HY000
13766	ER_INSTALL_PLUGIN_CONFLICT_LOG	HY000
13767	ER_DEPRECATED_PERSISTED_VARIABLE_WITH_ALIAS	HY000
13768	ER_LOG_COMPONENT_FLUSH_FAILED	HY000
13769	ER_IB_MSG_REENCRYPTED_TABLESPACE_KEY	HY000
13770	ER_IB_MSG_REENCRYPTED_GENERAL_TABLESPACE_KEY	HY000
13771	ER_IB_ERR_PAGE_ARCH_DBLWR_INIT_FAILED	HY000
13772	ER_IB_MSG_RECOVERY_NO_SPACE_IN_REDO_LOG__SKIP_IBUF_MERGES	HY000
13773	ER_IB_MSG_RECOVERY_NO_SPACE_IN_REDO_LOG__UNEXPECTED	HY000
13774	ER_WARN_AUDIT_LOG_FORMAT_UNIX_TIMESTAMP_ONLY_WHEN_JSON_LOG	HY000
13775	ER_PREPARE_FOR_PRIMARY_ENGINE	HY000
13776	ER_IB_MSG_PAR_RSEG_INIT_COMPLETE_MSG	HY000
13777	ER_IB_MSG_PAR_RSEG_INIT_TIME_MSG	HY000
13778	ER_DDL_MSG_1	HY000
13779	ER_MTR_MSG_1	HY000
13780	ER_GRP_RPL_MYSQL_NETWORK_PROVIDER_CLIENT_ERROR_CONN_ERR	HY000
13781	ER_GRP_RPL_MYSQL_NETWORK_PROVIDER_CLIENT_ERROR_COMMAND_ERR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13782	ER_GRP_RPL_FAILOVER_CONF_GET_EXCHANGEABLE_DATA	HY000
13783	ER_GRP_RPL_FAILOVER_CONF_DEFAULT_CONFIGURATION	HY000
13784	ER_GRP_RPL_FAILOVER_CONF_UNABLE_TO_SET_DEFAULT_CONFIGURATION	HY000
13785	ER_GRP_RPL_FAILOVER_CONF_PARSE_ON_MEMBER_JOIN	HY000
13786	ER_GRP_RPL_FAILOVER_CONF_CHANNEL_DOES_NOT_EXIST	HY000
13787	ER_GRP_RPL_FAILOVER_REGISTER_MESSAGE_LISTENER_SERVICE	HY000
13788	ER_GRP_RPL_FAILOVER_PRIMARY_WITHOUT_MAJORITY	HY000
13789	ER_GRP_RPL_FAILOVER_PRIMARY_BACK_TO_MAJORITY	HY000
13790	ER_RPL_INCREMENTING_MEMBER_ACTION_VERSION	HY000
13791	ER_GRP_RPL_REPLICA_THREAD_ERROR_ON_SECONDARY_MEMBER	HY000
13792	ER_IB_MSG_CLONE_DDL_NTFN	HY000
13793	ER_IB_MSG_CLONE_DDL_APPLY	HY000
13794	ER_IB_MSG_CLONE_DDL_INVALIDATE	HY000
13795	ER_IB_MSG_UNDO_ENCRYPTION_INFO_LOADED	HY000
13796	ER_IB_WRN_ENCRYPTION_INFO_SIZE_MISMATCH	HY000
13797	ER_INVALID_AUTHENTICATION_POLICY	HY000
13798	ER_AUTHENTICATION_PLUGIN_REGISTRATION_FAILED	HY000
13799	ER_AUTHENTICATION_PLUGIN_REGISTRATION_INSUFFICIENT_BUFFER	HY000
13800	ER_AUTHENTICATION_PLUGIN_AUTH_DATA_CORRUPT	HY000
13801	ER_AUTHENTICATION_PLUGIN_SIGNATURE_CORRUPT	HY000
13802	ER_AUTHENTICATION_PLUGIN_VERIFY_SIGNATURE_FAILED	HY000
13803	ER_AUTHENTICATION_PLUGIN_OOM	HY000
13804	ER_AUTHENTICATION_PLUGIN_LOG	HY000
13805	ER_WARN_REPLICA_GTID_ONLY_AND_GTID_MODE_NOT_ON	HY000
13806	ER_WARN_L_DISABLE_GTID_ONLY_WITH_SOURCE_AUTO_POS_INVALID_POS	HY000
13807	ER_RPL_CANNOT_OPEN_RELAY_LOG	HY000
13808	ER_AUTHENTICATION_OCI_PLUGIN_NOT_INITIALIZED	HY000
13809	ER_AUTHENTICATION_OCI_PRIVATE_KEY_ERROR	HY000
13810	ER_AUTHENTICATION_OCI_DOWNLOAD_PUBLIC_KEY	HY000
13811	ER_AUTHENTICATION_OCI_IMDS	HY000
13812	ER_AUTHENTICATION_OCI_IAM	HY000
13813	ER_AUTHENTICATION_OCI_INVALID_AUTHENTICATION_STRING	HY000
13814	ER_AUTHENTICATION_OCI_NO_MATCHING_GROUPS	HY000
13815	ER_AUTHENTICATION_OCI_NO_GROUPS_FOUND	HY000
13816	ER_AUTHENTICATION_OCI_NONCE	HY000
13817	ER_HEALTH_WARNING_MEMORY_USAGE_LEVEL_1	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13818	ER_HEALTH_WARNING_MEMORY_USAGE_LEVEL_2	HY000
13819	ER_HEALTH_WARNING_MEMORY_USAGE_LEVEL_3	HY000
13820	ER_GRP_RPL_SET_SINGLE_CONSENSUS_LEADER	HY000
13821	ER_GRP_RPL_ERROR_SET_SINGLE_CONSENSUS_LEADER	HY000
13822	ER_GRP_RPL_SET_MULTI_CONSENSUS_LEADER	HY000
13823	ER_GRP_RPL_ERROR_SET_MULTI_CONSENSUS_LEADER	HY000
13824	ER_GRP_RPL_PAXOS_SINGLE_LEADER_DIFF_FROM_GRP	HY000
13825	ER_MFA_USER_ATTRIBUTES_CORRUPT	HY000
13826	ER_MFA_PLUGIN_NOT_LOADED	HY000
13827	ER_WARN_DEPRECATED_CHARSET_OPTION	HY000
13828	ER_WARN_DEPRECATED_COLLATION_OPTION	HY000
13829	ER_REGEXP_MISSING_ICU_DATADIR	HY000
13830	ER_IB_WARN_MANY_NON_LRU_FILES_OPENED	HY000
13831	ER_IB_MSG_TRYING_TO_OPEN_FILE_FOR_LONG_TIME	HY000
13832	ER_GLOBAL_CONN_LIMIT	HY000
13833	ER_CONN_LIMIT	HY000
13834	ER_WARN_AUDIT_LOG_DISABLED	HY000
13835	ER_INVALID_TLS_VERSION	HY000
13836	ER_RPL_RELAY_LOG_RECOVERY_GTID_ONLY	HY000
13837	ER_KEYRING_OKV_STANDBY_SERVER_COUNT_EXCEEDED	HY000
13838	ER_WARN_MIGRATION_EMPTY_SOURCE_KEYRING	HY000
13839	ER_WARN_CANNOT_PERSIST_SENSITIVE_VARIABLES	HY000
13840	ER_CANNOT_INTERPRET_PERSISTED_SENSITIVE_VARIABLES	HY000
13841	ER_PERSISTED_VARIABLES_KEYRING_SUPPORT_REQUIRED	HY000
13842	ER_PERSISTED_VARIABLES_MASTER_KEY_NOT_FOUND	HY000
13843	ER_PERSISTED_VARIABLES_MASTER_KEY_CANNOT_BE_GENERATED	HY000
13844	ER_PERSISTED_VARIABLES_ENCRYPTION_FAILED	HY000
13845	ER_PERSISTED_VARIABLES_DECRYPTION_FAILED	HY000
13846	ER_PERSISTED_VARIABLES_LACK_KEYRING_SUPPORT	HY000
13847	ER_MY_MALLOC_USING_JEMALLOC	HY000
13848	ER_MY_MALLOC_USING_STD_MALLOC	HY000
13849	ER_MY_MALLOC_LOADLIBRARY_FAILED	HY000
13850	ER_MY_MALLOC_GETPROCADDRESS_FAILED	HY000
13851	ER_ACCOUNT_WITH_EXPIRED_PASSWORD	HY000
13852	ER_THREAD_POOL_PLUGIN_STARTED	HY000
13853	ER_THREAD_POOL_DEDICATED_LISTENERS_INVALID	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13854	ER_IB_DBLWR_BYTES_INFO	HY000
13855	ER_IB_RDBLWR_BYTES_INFO	HY000
13856	ER_IB_MSG_LOG_FILE_IS_EMPTY	HY000
13857	ER_IB_MSG_LOG_FILE_TOO_SMALL	HY000
13858	ER_IB_MSG_LOG_FILE_TOO_BIG	HY000
13859	ER_IB_MSG_LOG_FILE_HEADER_READ_FAILED	HY000
13860	ER_IB_MSG_LOG_INIT_DIR_NOT_EMPTY_WONT_INITIALIZE	HY000
13861	ER_IB_MSG_LOG_INIT_DIR_LIST_FAILED	HY000
13862	ER_IB_MSG_LOG_INIT_DIR_MISSING_SUBDIR	HY000
13863	ER_IB_MSG_LOG_FILES_CREATED_BY_CLONE_AND_READ_ONLY_MODE	HY000
13864	ER_IB_MSG_LOG_WRITER_WRITE_FAILED	HY000
13865	ER_IB_MSG_LOG_WRITER_WAIT_ON_NEW_LOG_FILE	HY000
13866	ER_IB_MSG_RECOVERY_CHECKPOINT_OUTSIDE_LOG_FILE	HY000
13867	ER_IB_MSG_LOG_WRITER_ENTERED_EXTRA_MARGIN	HY000
13868	ER_IB_MSG_LOG_WRITER_EXITED_EXTRA_MARGIN	HY000
13869	ER_IB_MSG_LOG_PARAMS_FILE_SIZE_UNUSED	HY000
13870	ER_IB_MSG_LOG_PARAMS_N_FILES_UNUSED	HY000
13871	ER_IB_MSG_LOG_UPGRADE_FORCED_RECV	HY000
13872	ER_IB_MSG_LOG_UPGRADE_IN_READ_ONLY_MODE	HY000
13873	ER_IB_MSG_LOG_UPGRADE_CLONED_DB	HY000
13874	ER_IB_MSG_LOG_UPGRADE_UNINITIALIZED_FILES	HY000
13875	ER_IB_MSG_LOG_UPGRADE_CORRUPTION__UNEXPECTED	HY000
13879	ER_IB_MSG_LOG_FILE_FOREIGN_UUID	HY000
13880	ER_IB_MSG_LOG_FILE_INVALID_START_LSN	HY000
13881	ER_IB_MSG_LOG_FILE_INVALID_LSN_RANGES	HY000
13882	ER_IB_MSG_LOG_FILE_MISSING_FOR_ID	HY000
13883	ER_IB_MSG_LOG_CHECKPOINT_FOUND	HY000
13884	ER_IB_MSG_LOG_FILES_CAPACITY_CHANGED	HY000
13885	ER_IB_MSG_LOG_FILES_RESIZE_REQUESTED	HY000
13886	ER_IB_MSG_LOG_FILES_RESIZE_CANCELLED	HY000
13887	ER_IB_MSG_LOG_FILES_RESIZE_FINISHED	HY000
13888	ER_IB_MSG_LOG_FILES_UPGRADE	HY000
13889	ER_IB_MSG_LOG_FILE_MARK_CURRENT_AS_INCOMPLETE	HY000
13890	ER_IB_MSG_LOG_FILE_REMOVE_FAILED	HY000
13891	ER_IB_MSG_LOG_FILE_RENAME_ON_CREATE_FAILED	HY000
13892	ER_IB_MSG_LOG_FILES_CREATED_BY_UNKNOWN_CREATOR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13893	ER_IB_MSG_LOG_FILES_FOUND_MISSING	HY000
13894	ER_IB_MSG_LOG_FILE_FORMAT_TOO_NEW	HY000
13895	ER_IB_MSG_LOG_FILE_FORMAT_TOO_OLD	HY000
13896	ER_IB_MSG_LOG_FILE_DIFFERENT_FORMATS	HY000
13897	ER_IB_MSG_LOG_PRE_8_0_30_MISSING_FILE0	HY000
13898	ER_IB_MSG_LOG_PFS_ACQUIRE_SERVICES_FAILED	HY000
13899	ER_IB_MSG_LOG_PFS_CREATE_TABLES_FAILED	HY000
13900	ER_IB_MSG_LOG_FILE_TRUNCATE	HY000
13901	ER_IB_MSG_LOG_FILE_UNUSED_RESIZE_FAILED	HY000
13902	ER_IB_MSG_LOG_FILE_UNUSED_REMOVE_FAILED	HY000
13903	ER_IB_MSG_LOG_FILE_UNUSED_RENAME_FAILED	HY000
13904	ER_IB_MSG_LOG_FILE_UNUSED_MARK_AS_IN_USE_FAILED	HY000
13905	ER_IB_MSG_LOG_FILE_MARK_AS_UNUSED_FAILED	HY000
13906	ER_IB_MSG_LOG_PARAMS_DEDICATED_SERVER_IGNORED	HY000
13907	ER_IB_MSG_LOG_PARAMS_LEGACY_USAGE	HY000
13908	ER_GRP_RPL_FAILED_TO_LOG_VIEW_CHANGE	HY000
13909	ER_BINLOG_CRASH_RECOVERY_MALFORMED_LOG	HY000
13910	ER_BINLOG_CRASH_RECOVERY_ERROR_RETURNED_SE	HY000
13911	ER_BINLOG_CRASH_RECOVERY_ENGINE_RESULTS	HY000
13912	ER_BINLOG_CRASH_RECOVERY_COMMIT_FAILED	HY000
13913	ER_BINLOG_CRASH_RECOVERY_ROLLBACK_FAILED	HY000
13914	ER_BINLOG_CRASH_RECOVERY_PREPARE_FAILED	HY000
13915	ER_COMPONENT_EE_SYS_VAR_REGISTRATION_FAILURE	HY000
13916	ER_COMPONENT_EE_SYS_VAR_DEREGISTRATION_FAILURE	HY000
13917	ER_COMPONENT_EE_FUNCTION_REGISTRATION_FAILURE	HY000
13918	ER_COMPONENT_EE_FUNCTION_DEREGISTRATION_FAILURE	HY000
13919	ER_COMPONENT_EE_FUNCTION_INVALID_ARGUMENTS	HY000
13920	ER_COMPONENT_EE_FUNCTION_INVALID_ALGORITHM	HY000
13921	ER_COMPONENT_EE_FUNCTION_KEY_LENGTH_OUT_OF_RANGE	HY000
13922	ER_COMPONENT_EE_FUNCTION_PRIVATE_KEY_GENERATION_FAILURE	HY000
13923	ER_COMPONENT_EE_FUNCTION_PUBLIC_KEY_GENERATION_FAILURE	HY000
13924	ER_COMPONENT_EE_DATA_LENGTH_OUT_OF_RAGE	HY000
13925	ER_COMPONENT_EE_DATA_ENCRYPTION_ERROR	HY000
13926	ER_COMPONENT_EE_DATA_DECRYPTION_ERROR	HY000
13927	ER_COMPONENT_EE_DATA_SIGN_ERROR	HY000
13928	ER_COMPONENT_EE_OPENSSL_ERROR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13929	ER_COMPONENT_EE_INSUFFICIENT_LENGTH	HY000
13930	ER_SYSTEMD_NOTIFY_DEBUG	HY000
13931	ER_TMP_SESSION_FOR_VAR	HY000
13932	ER_BUILD_ID	HY000
13933	ER_THREAD_POOL_CANNOT_REGISTER_DYNAMIC_PRIVILEGE	HY000
13934	ER_IB_MSG_LOG_WRITER_WAIT_ON_CONSUMER	HY000
13935	ER_CONDITIONAL_DEBUG	HY000
13936	ER_IB_MSG_PARSE_OLD_REDO_INDEX_VERSION	HY000
13941	ER_IB_MSG_CLEAR_INSTANT_DROP_COLUMN_METADATA	HY000
13942	ER_COMPONENT_KEYRING_OCI_OPEN_KEY_FILE	HY000
13943	ER_COMPONENT_KEYRING_OCI_CREATE_PRIVATE_KEY	HY000
13944	ER_COMPONENT_KEYRING_OCI_READ_KEY_FILE	HY000
13945	ER_NOTE_COMPONENT_KEYRING_OCI_MISSING_NAME_OR_TYPE	HY000
13946	ER_WARN_COMPONENT_KEYRING_OCI_DUPLICATE_KEY	HY000
13947	ER_KEYRING_OCI_PARSE_JSON	HY000
13948	ER_KEYRING_OCI_INVALID_JSON	HY000
13949	ER_KEYRING_OCI_HTTP_REQUEST	HY000
13950	ER_THREAD_POOL_SYSVAR_CHANGE	HY000
13951	ER_STACK_BACKTRACE	HY000
13952	ER_IB_MSG_BUF_POOL_RESIZE_COMPLETE_CUR_CODE	HY000
13953	ER_IB_MSG_BUF_POOL_RESIZE_PROGRESS_UPDATE	HY000
13954	ER_IB_MSG_BUF_POOL_RESIZE_CODE_STATUS	HY000
13955	ER_THREAD_POOL_QUERY_THREADS_PER_GROUP_INVALID	HY000
13956	ER_THREAD_POOL_QUERY_THRS_PER_GRP_EXCEEDS_TXN_THR_LIMIT	HY000
13957	ER_IB_MSG_INVALID_PAGE_TYPE	HY000
13958	ER_IB_PARALLEL_READER_WORKER_INFO	HY000
13959	ER_IB_BULK_LOAD_SUBTREE_INFO	HY000
13960	ER_IB_BULK_FLUSHER_INFO	HY000
13961	ER_IB_BUFFER_POOL_OVERUSE	HY000
13962	ER_IB_BUFFER_POOL_FULL	HY000
13963	ER_IB_DUPLICATE_KEY	HY000
13964	ER_REPLICATION_INCOMPATIBLE_TABLE_WITH_GIPK	HY000
13965	ER_BULK_EXECUTOR_INFO	HY000
13966	ER_BULK_LOADER_INFO	HY000
13967	ER_BULK_LOADER_FILE_CONTAINS_LESS_LINES_THAN_IGNORE_CLAUSE_LIMIT	HY000
13968	ER_BULK_READER_INFO	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
13969	ER_BULK_READER_LIBCURL_INIT_FAILED_LOG	HY000
13970	ER_BULK_READER_LIBCURL_ERROR_LOG	HY000
13971	ER_BULK_READER_SERVER_ERROR_LOG	HY000
13972	ER_BULK_READER_COMMUNICATION_ERROR_LOG	HY000
13973	ER_BULK_PARSER_MISSING_ENCLOSED_BY_LOG	HY000
13974	ER_BULK_PARSER_ROW_BUFFER_MAX_TOTAL_COLS_EXCEEDED_LOG	HY000
13975	ER_BULK_PARSER_COPY_BUFFER_SIZE_EXCEEDED_LOG	HY000
13976	ER_BULK_PARSER_UNEXPECTED_END_OF_INPUT_LOG	HY000
13977	ER_BULK_PARSER_UNEXPECTED_ROW_TERMINATOR_LOG	HY000
13978	ER_BULK_PARSER_UNEXPECTED_CHAR_AFTER_ENDING_ENCLOSED_BY_LOG	HY000
13979	ER_BULK_PARSER_UNEXPECTED_CHAR_AFTER_NULL_ESCAPE_LOG	HY000
13980	ER_BULK_PARSER_UNEXPECTED_CHAR_AFTER_COLUMN_TERMINATOR_LOG	HY000
13981	ER_BULK_PARSER_INCOMPLETE_ESCAPE_SEQUENCE_LOG	HY000
13982	ER_LOAD_BULK_DATA_WRONG_VALUE_FOR_FIELD_LOG	HY000
13983	ER_LOAD_BULK_DATA_WARN_NULL_TO_NOTNULL_LOG	HY000
13984	ER_IB_BULK_LOAD_THREAD_FAIL	HY000
13985	ER_IB_BULK_LOAD_MERGE_FAIL	HY000
13986	ER_IB_LOAD_BULK_CONCURRENCY_REDUCED	HY000
13987	ER_PLUGIN_EXCEPTION_OPERATION_FAILED	HY000
13988	ER_REQUIRE_TABLE_PRIMARY_KEY_CHECK_GENERATE_WITH_GR_IN_REPLICA	HY000
13989	ER_CHECK_TABLE_INSTANT_VERSION_BIT_SET	HY000
13990	ER_GRP_RPL_PAXOS_SINGLE_LEADER_DIFF_FROM_OLD_GRP	HY000
13991	ER_IB_WRN_IGNORE_REDO_LOG_CAPACITY	HY000
13992	ER_IB_PRIMARY_KEY_IS_INSTANT	HY000
13993	ER_THREAD_POOL_IDLE_CONNECTION_CLOSED	HY000
13994	ER_IB_HIDDEN_NAME_CONFLICT	HY000
13995	ER_IB_DICT_INVALID_COLUMN_POSITION	HY000
13996	ER_IB_DICT_LOG_TABLE_INFO	HY000
13997	ER_RPL_ASYNC_NEXT_FAILOVER_CHANNEL_SELECTED	HY000
13998	ER_RPL_REPLICA_SOURCE_UUID_HAS_NOT_CHANGED	HY000
13999	ER_RPL_REPLICA_SOURCE_UUID_HAS_CHANGED_HOST_PORT_UNCHANGED	HY000
14000	ER_RPL_REPLICA_SOURCE_UUID_HOST_PORT_HAS_CHANGED	HY000
14001	ER_RPL_REPLICA_CONNECTED_TO_SOURCE_RPL_STARTED_FILE_BASED	HY000
14002	ER_RPL_REPLICA_CONNECTED_TO_SOURCE_RPL_STARTED_GTID_BASED	HY000
14003	ER_IB_INDEX_LOADER_DONE	HY000
14004	ER_IB_INDEX_BUILDER_DONE	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
14005	ER_WARN_DEPRECATED_USER_DEFINED_COLLATIONS_OPTION	HY000
14006	ER_IB_INDEX_BUILDER_INIT	HY000
14007	ER_IB_SELECT_COUNT_STAR	HY000
14008	ER_IB_INDEX_LOG_VERSION_MISMATCH	HY000
14009	ER_WARN_COMPONENTS_INFRASTRUCTURE_MANIFEST_MULTIPLE_KEYRING	HY000
14010	ER_GRP_RPL_HAS_STARTED	HY000
14011	ER_CHECK_TABLE_MIN_REC_FLAG_SET	HY000
14012	ER_CHECK_TABLE_MIN_REC_FLAG_NOT_SET	HY000
14013	ER_NOTE_COMPONENT_SLOT_REGISTRATION_SUCCESS	HY000
14014	ER_NOTE_COMPONENT_SLOT_DEREGISTRATION_SUCCESS	HY000
14015	ER_WARN_CANNOT_FREE_COMPONENT_DATA_DEALLOCATION_FAILED	HY000
14016	ER_IB_RESURRECT_TRX_INSERT	HY000
14017	ER_IB_RESURRECT_TRX_UPDATE	HY000
14018	ER_IB_RESURRECT_IDENTIFY_TABLE_TO_LOCK	HY000
14019	ER_IB_RESURRECT_ACQUIRE_TABLE_LOCK	HY000
14020	ER_IB_RESURRECT_RECORD_PROGRESS	HY000
14021	ER_IB_RESURRECT_RECORD_COMPLETE	HY000
14022	ER_IB_RESURRECT_TRX_INSERT_COMPLETE	HY000
14023	ER_IB_RESURRECT_TRX_UPDATE_COMPLETE	HY000
14024	ER_AUTHENTICATION_OCI_INVALID_TOKEN	HY000
14025	ER_AUTHENTICATION_OCI_TOKEN_DETAILS_MISMATCH	HY000
14026	ER_AUTHENTICATION_OCI_TOKEN_NOT_VERIFIED	HY000
14027	ER_AUTHENTICATION_OCI_DOWNLOAD_IDDP_PUBLIC_KEY	HY000
14029	ER_SYS_VAR_REGISTRATION	HY000
14030	ER_SYS_VAR_DEREGISTRATION	HY000
14031	ER_UDF_REGISTRATION	HY000
14032	ER_UDF_DEREGISTRATION	HY000
14033	ER_PRIVILEGE_REGISTRATION	HY000
14034	ER_PRIVILEGE_DEREGISTRATION	HY000
14035	ER_UDF_EXEC_FAILURE	HY000
14036	ER_UDF_EXEC_FAILURE_REASON	HY000
14037	ER_COMPONENT_SERVICE_CALL	HY000
14038	ER_COMPONENT_SERVICE_CALL_RESULT	HY000
14039	ER_COMPONENT_LOCK	HY000
14040	ER_COMPONENT_UNLOCK	HY000
14041	ER_COMPONENT_MASKING_OTHER_ERROR	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
14042	ER_COMPONENT_MASKING_ABI	HY000
14043	ER_COMPONENT_MASKING_ABI_REASON	HY000
14044	ER_COMPONENT_MASKING_RANDOM_CREATE	HY000
14045	ER_COMPONENT_MASKING_RANDOM_CREATE_REASON	HY000
14046	ER_COMPONENT_MASKING_CANNOT_ACCESS_TABLE	HY000
14047	ER_REDUCED_DBLWR_FILE_CORRUPTED	HY000
14048	ER_REDUCED_DBLWR_PAGE_FOUND	HY000
14049	ER_CONN_INIT_CONNECT_IGNORED_MFA	HY000
14050	ER_SECONDARY_ENGINE_DDL_FAILED	HY000
14051	ER_THREAD_POOL_CONNECTION_REPORT	HY000
14052	ER_WARN_SCHEDULED_TASK_RUN_FAILED	HY000
14053	ER_AUDIT_LOG_INVALID_FLUSH_INTERVAL_VALUE	HY000
14054	ER_LOG_CANNOT_PURGE_BINLOG_WITH_BACKUP_LOCK	HY000
14055	ER_CONVERT_MULTI_VALUE	HY000
14056	ER_IB_DDL_CONVERT_HEAP_NOT_FOUND	HY000
14057	ER_SERVER_DOWNGRADE_FROM_VERSION	HY000
14058	ER_BEYOND_SERVER_DOWNGRADE_THRESHOLD	HY000
14059	ER_BEYOND_SERVER_UPGRADE_THRESHOLD	HY000
14060	ER_INVALID_SERVER_UPGRADE_NOT_LTS	HY000
14061	ER_INVALID_SERVER_DOWNGRADE_NOT_PATCH	HY000
14062	ER_FAILED_GET_DD_PROPERTY	HY000
14063	ER_FAILED_SET_DD_PROPERTY	HY000
14064	ER_SERVER_DOWNGRADE_STATUS	HY000
14065	ER_INFORMATION_SCHEMA_VERSION_CHANGE	HY000
14066	ER_PERFORMANCE_SCHEMA_VERSION_CHANGE	HY000
14067	ER_WARN_DEPRECATED_OR_BLOCKED_CIPHER	HY000
14068	ER_IB_MSG_DDL_FAIL_NO_BUILDER	HY000
14069	ER_GRP_RPL_MEMBER_INFO_DOES_NOT_EXIST	HY000
14070	ER_USAGE_DEPRECATION_COUNTER	HY000
15000	ER_LANGUAGE_COMPONENT_INFO	HY000
15001	ER_LANGUAGE_COMPONENT_WARNING	HY000
15002	ER_LANGUAGE_COMPONENT_ERROR	HY000
15003	ER_IB_BULK_FLUSHER_PUNCH_HOLE	HY000
15004	ER_GRP_RPL_CONN_KILLED	HY000
15005	ER_WARN_CANT_OPEN_CERTIFICATE	HY000
15006	ER_FAILED_TO_VALIDATE_CERTIFICATES_SERVER_EXIT	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
15007	ER_WARN_CA_CERT_VERIFY_FAILED	HY000
15008	ER_WARN_FAILED_TO_SETUP_TLS	HY000
15009	ER_TLS_LIBRARY_ERROR_INTERNAL	HY000
15010	ER_SERVER_CERT_VERIFY_FAILED	HY000
15011	ER_WARN_CERTIFICATE_ERROR_STRING	HY000
15012	ER_TELEMETRY_INFO	HY000
15013	ER_TELEMETRY_WARNING	HY000
15014	ER_TELEMETRY_ERROR	HY000
15015	ER_SRV_START	HY000
15016	ER_SRV_END	HY000
15017	ER_SRV_INIT_START	HY000
15018	ER_SRV_INIT_END	HY000
15019	ER_PLUGINS_SHUTDOWN_START	HY000
15020	ER_PLUGINS_SHUTDOWN_END	HY000
15021	ER_COMPONENTS_INFRASTRUCTURE_SHUTDOWN_START	HY000
15022	ER_COMPONENTS_INFRASTRUCTURE_SHUTDOWN_END	HY000
15023	ER_CONNECTIONS_SHUTDOWN_START	HY000
15024	ER_CONNECTIONS_SHUTDOWN_END	HY000
15025	ER_THREAD_STILL_ALIVE	HY000
15026	ER_NUM_THREADS_STILL_ALIVE	HY000
15027	ER_GRP_RPL_MYSQL_NETWORK_PROVIDER_SERVER_ERROR_COMMAND_ERROR	HY000
15028	ER_COMPONENT_KEYRING_OCI_INVALID_CONFIG_VAR	HY000
15029	ER_WARN_OPTION_RESET_AND_IGNORED_DURING_INITIALIZE	HY000
15030	ER_FIREWALL_SCHEDULER_REGISTER_FAILED	HY000
15031	ER_FIREWALL_INVALID_RELOAD_INTERVAL_VALUE	HY000
15032	ER_WARN_DEPRECATED_DYNAMIC_PRIV_FOR_USER	HY000
15033	ER_BULK_MULTI_READER_INFO	HY000
15034	ER_BULK_MERGE_LOADER_INFO	HY000
15035	ER_BULK_SORTING_LOADER_INFO	HY000
15036	ER_BULK_WRITER_INFO	HY000
15037	ER_BULK_WRITER_LIBCURL_INIT_FAILED_LOG	HY000
15038	ER_BULK_WRITER_LIBCURL_ERROR_LOG	HY000
15039	ER_IB_WRONG_PAGE_ID	HY000
15040	ER_IB_FIXED_PAGE_ID	HY000
15041	ER_IB_WRONG_PAGEID_AFTER_SYNC_READ	HY000
15042	ER_IB_SYNC_READ_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
15043	ER_START_REPLICA_CHANNEL_INVALID_CONFIGURATION_LOG	HY000
15044	ER_GROUP_REPLICATION_CERTIFIER_MESSAGE_LARGE	HY000
15045	ER_GROUP_REPLICATION_METADATA_SENDER_IS_REMOTE	HY000
15046	ER_GROUP_REPLICATION_METADATA_SENDER	HY000
15047	ER_GROUP_REPLICATION_ERROR_COMPRESS_INITIALIZE	HY000
15048	ER_GROUP_REPLICATION_COMPRESS_PROCESS	HY000
15049	ER_GROUP_REPLICATION_UNKOWN_COMPRESSION_TYPE	HY000
15050	ER_GROUP_REPLICATION_COMPRESS_EXCEEDS_MAX_SIZE	HY000
15051	ER_GROUP_REPLICATION_COMPRESS_OUT_OF_MEMORY	HY000
15052	ER_GROUP_REPLICATION_ERROR_DECOMPRESS_INITIALIZE	HY000
15053	ER_GROUP_REPLICATION_DECOMPRESS_EXCEEDS_MAX_SIZE	HY000
15054	ER_GROUP_REPLICATION_DECOMPRESS_OUT_OF_MEMORY	HY000
15055	ER_GROUP_REPLICATION_DECOMPRESS_TRUNCATED	HY000
15056	ER_GROUP_REPLICATION_DECOMPRESS_CORRUPTED	HY000
15057	ER_GROUP_REPLICATION_DECOMPRESS_END	HY000
15058	ER_GROUP_REPLICATION_DECOMPRESS_PROCESS	HY000
15059	ER_GRP_RPL_VCLE_NOT_BEING_LOGGED	HY000
15060	ER_GROUP_REPLICATION_METADATA_PROTOBUF_PARSING	HY000
15061	ER_GROUP_REPLICATION_PROTOBUF_SERIALIZING_ERROR	HY000
15062	ER_GROUP_REPLICATION_ERROR_RECEIVED_WAITING_METADATA	HY000
15063	ER_GROUP_REPLICATION_TIMEOUT_ERROR_FETCHING_METADATA	HY000
15064	ER_GROUP_REPLICATION_METADATA_PAYLOAD_EMPTY	HY000
15065	ER_GROUP_REPLICATION_METADATA_MESSAGE_PAYLOAD_EMPTY	HY000
15066	ER_GROUP_REPLICATION_METADATA_READ_GTID_EXECUTED	HY000
15067	ER_GROUP_REPLICATION_METADATA_PAYLOAD_DECODING	HY000
15068	ER_GROUP_REPLICATION_METADATA_SEND_ERROR	HY000
15069	ER_GROUP_REPLICATION_METADATA_SAVE_RECOVERY_COPY	HY000
15070	ER_GROUP_REPLICATION_METADATA_CERT_INFO_ERROR_PROCESSING	HY000
15071	ER_GROUP_REPLICATION_METADATA_CERT_INFO_PACKET_EMPTY	HY000
15072	ER_GROUP_REPLICATION_METADATA_ERROR_ON_SEND_ERROR_MESSAGE	HY000
15073	ER_GROUP_REPLICATION_METADATA_SET_IN_RECOVERY_FAILED	HY000
15074	ER_GROUP_REPLICATION_CERTIFICATION_MODULE_FAILURE	HY000
15075	ER_GROUP_REPLICATION_METADATA_INITIALIZATION_FAILURE	HY000
15076	ER_GROUP_REPLICATION_METADATA_MEMORY_ALLOC	HY000
15077	ER_GROUP_REPLICATION_RECOVERY_SKIPPED_GTID_PRESENT	HY000
15078	ER_GROUP_REPLICATION_RECOVERY_FETCHING_GTID_EXECUTED_SET	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
15079	ER_GROUP_REPLICATION_RECOVERY_ERROR_ADD_GTID_EXECUTED	HY000
15080	ER_GROUP_REPLICATION_RECOVERY_METADATA_SENDER_NOT_FOUND	HY000
15081	ER_GROUP_REPLICATION_METADATA_CERT_INFO_PACKET_COUNT_ERROR	HY000
15082	ER_GROUP_REPLICATION_METADATA_CERT_INFO_ENCODING_ERROR	HY000
15083	ER_GROUP_REPLICATION_METADATA_NO_VALID_DONOR	HY000
15084	ER_GROUP_REPLICATION_NO_CERTIFICATION_DONOR_AVAILABLE	HY000
15085	ER_GROUP_REPLICATION_RECOVERY_STOPPED_GTID_PRESENT	HY000
15086	ER_RPL_ASYNC_CHECK_CONNECTION_ERROR	HY000
15087	ER_RPL_ASYNC_MONITOR_IO_THD_FETCH_GROUP_MAJORITY_ERROR	HY000
15088	ER_RPL_ASYNC_REPLICA_IO_THD_FETCH_GROUP_MAJORITY_ERROR	HY000
15089	ER_RPL_ASYNC_GET_GROUP_MEMBERSHIP_DETAILS_ERROR	HY000
15090	ER_IB_ERR_CORRUPT_TABLESPACE_UNRECOVERABLE	HY000
15091	ER_IB_BULK_LOAD_STATS_WARN	HY000
15092	ER_COMPONENT_MASKING_INVALID_FLUSH_INTERVAL_VALUE	HY000
15093	ER_COMPONENT_MASKING_VAR_REGISTRATION_FAILURE	HY000
15094	ER_COMPONENT_MASKING_NOTIFICATION_REGISTRATION_FAILURE	HY000
15095	ER_GRP_RPL_MSG_DECODING_FAILED	HY000
15096	ER_GRP_RPL_APPLIER_ERROR_PACKET_RECEIVED	HY000
15097	ER_LANGUAGE_COMPONENT_INSTALL_ERROR	HY000
15098	ER_WARN_LANGUAGE_COMPONENT_CANNOT_UNINSTALL	HY000
15099	ER_LANGUAGE_COMPONENT_SERVER_ERROR	HY000
15100	ER_LANGUAGE_COMPONENT_INTERNAL_ERROR	HY000
15101	ER_LANGUAGE_COMPONENT_VM_API_FUNCTION_ERROR	HY000
15102	ER_LANGUAGE_COMPONENT_VM_INTERNAL_ERROR	HY000
15103	ER_WARN_LANGUAGE_COMPONENT_RESOURCE_LIMIT	HY000
15104	ER_BLOCKED_CIPHER	HY000
15105	ER_KEYRING_COMPONENT_KEYRING_FILE_NAME_EMPTY	HY000
15106	ER_KEYRING_COMPONENT_KEYRING_FILE_READ_FAILED	HY000
15107	ER_KEYRING_COMPONENT_KEYRING_FILE_DECRYPT_FAILED	HY000
15108	ER_KEYRING_COMPONENT_KEYRING_FILE_INVALID_FORMAT	HY000
15109	ER_KEYRING_COMPONENT_KEYRING_FILE_JSON_EXTRACT_FAILED	HY000
15110	ER_KEYRING_COMPONENT_KEYRING_FILE_KEY_EXTRACT_FAILED	HY000
15111	ER_NOTE_KEYRING_COMPONENT_NOT_INITIALIZED	HY000
15112	ER_NOTE_KEYRING_COMPONENT_EMPTY_DATA_ID	HY000
15113	ER_NOTE_KEYRING_COMPONENT_KEYS_METADATA_ITERATOR_INIT_FAILED	HY000
15114	ER_NOTE_KEYRING_COMPONENT_KEY_READ_ITERATOR_INIT_FAILED	HY000

MySQL Error Number	MySQL Error Name	SQL Standard SQLState
15115	ER_NOTE_KEYRING_COMPONENT_KEY_READ_ITERATOR_FETCH_FAILED	HY000
15116	ER_BACKGROUND_HISTOGRAM_UPDATE	HY000
15117	ER_IB_MSG_SUBMIT_DETAILED_BUG_REPORT	HY000
15118	ER_AUTO_INCREMENT_NOT_SUPPORTED_FOR_FLOAT_DOUBLE	HY000
15119	ER_SERVER_DOWNGRADE_SYS_SCHEMA	HY000
15120	ER_SERVER_DOWNGRADE_HELP_TABLE_STATUS	HY000
15121	ER_KEYRING_MIGRATE_SKIPPED_KEY	HY000
15122	ER_KEYRING_MIGRATE_MEMORY_DEALLOCATION_FAILED	HY000
15123	ER_LOG_CLIENT_INTERACTION_TIMEOUT	HY000
15124	ER_GRP_RPL_PREEMPTIVE_GARBAGE_COLLECTION_DIFF_FROM_GRP	HY000
15125	ER_IB_LONG_ROLLBACK_FULL	HY000
15126	ER_IB_LONG_ROLLBACK	HY000
15127	ER_INVALID_FILE_FORMAT	HY000
15128	ER_LOG_SANITIZATION	HY000
15129	ER_WARN_LOG_DEPRECATED_NON_STANDARD_KEY	HY000
15130	ER_THREAD_POOL_MTL_DISABLE	HY000
15131	ER_THREAD_POOL_MTL_REENABLE	HY000
15132	ER_LOG_PARTITION_PREFIX_KEY_NOT_SUPPORTED	HY000
15133	ER_LDAP_AUTH_INFO_USER_MAP	HY000
15134	ER_ACCESS_DENIED_NO_PROXY_GRANT_WITH_NAME	HY000
15135	ER_ACCESS_DENIED_NO_PROXY_WITH_NAME	HY000
15136	ER_GRP_RPL_RECOVERY_WAIT_APPLIER_BACKLOG_START	HY000
15137	ER_GRP_RPL_RECOVERY_WAIT_APPLIER_BACKLOG_FINISH	HY000
15138	ER_BULK_READER_ZSTD_ERROR_LOG	HY000
15139	ER_SECONDARY_ENGINE_DDL_TRACK_PROGRESS	HY000
15140	ER_IB_MSG_INNODB_FLUSH_METHOD	HY000

Chapter 7 JDBC Concepts

Table of Contents

This section provides some general JDBC background.

7.1 Connecting to MySQL Using the JDBC `DriverManager` Interface

When you are using JDBC outside of an application server, the `DriverManager` class manages the establishment of connections.

Specify to the `DriverManager` which JDBC drivers to try to make Connections with. The easiest way to do this is to use `Class.forName()` on the class that implements the `java.sql.Driver` interface. With MySQL Connector/J, the name of this class is `com.mysql.cj.jdbc.Driver`. With this method, you could use an external configuration file to supply the driver class name and driver parameters to use when connecting to a database.

The following section of Java code shows how you might register MySQL Connector/J from the `main()` method of your application. If testing this code, first read the installation section at Chapter 4, *Connector/J Installation*, to make sure you have connector installed correctly and the `CLASSPATH` set up. Also, ensure that MySQL is configured to accept external TCP/IP connections.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.cj.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
            // The newInstance() call is a work around for some
            // broken Java implementations

            Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

After the driver has been registered with the `DriverManager`, you can obtain a `Connection` instance that is connected to a particular database by calling `DriverManager.getConnection()`:

Example 7.1 Connector/J: Obtaining a connection from the `DriverManager`

If you have not already done so, please review the portion of Section 7.1, “Connecting to MySQL Using the JDBC `DriverManager` Interface” above before working with the example below.

This example shows how you can obtain a Connection instance from the DriverManager. There are a few different signatures for the getConnection() method. Consult the API documentation that comes with your JDK for more specific information on how to use them.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

Connection conn = null;
...
try {
    conn =
       DriverManager.getConnection("jdbc:mysql://localhost/test?" +
                                   "user=minty&password=greatsqldb");

    // Do something with the Connection

   ...
} catch (SQLException ex) {
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
```

Once a Connection is established, it can be used to create Statement and PreparedStatement objects, as well as retrieve metadata about the database. This is explained in the following sections.

When the user for the connection is unspecified, Connector/J's implementations of the authentication plugins use by default the name of the OS user who runs the application for authentication with the MySQL server (except when the Kerberos authentication plugin is being used; see Section 6.12.2, “Connecting Using Kerberos” for details).

> **Note**
>
> A user name is considered unspecified only when the following conditions are all met:
>
> 1. The method DriverManager.getConnection(String url, String user, String password) is not used.
> 2. The connection property user is not used in, for example, the connection URL,or elsewhere.
> 3. The user is not mentioned in the authority of the connection URL, as in jdbc:mysql://localhost:3306/test, or jdbc:mysql://@localhost:3306/test.
>
> Notice if (1) or (2) is not true and an empty string is passed, the user name is an empty string then, and is not considered unspecified.

7.2 Using JDBC Statement Objects to Execute SQL

Statement objects allow you to execute basic SQL queries and retrieve the results through the ResultSet class, which is described later.

To create a Statement instance, you call the createStatement() method on the Connection object you have retrieved using one of the DriverManager.getConnection() or DataSource.getConnection() methods described earlier.

Once you have a `Statement` instance, you can execute a `SELECT` query by calling the `executeQuery(String)` method with the SQL you want to use.

To update data in the database, use the `executeUpdate(String SQL)` method. This method returns the number of rows matched by the update statement, not the number of rows that were modified.

If you do not know ahead of time whether the SQL statement will be a `SELECT` or an `UPDATE`/`INSERT`, then you can use the `execute(String SQL)` method. This method will return true if the SQL query was a `SELECT`, or false if it was an `UPDATE`, `INSERT`, or `DELETE` statement. If the statement was a `SELECT` query, you can retrieve the results by calling the `getResultSet()` method. If the statement was an `UPDATE`, `INSERT`, or `DELETE` statement, you can retrieve the affected rows count by calling `getUpdateCount()` on the `Statement` instance.

Example 7.2 Connector/J: Using java.sql.Statement to execute a `SELECT` query

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;

// assume that conn is an already created JDBC connection (see previous examples)

Statement stmt = null;
ResultSet rs = null;

try {
    stmt = conn.createStatement();
    rs = stmt.executeQuery("SELECT foo FROM bar");

    // or alternatively, if you don't know ahead of time that
    // the query will be a SELECT...

    if (stmt.execute("SELECT foo FROM bar")) {
        rs = stmt.getResultSet();
    }

    // Now do something with the ResultSet ....
}
catch (SQLException ex){
    // handle any errors
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
}
finally {
    // it is a good idea to release
    // resources in a finally{} block
    // in reverse-order of their creation
    // if they are no-longer needed

    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException sqlEx) { } // ignore

        rs = null;
    }

    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException sqlEx) { } // ignore

        stmt = null;
```

```
    }
}
```

7.3 Using JDBC CallableStatements to Execute Stored Procedures

Connector/J fully implements the java.sql.CallableStatement interface.

For more information on MySQL stored procedures, please refer to Using Stored Routines.

Connector/J exposes stored procedure functionality through JDBC's CallableStatement interface.

The following example shows a stored procedure that returns the value of inOutParam incremented by 1, and the string passed in using inputParam as a ResultSet:

Example 7.3 Connector/J: Calling Stored Procedures

```
CREATE PROCEDURE demoSp(IN inputParam VARCHAR(255), \
                        INOUT inOutParam INT)
BEGIN
    DECLARE z INT;
    SET z = inOutParam + 1;
    SET inOutParam = z;

    SELECT inputParam;

    SELECT CONCAT('zyxw', inputParam);
END
```

To use the demoSp procedure with Connector/J, follow these steps:

1. Prepare the callable statement by using Connection.prepareCall().

   Notice that you have to use JDBC escape syntax, and that the parentheses surrounding the parameter placeholders are not optional:

   Example 7.4 Connector/J: Using Connection.prepareCall()

   ```
   import java.sql.CallableStatement;

   ...

       //
       // Prepare a call to the stored procedure 'demoSp'
       // with two parameters
       //
       // Notice the use of JDBC-escape syntax ({call ...})
       //

       CallableStatement cStmt = conn.prepareCall("{call demoSp(?, ?)}");



       cStmt.setString(1, "abcdefg");
   ```

   > **Note**
   >
   > Connection.prepareCall() is an expensive method, due to the metadata retrieval that the driver performs to support output parameters. For performance reasons, minimize unnecessary calls to Connection.prepareCall() by reusing CallableStatement instances in your code.

2. Register the output parameters (if any exist)

   To retrieve the values of output parameters (parameters specified as OUT or INOUT when you created the stored procedure), JDBC requires that they be specified before statement execution using the various registerOutputParameter() methods in the CallableStatement interface:

   Example 7.5 Connector/J: Registering output parameters

```
import java.sql.Types;
...
//
// Connector/J supports both named and indexed
// output parameters. You can register output
// parameters using either method, as well
// as retrieve output parameters using either
// method, regardless of what method was
// used to register them.
//
// The following examples show how to use
// the various methods of registering
// output parameters (you should of course
// use only one registration per parameter).
//

//
// Registers the second parameter as output, and
// uses the type 'INTEGER' for values returned from
// getObject()
//

cStmt.registerOutParameter(2, Types.INTEGER);

//
// Registers the named parameter 'inOutParam', and
// uses the type 'INTEGER' for values returned from
// getObject()
//

cStmt.registerOutParameter("inOutParam", Types.INTEGER);
...
```

3. Set the input parameters (if any exist)

   Input and in/out parameters are set as for PreparedStatement objects. However, CallableStatement also supports setting parameters by name:

   Example 7.6 Connector/J: Setting CallableStatement input parameters

```
...

    //
    // Set a parameter by index
    //

    cStmt.setString(1, "abcdefg");

    //
    // Alternatively, set a parameter using
    // the parameter name
    //

    cStmt.setString("inputParam", "abcdefg");

    //
    // Set the 'in/out' parameter using an index
```

```
    //

    cStmt.setInt(2, 1);

    //
    // Alternatively, set the 'in/out' parameter
    // by name
    //

    cStmt.setInt("inOutParam", 1);

...
```

4. Execute the `CallableStatement`, and retrieve any result sets or output parameters.

   Although `CallableStatement` supports calling any of the `Statement` execute methods (`executeUpdate()`, `executeQuery()` or `execute()`), the most flexible method to call is `execute()`, as you do not need to know ahead of time if the stored procedure returns result sets:

   Example 7.7 Connector/J: Retrieving results and output parameter values

```
...

    boolean hadResults = cStmt.execute();

    //
    // Process all returned result sets
    //

    while (hadResults) {
        ResultSet rs = cStmt.getResultSet();

        // process result set
        ...

        hadResults = cStmt.getMoreResults();
    }

    //
    // Retrieve output parameters
    //
    // Connector/J supports both index-based and
    // name-based retrieval
    //

    int outputValue = cStmt.getInt(2); // index-based

    outputValue = cStmt.getInt("inOutParam"); // name-based

...
```

7.4 Retrieving `AUTO_INCREMENT` Column Values through JDBC

`getGeneratedKeys()` is the preferred method to use if you need to retrieve `AUTO_INCREMENT` keys and through JDBC; this is illustrated in the first example below. The second example shows how you can retrieve the same value using a standard `SELECT LAST_INSERT_ID()` query. The final example shows how updatable result sets can retrieve the `AUTO_INCREMENT` value when using the `insertRow()` method.

Example 7.8 Connector/J: Retrieving `AUTO_INCREMENT` column values using `Statement.getGeneratedKeys()`

```
Statement stmt = null;
ResultSet rs = null;
```

```
try {

    //
    // Create a Statement instance that we can use for
    // 'normal' result sets assuming you have a
    // Connection 'conn' to a MySQL database already
    // available

    stmt = conn.createStatement();

    //
    // Issue the DDL queries for the table for this example
    //

    stmt.executeUpdate("DROP TABLE IF EXISTS autoIncTutorial");
    stmt.executeUpdate(
            "CREATE TABLE autoIncTutorial ("
            + "priKey INT NOT NULL AUTO_INCREMENT, "
            + "dataField VARCHAR(64), PRIMARY KEY (priKey))");

    //
    // Insert one row that will generate an AUTO INCREMENT
    // key in the 'priKey' field
    //

    stmt.executeUpdate(
            "INSERT INTO autoIncTutorial (dataField) "
            + "values ('Can I Get the Auto Increment Field?')",
            Statement.RETURN_GENERATED_KEYS);

    //
    // Example of using Statement.getGeneratedKeys()
    // to retrieve the value of an auto-increment
    // value
    //

    int autoIncKeyFromApi = -1;

    rs = stmt.getGeneratedKeys();

    if (rs.next()) {
        autoIncKeyFromApi = rs.getInt(1);
    } else {

        // throw an exception from here
    }

    System.out.println("Key returned from getGeneratedKeys():"
        + autoIncKeyFromApi);
} finally {

    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException ex) {
            // ignore
        }
    }

    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException ex) {
            // ignore
        }
    }
```

```
}
```

Example 7.9 Connector/J: Retrieving AUTO_INCREMENT column values using SELECT LAST_INSERT_ID()

```
Statement stmt = null;
ResultSet rs = null;

try {

    //
    // Create a Statement instance that we can use for
    // 'normal' result sets.

    stmt = conn.createStatement();

    //
    // Issue the DDL queries for the table for this example
    //

    stmt.executeUpdate("DROP TABLE IF EXISTS autoIncTutorial");
    stmt.executeUpdate(
            "CREATE TABLE autoIncTutorial ("
            + "priKey INT NOT NULL AUTO_INCREMENT, "
            + "dataField VARCHAR(64), PRIMARY KEY (priKey))");

    //
    // Insert one row that will generate an AUTO INCREMENT
    // key in the 'priKey' field
    //

    stmt.executeUpdate(
            "INSERT INTO autoIncTutorial (dataField) "
            + "values ('Can I Get the Auto Increment Field?')");

    //
    // Use the MySQL LAST_INSERT_ID()
    // function to do the same thing as getGeneratedKeys()
    //

    int autoIncKeyFromFunc = -1;
    rs = stmt.executeQuery("SELECT LAST_INSERT_ID()");

    if (rs.next()) {
        autoIncKeyFromFunc = rs.getInt(1);
    } else {
        // throw an exception from here
    }

    System.out.println("Key returned from " +
                       "'SELECT LAST_INSERT_ID()': " +
                       autoIncKeyFromFunc);

} finally {

    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException ex) {
            // ignore
        }
    }

    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException ex) {
```

```
            // ignore
        }
    }
}
```

Example 7.10 Connector/J: Retrieving AUTO_INCREMENT column values in Updatable ResultSets

```
Statement stmt = null;
ResultSet rs = null;

try {

    //
    // Create a Statement instance that we can use for
    // 'normal' result sets as well as an 'updatable'
    // one, assuming you have a Connection 'conn' to
    // a MySQL database already available
    //

    stmt = conn.createStatement(java.sql.ResultSet.TYPE_FORWARD_ONLY,
                                java.sql.ResultSet.CONCUR_UPDATABLE);

    //
    // Issue the DDL queries for the table for this example
    //

    stmt.executeUpdate("DROP TABLE IF EXISTS autoIncTutorial");
    stmt.executeUpdate(
            "CREATE TABLE autoIncTutorial ("
            + "priKey INT NOT NULL AUTO_INCREMENT, "
            + "dataField VARCHAR(64), PRIMARY KEY (priKey))");

    //
    // Example of retrieving an AUTO INCREMENT key
    // from an updatable result set
    //

    rs = stmt.executeQuery("SELECT priKey, dataField "
       + "FROM autoIncTutorial");

    rs.moveToInsertRow();

    rs.updateString("dataField", "AUTO INCREMENT here?");
    rs.insertRow();

    //
    // the driver adds rows at the end
    //

    rs.last();

    //
    // We should now be on the row we just inserted
    //

    int autoIncKeyFromRS = rs.getInt("priKey");

    System.out.println("Key returned for inserted row: "
        + autoIncKeyFromRS);

} finally {

    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException ex) {
            // ignore
```

```
        }
    }

    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException ex) {
            // ignore
        }
    }
}
```

Running the preceding example code should produce the following output:

```
Key returned from getGeneratedKeys(): 1
Key returned from SELECT LAST_INSERT_ID(): 1
Key returned for inserted row: 1
```

At times, it can be tricky to use the `SELECT LAST_INSERT_ID()` query, as that function's value is scoped to a connection. So, if some other query happens on the same connection, the value is overwritten. On the other hand, the `getGeneratedKeys()` method is scoped by the `Statement` instance, so it can be used even if other queries happen on the same connection, but not on the same `Statement` instance.

Chapter 8 Connection Pooling with Connector/J

Connection pooling is a technique of creating and managing a pool of connections that are ready for use by any thread that needs them. Connection pooling can greatly increase the performance of your Java application, while reducing overall resource usage.

How Connection Pooling Works

Most applications only need a thread to have access to a JDBC connection when they are actively processing a transaction, which often takes only milliseconds to complete. When not processing a transaction, the connection sits idle. Connection pooling enables the idle connection to be used by some other thread to do useful work.

In practice, when a thread needs to do work against a MySQL or other database with JDBC, it requests a connection from the pool. When the thread is finished using the connection, it returns it to the pool, so that it can be used by any other threads.

When the connection is loaned out from the pool, it is used exclusively by the thread that requested it. From a programming point of view, it is the same as if your thread called `DriverManager.getConnection()` every time it needed a JDBC connection. With connection pooling, your thread may end up using either a new connection or an already-existing connection.

Benefits of Connection Pooling

The main benefits to connection pooling are:

- Reduced connection creation time.

  Although this is not usually an issue with the quick connection setup that MySQL offers compared to other databases, creating new JDBC connections still incurs networking and JDBC driver overhead that will be avoided if connections are recycled.

- Simplified programming model.

  When using connection pooling, each individual thread can act as though it has created its own JDBC connection, allowing you to use straightforward JDBC programming techniques.

- Controlled resource usage.

  If you create a new connection every time a thread needs one rather than using connection pooling, your application's resource usage can be wasteful, and it could lead to unpredictable behaviors for your application when it is under a heavy load.

Using Connection Pooling with Connector/J

The concept of connection pooling in JDBC has been standardized through the JDBC 2.0 Optional interfaces, and all major application servers have implementations of these APIs that work with MySQL Connector/J.

Generally, you configure a connection pool in your application server configuration files, and access it through the Java Naming and Directory Interface (JNDI). The following code shows how you might use a connection pool from an application deployed in a J2EE application server:

Example 8.1 Connector/J: Using a connection pool with a J2EE application server

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import javax.naming.InitialContext;
import javax.sql.DataSource;


public class MyServletJspOrEjb {

    public void doSomething() throws Exception {
        /*
         * Create a JNDI Initial context to be able to
         *  lookup  the DataSource
         *
         * In production-level code, this should be cached as
         * an instance or static variable, as it can
         * be quite expensive to create a JNDI context.
         *
         * Note: This code only works when you are using servlets
         * or EJBs in a J2EE application server. If you are
         * using connection pooling in standalone Java code, you
         * will have to create/configure datasources using whatever
         * mechanisms your particular connection pooling library
         * provides.
         */

        InitialContext ctx = new InitialContext();

         /*
          * Lookup the DataSource, which will be backed by a pool
          * that the application server provides. DataSource instances
          * are also a good candidate for caching as an instance
          * variable, as JNDI lookups can be expensive as well.
          */

        DataSource ds =
          (DataSource)ctx.lookup("java:comp/env/jdbc/MySQLDB");

        /*
         * The following code is what would actually be in your
         * Servlet, JSP or EJB 'service' method...where you need
         * to work with a JDBC connection.
         */

        Connection conn = null;
        Statement stmt = null;

        try {
            conn = ds.getConnection();

            /*
             * Now, use normal JDBC programming to work with
             * MySQL, making sure to close each resource when you're
             * finished with it, which permits the connection pool
             * resources to be recovered as quickly as possible
             */

            stmt = conn.createStatement();
            stmt.execute("SOME SQL QUERY");

            stmt.close();
            stmt = null;

            conn.close();
            conn = null;
        } finally {
            /*
             * close any jdbc instances here that weren't
             * explicitly closed during normal code path, so
```

```
             * that we don't 'leak' resources...
             */

            if (stmt != null) {
                try {
                    stmt.close();
                } catch (sqlexception sqlex) {
                    // ignore, as we can't do anything about it here
                }

                stmt = null;
            }

            if (conn != null) {
                try {
                    conn.close();
                } catch (sqlexception sqlex) {
                    // ignore, as we can't do anything about it here
                }

                conn = null;
            }
        }
    }
}
```

As shown in the example above, after obtaining the JNDI `InitialContext`, and looking up the `DataSource`, the rest of the code follows familiar JDBC conventions.

When using connection pooling, always make sure that connections, and anything created by them (such as statements or result sets) are closed. This rule applies no matter what happens in your code (exceptions, flow-of-control, and so forth). When these objects are closed, they can be re-used; otherwise, they will be stranded, which means that the MySQL server resources they represent (such as buffers, locks, or sockets) are tied up for some time, or in the worst case can be tied up forever.

Sizing the Connection Pool

Each connection to MySQL has overhead (memory, CPU, context switches, and so forth) on both the client and server side. Every connection limits how many resources there are available to your application as well as the MySQL server. Many of these resources will be used whether or not the connection is actually doing any useful work! Connection pools can be tuned to maximize performance, while keeping resource utilization below the point where your application will start to fail rather than just run slower.

The optimal size for the connection pool depends on anticipated load and average database transaction time. In practice, the optimal connection pool size can be smaller than you might expect. If you take Oracle's Java Petstore blueprint application for example, a connection pool of 15-20 connections can serve a relatively moderate load (600 concurrent users) using MySQL and Tomcat with acceptable response times.

To correctly size a connection pool for your application, create load test scripts with tools such as Apache JMeter or The Grinder, and load test your application.

An easy way to determine a starting point is to configure your connection pool's maximum number of connections to be unbounded, run a load test, and measure the largest amount of concurrently used connections. You can then work backward from there to determine what values of minimum and maximum pooled connections give the best performance for your particular application.

Validating Connections

MySQL Connector/J can validate the connection by executing a lightweight ping against a server. In the case of load-balanced connections, this is performed against all active pooled internal connections that are

retained. This is beneficial to Java applications using connection pools, as the pool can use this feature to validate connections. Depending on your connection pool and configuration, this validation can be carried out at different times:

1. Before the pool returns a connection to the application.
2. When the application returns a connection to the pool.
3. During periodic checks of idle connections.

To use this feature, specify a validation query in your connection pool that starts with `/* ping */`. Note that the syntax must be exactly as specified. This will cause the driver send a ping to the server and return a dummy lightweight result set. When using a `ReplicationConnection` or `LoadBalancedConnection`, the ping will be sent across all active connections.

It is critical that the syntax be specified correctly. The syntax needs to be exact for reasons of efficiency, as this test is done for every statement that is executed:

```
protected static final String PING_MARKER = "/* ping */";
...
if (sql.charAt(0) == '/') {
if (sql.startsWith(PING_MARKER)) {
doPingInstead();
...
```

None of the following snippets will work, because the ping syntax is sensitive to whitespace, capitalization, and placement:

```
sql = "/* PING */ SELECT 1";
sql = "SELECT 1 /* ping*/";
sql = "/*ping*/ SELECT 1";
sql = " /* ping */ SELECT 1";
sql = "/*to ping or not to ping*/ SELECT 1";
```

All of the previous statements will issue a normal `SELECT` statement and will **not** be transformed into the lightweight ping. Further, for load-balanced connections, the statement will be executed against one connection in the internal pool, rather than validating each underlying physical connection. This results in the non-active physical connections assuming a stale state, and they may die. If Connector/J then re-balances, it might select a dead connection, resulting in an exception being passed to the application. To help prevent this, you can use `loadBalanceValidateConnectionOnSwapServer` to validate the connection before use.

If your Connector/J deployment uses a connection pool that allows you to specify a validation query, take advantage of it, but ensure that the query starts *exactly* with `/* ping */`. This is particularly important if you are using the load-balancing or replication-aware features of Connector/J, as it will help keep alive connections which otherwise will go stale and die, causing problems later.

Chapter 9 Multi-Host Connections

Table of Contents

The following sections discuss a number of topics that involve multi-host connections, namely, server load-balancing, failover, and replication.

Developers should know the following things about multi-host connections that are managed through Connector/J:

- Each multi-host connection is a wrapper of the underlying physical connections.
- Each of the underlying physical connections has its own session. Sessions cannot be tracked, shared, or copied, given the MySQL architecture.
- Every switch between physical connections means a switch between sessions.
- Within a transaction boundary, there are no switches between physical connections. Beyond a transaction boundary, there is no guarantee that a switch does not occur.

> **Note**
>
> If an application reuses session-scope data (for example, variables, SSPs) beyond a transaction boundary, failures are possible, as a switch between the physical connections (which is also a switch between sessions) might occur. Therefore, the application should re-prepare the session data and also restart the last transaction in case of an exception, or it should re-prepare session data for each new transaction if it does not want to deal with exception handling.

9.1 Configuring Server Failover for Connections Using JDBC

MySQL Connector/J supports server failover. A failover happens when connection-related errors occur for an underlying, active connection. The connection errors are, by default, propagated to the client, which has to handle them by, for example, recreating the working objects (`Statement`, `ResultSet`, etc.) and restarting the processes. Sometimes, the driver might eventually fall back to the original host automatically before the client application continues to run, in which case the host switch is transparent and the client application will not even notice it.

A connection using failover support works just like a standard connection: the client does not experience any disruptions in the failover process. This means the client can rely on the same connection instance even if two successive statements might be executed on two different physical hosts. However, this does not mean the client does not have to deal with the exception that triggered the server switch.

The failover is configured at the initial setup stage of the server connection by the connection URL (see explanations for its format here):

```
jdbc:mysql://[primary host][:port],[secondary host 1][:port][,[secondary host 2][:port]]...[/[database]]»
[?propertyName1=propertyValue1[&propertyName2=propertyValue2]...]
```

The host list in the connection URL comprises of two types of hosts, the primary and the secondary. When starting a new connection, the driver always tries to connect to the primary host first and, if required, fails over to the secondary hosts on the list sequentially when communication problems are experienced. Even if the initial connection to the primary host fails and the driver gets connected to a secondary host, the primary host never loses its special status: for example, it can be configured with an access mode distinct from those of the secondary hosts, and it can be put on a higher priority when a host is to be picked during a failover process.

The failover support is configured by the following connection properties (their functions are explained in the paragraphs below):

- `failOverReadOnly`
- `secondsBeforeRetrySource`
- `queriesBeforeRetrySource`
- `retriesAllDown`
- `autoReconnect`
- `autoReconnectForPools`

Configuring Connection Access Mode

As with any standard connection, the initial connection to the primary host is in read/write mode. However, if the driver fails to establish the initial connection to the primary host and it automatically switches to the next host on the list, the access mode now depends on the value of the property `failOverReadOnly`, which is “true” by default. The same happens if the driver is initially connected to the primary host and, because of some connection failure, it fails over to a secondary host. Every time the connection falls back to the primary host, its access mode will be read/write, irrespective of whether or not the primary host has been connected to before. The connection access mode can be changed any time at runtime by calling the method `Connection.setReadOnly(boolean)`, which partially overrides the property `failOverReadOnly`. When `failOverReadOnly=false` and the access mode is explicitly set to either true or false, it becomes the mode for every connection after a host switch, no matter what host type are being connected to; but, if `failOverReadOnly=true`, changing the access mode to read/write is only possible if the driver is connecting to the primary host; however, even if the access mode cannot be changed for the current connection, the driver remembers the client's last intention and, when falling back to the primary host, that is the mode that will be used. For an illustration, see the following successions of events with a two-host connection.

- Sequence A, with `failOverReadOnly=true`:
  1. Connects to primary host in read/write mode
  2. Sets `Connection.setReadOnly(true)`; primary host now in read-only mode
  3. Failover event; connects to secondary host in read-only mode
  4. Sets `Connection.setReadOnly(false)`; secondary host remains in read-only mode
  5. Falls back to primary host; connection now in read/write mode
- Sequence B, with `failOverReadOnly=false`

1. Connects to primary host in read/write mode
2. Sets `Connection.setReadOnly(true)`; primary host now in read-only mode
3. Failover event; connects to secondary host in read-only mode
4. Set `Connection.setReadOnly(false)`; connection to secondary host switches to read/write mode
5. Falls back to primary host; connection now in read/write mode

The difference between the two scenarios is in step 4: the access mode for the secondary host in sequence A does not change at that step, but the driver remembers and uses the set mode when falling back to the primary host, which would be read-only otherwise; but in sequence B, the access mode for the secondary host changes immediately.

Configuring Fallback to Primary Host

As already mentioned, the primary host is special in the failover arrangement when it comes to the host's access mode. Additionally, the driver tries to fall back to the primary host as soon as possible by default, even if no communication exception occurs. Two properties, `secondsBeforeRetrySource` and `queriesBeforeRetrySource`, determine when the driver is ready to retry a reconnection to the primary host (the `Source` in the property names stands for the primary host of our connection URL, which is not necessarily a source host in a replication setup):

- `secondsBeforeRetrySource` determines how much time the driver waits before trying to fall back to the primary host
- `queriesBeforeRetrySource` determines the number of queries that are executed before the driver tries to fall back to the primary host. Note that for the driver, each call to a `Statement.execute*()` method increments the query execution counter; therefore, when calls are made to `Statement.executeBatch()` or if `allowMultiQueries` or `rewriteBatchStatements` are enabled, the driver may not have an accurate count of the actual number of queries executed on the server. Also, the driver calls the `Statement.execute*()` methods internally in several occasions. All these mean you can only use `queriesBeforeRetrySource` only as a coarse specification for when to fall back to the primary host.

In general, an attempt to fallback to the primary host is made when at least one of the conditions specified by the two properties is met, and the attempt always takes place at transaction boundaries. However, if auto-commit is turned off, the check happens only when the method `Connection.commit()` or `Connection.rollback()` is called. The automatic fallback to the primary host can be turned off by setting simultaneously `secondsBeforeRetrySource` and `queriesBeforeRetrySource` to “0”. Setting only one of the properties to “0” only disables one part of the check.

Configuring Reconnection Attempts

When establishing a new connection or when a failover event occurs, the driver tries to connect successively to the next candidate on the host list. When the end of the list has been reached, it restarts all over again from the beginning of the list; however, the primary host is skipped over, if (a) NOT all the secondary hosts have already been tested at least once, AND (b) the fallback conditions defined by `secondsBeforeRetrySource` and `queriesBeforeRetrySource` are not yet fulfilled. Each run-through of the whole host list, (which is not necessarily completed at the end of the host list) counts as a single connection attempt. The driver tries as many connection attempts as specified by the value of the property `retriesAllDown`.

Seamless Reconnection

Although not recommended, you can make the driver perform failovers without invalidating the active `Statement` or `ResultSet` instances by setting either the parameter `autoReconnect` or `autoReconnectForPools` to `true`. This allows the client to continue using the same object instances after a failover event, without taking any exceptional measures. This, however, may lead to unexpected results: for example, if the driver is connected to the primary host with read/write access mode and it fails-over to a secondary host in read-only mode, further attempts to issue data-changing queries will result in errors, and the client will not be aware of that. This limitation is particularly relevant when using data streaming: after the failover, the `ResultSet` looks to be alright, but the underlying connection may have changed already, and no backing cursor is available anymore.

Configuring Server Failover Using JDBC with DNS SRV

See Section 6.14, “Support for DNS SRV Records” for details.

9.2 Configuring Server Failover for Connections Using X DevAPI

When using the X Protocol, Connector/J supports a client-side failover feature for establishing a Session. If multiple hosts are specified in the connection URL, when Connector/J fails to connect to a listed host, it tries to connect to another one. This is a sample X DevAPI URL for configuring client-side failover:

```
mysqlx://sandy:mypassword@[host1:33060,host2:33061]/test
```

With the client-side failover configured, when there is a failure to establish a connection, Connector/J keeps attempting to connect to a host on the host list. The order in which the hosts are attempted for connection is as follows:

- *For connections with the `priority` property set for each host in the connection URL,* hosts are attempted according to the set priorities for the hosts, which are specified by any numbers between 0 to 100, with a larger number indicating a higher priority for connection. For example:

  ```
  mysqlx://sandy:mypassword@[(address=host1:33060,priority=2),(address=host2:33061,priority=1)]/test
  ```

  In this example, `host1` is always attempted before `host2` when new sessions are created.

  Priorities should either be set for all or no hosts.

- *For connections with the `priority` property NOT set for each host in the connection URL,* hosts are attempted one after another in a random order.

Notice that the server failover feature for X DevAPI only allows for a failover when Connector/J is trying to establish a connection, but not during operations after a connection has already been made.

Connection Pooling Using X DevAPI. When using connection pooling with X DevAPI, Connector/J keeps track of any host it failed to connect to and, for a short waiting period after the failure, avoids connecting to it during the creation or retrieval of a `Session`. However, if all other hosts have already been tried, those excluded hosts will be retried without waiting. Once all hosts have been tried and no connections can be established, Connector/J throws a `com.mysql.cj.exceptions.CJCommunicationsException` and returns the message `Unable to connect to any of the target hosts`.

Configuring Server Failover Using X DevAPI with DNS SRV

See Section 6.14, “Support for DNS SRV Records” for details.

9.3 Configuring Load Balancing with Connector/J

Connector/J has long provided an effective means to distribute read/write load across multiple MySQL server instances for Cluster or source-source replication deployments. You can dynamically configure load-balanced connections, with no service outage. In-process transactions are not lost, and no application exceptions are generated if any application is trying to use that particular server instance.

The load balancing is configured at the initial setup stage of the server connection by the following connection URL, which has a similar format as the general JDBC URL for MySQL connection, but a specialized scheme:

```
jdbc:mysql:loadbalance://[host1][:port],[host2][:port][,[host3][:port]]...[/[database]] »
[?propertyName1=propertyValue1[&propertyName2=propertyValue2]...]
```

There are two configuration properties associated with this functionality:

- `loadBalanceConnectionGroup` – This provides the ability to group connections from different sources. This allows you to manage these JDBC sources within a single class loader in any combination you choose. If they use the same configuration, and you want to manage them as a logical single group, give them the same name. This is the key property for management: if you do not define a name (string) for `loadBalanceConnectionGroup`, you cannot manage the connections. All load-balanced connections sharing the same `loadBalanceConnectionGroup` value, regardless of how the application creates them, will be managed together.
- `ha.enableJMX` – The ability to manage the connections is exposed when you define a `loadBalanceConnectionGroup`; but if you want to manage this externally, enable JMX by setting this property to `true`. This enables a JMX implementation, which exposes the management and monitoring operations of a connection group. Further, start your application with the `-Dcom.sun.management.jmxremote` JVM flag. You can then perform connect and perform operations using a JMX client such as `jconsole`.

Once a connection has been made using the correct connection properties, a number of monitoring properties are available:

- Current active host count.
- Current active physical connection count.
- Current active logical connection count.
- Total logical connections created.
- Total transaction count.

The following management operations can also be performed:

- Add host.
- Remove host.

The JMX interface, `com.mysql.cj.jdbc.jmx.LoadBalanceConnectionGroupManagerMBean`, has the following methods:

- `int getActiveHostCount(String group);`
- `int getTotalHostCount(String group);`
- `long getTotalLogicalConnectionCount(String group);`

- `long getActiveLogicalConnectionCount(String group);`
- `long getActivePhysicalConnectionCount(String group);`
- `long getTotalPhysicalConnectionCount(String group);`
- `long getTotalTransactionCount(String group);`
- `void removeHost(String group, String host) throws SQLException;`
- `void stopNewConnectionsToHost(String group, String host) throws SQLException;`
- `void addHost(String group, String host, boolean forExisting);`
- `String getActiveHostsList(String group);`
- `String getRegisteredConnectionGroups();`

The `getRegisteredConnectionGroups()` method returns the names of all connection groups defined in that class loader.

You can test this setup with the following code:

```
public class Test {

    private static String URL = "jdbc:mysql:loadbalance://" +
        "localhost:3306,localhost:3310/test?" +
        "loadBalanceConnectionGroup=first&ha.enableJMX=true";

    public static void main(String[] args) throws Exception {
        new Thread(new Repeater()).start();
        new Thread(new Repeater()).start();
        new Thread(new Repeater()).start();
    }

    static Connection getNewConnection() throws SQLException, ClassNotFoundException {
        Class.forName("com.mysql.cj.jdbc.Driver");
        return DriverManager.getConnection(URL, "root", "");
    }

    static void executeSimpleTransaction(Connection c, int conn, int trans){
        try {
            c.setAutoCommit(false);
            Statement s = c.createStatement();
            s.executeQuery("SELECT SLEEP(1) /* Connection: " + conn + ", transaction: " + trans + " */");
            c.commit();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static class Repeater implements Runnable {
        public void run() {
            for(int i=0; i < 100; i++){
                try {
                    Connection c = getNewConnection();
                    for(int j=0; j < 10; j++){
                        executeSimpleTransaction(c, i, j);
                        Thread.sleep(Math.round(100 * Math.random()));
                    }
                    c.close();
                    Thread.sleep(100);
                } catch (Exception e) {
                    e.printStackTrace();
```

```
                }
            }
        }
    }
}
```

After compiling, the application can be started with the `-Dcom.sun.management.jmxremote` flag, to enable remote management. `jconsole` can then be started. The `Test` main class will be listed by `jconsole`. Select this and click **Connect**. You can then navigate to the `com.mysql.cj.jdbc.jmx.LoadBalanceConnectionGroupManager` bean. At this point, you can click on various operations and examine the returned result.

If you now had an additional instance of MySQL running on port 3309, you could ensure that Connector/J starts using it by using the `addHost()`, which is exposed in `jconsole`. Note that these operations can be performed dynamically without having to stop the application running.

For further information on the combination of load balancing and failover, see Section 9.5, “Advanced Load-balancing and Failover Configuration”.

Configuring Load Balancing with DNS SRV

See Section 6.14, “Support for DNS SRV Records” for details.

9.4 Configuring Source/Replica Replication with Connector/J

This section describe a number of features of Connector/J's support for replication-aware deployments.

The replication is configured at the initial setup stage of the server connection by the connection URL, which has a similar format as the general JDBC URL for MySQL connection, but a specialized scheme:

```
jdbc:mysql:replication://[source host][:port],[replica host 1][:port][,[replica host 2][:port]]...[/[databa
[?propertyName1=propertyValue1[&propertyName2=propertyValue2]...]
```

Users may specify the property `allowSourceDownConnections=true` to allow `Connection` objects to be created even though no source hosts are reachable. Such `Connection` objects report they are read-only, and `isSourceConnection()` returns false for them. The `Connection` tests for available source hosts when `Connection.setReadOnly(false)` is called, throwing an SQLException if it cannot establish a connection to a source, or switching to a source connection if the host is available.

Users may specify the property `allowReplicasDownConnections=true` to allow `Connection` objects to be created even though no replica hosts are reachable. A `Connection` then, at runtime, tests for available replica hosts when `Connection.setReadOnly(true)` is called (see explanation for the method below), throwing an SQLException if it cannot establish a connection to a replica, unless the property `readFromSourceWhenNoReplicas` is set to be “true” (see below for a description of the property).

Scaling out Read Load by Distributing Read Traffic to Replicas

Connector/J supports replication-aware connections. It can automatically send queries to a read/write source host, or a failover or round-robin loadbalanced set of replicas based on the state of `Connection.getReadOnly()`.

An application signals that it wants a transaction to be read-only by calling `Connection.setReadOnly(true)`. The replication-aware connection will use one of the replica connections, which are load-balanced per replica host using a round-robin scheme. A given connection is sticky to a replica until a transaction boundary command

(a commit or rollback) is issued, or until the replica is removed from service. After calling `Connection.setReadOnly(true)`, if you want to allow connection to a source when no replicas are available, set the property `readFromSourceWhenNoReplicas` to “true.” Notice that the source host will be used in read-only state in those cases, as if it is a replica host. Also notice that setting `readFromSourceWhenNoReplicas=true` might result in an extra load for the source host in a transparent manner.

If you have a write transaction, or if you have a read that is time-sensitive (remember, replication in MySQL is asynchronous), set the connection to be not read-only, by calling `Connection.setReadOnly(false)` and the driver will ensure that further calls are sent to the source MySQL server. The driver takes care of propagating the current state of autocommit, isolation level, and catalog between all of the connections that it uses to accomplish this load balancing functionality.

To enable this functionality, use the specialized replication scheme (`jdbc:mysql:replication://`) when connecting to the server.

Here is a short example of how a replication-aware connection might be used in a standalone application:

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.util.Properties;
import java.sql.DriverManager;

public class ReplicationDemo {

  public static void main(String[] args) throws Exception {

    Properties props = new Properties();

    // We want this for failover on the replicas
    props.put("autoReconnect", "true");

    // We want to load balance between the replicas
    props.put("roundRobinLoadBalance", "true");

    props.put("user", "foo");
    props.put("password", "password");

    //
    // Looks like a normal MySQL JDBC url, with a
    // comma-separated list of hosts, the first
    // being the 'source', the rest being any number
    // of replicas that the driver will load balance against
    //

    Connection conn =
        DriverManager.getConnection("jdbc:mysql:replication://source,replica1,replica2,replica3/test",
            props);

    //
    // Perform read/write work on the source
    // by setting the read-only flag to "false"
    //

    conn.setReadOnly(false);
    conn.setAutoCommit(false);
    conn.createStatement().executeUpdate("UPDATE some_table ....");
    conn.commit();

    //
    // Now, do a query from a replica, the driver automatically picks one
    // from the list
    //
```

```
    conn.setReadOnly(true);

    ResultSet rs =
      conn.createStatement().executeQuery("SELECT a,b FROM alt_table");

      .......
  }
}
```

Consider using the Load Balancing JDBC Pool (`lbpool`) tool, which provides a wrapper around the standard JDBC driver and enables you to use DB connection pools that includes checks for system failures and uneven load distribution. For more information, see Load Balancing JDBC Driver for MySQL (mysql-lbpool).

Support for Multiple-Source Replication Topographies

Connector/J supports multi-source replication topographies.

The connection URL for replication discussed earlier (i.e., in the format of `jdbc:mysql:replication://source,replica1,replica2,replica3/test`) assumes that the first (and only the first) host is the source host. Supporting deployments with an arbitrary number of sources and replicas requires the "address-equals" URL syntax for multiple host connection discussed in Section 6.2, “Connection URL Syntax”, with the property `type=[source|replica]`; for example:

```
jdbc:mysql:replication://address=(type=source)(host=source1host),address=(type=source)(host=source2host),a
```

Connector/J uses a load-balanced connection internally for management of the source connections, which means that `ReplicationConnection`, when configured to use multiple sources, exposes the same options to balance load across source hosts as described in Section 9.3, “Configuring Load Balancing with Connector/J”.

Live Reconfiguration of Replication Topography

Connector/J also supports live management of replication host (single or multi-source) topographies. This enables users to promote replicas for Java applications without requiring an application restart.

The replication hosts are most effectively managed in the context of a replication connection group. A ReplicationConnectionGroup class represents a logical grouping of connections which can be managed together. There may be one or more such replication connection groups in a given Java class loader (there can be an application with two different JDBC resources needing to be managed independently). This key class exposes host management methods for replication connections, and `ReplicationConnection` objects register themselves with the appropriate `ReplicationConnectionGroup` if a value for the new `replicationConnectionGroup` property is specified. The `ReplicationConnectionGroup` object tracks these connections until they are closed, and it is used to manipulate the hosts associated with these connections.

Some important methods related to host management include:

- `getSourceHosts()`: Returns a collection of strings representing the hosts configured as source hosts
- `getReplicaHosts()`: Returns a collection of strings representing the hosts configured as replica hosts
- `addReplicaHost(String host)`: Adds new host to pool of possible replica hosts for selection at start of new read-only workload
- `promoteReplicaToSource(String host)`: Removes the host from the pool of potential replica hosts for future read-only processes (existing read-only process is allowed to continue to completion) and adds the host to the pool of potential source hosts

- `removeReplicaHost(String host, boolean closeGently)`: Removes the host (host name match must be exact) from the list of configured replica hosts; if `closeGently` is false, existing connections which have this host as currently active will be closed hardly (application should expect exceptions)
- `removeSourceHost(String host, boolean closeGently)`: Same as `removeReplicaHost()`, but removes the host from the list of configured source hosts

Some useful management metrics include:

- `getConnectionCountWithHostAsReplica(String host)`: Returns the number of ReplicationConnection objects that have the given host configured as a possible replica host
- `getConnectionCountWithHostAsSource(String host)`: Returns the number of ReplicationConnection objects that have the given host configured as a possible source host
- `getNumberOfReplicasAdded()`: Returns the number of times a replica host has been dynamically added to the group pool
- `getNumberOfReplicasRemoved()`: Returns the number of times a replica host has been dynamically removed from the group pool
- `getNumberOfReplicaPromotions()`: Returns the number of times a replica host has been promoted to be a source host
- `getTotalConnectionCount()`: Returns the number of ReplicationConnection objects which have been registered with this group
- `getActiveConnectionCount()`: Returns the number of ReplicationConnection objects currently being managed by this group

ReplicationConnectionGroupManager

`com.mysql.cj.jdbc.ha.ReplicationConnectionGroupManager` provides access to the replication connection groups, together with some utility methods.

- `getConnectionGroup(String groupName)`: Returns the `ReplicationConnectionGroup` object matching the groupName provided

The other methods in `ReplicationConnectionGroupManager` mirror those of `ReplicationConnectionGroup`, except that the first argument is a String group name. These methods will operate on all matching ReplicationConnectionGroups, which are helpful for removing a server from service and have it decommissioned across all possible `ReplicationConnectionGroups`.

These methods might be useful for in-JVM management of replication hosts if an application triggers topography changes. For managing host configurations from outside the JVM, JMX can be used.

Using JMX for Managing Replication Hosts

When Connector/J is started with `ha.enableJMX=true` and a value set for the property `replicationConnectionGroup`, a JMX MBean will be registered, allowing manipulation of replication hosts by a JMX client. The MBean interface is defined in `com.mysql.cj.jdbc.jmx.ReplicationGroupManagerMBean`, and leverages the `ReplicationConnectionGroupManager` static methods:

```
public abstract void addReplicaHost(String groupFilter, String host) throws SQLException;
public abstract void removeReplicaHost(String groupFilter, String host) throws SQLException;
```

```
public abstract void promoteReplicaToSource(String groupFilter, String host) throws SQLException;
public abstract void removeSourceHost(String groupFilter, String host) throws SQLException;
public abstract String getSourceHostsList(String group);
public abstract String getReplicaHostsList(String group);
public abstract String getRegisteredConnectionGroups();
public abstract int getActiveSourceHostCount(String group);
public abstract int getActiveReplicaHostCount(String group);
public abstract int getReplicaPromotionCount(String group);
public abstract long getTotalLogicalConnectionCount(String group);
public abstract long getActiveLogicalConnectionCount(String group);
```

Configuring Source/Replica Replication with DNS SRV

See Section 6.14, “Support for DNS SRV Records” for details.

9.5 Advanced Load-balancing and Failover Configuration

Connector/J provides a useful load-balancing implementation for MySQL Cluster or multi-source deployments, as explained in Section 9.3, “Configuring Load Balancing with Connector/J” and Support for Multiple-Source Replication Topographies. This same implementation is used for balancing load between read-only replicas for replication-aware connections.

When trying to balance workload between multiple servers, the driver has to determine when it is safe to swap servers, doing so in the middle of a transaction, for example, could cause problems. It is important not to lose state information. For this reason, Connector/J will only try to pick a new server when one of the following happens:

1. At transaction boundaries (transactions are explicitly committed or rolled back).
2. A communication exception (SQL State starting with "08") is encountered.
3. When a `SQLException` matches conditions defined by user, using the extension points defined by the `loadBalanceSQLStateFailover`, `loadBalanceSQLExceptionSubclassFailover` or `loadBalanceExceptionChecker` properties.

The third condition revolves around three properties, which allow you to control which `SQLException`s trigger failover:

- `loadBalanceExceptionChecker` - The `loadBalanceExceptionChecker` property is really the key. This takes a fully-qualified class name which implements the new `com.mysql.cj.jdbc.ha.LoadBalanceExceptionChecker` interface. This interface is very simple, and you only need to implement the following method:

  ```
  public boolean shouldExceptionTriggerFailover(SQLException ex)
  ```

  A `SQLException` is passed in, and a boolean returned. A value of `true` triggers a failover, `false` does not.

  You can use this to implement your own custom logic. An example where this might be useful is when dealing with transient errors when using MySQL Cluster, where certain buffers may become overloaded. The following code snippet illustrates this:

  ```
  public class NdbLoadBalanceExceptionChecker
   extends StandardLoadBalanceExceptionChecker {

   public boolean shouldExceptionTriggerFailover(SQLException ex) {
    return super.shouldExceptionTriggerFailover(ex)
      ||  checkNdbException(ex);
  ```

```
 }

 private boolean checkNdbException(SQLException ex){
 // Have to parse the message since most NDB errors
 // are mapped to the same DEMC.
  return (ex.getMessage().startsWith("Lock wait timeout exceeded") ||
  (ex.getMessage().startsWith("Got temporary error")
  && ex.getMessage().endsWith("from NDB")));
 }
}
```

The code above extends `com.mysql.cj.jdbc.ha.StandardLoadBalanceExceptionChecker`, which is the default implementation. There are a few convenient shortcuts built into this, for those who want to have some level of control using properties, without writing Java code. This default implementation uses the two remaining properties: `loadBalanceSQLStateFailover` and `loadBalanceSQLExceptionSubclassFailover`.

- `loadBalanceSQLStateFailover` - allows you to define a comma-delimited list of `SQLState` code prefixes, against which a `SQLException` is compared. If the prefix matches, failover is triggered. So, for example, the following would trigger a failover if a given `SQLException` starts with "00", or is "12345":

```
loadBalanceSQLStateFailover=00,12345
```

- `loadBalanceSQLExceptionSubclassFailover` - can be used in conjunction with `loadBalanceSQLStateFailover` or on its own. If you want certain subclasses of `SQLException` to trigger failover, simply provide a comma-delimited list of fully-qualified class or interface names to check against. For example, if you want all `SQLTransientConnectionExceptions` to trigger failover, you would specify:

```
loadBalanceSQLExceptionSubclassFailover=java.sql.SQLTransientConnectionException
```

While the three failover conditions enumerated earlier suit most situations, if `autocommit` is enabled, Connector/J never re-balances, and continues using the same physical connection. This can be problematic, particularly when load-balancing is being used to distribute read-only load across multiple replicas. However, Connector/J can be configured to re-balance after a certain number of statements are executed, when `autocommit` is enabled. This functionality is dependent upon the following properties:

- `loadBalanceAutoCommitStatementThreshold` – defines the number of matching statements which will trigger the driver to potentially swap physical server connections. The default value, 0, retains the behavior that connections with `autocommit` enabled are never balanced.

- `loadBalanceAutoCommitStatementRegex` – the regular expression against which statements must match. The default value, blank, matches all statements. So, for example, using the following properties will cause Connector/J to re-balance after every third statement that contains the string “test”:

```
loadBalanceAutoCommitStatementThreshold=3
loadBalanceAutoCommitStatementRegex=.*test.*
```

`loadBalanceAutoCommitStatementRegex` can prove useful in a number of situations. Your application may use temporary tables, server-side session state variables, or connection state, where letting the driver arbitrarily swap physical connections before processing is complete could cause data loss or other problems. This allows you to identify a trigger statement that is only executed when it is safe to swap physical connections.

Configuring Load Balancing and Failover with DNS SRV

See Section 6.14, “Support for DNS SRV Records” for details.

Chapter 10 Using the X DevAPI with Connector/J: Special Topics

Table of Contents

Connector/J 9.0 supports the X DevAPI, through which native support by MySQL for JSON, NoSQL, document collection, and other features are provided to Java applications. See Using MySQL as a Document Store, the X DevAPI User Guide, and the *Connector/J X DevAPI Reference* available at Connectors and APIs for details.

Information on using the X DevAPI with Connector/J can be found in different chapters in this manual. This chapter explores some special topics that are not covered elsewhere.

10.1 Connection Compression Using X DevAPI

Connector/J supports data compression for X DevAPI connections when working with MySQL Server 8.0.19 and later. General details about this feature can be found in Connection Compression with X Plugin. For details on how to configure connection compression for Connector/J, see the descriptions for the connection properties `xdevapi.compression`, `xdevapi.compression-algorithms`, and `xdevapi.compression-extensions` in Section 6.3, "Configuration Properties". The following is a summary of the feature:

The compression algorithms to be negotiated with the server and the priority of negotiation can be specified using the connection property `xdevapi.compression-algorithms`. It accepts a list of *`[algorithm-name]_[operation-mode]`*, separated by commas (,). If the property is not set, the default value of "`zstd_stream,lz4_message,deflate_stream`" is used. The priority for negotiation follows the order the algorithms appear in the list. Setting an empty string explicitly for the property means compression should be disabled for the connection.

> **Note**
>
> When specifying compression algorithms with `xdevapi.compression-algorithms`, the aliases `zstd`, `lz4`, and `deflate` can be used in place of `zstd_stream`, `lz4_message`, and `deflate_stream`, respectively.

Out of all the compression algorithms now supported by MySQL Server for X DevAPI connections, Connector/J provides out-of-the-box support for Deflate only; this is because none of the other compression algorithms (LZ4 and zstd, for now) are natively supported by the existing JREs. To support those algorithms, the client application must provide implementations for the corresponding deflate and inflate operations in the form of an `OutputStream` and an `InputStream` object, respectively. The easiest way to accomplish this is by using a third-party library such as the Apache Commons Compress library, which supports LZ4 and zstd. The connection option `xdevapi.compression-extensions` allows users to configure Connector/J to use any compression algorithm that is supported by MySQL Server, as long as there is a Java implementation for that algorithm. The option takes a list of triplets separated by commas (,), and each triplet in turn contains the following elements, separated by colons (:):

- The compression algorithm name, indicated by the identifier used by the server (see Connection Compression with X Plugin; aliases mentioned in the Note above can be used).
- A fully-qualified name of a class implementing the interface `java.io.InputStream` that will be used to *inflate* data compressed with the named algorithm.

- A fully-qualified name of a class implementing the interface `java.io.OutputStream` that will be used to *deflate* data using the named algorithm.

Here is an example that sets up the support for the algorithms `lz4_message` and `zstd_stream` using the Apache Commons Compress library:

```
String connStr = "jdbc:mysql://johndoe:secret@localhost:33060/mydb?"
    + "xdevapi.compression-extensions="
    + "lz4_message"+":" // LZ4 triplet
    + FramedLZ4CompressorInputStream.class.getName() + ":"
    + FramedLZ4CompressorOutputStream.class.getName() + ","
    + "zstd_stream"+":" // zstd triplet
    + ZstdCompressorInputStream.class.getName() + ":"
    + ZstdCompressorOutputStream.class.getName();
SessionFactory sessFact = new SessionFactory();
Session sess = sessFact.getSession(connStr);
Collection col = sess.getDefaultSchema().getCollection("myCollection");
// (...)
sess.close();
```

> **Note**
>
> *For Connector/J 8.0.21 and earlier:* The connection property `xdevapi.compression-extensions` described above is named `xdevapi.compression-algorithm` for Connector/J 8.0.21 and earlier, and the elements in each triplet should be separated by commas (,) instead of colons (:).

Negotiation for a compression algorithm is attempted by default (`xdevapi.compression`=`Preferred` by default), unless the connection property `xdevapi.compression` is set to `DISABLED`. The final choice of compression algorithm depends on what algorithms are enabled on the server. By default, because compression is not required, if the negotiation fails, the connection will not be compressed, but the client will still be able to communicate with the server; however, if the connection property `xdevapi.compression` is set to `REQUIRED`, the connection attempt fails with an error if no algorithm can be negotiated for successfully.

10.2 Schema Validation

When working with MySQL Server 8.0.19 and later: Schema validation can be configured for a `Collection`, so that documents in the `Collection` are validated against a schema before they can be inserted or updated. This is done by specifying a JSON Schema during `Collection` creation or modification; schema validation is then performed by the server at a document creation or update, and an error is returned if the document does not validate against the assigned schema. For more information on JSON schema validation in MySQL, see JSON Schema Validation Functions. This section describes how to configure schema validation for a `Collection` with Connector/J.

To configure schema validation during the creation of a `Collection`, pass to the `createCollection()` method a `CreateCollectionOptions` object, which has these fields:

- `reuse`: a boolean set by the `setReuseExisting` method. If it is `true`, when the `Collection` to be created already exists within the `Schema` that is to contain it, Connector/J returns success (without any attempt to apply JSON schema to the existing `Collection`); in the same case, Connector/J returns an error if the parameter is set to `false`. If `reuse` is not set, it is taken to be `false`.
- `validation`: a `Validation` object set by the `setValidation()` method. A `Validation` object in turns contains these fields:
  - `level`: a enumeration of the class `ValidationLevel`, set by the `setLevel()` method; it can be one of the following two values:

  - `STRICT`: Strict validation. Attempting to insert or modify a document that violates the validation schema results in a server error being raised.
  - `OFF`: No validation. Schema validation is turned off.

  If `level` is not set, it is taken as `OFF` for MySQL Server 8.0.19, and `STRICT` for 8.0.20 and later.

- `schema`: A string representing a JSON Schema to be used to validate a `Document` in the `Collection`; set by the `setSchema()` method.

  If `schema` is not provided but `level` is set to STRICT, the `Collection` is validated against the default schema `{"type" : "object"}`.

This is an example of how to configure schema validation at the creation of a `Collection`:

```
Collection coll = this.schema.createCollection(collName,
    new CreateCollectionOptions()
        .setReuseExisting(false)
        .setValidation(new Validation()
           .setLevel(ValidationLevel.STRICT)
           .setSchema(
               "{\"id\": \"http://json-schema.org/geo\","
               + "\"$schema\": \"http://json-schema.org/draft-06/schema#\","
               + "       \"description\": \"A geographical coordinate\","
               + "       \"type\": \"object\","
               + "       \"properties\": {"
               + "          \"latitude\": {"
               + "             \"type\": \"number\""
               + "          },"
               + "          \"longitude\": {"
               + "             \"type\": \"number\""
               + "          }"
               + "       },"
               + "       \"required\": [\"latitude\", \"longitude\"]"
               + "  }"
)));
```

The set fields are accessible by the corresponding getter methods.

To modify the schema validation configuration for a `Collection`, use the `modifyCollection()` method and pass to it a `ModifyCollectionOptions` object, which has the same fields as the `CreateCollectionOptions` object except for the `reuse` field, which does not exist for a `ModifyCollectionOptions` object. For the `Validation` object of a `ModifyCollectionOptions` object, users can set either its `level` or `schema`, or both. Here is an example of using the `modifyCollection()` to change the schema validation configuration:

```
schema.modifyCollection(collName,
    new ModifyCollectionOptions()
        .setValidation(new Validation()
           .setLevel(ValidationLevel.OFF)
           .setSchema(
               "{\"id\": \"http://json-schema.org/geo\","
               + "\"$schema\": \"http://json-schema.org/draft-06/schema#\","
               + "       \"description\": \"NEW geographical coordinate\","
               + "       \"type\": \"object\","
               + "       \"properties\": {"
               + "          \"latitude\": {"
               + "             \"type\": \"number\""
               + "          },"
               + "          \"longitude\": {"
               + "             \"type\": \"number\""
```

```
                + "          }"
                + "       },"
                + "       \"required\": [\"latitude\", \"longitude\"]"
                + "  }"
)));
```

If the Collection contains documents that do not validate against the new JSON schema supplied through `ModifyCollectionOptions`, the server will reject the schema modification with the error `ERROR 5180 (HY000) Document is not valid according to the schema assigned to collection`.

> **Note**
>
> `createCollection()` and `modifyCollection()` are overloaded: they can be called without passing to them the `CreateCollectionOptions` or the `ModifyCollectionOptions`, respectively, in which case schema validation will not be applied to the `Collection`.

Chapter 11 Using the Connector/J Interceptor Classes

An interceptor is a software design pattern that provides a transparent way to extend or modify some aspect of a program, similar to a user exit. No recompiling is required. With Connector/J, the interceptors are enabled and disabled by updating the connection string to refer to different sets of interceptor classes that you instantiate.

The connection properties that control the interceptors are explained in Section 6.3, “Configuration Properties”:

- `connectionLifecycleInterceptors`, where you specify the fully qualified names of classes that implement the `com.mysql.cj.jdbc.interceptors.ConnectionLifecycleInterceptor` interface. In these kinds of interceptor classes, you might log events such as rollbacks, measure the time between transaction start and end, or count events such as calls to `setAutoCommit()`.
- `exceptionInterceptors`, where you specify the fully qualified names of classes that implement the `com.mysql.cj.exceptions.ExceptionInterceptor` interface. In these kinds of interceptor classes, you might add extra diagnostic information to exceptions that can have multiple causes or indicate a problem with server settings. `exceptionInterceptors` classes are called when handling an `Exception` thrown from Connector/J code.
- `queryInterceptors`, where you specify the fully qualified names of classes that implement the `com.mysql.cj.interceptors.QueryInterceptor` interface. In these kinds of interceptor classes, you might change or augment the processing done by certain kinds of statements, such as automatically checking for queried data in a `memcached` server, rewriting slow queries, logging information about statement execution, or route requests to remote servers.

Chapter 12 Using Logging Frameworks with SLF4J

Besides its default logger `com.mysql.cj.log.StandardLogger`, which logs to `stderr`, Connector/J supports the SLF4J logging facade, allowing end users of applications using Connector/J to plug in logging frameworks of their own choices at deployment time. Popular logging frameworks such as `java.util.logging`, `logback`, and `log4j` are supported by SLF4J. Follow these requirements to use a logging framework with SLF4J and Connector/J:

- In the development environment:
  - Install on your system `slf4j-api-x.y.z.jar` (available at https://www.slf4j.org/download.html) and add it to the Java classpath.
  - In the code of your application, obtain an `SLF4JLogger` as a `Log` instantiated within a `MysqlConnection` `Session`, and then use the `Log` instance for your logging.
- On the deployment system:
  - Install on your system `slf4j-api-x.y.z.jar` and add it to the Java classpath
  - Install on your system the SLF4J binding for the logging framework of your choice and add it to your Java classpath. SLF4J bindings are available at, for example, https://www.slf4j.org/manual.html#swapping.

    Note

    Do not put more than one SLF4J binding in you Java classpath. Switch from one logging framework to another by removing a binding and adding a new one to the classpath.
  - Install the logging framework of your choice on your system and add it to the Java classpath.
  - Configure the logging framework of your choice. This often consists of setting up appenders or handlers for log messages using a configuration file; see your logging framework's documentation for details.
  - When connecting the application to the MySQL Server, set the Connector/J connection property `logger` to `Slf4JLogger`.

The log category name used by Connector/J with SLF4J is `MySQL`. See the SLF4J user manual for more details about using SLF4J, including discussions on Maven dependency and bindings. Here is a sample code for using SLF4J with Connector/J:

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import com.mysql.cj.jdbc.JdbcConnection;
import com.mysql.cj.log.Log;

public class JDBCDemo {

  public static void main(String[] args) {

    Connection conn = null;
    Statement statement = null;
    ResultSet resultSet = null;
    Log logger = null;
```

```
  try {
     // Database parameters
     String url = "jdbc:mysql://myexample.com:3306/pets?logger=Slf4JLogger&explainSlowQueries=true";
     String user = "user";
     String password = "password";
     // create a connection to the database
     conn = DriverManager.getConnection(url, user, password);
     logger = ((JdbcConnection)conn).getSession().getLog();
  }
  catch (SQLException e) {
     System.err.println(e.getMessage());
     System.exit(1);
   }

  try {
     statement = conn.createStatement();
     resultSet = statement.executeQuery("SELECT * FROM pets.dogs");
     while(resultSet.next()){
       System.out.printf("%d\t%s\t%s\t %4$ty.%4$tm.%4$td \n",
       resultSet.getInt(1),
       resultSet.getString(2),
       resultSet.getString(3),
       resultSet.getDate(4));
     }
  }
  catch(SQLException e) {
     logger.logWarn("Warning: Select failed!");
  }

}

}
```

If you want to use, for example, Log4j 2.17.1 as your logging framework when running this program, put these JAR files in your Java classpath:

- `slf4j-api-2.0.3.jar` (SLF4J API module, available at, for example, https://central.sonatype.com/artifact/org.slf4j/slf4j-api/2.0.3/jar).
- `log4j-api-2.17.1.jar` and `log4j-core-2.17.1.jar` (Log4J library, available at, for example, https://central.sonatype.com/artifact/org.apache.logging.log4j/log4j-api/2.17.1/jar and https://central.sonatype.com/artifact/org.apache.logging.log4j/log4j-core/2.17.1/jar).
- `log4j-slf4j-impl-2.17.1.jar` (SLF4J's binding for Log4J 2.17.1, available at, for example, https://central.sonatype.com/artifact/org.apache.logging.log4j/log4j-slf4j-impl/2.17.1/jar).

Here is output of the program when the SELECT statement failed:

```
[2021-09-05 12:06:19,624] WARN     0[main] - WARN MySQL - Warning: Select failed!
```

Chapter 13 Using Connector/J with OpenTelemetry

OpenTelemetry is a set of APIs, libraries, agents, and instrumentation to provide observability for applications and their interactions with one another. It enables developers to instrument their code so that they can export observability data including traces, metrics, and logs, enabling increased granularity of profiling, debugging, and testing.

The OpenTelemetry project provides automatic instrumentation for JDBC libraries. However, when it comes to distributed tracing, the automatic instrumentation is not able to propagate the context to the database layer, causing the trace chain to be broken. Also, the automatic instrumentation applies only to the visible layer of the JDBC instrumentation, keeping out any internal operations that are worth tracing as well.

MySQL Enterprise Server 8.4.0 has the capability of collecting observability data for the server operations in the OpenTelemetry format (see Telemetry for details). The feature is supported by component_telemetry. MySQL Connector/J 8.4.0 introduces the client-side counterpart feature, with the capability of propagating the context to the MySQL Server it connects to and allowing a more complete observability for an application stack.

Applications using Connector/J that wish to enable OpenTelemetry tracing need the following 3rd-party libraries:

- The opentelemetry-api library.
- The opentelemetry-context library
- The opentelemetry-sdk library if you choose manual instrumentation, or the opentelemetry-javaagent if you prefer automatic instrumentation.

The Connector/J connection property openTelemetry controls how observability data production is to be handled on a per connection basis. This option accepts the following values:

- `REQUIRED`: An OpenTelemetry library must be available at run time, or connections to the MySQL Server will fail. Note that the `opentelemtry-api` library alone does not produce any output traces.
- `PREFERRED`: Enables generating OpenTelemetry instrumentation, provided that an OpenTelemetry library is available at run time; a warning is issued if a library is not available.
- `DISABLED`: Turns off generating OpenTelemetry instrumentation by Connector/J. However, this does not prevent external means of instrumentation, such as the automatic instrumentation provided by the OpenTelemetry Java agent.

Not setting a value for the property is equivalent to setting it as `PREFERRED`, except that no warning is issued when no OpenTelemetry library is available at runtime.

Notice that MySQL Connector/J does not provide any means for configuring its own OpenTelemetry exporters—it relies entirely on the calling application for the exporter configuration.

The following is a demonstration for how to use OpenTelemetry. It assumes that client application, MySQL Server, and the observability backend are all running on the same machine. It also assumes that component_telemetry is enabled and is properly configured on the MySQL Server. The Java agent is used for the sake of simplicity. The opentelemetry-instrumentation-annotations library is also used, so there is no need to write any OpenTelemetry code in the sample class.

This simple demonstration contains a class `OTelDemo`, which creates a connection to the Sakila database and executes an SQL `SELECT` statement that returns five rows from the table `film`. The purpose of the

demonstration is just to generate a sequence of traces, not to produce anything practically useful. Here are the contents of the source file `src/demo/OTelDemo.java`

```
package demo;
import java.sql.*;
import io.opentelemetry.instrumentation.annotations.WithSpan;
public class OTelDemo {
    public static void main(String[] args) throws Exception {
        listFiveFilms();
    }
    @WithSpan
    private static void listFiveFilms() throws Exception {
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://johndoe:s3cr3t@localhost:3306/sakila"
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM film LIMIT 5");
            while (rs.next()) {
                System.out.println(rs.getString(2));
            }
        }
    }
}
```

The top level trace that will be observed is created within the application code via the annotation `@WithSpan`. This trace becomes the current context (parent) for any subsequent spans created inside the JDBC driver. Subsequently, via context propagation with the executed commands, a trace created by the driver becomes the parent of the spans created within the MySQL Server.

The code can be compiled by issuing the command:

```
$ javac -classpath "lib/*" -d bin src/demo/OTelDemo.java
```

Before you run the demonstration, set up an observability backend with, for example, Jaeger. You can then execute the code with the command:

```
$ java -javaagent:agent/opentelemetry-javaagent.jar \
  -Dotel.traces.exporter=jaeger \
  -Dotel.metrics.exporter=none \
  -Dotel.service.name=OTelDemo \
  -Dotel.instrumentation.common.default-enabled=false \
  -Dotel.instrumentation.opentelemetry-api.enabled=true \
  -Dotel.instrumentation.opentelemetry-instrumentation-annotations.enabled=true \
  -classpath "bin:lib/*" \
  demo.OTelDemo
 [otel.javaagent 2024-04-12 16:10:32:140 +0100]
 [main] INFO io.opentelemetry.javaagent.tooling.VersionLogger - opentelemetry-javaagent - version: 1.32.0
 ACADEMY DINOSAUR
 ACE GOLDFINGER
 ADAPTATION HOLES
 AFFAIR PREJUDICE
 AFRICAN EGG
```

You can now open the Jaeger backend in your web browser and search for the traces of the `OTelDemo` service.

Distributed tracing in MySQL is limited to statement executions. This limitation comes from the fact that context propagation is implemented through query attributes and is only supported for query executions. While running, the server generates spans for other operations as well. Those spans can be seen in the observability backend too, but they are unlinked from any of the traces started by the client application or library. Similarly, spans created by Connector/J that do not produce server commands or those producing commands lacking support for query attributes are depicted as terminal nodes in the trace graphs. This is exemplified by operations such as the `PING` command. Nonetheless, the server still generates a span for the corresponding operation, just that it appears unlinked from the originating trace.

Chapter 14 Using Connector/J with Tomcat

The following instructions are based on the instructions for Tomcat-5.x, available at http://tomcat.apache.org/tomcat-5.5-doc/jndi-datasource-examples-howto.html which is current at the time this document was written.

First, install the `.jar` file that comes with Connector/J in `$CATALINA_HOME/common/lib` so that it is available to all applications installed in the container.

Next, configure the JNDI DataSource by adding a declaration resource to `$CATALINA_HOME/conf/server.xml` in the context that defines your web application:

```
<Context ....>

...

<Resource name="jdbc/MySQLDB"
            auth="Container"
            type="javax.sql.DataSource"/>

<ResourceParams name="jdbc/MySQLDB">
  <parameter>
    <name>factory</name>
    <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
  </parameter>

  <parameter>
    <name>maxActive</name>
    <value>10</value>
  </parameter>

  <parameter>
    <name>maxIdle</name>
    <value>5</value>
  </parameter>

  <parameter>
    <name>validationQuery</name>
    <value>SELECT 1</value>
  </parameter>

  <parameter>
    <name>testOnBorrow</name>
    <value>true</value>
  </parameter>

  <parameter>
    <name>testWhileIdle</name>
    <value>true</value>
  </parameter>

  <parameter>
    <name>timeBetweenEvictionRunsMillis</name>
    <value>10000</value>
  </parameter>

  <parameter>
    <name>minEvictableIdleTimeMillis</name>
    <value>60000</value>
  </parameter>

  <parameter>
   <name>username</name>
   <value>someuser</value>
```

```
    </parameter>

    <parameter>
     <name>password</name>
     <value>somepass</value>
    </parameter>

    <parameter>
       <name>driverClassName</name>
       <value>com.mysql.cj.jdbc.Driver</value>
    </parameter>

    <parameter>
      <name>url</name>
      <value>jdbc:mysql://localhost:3306/test</value>
    </parameter>

  </ResourceParams>
</Context>
```

Connector/J introduces a facility whereby, rather than use a `validationQuery` value of `SELECT 1`, it is possible to use `validationQuery` with a value set to `/* ping */`. This sends a ping to the server which then returns a fake result set. This is a lighter weight solution. It also has the advantage that if using `ReplicationConnection` or `LoadBalancedConnection` type connections, the ping will be sent across all active connections. The following XML snippet illustrates how to select this option:

```
<parameter>
 <name>validationQuery</name>
 <value>/* ping */</value>
</parameter>
```

Note that `/* ping */` has to be specified exactly.

In general, follow the installation instructions that come with your version of Tomcat, as the way you configure datasources in Tomcat changes from time to time, and if you use the wrong syntax in your XML file, you will most likely end up with an exception similar to the following:

```
Error: java.sql.SQLException: Cannot load JDBC driver class 'null ' SQL
state: null
```

Note that the auto-loading of drivers having the `META-INF/service/java.sql.Driver` class in JDBC 4.0 and later causes an improper undeployment of the Connector/J driver in Tomcat on Windows. Namely, the Connector/J jar remains locked. This is an initialization problem that is not related to the driver. The possible workarounds, if viable, are as follows: use "`antiResourceLocking=true`" as a Tomcat Context attribute, or remove the `META-INF/` directory.

Chapter 15 Using Connector/J with Spring

Table of Contents

The Spring Framework is a Java-based application framework designed for assisting in application design by providing a way to configure components. The technique used by Spring is a well known design pattern called Dependency Injection (see Inversion of Control Containers and the Dependency Injection pattern). This article will focus on Java-oriented access to MySQL databases with Spring 2.0. For those wondering, there is a .NET port of Spring appropriately named Spring.NET.

Spring is not only a system for configuring components, but also includes support for aspect oriented programming (AOP). This is one of the main benefits and the foundation for Spring's resource and transaction management. Spring also provides utilities for integrating resource management with JDBC and Hibernate.

For the examples in this section the MySQL world sample database will be used. The first task is to set up a MySQL data source through Spring. Components within Spring use the “bean” terminology. For example, to configure a connection to a MySQL server supporting the world sample database, you might use:

```
<util:map id="dbProps">
    <entry key="db.driver" value="com.mysql.cj.jdbc.Driver"/>
    <entry key="db.jdbcurl" value="jdbc:mysql://localhost/world"/>
    <entry key="db.username" value="myuser"/>
    <entry key="db.password" value="mypass"/>
</util:map>
```

In the above example, we are assigning values to properties that will be used in the configuration. For the datasource configuration:

```
<bean id="dataSource"
       class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${db.driver}"/>
    <property name="url" value="${db.jdbcurl}"/>
    <property name="username" value="${db.username}"/>
    <property name="password" value="${db.password}"/>
</bean>
```

The placeholders are used to provide values for properties of this bean. This means that we can specify all the properties of the configuration in one place instead of entering the values for each property on each bean. We do, however, need one more bean to pull this all together. The last bean is responsible for actually replacing the placeholders with the property values.

```
<bean
 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="properties" ref="dbProps"/>
</bean>
```

Now that we have our MySQL data source configured and ready to go, we write some Java code to access it. The example below will retrieve three random cities and their corresponding country using the data source we configured with Spring.

```
// Create a new application context. this processes the Spring config
ApplicationContext ctx =
    new ClassPathXmlApplicationContext("ex1appContext.xml");
// Retrieve the data source from the application context
    DataSource ds = (DataSource) ctx.getBean("dataSource");
// Open a database connection using Spring's DataSourceUtils
Connection c = DataSourceUtils.getConnection(ds);
try {
    // retrieve a list of three random cities
    PreparedStatement ps = c.prepareStatement(
        "select City.Name as 'City', Country.Name as 'Country' " +
        "from City inner join Country on City.CountryCode = Country.Code " +
        "order by rand() limit 3");
    ResultSet rs = ps.executeQuery();
    while(rs.next()) {
        String city = rs.getString("City");
        String country = rs.getString("Country");
        System.out.printf("The city %s is in %s%n", city, country);
    }
} catch (SQLException ex) {
    // something has failed and we print a stack trace to analyse the error
    ex.printStackTrace();
    // ignore failure closing connection
    try { c.close(); } catch (SQLException e) { }
} finally {
    // properly release our connection
    DataSourceUtils.releaseConnection(c, ds);
}
```

This is very similar to normal JDBC access to MySQL with the main difference being that we are using DataSourceUtils instead of the DriverManager to create the connection.

While it may seem like a small difference, the implications are somewhat far reaching. Spring manages this resource in a way similar to a container managed data source in a J2EE application server. When a connection is opened, it can be subsequently accessed in other parts of the code if it is synchronized with a transaction. This makes it possible to treat different parts of your application as transactional instead of passing around a database connection.

15.1 Using `JdbcTemplate`

Spring makes extensive use of the Template method design pattern (see Template Method Pattern). Our immediate focus will be on the `JdbcTemplate` and related classes, specifically `NamedParameterJdbcTemplate`. The template classes handle obtaining and releasing a connection for data access when one is needed.

The next example shows how to use `NamedParameterJdbcTemplate` inside of a DAO (Data Access Object) class to retrieve a random city given a country code.

```
public class Ex2JdbcDao {
     /**
     * Data source reference which will be provided by Spring.
     */
     private DataSource dataSource;

     /**
     * Our query to find a random city given a country code. Notice
     * the ":country" parameter toward the end. This is called a
     * named parameter.
     */
```

```
    private String queryString = "select Name from City " +
       "where CountryCode = :country order by rand() limit 1";

    /**
    * Retrieve a random city using Spring JDBC access classes.
    */
    public String getRandomCityByCountryCode(String cntryCode) {
        // A template that permits using queries with named parameters
        NamedParameterJdbcTemplate template =
        new NamedParameterJdbcTemplate(dataSource);
        // A java.util.Map is used to provide values for the parameters
        Map params = new HashMap();
        params.put("country", cntryCode);
        // We query for an Object and specify what class we are expecting
        return (String)template.queryForObject(queryString, params, String.class);
    }

   /**
   * A JavaBean setter-style method to allow Spring to inject the data source.
   * @param dataSource
   */
   public void setDataSource(DataSource dataSource) {
       this.dataSource = dataSource;
   }
}
```

The focus in the above code is on the `getRandomCityByCountryCode()` method. We pass a country code and use the `NamedParameterJdbcTemplate` to query for a city. The country code is placed in a Map with the key "country", which is the parameter is named in the SQL query.

To access this code, you need to configure it with Spring by providing a reference to the data source.

```
<bean id="dao" class="code.Ex2JdbcDao">
    <property name="dataSource" ref="dataSource"/>
</bean>
```

At this point, we can just grab a reference to the DAO from Spring and call `getRandomCityByCountryCode()`.

```
    // Create the application context
    ApplicationContext ctx =
    new ClassPathXmlApplicationContext("ex2appContext.xml");
    // Obtain a reference to our DAO
    Ex2JdbcDao dao = (Ex2JdbcDao) ctx.getBean("dao");

    String countryCode = "USA";

    // Find a few random cities in the US
    for(int i = 0; i < 4; ++i)
        System.out.printf("A random city in %s is %s%n", countryCode,
            dao.getRandomCityByCountryCode(countryCode));
```

This example shows how to use Spring's JDBC classes to completely abstract away the use of traditional JDBC classes including `Connection` and `PreparedStatement`.

15.2 Transactional JDBC Access

Spring allows us to add transactions into our code without having to deal directly with the JDBC classes. For that purpose, Spring provides a transaction management package that not only replaces JDBC transaction management, but also enables declarative transaction management (configuration instead of code).

To use transactional database access, we will need to change the storage engine of the tables in the world database. The downloaded script explicitly creates MyISAM tables, which do not support transactional semantics. The InnoDB storage engine does support transactions and this is what we will be using. We can change the storage engine with the following statements.

```
ALTER TABLE City ENGINE=InnoDB;
ALTER TABLE Country ENGINE=InnoDB;
ALTER TABLE CountryLanguage ENGINE=InnoDB;
```

A good programming practice emphasized by Spring is separating interfaces and implementations. What this means is that we can create a Java interface and only use the operations on this interface without any internal knowledge of what the actual implementation is. We will let Spring manage the implementation and with this it will manage the transactions for our implementation.

First you create a simple interface:

```
public interface Ex3Dao {
    Integer createCity(String name, String countryCode,
    String district, Integer population);
}
```

This interface contains one method that will create a new city record in the database and return the id of the new record. Next you need to create an implementation of this interface.

```
public class Ex3DaoImpl implements Ex3Dao {
    protected DataSource dataSource;
    protected SqlUpdate updateQuery;
    protected SqlFunction idQuery;

    public Integer createCity(String name, String countryCode,
        String district, Integer population) {
            updateQuery.update(new Object[] { name, countryCode,
                   district, population });
            return getLastId();
        }

    protected Integer getLastId() {
        return idQuery.run();
    }
}
```

You can see that we only operate on abstract query objects here and do not deal directly with the JDBC API. Also, this is the complete implementation. All of our transaction management will be dealt with in the configuration. To get the configuration started, we need to create the DAO.

```
<bean id="dao" class="code.Ex3DaoImpl">
    <property name="dataSource" ref="dataSource"/>
    <property name="updateQuery">...</property>
    <property name="idQuery">...</property>
</bean>
```

Now we need to set up the transaction configuration. The first thing we must do is create transaction manager to manage the data source and a specification of what transaction properties are required for the dao methods.

```
<bean id="transactionManager"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>
```

```
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="*"/>
    </tx:attributes>
</tx:advice>
```

The preceding code creates a transaction manager that handles transactions for the data source provided to it. The `txAdvice` uses this transaction manager and the attributes specify to create a transaction for all methods. Finally we need to apply this advice with an AOP pointcut.

```
<aop:config>
    <aop:pointcut id="daoMethods"
        expression="execution(* code.Ex3Dao.*(..))"/>
     <aop:advisor advice-ref="txAdvice" pointcut-ref="daoMethods"/>
</aop:config>
```

This basically says that all methods called on the `Ex3Dao` interface will be wrapped in a transaction. To make use of this, we only have to retrieve the `dao` from the application context and call a method on the `dao` instance.

```
Ex3Dao dao = (Ex3Dao) ctx.getBean("dao");
Integer id = dao.createCity(name,  countryCode, district, pop);
```

We can verify from this that there is no transaction management happening in our Java code and it is all configured with Spring. This is a very powerful notion and regarded as one of the most beneficial features of Spring.

15.3 Connection Pooling with Spring

In many situations, such as web applications, there will be a large number of small database transactions. When this is the case, it usually makes sense to create a pool of database connections available for web requests as needed. Although MySQL does not spawn an extra process when a connection is made, there is still a small amount of overhead to create and set up the connection. Pooling of connections also alleviates problems such as collecting large amounts of sockets in the `TIME_WAIT` state.

Setting up pooling of MySQL connections with Spring is as simple as changing the data source configuration in the application context. There are a number of configurations that we can use. The first example is based on the Jakarta Commons DBCP library. The example below replaces the source configuration that was based on `DriverManagerDataSource` with DBCP's BasicDataSource.

```
<bean id="dataSource" destroy-method="close"
  class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="${db.driver}"/>
    <property name="url" value="${db.jdbcurl}"/>
    <property name="username" value="${db.username}"/>
    <property name="password" value="${db.password}"/>
    <property name="initialSize" value="3"/>
</bean>
```

The configuration of the two solutions is very similar. The difference is that DBCP will pool connections to the database instead of creating a new connection every time one is requested. We have also set a parameter here called `initialSize`. This tells DBCP that we want three connections in the pool when it is created.

Chapter 16 Troubleshooting Connector/J Applications

This section explains the symptoms and resolutions for the most commonly encountered issues with applications using MySQL Connector/J.

Questions

Questions and Answers

16.1: When I try to connect to the database with MySQL Connector/J, I get the following exception:

```
SQLException: Server configuration denies access to data source
SQLState: 08001
VendorError: 0
```

What is going on? I can connect just fine with the MySQL command-line client.

Connector/J normally uses TCP/IP sockets to connect to MySQL (see Section 6.10, “Connecting Using Unix Domain Sockets” and Section 6.11, “Connecting Using Named Pipes” for exceptions). The security manager on the MySQL server uses its grant tables to determine whether a TCP/IP connection is permitted. You must therefore add the necessary security credentials to the MySQL server for the connection by issuing a `GRANT` statement to your MySQL Server. See GRANT Statement, for more information.

> **Warning**
>
> Changing privileges and permissions improperly on MySQL can potentially cause your server installation to have non-optimal security properties.

> **Note**
>
> Testing your connectivity with the `mysql` command-line client will not work unless you add the `--host` flag, and use something other than `localhost` for the host. The `mysql` command-line client will try to use Unix domain sockets if you use the special host name `localhost`. If you are testing TCP/IP connectivity to `localhost`, use `127.0.0.1` as the host name instead.

16.2: My application throws an SQLException 'No Suitable Driver'. Why is this happening?

There are three possible causes for this error:

- The Connector/J driver is not in your `CLASSPATH`, see Chapter 4, *Connector/J Installation*.
- The format of your connection URL is incorrect, or you are referencing the wrong JDBC driver.
- When using DriverManager, the `jdbc.drivers` system property has not been populated with the location of the Connector/J driver.

16.3: I'm trying to use MySQL Connector/J in an applet or application and I get an exception similar to:

```
SQLException: Cannot connect to MySQL server on host:3306.
Is there a MySQL server running on the machine/port you
are trying to connect to?

(java.security.AccessControlException)
SQLState: 08S01
VendorError: 0
```

Either you're running an Applet, your MySQL server has been installed with the `skip_networking` system variable enabled, or your MySQL server has a firewall sitting in front of it.

Applets can only make network connections back to the machine that runs the web server that served the .class files for the applet. This means that MySQL must run on the same machine (or you must have some sort of port re-direction) for this to work. This also means that you will not be able to test applets from your local file system, but must always deploy them to a web server.

Connector/J normally uses TCP/IP sockets to connect to MySQL (see Section 6.10, “Connecting Using Unix Domain Sockets” and Section 6.11, “Connecting Using Named Pipes” for exceptions). TCP/IP communication with MySQL can be affected by the `skip_networking` system variable or the server firewall. If MySQL has been started with `skip_networking` enabled, you need to comment it out in the file `/etc/mysql/my.cnf` or `/etc/my.cnf` for TCP/IP connections to work. (Note that your server configuration file might also exist in the `data` directory of your MySQL server, or somewhere else, depending on how MySQL was compiled; binaries created by Oracle always look for `/etc/my.cnf` and *`datadir`*`/my.cnf`; see Using Option Files for details.) If your MySQL server has been firewalled, you will

need to have the firewall configured to allow TCP/IP connections from the host where your Java code is running to the MySQL server on the port that MySQL is listening to (by default, 3306).

16.4: I have a servlet/application that works fine for a day, and then stops working overnight

MySQL closes connections after 8 hours of inactivity. You either need to use a connection pool that handles stale connections or use the `autoReconnect` parameter (see Section 6.3, “Configuration Properties”).

Also, catch `SQLExceptions` in your application and deal with them, rather than propagating them all the way until your application exits. This is just good programming practice. MySQL Connector/J will set the `SQLState` (see `java.sql.SQLException.getSQLState()` in your API docs) to `08S01` when it encounters network-connectivity issues during the processing of a query. Attempt to reconnect to MySQL at this point.

The following (simplistic) example shows what code that can handle these exceptions might look like:

Example 16.1 Connector/J: Example of transaction with retry logic

```
public void doBusinessOp() throws SQLException {
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;

    //
    // How many times do you want to retry the transaction
    // (or at least _getting_ a connection)?
    //
    int retryCount = 5;

    boolean transactionCompleted = false;

    do {
        try {
            conn = getConnection(); // assume getting this from a
                                    // javax.sql.DataSource, or the
                                    // java.sql.DriverManager

            conn.setAutoCommit(false);

            //
            // Okay, at this point, the 'retry-ability' of the
            // transaction really depends on your application logic,
            // whether or not you're using autocommit (in this case
            // not), and whether you're using transactional storage
            // engines
            //
            // For this example, we'll assume that it's _not_ safe
            // to retry the entire transaction, so we set retry
            // count to 0 at this point
            //
            // If you were using exclusively transaction-safe tables,
            // or your application could recover from a connection going
            // bad in the middle of an operation, then you would not
            // touch 'retryCount' here, and just let the loop repeat
            // until retryCount == 0.
            //
            retryCount = 0;

            stmt = conn.createStatement();

            String query = "SELECT foo FROM bar ORDER BY baz";

            rs = stmt.executeQuery(query);
```

```
            while (rs.next()) {
            }

            rs.close();
            rs = null;

            stmt.close();
            stmt = null;

            conn.commit();
            conn.close();
            conn = null;

            transactionCompleted = true;
        } catch (SQLException sqlEx) {

            //
            // The two SQL states that are 'retry-able' are 08S01
            // for a communications error, and 40001 for deadlock.
            //
            // Only retry if the error was due to a stale connection,
            // communications problem or deadlock
            //

            String sqlState = sqlEx.getSQLState();

            if ("08S01".equals(sqlState) || "40001".equals(sqlState)) {
                retryCount -= 1;
            } else {
                retryCount = 0;
            }
        } finally {
            if (rs != null) {
                try {
                    rs.close();
                } catch (SQLException sqlEx) {
                    // You'd probably want to log this...
                }
            }

            if (stmt != null) {
                try {
                    stmt.close();
                } catch (SQLException sqlEx) {
                    // You'd probably want to log this as well...
                }
            }

            if (conn != null) {
                try {
                    //
                    // If we got here, and conn is not null, the
                    // transaction should be rolled back, as not
                    // all work has been done

                    try {
                        conn.rollback();
                    } finally {
                        conn.close();
                    }
                } catch (SQLException sqlEx) {
                    //
                    // If we got an exception here, something
                    // pretty serious is going on, so we better
                    // pass it up the stack, rather than just
                    // logging it...
```

```
                    throw sqlEx;
                }
            }
        }
    } while (!transactionCompleted && (retryCount > 0));
}
```

> **Note**
>
> Use of the `autoReconnect` option is not recommended because there is no safe method of reconnecting to the MySQL server without risking some corruption of the connection state or database state information. Instead, use a connection pool, which will enable your application to connect to the MySQL server using an available connection from the pool. The `autoReconnect` facility is deprecated, and may be removed in a future release.

16.5: I cannot connect to the MySQL server using Connector/J, and I'm sure the connection parameters are correct.

Make sure that the `skip_networking` system variable has not been enabled on your server. Connector/J must be able to communicate with your server over TCP/IP; named sockets are not supported. Also ensure that you are not filtering connections through a firewall or other network security system. For more information, see Can't connect to [local] MySQL server.

16.6: Updating a table that contains a primary key that is either `FLOAT` or compound primary key that uses `FLOAT` fails to update the table and raises an exception.

Connector/J adds conditions to the `WHERE` clause during an `UPDATE` to check the old values of the primary key. If there is no match, then Connector/J considers this a failure condition and raises an exception.

The problem is that rounding differences between supplied values and the values stored in the database may mean that the values never match, and hence the update fails. The issue will affect all queries, not just those from Connector/J.

To prevent this issue, use a primary key that does not use `FLOAT`. If you have to use a floating point column in your primary key, use `DOUBLE` or `DECIMAL` types in place of `FLOAT`.

16.7: I get an `ER_NET_PACKET_TOO_LARGE` exception, even though the binary blob size I want to insert using JDBC is safely below the `max_allowed_packet` size.

This is because the `hexEscapeBlock()` method in `com.mysql.cj.AbstractPreparedQuery.streamToBytes()` may almost double the size of your data.

16.8: What should I do if I receive error messages similar to the following: “Communications link failure – Last packet sent to the server was X ms ago”?

Generally speaking, this error suggests that the network connection has been closed. There can be several root causes:

- Firewalls or routers may clamp down on idle connections (the MySQL client/server protocol does not ping).
- The MySQL Server may be closing idle connections that exceed the `wait_timeout` or `interactive_timeout` threshold.

Although network connections can be volatile, the following can be helpful in avoiding problems:

- Ensure connections are valid when used from the connection pool. Use a query that starts with `/* ping */` to execute a lightweight ping instead of full query. Note, the syntax of the ping needs to be exactly as specified here.
- Minimize the duration a connection object is left idle while other application logic is executed.
- Explicitly validate the connection before using it if the connection has been left idle for an extended period of time.
- Ensure that `wait_timeout` and `interactive_timeout` are set sufficiently high.
- Ensure that `tcpKeepalive` is enabled.
- Ensure that any configurable firewall or router timeout settings allow for the maximum expected connection idle time.

> **Note**
>
> Do not expect to be able to reuse a connection without problems if it has being lying idle for a period. If a connection is to be reused after being idle for any length of time, ensure that you explicitly test it before reusing it.

16.9: Why does Connector/J not reconnect to MySQL and re-issue the statement after a communication failure instead of throwing an Exception, even though I use the `autoReconnect` connection string option?

There are several reasons for this. The first is transactional integrity. The MySQL Reference Manual states that “there is no safe method of reconnecting to the MySQL server without risking some corruption of the connection state or database state information”. Consider the following series of statements for example:

```
conn.createStatement().execute(
  "UPDATE checking_account SET balance = balance - 1000.00 WHERE customer='Smith'");
conn.createStatement().execute(
  "UPDATE savings_account SET balance = balance + 1000.00 WHERE customer='Smith'");
conn.commit();
```

Consider the case where the connection to the server fails after the `UPDATE` to `checking_account`. If no exception is thrown, and the application never learns about the problem, it will continue executing. However, the server did not commit the first transaction in this case, so that will get rolled back. But execution continues with the next transaction, and increases the `savings_account` balance by 1000. The application did not receive an exception, so it continued regardless, eventually committing the second transaction, as the commit only applies to the changes made in the new connection. Rather than a transfer taking place, a deposit was made in this example.

Note that running with `autocommit` enabled does not solve this problem. When Connector/J encounters a communication problem, there is no means to determine whether the server processed the currently executing statement or not. The following theoretical states are equally possible:

- The server never received the statement, and therefore no related processing occurred on the server.
- The server received the statement, executed it in full, but the response was not received by the client.

If you are running with `autocommit` enabled, it is not possible to guarantee the state of data on the server when a communication exception is encountered. The statement may have reached the server, or it may not. All you know is that communication failed at some point, before the client received confirmation (or data) from the server. This does not only affect `autocommit` statements though. If the communication

problem occurred during `Connection.commit()`, the question arises of whether the transaction was committed on the server before the communication failed, or whether the server received the commit request at all.

The second reason for the generation of exceptions is that transaction-scoped contextual data may be vulnerable, for example:

- Temporary tables.
- User-defined variables.
- Server-side prepared statements.

These items are lost when a connection fails, and if the connection silently reconnects without generating an exception, this could be detrimental to the correct execution of your application.

In summary, communication errors generate conditions that may well be unsafe for Connector/J to simply ignore by silently reconnecting. It is necessary for the application to be notified. It is then for the application developer to decide how to proceed in the event of connection errors and failures.

16.10: How can I use 3-byte UTF8 with Connector/J?

Because there is no Java-style character set name for `utfmb3` that you can use with the connection option `charaterEncoding`, the only way to use `utf8mb3` as your connection character set is to use a `utf8mb3` collation (for example, `utf8_general_ci`) for the connection option `connectionCollation`, which forces a `utf8mb3` character set to be used. See Section 6.7, “Using Character Sets and Unicode” for details.

16.11: How can I use 4-byte UTF8 (`utf8mb4`) with Connector/J?

To use 4-byte UTF8 with Connector/J configure the MySQL server with `character_set_server=utf8mb4`. Connector/J will then use that setting, if `characterEncoding` and `connectionCollation` have not been set in the connection string. This is equivalent to autodetection of the character set. See Section 6.7, “Using Character Sets and Unicode” for details. You can use `characterEncoding=UTF-8` to use `utf8mb4`, even if `character_set_server` on the server has been set to something else.

16.12: Using `useServerPrepStmts=false` and certain character encodings can lead to corruption when inserting BLOBs. How can this be avoided?

When using certain character encodings, such as SJIS, CP932, and BIG5, it is possible that BLOB data contains characters that can be interpreted as control characters, for example, backslash, '\'. This can lead to corrupted data when inserting BLOBs into the database. There are two things that need to be done to avoid this:

1. Set the connection string option `useServerPrepStmts` to `true`.
2. Set `SQL_MODE` to `NO_BACKSLASH_ESCAPES`.

Chapter 17 Known Issues and Limitations

The following are some known issues and limitations for MySQL Connector/J:

- When Connector/J retrieves timestamps for a daylight saving time (DST) switch day using the `getTimeStamp()` method on the result set, some of the returned values might be wrong. In order to avoid such errors, we recommend setting a connection time zone that uses a monotonic clock by, for example, setting `connectionTimeZone=UTC`, and configuring other date-time connection properties according to your needs; see Section 6.6, “Handling of Date-Time Values” for details.
- The functionality of the property `elideSetAutoCommits` has been disabled due to Bug# 66884. Any value given for the property is ignored by Connector/J.
- MySQL Server uses a proleptic Gregorian calendar internally. However, Connector/J uses `java.sql.Date`, which is non-proleptic. Therefore, when setting and retrieving dates that were before the Julian-Gregorian cutover (October 15, 1582) using the `PreparedStatement` methods, always supply explicitly a proleptic Gregorian calendar to the `setDate()` and `getDate()` methods, in order to avoid possible errors with dates stored to and calculated by the server.
- To use Windows named pipes for connections, the MySQL Server that Connector/J wants to connect to must be started with the system variable `named_pipe_full_access_group`; see Section 6.11, “Connecting Using Named Pipes” for details.

Chapter 18 Connector/J Support

Table of Contents

18.1 Connector/J Community Support

You can join the `#connectors` channel in the MySQL Community Slack workspace, where you can get help directly from MySQL developers and other users.

18.2 How to Report Connector/J Bugs or Problems

The normal place to report bugs is http://bugs.mysql.com/, which is the address for our bugs database. This database is public, and can be browsed and searched by anyone. If you log in to the system, you will also be able to enter new reports.

If you find a sensitive security bug in MySQL Server, please let us know immediately by sending an email message to `<secalert_us@oracle.com>`. Exception: Support customers should report all problems, including security bugs, to Oracle Support at http://support.oracle.com/.

Writing a good bug report takes patience, but doing it right the first time saves time both for us and for yourself. A good bug report, containing a full test case for the bug, makes it very likely that we will fix sooner rather than later.

This section will help you write your report correctly so that you do not waste your time doing things that may not help us much or at all.

If you have a repeatable bug report, please report it to the bugs database at http://bugs.mysql.com/. Any bug that we are able to repeat has a high chance of being fixed sooner rather than later.

To report other problems, you can use one of the MySQL mailing lists.

Remember that it is possible for us to respond to a message containing too much information, but not to one containing too little. People often omit facts because they think they know the cause of a problem and assume that some details do not matter.

A good principle is this: If you are in doubt about stating something, state it. It is faster and less troublesome to write a couple more lines in your report than to wait longer for the answer if we must ask you to provide information that was missing from the initial report.

The most common errors made in bug reports are (a) not including the version number of Connector/J or MySQL used, and (b) not fully describing the platform on which Connector/J is installed (including the JVM version, and the platform type and version number that MySQL itself is installed on).

This is highly relevant information, and in 99 cases out of 100, the bug report is useless without it. Very often we get questions like, “Why doesn't this work for me?” Then we find that the feature requested was not implemented in that MySQL version, or that a bug described in a report has already been fixed in newer MySQL versions.

Sometimes the error is platform-dependent; in such cases, it is next to impossible for us to fix anything without knowing the operating system and the version number of the platform.

If at all possible, create a repeatable, standalone testcase that doesn't involve any third-party classes.

To streamline this process, we ship a base class for testcases with Connector/J, named '`com.mysql.cj.jdbc.util.BaseBugReport`'. To create a testcase for Connector/J using this class, create your own class that inherits from `com.mysql.cj.jdbc.util.BaseBugReport` and override the methods `setUp()`, `tearDown()` and `runTest()`.

In the `setUp()` method, create code that creates your tables, and populates them with any data needed to demonstrate the bug.

In the `runTest()` method, create code that demonstrates the bug using the tables and data you created in the `setUp` method.

In the `tearDown()` method, drop any tables you created in the `setUp()` method.

In any of the above three methods, use one of the variants of the `getConnection()` method to create a JDBC connection to MySQL:

- `getConnection()` - Provides a connection to the JDBC URL specified in `getUrl()`. If a connection already exists, that connection is returned, otherwise a new connection is created.
- `getNewConnection()` - Use this if you need to get a new connection for your bug report (that is, there is more than one connection involved).
- `getConnection(String url)` - Returns a connection using the given URL.
- `getConnection(String url, Properties props)` - Returns a connection using the given URL and properties.

If you need to use a JDBC URL that is different from 'jdbc:mysql:///test', override the method `getUrl()` as well.

Use the `assertTrue(boolean expression)` and `assertTrue(String failureMessage, boolean expression)` methods to create conditions that must be met in your testcase demonstrating the behavior you are expecting (vs. the behavior you are observing, which is why you are most likely filing a bug report).

Finally, create a `main()` method that creates a new instance of your testcase, and calls the `run` method:

```
public static void main(String[] args) throws Exception {
      new MyBugReport().run();
 }
```

Once you have finished your testcase, and have verified that it demonstrates the bug you are reporting, upload it with your bug report to http://bugs.mysql.com/.

Index

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z