# MySQL 8.0 Reference Manual

## Including MySQL NDB Cluster 8.0

**Abstract**

This is the MySQL Reference Manual. It documents MySQL 8.0 through 8.0.34, as well as NDB Cluster releases based on version 8.0 of NDB through 8.0.34-ndb-8.0.34, respectively. It may include documentation of features of MySQL versions that have not yet been released. For information about which versions have been released, see the MySQL 8.0 Release Notes.

**MySQL 8.0 features.** This manual describes features that are not included in every edition of MySQL 8.0; such features may not be included in the edition of MySQL 8.0 licensed to you. If you have any questions about the features included in your edition of MySQL 8.0, refer to your MySQL 8.0 license agreement or contact your Oracle sales representative.

For notes detailing the changes in each release, see the MySQL 8.0 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: 2023-04-04 (revision: 75301)

# Table of Contents

# Preface and Legal Notices

This is the Reference Manual for the MySQL Database System, version 8.0, through release 8.0.32. Differences between minor versions of MySQL 8.0 are noted in the present text with reference to release numbers (8.0.*x*). For license information, see the Legal Notices.

This manual is not intended for use with older versions of the MySQL software due to the many functional and other differences between MySQL 8.0 and previous versions. If you are using an earlier release of the MySQL software, please refer to the appropriate manual. For example, *MySQL 5.7 Reference Manual* covers the 5.7 series of MySQL software releases.

**Licensing information—MySQL 8.0.** This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 8.0, see the MySQL 8.0 Commercial Release 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 8.0, see the MySQL 8.0 Community Release License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

**Licensing information—MySQL NDB Cluster 8.0.** If you are using a *Commercial* release of MySQL NDB Cluster 8.0, see the MySQL NDB Cluster 8.0 Commercial Release 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 NDB Cluster 8.0, see the MySQL NDB Cluster 8.0 Community Release 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

applications that may create a risk of personal injury. If you use this software or hardware in dangerous 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.

Oracle, Java, and MySQL 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.

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.

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
https://www.oracle.com/corporate/accessibility/.

# Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

# Chapter 1 General Information

## Table of Contents



The MySQL software delivers a very fast, multithreaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. MySQL is a trademark of Oracle Corporation and/or its affiliates, and shall not be used by Customer without Oracle's express written authorization. Other names may be trademarks of their respective owners.

The MySQL software is Dual Licensed. Users can choose to use the MySQL software as an Open Source product under the terms of the GNU General Public License (http://www.fsf.org/licenses/) or can purchase a standard commercial license from Oracle. See http://www.mysql.com/company/legal/licensing/ for more information on our licensing policies.

The following list describes some sections of particular interest in this manual:

- For a discussion of MySQL Database Server capabilities, see Section 1.2.2, “The Main Features of MySQL”.
- For an overview of new MySQL features, see Section 1.3, “What Is New in MySQL 8.0”. For information about the changes in each version, see the Release Notes.
- For installation instructions, see Chapter 2, *Installing and Upgrading MySQL*. For information about upgrading MySQL, see Section 2.10, “Upgrading MySQL”.
- For a tutorial introduction to the MySQL Database Server, see Chapter 3, *Tutorial*.
- For information about configuring and administering MySQL Server, see Chapter 5, *MySQL Server Administration*.
- For information about security in MySQL, see Chapter 6, *Security*.
- For information about setting up replication servers, see Chapter 17, *Replication*.
- For information about MySQL Enterprise, the commercial MySQL release with advanced features and management tools, see Chapter 30, *MySQL Enterprise Edition*.

- For answers to a number of questions that are often asked concerning the MySQL Database Server and its capabilities, see Appendix A, *MySQL 8.0 Frequently Asked Questions*.
- For a history of new features and bug fixes, see the Release Notes.

**Important**

To report problems or bugs, please use the instructions at Section 1.5, "How to Report Bugs or Problems". If you find a 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.

# 1.1 About This Manual

This is the Reference Manual for the MySQL Database System, version 8.0, through release 8.0.32. Differences between minor versions of MySQL 8.0 are noted in the present text with reference to release numbers (8.0.*x*). For license information, see the Legal Notices.

This manual is not intended for use with older versions of the MySQL software due to the many functional and other differences between MySQL 8.0 and previous versions. If you are using an earlier release of the MySQL software, please refer to the appropriate manual. For example, *MySQL 5.7 Reference Manual* covers the 5.7 series of MySQL software releases.

Because this manual serves as a reference, it does not provide general instruction on SQL or relational database concepts. It also does not teach you how to use your operating system or command-line interpreter.

The MySQL Database Software is under constant development, and the Reference Manual is updated frequently as well. The most recent version of the manual is available online in searchable form at https://dev.mysql.com/doc/. Other formats also are available there, including downloadable HTML and PDF versions.

The source code for MySQL itself contains internal documentation written using Doxygen. The generated Doxygen content is available from https://dev.mysql.com/doc/index-other.html. It is also possible to generate this content locally from a MySQL source distribution using the instructions at Section 2.8.10, "Generating MySQL Doxygen Documentation Content".

If you have questions about using MySQL, join the MySQL Community Slack. If you have suggestions concerning additions or corrections to the manual itself, please send them to the http://www.mysql.com/company/contact/.

## Typographical and Syntax Conventions

This manual uses certain typographical conventions:

- `Text in this style` is used for SQL statements; database, table, and column names; program listings and source code; and environment variables. Example: "To reload the grant tables, use the `FLUSH PRIVILEGES` statement."
- **`Text in this style`** indicates input that you type in examples.
- `Text in this style` indicates the names of executable programs and scripts, examples being `mysql` (the MySQL command-line client program) and `mysqld` (the MySQL server executable).
- *`Text in this style`* is used for variable input for which you should substitute a value of your own choosing.
- *Text in this style* is used for emphasis.

- **Text in this style** is used in table headings and to convey especially strong emphasis.
- `Text in this style` is used to indicate a program option that affects how the program is executed, or that supplies information that is needed for the program to function in a certain way. *Example*: "The `--host` option (short form `-h`) tells the `mysql` client program the hostname or IP address of the MySQL server that it should connect to".
- File names and directory names are written like this: "The global `my.cnf` file is located in the `/etc` directory."
- Character sequences are written like this: "To specify a wildcard, use the '`%`' character."

When commands or statements are prefixed by a prompt, we use these:

```
$> type a command here
#> type a command as root here
C:\> type a command here (Windows only)
mysql> type a mysql statement here
```

Commands are issued in your command interpreter. On Unix, this is typically a program such as `sh`, `csh`, or `bash`. On Windows, the equivalent program is `command.com` or `cmd.exe`, typically run in a console window. Statements prefixed by `mysql` are issued in the `mysql` command-line client.

**Note**

When you enter a command or statement shown in an example, do not type the prompt shown in the example.

In some areas different systems may be distinguished from each other to show that commands should be executed in two different environments. For example, while working with replication the commands might be prefixed with `source` and `replica`:

```
source> type a mysql statement on the replication source here
replica> type a mysql statement on the replica here
```

Database, table, and column names must often be substituted into statements. To indicate that such substitution is necessary, this manual uses *`db_name`*, *`tbl_name`*, and *`col_name`*. For example, you might see a statement like this:

```
mysql> SELECT col_name FROM db_name.tbl_name;
```

This means that if you were to enter a similar statement, you would supply your own database, table, and column names, perhaps like this:

```
mysql> SELECT author_name FROM biblio_db.author_list;
```

SQL keywords are not case-sensitive and may be written in any lettercase. This manual uses uppercase.

In syntax descriptions, square brackets ("`[`" and "`]`") indicate optional words or clauses. For example, in the following statement, `IF EXISTS` is optional:

```
DROP TABLE [IF EXISTS] tbl_name
```

When a syntax element consists of a number of alternatives, the alternatives are separated by vertical bars ("`|`"). When one member from a set of choices *may* be chosen, the alternatives are listed within square brackets ("`[`" and "`]`"):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

When one member from a set of choices *must* be chosen, the alternatives are listed within braces ("`{`" and "`}`"):

```
{DESCRIBE | DESC} tbl_name [col_name | wild]
```

An ellipsis (`...`) indicates the omission of a section of a statement, typically to provide a shorter version of more complex syntax. For example, `SELECT ... INTO OUTFILE` is shorthand for the form of `SELECT` statement that has an `INTO OUTFILE` clause following other parts of the statement.

An ellipsis can also indicate that the preceding syntax element of a statement may be repeated. In the following example, multiple *`reset_option`* values may be given, with each of those after the first preceded by commas:

```
RESET reset_option [,reset_option] ...
```

Commands for setting shell variables are shown using Bourne shell syntax. For example, the sequence to set the `CC` environment variable and run the `configure` command looks like this in Bourne shell syntax:

```
$> CC=gcc ./configure
```

If you are using `csh` or `tcsh`, you must issue commands somewhat differently:

```
$> setenv CC gcc
$> ./configure
```

## Manual Authorship

The Reference Manual source files are written in DocBook XML format. The HTML version and other formats are produced automatically, primarily using the DocBook XSL stylesheets. For information about DocBook, see http://docbook.org/

This manual was originally written by David Axmark and Michael “Monty” Widenius. It is maintained by the MySQL Documentation Team, consisting of Chris Cole, Aijaz Fatima, Edward Gilmore, Stefan Hinz, David Hollis, Philip Olson, Daniel So, and Jon Stephens.

# 1.2 Overview of the MySQL Database Management System

## 1.2.1 What is MySQL?

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

The MySQL website (http://www.mysql.com/) provides the latest information about MySQL software.

- **MySQL is a database management system.**

  A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**

  A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

  The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment,

you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- **MySQL software is Open Source.**

  Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), http://www.fsf.org/licenses/, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information (http://www.mysql.com/company/legal/licensing/).

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

  If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

  MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

- **MySQL Server works in client/server or embedded systems.**

  The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

  We also provide MySQL Server as an embedded multithreaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

- **A large amount of contributed MySQL software is available.**

  MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way.

## 1.2.2 The Main Features of MySQL

This section describes some of the important characteristics of the MySQL Database Software. In most respects, the roadmap applies to all versions of MySQL. For information about features as they are introduced into MySQL on a series-specific basis, see the “In a Nutshell” section of the appropriate Manual:

- MySQL 8.0: Section 1.3, “What Is New in MySQL 8.0”
- MySQL 5.7: What Is New in MySQL 5.7

### Internals and Portability

- Written in C and C++.
- Tested with a broad range of different compilers.
- Works on many different platforms. See https://www.mysql.com/support/supportedplatforms/database.html.
- For portability, configured using `CMake`.
- Tested with Purify (a commercial memory leakage detector) as well as with Valgrind, a GPL tool (http://developer.kde.org/~sewardj/).
- Uses multi-layered server design with independent modules.
- Designed to be fully multithreaded using kernel threads, to easily use multiple CPUs if they are available.
- Provides transactional and nontransactional storage engines.
- Uses very fast B-tree disk tables (`MyISAM`) with index compression.
- Designed to make it relatively easy to add other storage engines. This is useful if you want to provide an SQL interface for an in-house database.
- Uses a very fast thread-based memory allocation system.
- Executes very fast joins using an optimized nested-loop join.
- Implements in-memory hash tables, which are used as temporary tables.
- Implements SQL functions using a highly optimized class library that should be as fast as possible. Usually there is no memory allocation at all after query initialization.
- Provides the server as a separate program for use in a client/server networked environment.

### Data Types

- Many data types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, `FLOAT`, `DOUBLE`, `CHAR`, `VARCHAR`, `BINARY`, `VARBINARY`, `TEXT`, `BLOB`, `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`, `SET`, `ENUM`, and OpenGIS spatial types. See Chapter 11, *Data Types*.
- Fixed-length and variable-length string types.

### Statements and Functions

- Full operator and function support in the `SELECT` list and `WHERE` clause of queries. For example:

```
mysql> SELECT CONCAT(first_name, ' ', last_name)
    -> FROM citizen
    -> WHERE income/dependents > 10000 AND age > 30;
```

- Full support for SQL `GROUP BY` and `ORDER BY` clauses. Support for group functions (`COUNT()`, `AVG()`, `STD()`, `SUM()`, `MAX()`, `MIN()`, and `GROUP_CONCAT()`).
- Support for `LEFT OUTER JOIN` and `RIGHT OUTER JOIN` with both standard SQL and ODBC syntax.
- Support for aliases on tables and columns as required by standard SQL.
- Support for `DELETE`, `INSERT`, `REPLACE`, and `UPDATE` to return the number of rows that were changed (affected), or to return the number of rows matched instead by setting a flag when connecting to the server.

- Support for MySQL-specific `SHOW` statements that retrieve information about databases, storage engines, tables, and indexes. Support for the `INFORMATION_SCHEMA` database, implemented according to standard SQL.
- An `EXPLAIN` statement to show how the optimizer resolves a query.
- Independence of function names from table or column names. For example, `ABS` is a valid column name. The only restriction is that for a function call, no spaces are permitted between the function name and the "`(`" that follows it. See Section 9.3, "Keywords and Reserved Words".
- You can refer to tables from different databases in the same statement.

## Security

- A privilege and password system that is very flexible and secure, and that enables host-based verification.
- Password security by encryption of all password traffic when you connect to a server.

## Scalability and Limits

- Support for large databases. We use MySQL Server with databases that contain 50 million records. We also know of users who use MySQL Server with 200,000 tables and about 5,000,000,000 rows.
- Support for up to 64 indexes per table. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width for `InnoDB` tables is either 767 bytes or 3072 bytes. See Section 15.22, "InnoDB Limits". The maximum index width for `MyISAM` tables is 1000 bytes. See Section 16.2, "The MyISAM Storage Engine". An index may use a prefix of a column for `CHAR`, `VARCHAR`, `BLOB`, or `TEXT` column types.

## Connectivity

- Clients can connect to MySQL Server using several protocols:
  - Clients can connect using TCP/IP sockets on any platform.
  - On Windows systems, clients can connect using named pipes if the server is started with the `named_pipe` system variable enabled. Windows servers also support shared-memory connections if started with the `shared_memory` system variable enabled. Clients can connect through shared memory by using the `--protocol=memory` option.
  - On Unix systems, clients can connect using Unix domain socket files.
- MySQL client programs can be written in many languages. A client library written in C is available for clients written in C or C++, or for any language that provides C bindings.
- APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available, enabling MySQL clients to be written in many languages. See Chapter 29, *Connectors and APIs*.
- The Connector/ODBC (MyODBC) interface provides MySQL support for client programs that use ODBC (Open Database Connectivity) connections. For example, you can use MS Access to connect to your MySQL server. Clients can be run on Windows or Unix. Connector/ODBC source is available. All ODBC 2.5 functions are supported, as are many others. See MySQL Connector/ODBC Developer Guide.
- The Connector/J interface provides MySQL support for Java client programs that use JDBC connections. Clients can be run on Windows or Unix. Connector/J source is available. See MySQL Connector/J 8.0 Developer Guide.
- MySQL Connector/NET enables developers to easily create .NET applications that require secure, high-performance data connectivity with MySQL. It implements the required ADO.NET interfaces and integrates into ADO.NET aware tools. Developers can build applications using their choice of .NET

languages. MySQL Connector/NET is a fully managed ADO.NET driver written in 100% pure C#. See MySQL Connector/NET Developer Guide.

### Localization

- The server can provide error messages to clients in many languages. See Section 10.12, “Setting the Error Message Language”.
- Full support for several different character sets, including `latin1` (cp1252), `german`, `big5`, `ujis`, several Unicode character sets, and more. For example, the Scandinavian characters “å”, “ä” and “ö” are permitted in table and column names.
- All data is saved in the chosen character set.
- Sorting and comparisons are done according to the default character set and collation. It is possible to change this when the MySQL server is started (see Section 10.3.2, “Server Character Set and Collation”). To see an example of very advanced sorting, look at the Czech sorting code. MySQL Server supports many different character sets that can be specified at compile time and runtime.
- The server time zone can be changed dynamically, and individual clients can specify their own time zone. See Section 5.1.15, “MySQL Server Time Zone Support”.

### Clients and Tools

- MySQL includes several client and utility programs. These include both command-line programs such as `mysqldump` and `mysqladmin`, and graphical programs such as MySQL Workbench.
- MySQL Server has built-in support for SQL statements to check, optimize, and repair tables. These statements are available from the command line through the `mysqlcheck` client. MySQL also includes `myisamchk`, a very fast command-line utility for performing these operations on `MyISAM` tables. See Chapter 4, *MySQL Programs*.
- MySQL programs can be invoked with the `--help` or `-?` option to obtain online assistance.

## 1.2.3 History of MySQL

We started out with the intention of using the `mSQL` database system to connect to our tables using our own fast low-level (ISAM) routines. However, after some testing, we came to the conclusion that `mSQL` was not fast enough or flexible enough for our needs. This resulted in a new SQL interface to our database but with almost the same API interface as `mSQL`. This API was designed to enable third-party code that was written for use with `mSQL` to be ported easily for use with MySQL.

MySQL is named after co-founder Monty Widenius's daughter, My.

The name of the MySQL Dolphin (our logo) is “Sakila,” which was chosen from a huge list of names suggested by users in our “Name the Dolphin” contest. The winning name was submitted by Ambrose Twebaze, an Open Source software developer from Eswatini (formerly Swaziland), Africa. According to Ambrose, the feminine name Sakila has its roots in SiSwati, the local language of Eswatini. Sakila is also the name of a town in Arusha, Tanzania, near Ambrose's country of origin, Uganda.

# 1.3 What Is New in MySQL 8.0

This section summarizes what has been added to, deprecated in, and removed from MySQL 8.0. A companion section lists MySQL server options and variables that have been added, deprecated, or removed in MySQL 8.0; see Section 1.4, “Server and Status Variables and Options Added, Deprecated, or Removed in MySQL 8.0”.

- Features Added in MySQL 8.0
- Features Deprecated in MySQL 8.0
- Features Removed in MySQL 8.0

# Features Added in MySQL 8.0

The following features have been added to MySQL 8.0:

- **Data dictionary.** MySQL now incorporates a transactional data dictionary that stores information about database objects. In previous MySQL releases, dictionary data was stored in metadata files and nontransactional tables. For more information, see Chapter 14, *MySQL Data Dictionary*.

- **Atomic data definition statements (Atomic DDL).** An atomic DDL statement combines the data dictionary updates, storage engine operations, and binary log writes associated with a DDL operation into a single, atomic transaction. For more information, see Section 13.1.1, “Atomic Data Definition Statement Support”.

- **Upgrade procedure.** Previously, after installation of a new version of MySQL, the MySQL server automatically upgrades the data dictionary tables at the next startup, after which the DBA is expected to invoke `mysql_upgrade` manually to upgrade the system tables in the `mysql` schema, as well as objects in other schemas such as the `sys` schema and user schemas.

  As of MySQL 8.0.16, the server performs the tasks previously handled by `mysql_upgrade`. After installation of a new MySQL version, the server now automatically performs all necessary upgrade tasks at the next startup and is not dependent on the DBA invoking `mysql_upgrade`. In addition, the server updates the contents of the help tables (something `mysql_upgrade` did not do). A new `--upgrade` server option provides control over how the server performs automatic data dictionary and server upgrade operations. For more information, see Section 2.10.3, “What the MySQL Upgrade Process Upgrades”.

- **Session Reuse.** MySQL Server now supports SSL session reuse by default with a timeout setting to control how long the server maintains a session cache that establishes the period during which a client is permitted to request session reuse for new connections. All MySQL client programs support session reuse. For server-side and client-side configuration information, see Section 6.3.5, “Reusing SSL Sessions”.

  In addition, C applications now can use the C API capabilities to enable session reuse for encrypted connections (see SSL Session Reuse).

- **Security and account management.** These enhancements were added to improve security and enable greater DBA flexibility in account management:

  - The grant tables in the `mysql` system database are now `InnoDB` (transactional) tables. Previously, these were `MyISAM` (nontransactional) tables. The change of grant table storage engine underlies an accompanying change to the behavior of account-management statements. Previously, an account-management statement (such as `CREATE USER` or `DROP USER`) that named multiple users could succeed for some users and fail for others. Now, each statement is transactional and either succeeds for all named users or rolls back and has no effect if any error occurs. The statement is written to the binary log if it succeeds, but not if it fails; in that case, rollback occurs and no changes are made. For more information, see Section 13.1.1, “Atomic Data Definition Statement Support”.

  - A new `caching_sha2_password` authentication plugin is available. Like the `sha256_password` plugin, `caching_sha2_password` implements SHA-256 password hashing, but uses caching to address latency issues at connect time. It also supports more transport protocols and does not require linking against OpenSSL for RSA key pair-based password-exchange capabilities. See Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

    The `caching_sha2_password` and `sha256_password` authentication plugins provide more secure password encryption than the `mysql_native_password` plugin, and `caching_sha2_password` provides better performance than `sha256_password`. Due to these superior security and performance characteristics of `caching_sha2_password`, it is now the preferred authentication plugin, and is also the default authentication plugin rather than `mysql_native_password`. For information about the implications of this change of default

plugin for server operation and compatibility of the server with clients and connectors, see caching_sha2_password as the Preferred Authentication Plugin.

- The MySQL Enterprise Edition SASL LDAP authentication plugin now supports GSSAPI/Kerberos as an authentication method for MySQL clients and servers on Linux. This is useful in Linux environments where applications access LDAP using Microsoft Active Directory, which has Kerberos enabled by default. See LDAP Authentication Methods.

- MySQL Enterprise Edition now supports an authentication method that enables users to authenticate to MySQL Server using Kerberos, provided that appropriate Kerberos tickets are available or can be obtained. For details, see Section 6.4.1.8, “Kerberos Pluggable Authentication”.

- MySQL now supports roles, which are named collections of privileges. Roles can be created and dropped. Roles can have privileges granted to and revoked from them. Roles can be granted to and revoked from user accounts. The active applicable roles for an account can be selected from among those granted to the account, and can be changed during sessions for that account. For more information, see Section 6.2.10, “Using Roles”.

- MySQL now incorporates the concept of user account categories, with system and regular users distinguished according to whether they have the `SYSTEM_USER` privilege. See Section 6.2.11, “Account Categories”.

- Previously, it was not possible to grant privileges that apply globally except for certain schemas. This is now possible if the `partial_revokes` system variable is enabled. See Section 6.2.12, “Privilege Restriction Using Partial Revokes”.

- The `GRANT` statement has an `AS *user* [WITH ROLE]` clause that specifies additional information about the privilege context to use for statement execution. This syntax is visible at the SQL level, although its primary purpose is to enable uniform replication across all nodes of grantor privilege restrictions imposed by partial revokes, by causing those restrictions to appear in the binary log. See Section 13.7.1.6, “GRANT Statement”.

- MySQL now maintains information about password history, enabling restrictions on reuse of previous passwords. DBAs can require that new passwords not be selected from previous passwords for some number of password changes or period of time. It is possible to establish password-reuse policy globally as well as on a per-account basis.

  It is now possible to require that attempts to change account passwords be verified by specifying the current password to be replaced. This enables DBAs to prevent users from changing password without proving that they know the current password. It is possible to establish password-verification policy globally as well as on a per-account basis.

  Accounts are now permitted to have dual passwords, which enables phased password changes to be performed seamlessly in complex multiple-server systems, without downtime.

  MySQL now enables administrators to configure user accounts such that too many consecutive login failures due to incorrect passwords cause temporary account locking. The required number of failures and the lock time are configurable per account.

  These new capabilities provide DBAs more complete control over password management. For more information, see Section 6.2.15, “Password Management”.

- MySQL now supports FIPS mode, if compiled using OpenSSL, and an OpenSSL library and FIPS Object Module are available at runtime. FIPS mode imposes conditions on cryptographic operations such as restrictions on acceptable encryption algorithms or requirements for longer key lengths. See Section 6.8, “FIPS Support”.

- The TLS context the server uses for new connections now is reconfigurable at runtime. This capability may be useful, for example, to avoid restarting a MySQL server that has been running

so long that its SSL certificate has expired. See Server-Side Runtime Configuration and Monitoring for Encrypted Connections.

- OpenSSL 1.1.1 supports the TLS v1.3 protocol for encrypted connections, and MySQL 8.0.16 and higher supports TLS v1.3 as well, if both the server and client are compiled using OpenSSL 1.1.1 or higher. See Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- MySQL now sets the access control granted to clients on the named pipe to the minimum necessary for successful communication on Windows. Newer MySQL client software can open named pipe connections without any additional configuration. If older client software cannot be upgraded immediately, the new `named_pipe_full_access_group` system variable can be used to give a Windows group the necessary permissions to open a named pipe connection. Membership in the full-access group should be restricted and temporary.

- Previously, MySQL user accounts authenticated to the server using a single authentication method. As of MySQL 8.0.27, MySQL supports multifactor authentication (MFA), which makes it possible to create accounts that have up to three authentication methods. MFA support entails these changes:

  - `CREATE USER` and `ALTER USER` syntax has been extended to permit specification of multiple authentication methods.

  - The `authentication_policy` system variable enables MFA policy to be established by controlling how many factors can be used and the types of authentication permitted for each factor. This places constraints on how the authentication-related clauses of `CREATE USER` and `ALTER USER` statements may be used.

  - Client programs have new `--password1`, `--password2`, and `--password3` command-line options for specifying multiple passwords. For applications that use the C API, the new `MYSQL_OPT_USER_PASSWORD` option for the `mysql_options4()` C API function enables the same capability.

  In addition, MySQL Enterprise Edition now supports authentication to MySQL Server using devices such as smart cards, security keys, and biometric readers. This authentication method is based on the Fast Identity Online (FIDO) standard, and uses a pair of plugins, `authentication_fido` on the server side and `authentication_fido_client` on the client side. The server-side FIDO authentication plugin is included only in MySQL Enterprise Edition distributions. It is not included in MySQL community distributions. However, the client-side plugin is included in all distributions, including community distributions. This enables clients from any distribution to connect to a server that has the server-side plugin loaded.

  Multifactor authentication can use existing MySQL authentication methods, the new FIDO authentication method, or a combination of both. For more information, see Section 6.2.18, “Multifactor Authentication”, and Section 6.4.1.11, “FIDO Pluggable Authentication”.

- **Resource management.** MySQL now supports creation and management of resource groups, and permits assigning threads running within the server to particular groups so that threads execute according to the resources available to the group. Group attributes enable control over its resources, to enable or restrict resource consumption by threads in the group. DBAs can modify these attributes as appropriate for different workloads. Currently, CPU time is a manageable resource, represented by the concept of “virtual CPU” as a term that includes CPU cores, hyperthreads, hardware threads, and so forth. The server determines at startup how many virtual CPUs are available, and database administrators with appropriate privileges can associate these CPUs with resource groups and assign threads to groups. For more information, see Section 5.1.16, “Resource Groups”.

- **Table encryption management.** Table encryption can now be managed globally by defining and enforcing encryption defaults. The `default_table_encryption` variable defines an encryption default for newly created schemas and general tablespace. The encryption default for a schema can also be defined using the `DEFAULT ENCRYPTION` clause when creating a schema. By default, a table inherits the encryption of the schema or general tablespace it is created in.

Encryption defaults are enforced by enabling the `table_encryption_privilege_check` variable. The privilege check occurs when creating or altering a schema or general tablespace with an encryption setting that differs from the `default_table_encryption` setting, or when creating or altering a table with an encryption setting that differs from the default schema encryption. The `TABLE_ENCRYPTION_ADMIN` privilege permits overriding default encryption settings when `table_encryption_privilege_check` is enabled. For more information, see Defining an Encryption Default for Schemas and General Tablespaces.

- **InnoDB enhancements.** These `InnoDB` enhancements were added:

  - The current maximum auto-increment counter value is written to the redo log each time the value changes, and saved to an engine-private system table on each checkpoint. These changes make the current maximum auto-increment counter value persistent across server restarts. Additionally:

    - A server restart no longer cancels the effect of the `AUTO_INCREMENT = N` table option. If you initialize the auto-increment counter to a specific value, or if you alter the auto-increment counter value to a larger value, the new value is persisted across server restarts.

    - A server restart immediately following a `ROLLBACK` operation no longer results in the reuse of auto-increment values that were allocated to the rolled-back transaction.

    - If you modify an `AUTO_INCREMENT` column value to a value larger than the current maximum auto-increment value (in an `UPDATE` operation, for example), the new value is persisted, and subsequent `INSERT` operations allocate auto-increment values starting from the new, larger value.

    For more information, see Section 15.6.1.6, “AUTO_INCREMENT Handling in InnoDB”, and InnoDB AUTO_INCREMENT Counter Initialization.

  - When encountering index tree corruption, `InnoDB` writes a corruption flag to the redo log, which makes the corruption flag crash safe. `InnoDB` also writes in-memory corruption flag data to an engine-private system table on each checkpoint. During recovery, `InnoDB` reads corruption flags from both locations and merges results before marking in-memory table and index objects as corrupt.

  - The `InnoDB` `memcached` plugin supports multiple `get` operations (fetching multiple key-value pairs in a single `memcached` query) and range queries. See Section 15.20.4, “InnoDB memcached Multiple get and Range Query Support”.

  - A new dynamic variable, `innodb_deadlock_detect`, may be used to disable deadlock detection. On high concurrency systems, deadlock detection can cause a slowdown when numerous threads wait for the same lock. At times, it may be more efficient to disable deadlock detection and rely on the `innodb_lock_wait_timeout` setting for transaction rollback when a deadlock occurs.

  - The new Information Schema `INNODB_CACHED_INDEXES` table reports the number of index pages cached in the `InnoDB` buffer pool for each index.

  - `InnoDB` temporary tables are now created in the shared temporary tablespace, `ibtmp1`.

  - The `InnoDB` tablespace encryption feature supports encryption of redo log and undo log data. See Redo Log Encryption, and Undo Log Encryption.

  - `InnoDB` supports `NOWAIT` and `SKIP LOCKED` options with `SELECT ... FOR SHARE` and `SELECT ... FOR UPDATE` locking read statements. `NOWAIT` causes the statement to return immediately if a requested row is locked by another transaction. `SKIP LOCKED` removes locked rows from the result set. See Locking Read Concurrency with NOWAIT and SKIP LOCKED.

    `SELECT ... FOR SHARE` replaces `SELECT ... LOCK IN SHARE MODE`, but `LOCK IN SHARE MODE` remains available for backward compatibility. The statements are equivalent.

However, `FOR UPDATE` and `FOR SHARE` support `NOWAIT`, `SKIP LOCKED`, and `OF tbl_name` options. See Section 13.2.13, “SELECT Statement”.

`OF tbl_name` applies locking queries to named tables.

- `ADD PARTITION`, `DROP PARTITION`, `COALESCE PARTITION`, `REORGANIZE PARTITION`, and `REBUILD PARTITION` `ALTER TABLE` options are supported by native partitioning in-place APIs and may be used with `ALGORITHM={COPY|INPLACE}` and `LOCK` clauses.

  `DROP PARTITION` with `ALGORITHM=INPLACE` deletes data stored in the partition and drops the partition. However, `DROP PARTITION` with `ALGORITHM=COPY` or `old_alter_table=ON` rebuilds the partitioned table and attempts to move data from the dropped partition to another partition with a compatible `PARTITION ... VALUES` definition. Data that cannot be moved to another partition is deleted.

- The `InnoDB` storage engine now uses the MySQL data dictionary rather than its own storage engine-specific data dictionary. For information about the data dictionary, see Chapter 14, *MySQL Data Dictionary*.

- `mysql` system tables and data dictionary tables are now created in a single `InnoDB` tablespace file named `mysql.ibd` in the MySQL data directory. Previously, these tables were created in individual `InnoDB` tablespace files in the `mysql` database directory.

- The following undo tablespace changes are introduced in MySQL 8.0:

  - By default, undo logs now reside in two undo tablespaces that are created when the MySQL instance is initialized. Undo logs are no longer created in the system tablespace.

  - As of MySQL 8.0.14, additional undo tablespaces can be created in a chosen location at runtime using `CREATE UNDO TABLESPACE` syntax.

    ```
    CREATE UNDO TABLESPACE tablespace_name ADD DATAFILE 'file_name.ibu';
    ```

    Undo tablespaces created using `CREATE UNDO TABLESPACE` syntax can be dropped at runtime using `DROP UNDO TABLESPACE` syntax.

    ```
    DROP UNDO TABLESPACE tablespace_name;
    ```

    `ALTER UNDO TABLESPACE` syntax can be used to mark an undo tablespace as active or inactive.

    ```
    ALTER UNDO TABLESPACE tablespace_name SET {ACTIVE|INACTIVE};
    ```

    A `STATE` column that shows the state of a tablespace was added to the Information Schema `INNODB_TABLESPACES` table. An undo tablespace must be in an `empty` state before it can be dropped.

  - The `innodb_undo_log_truncate` variable is enabled by default.

  - The `innodb_rollback_segments` variable defines the number of rollback segments per undo tablespace. Previously, `innodb_rollback_segments` specified the total number of rollback segments for the MySQL instance. This change increases the number of rollback segments available for concurrent transactions. More rollback segments increases the likelihood that concurrent transactions use separate rollback segments for undo logs, resulting in less resource contention.

- Default values for variables that affect buffer pool preflushing and flushing behavior were modified:

  - The `innodb_max_dirty_pages_pct_lwm` default value is now 10. The previous default value of 0 disables buffer pool preflushing. A value of 10 enables preflushing when the

percentage of dirty pages in the buffer pool exceeds 10%. Enabling preflushing improves performance consistency.

- The `innodb_max_dirty_pages_pct` default value was increased from 75 to 90. `InnoDB` attempts to flush data from the buffer pool so that the percentage of dirty pages does not exceed this value. The increased default value permits a greater percentage of dirty pages in the buffer pool.

- The default `innodb_autoinc_lock_mode` setting is now 2 (interleaved). Interleaved lock mode permits the execution of multi-row inserts in parallel, which improves concurrency and scalability. The new `innodb_autoinc_lock_mode` default setting reflects the change from statement-based replication to row based replication as the default replication type in MySQL 5.7. Statement-based replication requires the consecutive auto-increment lock mode (the previous default) to ensure that auto-increment values are assigned in a predictable and repeatable order for a given sequence of SQL statements, whereas row-based replication is not sensitive to the execution order of SQL statements. For more information, see InnoDB AUTO_INCREMENT Lock Modes.

  For systems that use statement-based replication, the new `innodb_autoinc_lock_mode` default setting may break applications that depend on sequential auto-increment values. To restore the previous default, set `innodb_autoinc_lock_mode` to 1.

- Renaming a general tablespace is supported by `ALTER TABLESPACE ... RENAME TO` syntax.

- The new `innodb_dedicated_server` variable, which is disabled by default, can be used to have `InnoDB` automatically configure the following options according to the amount of memory detected on the server:

  - `innodb_buffer_pool_size`
  - `innodb_log_file_size`
  - `innodb_flush_method`

  This option is intended for MySQL server instances that run on a dedicated server. For more information, see Section 15.8.12, “Enabling Automatic Configuration for a Dedicated MySQL Server”.

- The new Information Schema `INNODB_TABLESPACES_BRIEF` view provides space, name, path, flag, and space type data for `InnoDB` tablespaces.

- The zlib library version bundled with MySQL was raised from version 1.2.3 to version 1.2.11. MySQL implements compression with the help of the zlib library.

  If you use `InnoDB` compressed tables, see Section 2.10.4, “Changes in MySQL 8.0” for related upgrade implications.

- Serialized dictionary information (SDI) is present in all `InnoDB` tablespace files except for global temporary tablespace and undo tablespace files. SDI is serialized metadata for table and tablespace objects. The presence of SDI data provides metadata redundancy. For example, dictionary object metadata may be extracted from tablespace files if the data dictionary becomes

unavailable. SDI extraction is performed using the `ibd2sdi` tool. SDI data is stored in `JSON` format.

The inclusion of SDI data in tablespace files increases tablespace file size. An SDI record requires a single index page, which is 16KB in size by default. However, SDI data is compressed when it is stored to reduce the storage footprint.

- The `InnoDB` storage engine now supports atomic DDL, which ensures that DDL operations are either fully committed or rolled back, even if the server halts during the operation. For more information, see Section 13.1.1, “Atomic Data Definition Statement Support”.
- Tablespace files can be moved or restored to a new location while the server is offline using the `innodb_directories` option. For more information, see Section 15.6.3.6, “Moving Tablespace Files While the Server is Offline”.
- The following redo logging optimizations were implemented:
  - User threads can now write concurrently to the log buffer without synchronizing writes.
  - User threads can now add dirty pages to the flush list in a relaxed order.
  - A dedicated log thread is now responsible for writing the log buffer to the system buffers, flushing system buffers to disk, notifying user threads about written and flushed redo, maintaining the lag required for the relaxed flush list order, and write checkpoints.
  - System variables were added for configuring the use of spin delay by user threads waiting for flushed redo:
    - `innodb_log_wait_for_flush_spin_hwm`: Defines the maximum average log flush time beyond which user threads no longer spin while waiting for flushed redo.
    - `innodb_log_spin_cpu_abs_lwm`: Defines the minimum amount of CPU usage below which user threads no longer spin while waiting for flushed redo.
    - `innodb_log_spin_cpu_pct_hwm`: Defines the maximum amount of CPU usage above which user threads no longer spin while waiting for flushed redo.
  - The `innodb_log_buffer_size` variable is now dynamic, which permits resizing of the log buffer while the server is running.

  For more information, see Section 8.5.4, “Optimizing InnoDB Redo Logging”.
- As of MySQL 8.0.12, undo logging is supported for small updates to large object (LOB) data, which improves performance of LOB updates that are 100 bytes in size or less. Previously, LOB updates were a minimum of one LOB page in size, which is less than optimal for updates that might only

modify a few bytes. This enhancement builds upon support added in MySQL 8.0.4 for partial update of LOB data.

- As of MySQL 8.0.12, `ALGORITHM=INSTANT` is supported for the following `ALTER TABLE` operations:

  - Adding a column. This feature is also referred to as "Instant `ADD COLUMN`". Limitations apply. See Section 15.12.1, "Online DDL Operations".
  - Adding or dropping a virtual column.
  - Adding or dropping a column default value.
  - Modifying the definition of an `ENUM` or `SET` column.
  - Changing the index type.
  - Renaming a table.

  Operations that support `ALGORITHM=INSTANT` only modify metadata in the data dictionary. No metadata locks are taken on the table, and table data is unaffected, making the operations instantaneous. If not specified explicitly, `ALGORITHM=INSTANT` is used by default by operations that support it. If `ALGORITHM=INSTANT` is specified but not supported, the operation fails immediately with an error.

  For more information about operations that support `ALGORITHM=INSTANT`, see Section 15.12.1, "Online DDL Operations".

- As of MySQL 8.0.13, the `TempTable` storage engine supports storage of binary large object (BLOB) type columns. This enhancement improves performance for queries that use temporary tables containing BLOB data. Previously, temporary tables that contained BLOB data were stored in the on-disk storage engine defined by `internal_tmp_disk_storage_engine`. For more information, see Section 8.4.4, "Internal Temporary Table Use in MySQL".

- As of MySQL 8.0.13, the `InnoDB` data-at-rest encryption feature supports general tablespaces. Previously, only file-per-table tablespaces could be encrypted. To support encryption of general tablespaces, `CREATE TABLESPACE` and `ALTER TABLESPACE` syntax was extended to include an `ENCRYPTION` clause.

  The Information Schema `INNODB_TABLESPACES` table now includes an `ENCRYPTION` column that indicates whether or not a tablespace is encrypted.

  The `stage/innodb/alter tablespace (encryption)` Performance Schema stage instrument was added to permit monitoring of general tablespace encryption operations.

- Disabling the `innodb_buffer_pool_in_core_file` variable reduces the size of core files by excluding `InnoDB` buffer pool pages. To use this variable, the `core_file` variable must be enabled and the operating system must support the `MADV_DONTDUMP` non-POSIX extension to `madvise()`, which is supported in Linux 3.4 and later. For more information, see Section 15.8.3.7, "Excluding Buffer Pool Pages from Core Files".

- As of MySQL 8.0.13, user-created temporary tables and internal temporary tables created by the optimizer are stored in session temporary tablespaces that are allocated to a session from a pool of temporary tablespaces. When a session disconnects, its temporary tablespaces are truncated and released back to the pool. In previous releases, temporary tables were created in the global

temporary tablespace (`ibtmp1`), which did not return disk space to the operating system after temporary tables were dropped.

The `innodb_temp_tablespaces_dir` variable defines the location where session temporary tablespaces are created. The default location is the `#innodb_temp` directory in the data directory.

The `INNODB_SESSION_TEMP_TABLESPACES` table provides metadata about session temporary tablespaces.

The global temporary tablespace (`ibtmp1`) now stores rollback segments for changes made to user-created temporary tables.

- As of MySQL 8.0.14, `InnoDB` supports parallel clustered index reads, which can improve `CHECK TABLE` performance. This feature does not apply to secondary index scans. The `innodb_parallel_read_threads` session variable must be set to a value greater than 1 for parallel clustered index reads to occur. The default value is 4. The actual number of threads used to perform a parallel clustered index read is determined by the `innodb_parallel_read_threads` setting or the number of index subtrees to scan, whichever is smaller.

- As of 8.0.14, when the `innodb_dedicated_server` variable is enabled, the size and number of log files are configured according to the automatically configured buffer pool size. Previously, log file size was configured according to the amount of memory detected on the server, and the number of log files was not configured automatically.

- As of 8.0.14, the `ADD DATAFILE` clause of the `CREATE TABLESPACE` statement is optional, which permits users without the `FILE` privilege to create tablespaces. A `CREATE TABLESPACE` statement executed without an `ADD DATAFILE` clause implicitly creates a tablespace data file with a unique file name.

- By default, when the amount of memory occupied by the TempTable storage engine exceeds the memory limit defined by the `temptable_max_ram` variable, the TempTable storage engine begins allocating memory-mapped temporary files from disk. As of MySQL 8.0.16, this behavior is controlled by the `temptable_use_mmap` variable. Disabling `temptable_use_mmap` causes the TempTable storage engine to use `InnoDB` on-disk internal temporary tables instead of memory-mapped files as its overflow mechanism. For more information, see Internal Temporary Table Storage Engine.

- As of MySQL 8.0.16, the `InnoDB` data-at-rest encryption feature supports encryption of the `mysql` system tablespace. The `mysql` system tablespace contains the `mysql` system database and the MySQL data dictionary tables. For more information, see Section 15.13, “InnoDB Data-at-Rest Encryption”.

- The `innodb_spin_wait_pause_multiplier` variable, introduced in MySQL 8.0.16, provides greater control over the duration of spin-lock polling delays that occur when a thread waits to acquire a mutex or rw-lock. Delays can be tuned more finely to account for differences in PAUSE

instruction duration on different processor architectures. For more information, see Section 15.8.8, “Configuring Spin Lock Polling”.

- `InnoDB` parallel read thread performance for large data sets was improved in MySQL 8.0.17 through better utilization of read threads, through a reduction in read thread I/O for prefetch activity that occurs during parallel scans, and through support for parallel scanning of partitions.

  The parallel read thread feature is controlled by the `innodb_parallel_read_threads` variable. The maximum setting is now 256, which is the total number of threads for all client connections. If the thread limit is reached, connections fall back to using a single thread.

- The `innodb_idle_flush_pct` variable, introduced in MySQL 8.0.18, permits placing a limit on page flushing during idle periods, which can help extend the life of solid state storage devices. See Limiting Buffer Flushing During Idle Periods.

- Efficient sampling of `InnoDB` data for the purpose of generating histogram statistics is supported as of MySQL 8.0.19. See Histogram Statistics Analysis.

- As of MySQL 8.0.20, the doublewrite buffer storage area resides in doublewrite files. In previous releases, the storage area resided in the system tablespace. Moving the storage area out of the system tablespace reduces write latency, increases throughput, and provides flexibility with respect to placement of doublewrite buffer pages. The following system variables were introduced for advanced doublewrite buffer configuration:

  - `innodb_doublewrite_dir`

    Defines the doublewrite buffer file directory.

  - `innodb_doublewrite_files`

    Defines the number of doublewrite files.

  - `innodb_doublewrite_pages`

    Defines the maximum number of doublewrite pages per thread for a batch write.

  - `innodb_doublewrite_batch_size`

    Defines the number of doublewrite pages to write in a batch.

  For more information, see Section 15.6.4, “Doublewrite Buffer”.

- The Contention-Aware Transaction Scheduling (CATS) algorithm, which prioritizes transactions that are waiting for locks, was improved in MySQL 8.0.20. Transaction scheduling weight computation is now performed a separate thread entirely, which improves computation performance and accuracy.

  The First In First Out (FIFO) algorithm, which had also been used for transaction scheduling, was removed. The FIFO algorithm was rendered redundant by CATS algorithm enhancements.

Transaction scheduling previously performed by the FIFO algorithm is now performed by the CATS algorithm.

A `TRX_SCHEDULE_WEIGHT` column was added to the `INFORMATION_SCHEMA.INNODB_TRX` table, which permits querying transaction scheduling weights assigned by the CATS algorithm.

The following `INNODB_METRICS` counters were added for monitoring code-level transaction scheduling events:

- `lock_rec_release_attempts`

  The number of attempts to release record locks.

- `lock_rec_grant_attempts`

  The number of attempts to grant record locks.

- `lock_schedule_refreshes`

  The number of times the wait-for graph was analyzed to update transaction schedule weights.

For more information, see Section 15.7.6, “Transaction Scheduling”.

- As of MySQL 8.0.21, to improve concurrency for operations that require access to lock queues for table and row resources, the lock system mutex (`lock_sys->mutex`) was replaced in by sharded latches, and lock queues were grouped into table and page *lock queue shards*, with each shard protected by a dedicated mutex. Previously, the single lock system mutex protected all lock queues, which was a point of contention on high-concurrency systems. The new sharded implementation permits more granular access to lock queues.

  The lock system mutex (`lock_sys->mutex`) was replaced by the following sharded latches:

  - A global latch (`lock_sys->latches.global_latch`) consisting of 64 read-write lock objects (`rw_lock_t`). Access to an individual lock queue requires a shared global latch and a latch on the lock queue shard. Operations that require access to all lock queues take an exclusive global latch, which latches all table and page lock queue shards.

  - Table shard latches (`lock_sys->latches.table_shards.mutexes`), implemented as an array of 512 mutexes, with each mutex dedicated to one of 512 table lock queue shards.

  - Page shard latches (`lock_sys->latches.page_shards.mutexes`), implemented as an array of 512 mutexes, with each mutex dedicated to one of 512 page lock queue shards.

  The Performance Schema `wait/synch/mutex/innodb/lock_mutex` instrument for monitoring the single lock system mutex was replaced by instruments for monitoring the new global, table shard, and page shard latches:

  - `wait/synch/sxlock/innodb/lock_sys_global_rw_lock`

  - `wait/synch/mutex/innodb/lock_sys_table_mutex`

  - `wait/synch/mutex/innodb/lock_sys_page_mutex`

- As of MySQL 8.0.21, table and table partition data files created outside of the data directory using the `DATA DIRECTORY` clause are restricted to directories known to `InnoDB`. This change permits

database administrators to control where tablespace data files are created and ensures that the data files can be found during recovery.

General and file-per-table tablespaces data files (`.ibd` files) can no longer be created in the undo tablespace directory (`innodb_undo_directory`) unless that directly is known to `InnoDB`.

Known directories are those defined by the `datadir`, `innodb_data_home_dir`, and `innodb_directories` variables.

Truncating an `InnoDB` table that resides in a file-per-table tablespace drops the existing tablespace and creates a new one. As of MySQL 8.0.21, `InnoDB` creates the new tablespace in the default location and writes a warning to the error log if the current tablespace directory is unknown. To have `TRUNCATE TABLE` create the tablespace in its current location, add the directory to the `innodb_directories` setting before running `TRUNCATE TABLE`.

- As of MySQL 8.0.21, redo logging can be enabled and disabled using `ALTER INSTANCE {ENABLE|DISABLE} INNODB REDO_LOG` syntax. This functionality is intended for loading data into a new MySQL instance. Disabling redo logging helps speed up data loading by avoiding redo log writes.

  The new `INNODB_REDO_LOG_ENABLE` privilege permits enabling and disabling redo logging.

  The new `Innodb_redo_log_enabled` status variable permits monitoring redo logging status.

  See Disabling Redo Logging.

- At startup, `InnoDB` validates the paths of known tablespace files against tablespace file paths stored in the data dictionary in case tablespace files have been moved to a different location. The new `innodb_validate_tablespace_paths` variable, introduced in MySQL 8.0.21, permits disabling tablespace path validation. This feature is intended for environments where tablespaces files are not moved. Disabling tablespace path validation improves startup time on systems with a large number of tablespace files.

  For more information, see Section 15.6.3.7, “Disabling Tablespace Path Validation”.

- As of MySQL 8.0.21, on storage engines that support atomic DDL, the `CREATE TABLE ... SELECT` statement is logged as one transaction in the binary log when row-based replication is in use. Previously, it was logged as two transactions, one to create the table, and the other to insert data. With this change, `CREATE TABLE ... SELECT` statements are now safe for row-based replication and permitted for use with GTID-based replication. For more information, see Section 13.1.1, “Atomic Data Definition Statement Support”.

- Truncating an undo tablespace on a busy system could affect performance due to associated flushing operations that remove old undo tablespace pages from the buffer pool and flush the initial pages of the new undo tablespace to disk. To address this issue, the flushing operations are removed as of MySQL 8.0.21.

  Old undo tablespace pages are released passively as they become least recently used, or are removed at the next full checkpoint. The initial pages of the new undo tablespace are now redo logged instead of flushed to disk during the truncate operation, which also improves durability of the undo tablespace truncate operation.

  To prevent potential issues caused by an excessive number of undo tablespace truncate operations, truncate operations on the same undo tablespace between checkpoints are now limited to 64. If the limit is exceeded, an undo tablespace can still be made inactive, but it is not truncated until after the next checkpoint.

  `INNODB_METRICS` counters associated with defunct undo truncate flushing operations were removed. Removed counters include: `undo_truncate_sweep_count`,

`undo_truncate_sweep_usec`, `undo_truncate_flush_count`, and `undo_truncate_flush_usec`.

See Section 15.6.3.4, “Undo Tablespaces”.

- As of MySQL 8.0.22, the new `innodb_extend_and_initialize` variable permits configuring how `InnoDB` allocates space to file-per-table and general tablespaces on Linux. By default, when an operation requires additional space in a tablespace, `InnoDB` allocates pages to the tablespace and physically writes NULLs to those pages. This behavior affects performance if new pages are allocated frequently. You can disable `innodb_extend_and_initialize` on Linux systems to avoid physically writing NULLs to newly allocated tablespace pages. When `innodb_extend_and_initialize` is disabled, space is allocated using `posix_fallocate()` calls, which reserve space without physically writing NULLs.

  A `posix_fallocate()` operation is not atomic, which makes it possible for a failure to occur between allocating space to a tablespace file and updating the file metadata. Such a failure can leave newly allocated pages in an uninitialized state, resulting in a failure when `InnoDB` attempts to access those pages. To prevent this scenario, `InnoDB` writes a redo log record before allocating a new tablespace page. If a page allocation operation is interrupted, the operation is replayed from the redo log record during recovery.

- As of MySQL 8.0.23, `InnoDB` supports encryption of doublewrite file pages belonging to encrypted tablespaces. The pages are encrypted using the encryption key of the associated tablespace. For more information, see Section 15.13, “InnoDB Data-at-Rest Encryption”.

- The `temptable_max_mmap` variable, introduced in MySQL 8.0.23, defines the maximum amount of memory the TempTable storage engine is permitted to allocate from memory-mapped (MMAP) files before it starts storing internal temporary table data on disk. A setting of 0 disables allocation from MMAP files. For more information, see Section 8.4.4, “Internal Temporary Table Use in MySQL”.

- The `AUTOEXTEND_SIZE` option, introduced in MySQL 8.0.23, defines the amount by which `InnoDB` extends the size of a tablespace when it becomes full, making it possible to extend tablespace size in larger increments. The `AUTOEXTEND_SIZE` option is supported with the `CREATE TABLE`, `ALTER TABLE`, `CREATE TABLESPACE`, and `ALTER TABLESPACE` statements. For more information, see Section 15.6.3.9, “Tablespace AUTOEXTEND_SIZE Configuration”.

  An `AUTOEXTEND_SIZE` size column was added to the Information Schema `INNODB_TABLESPACES` table.

- The `innodb_segment_reserve_factor` system variable, introduced in MySQL 8.0.26, permits configuring the percentage of tablespace file segment pages that are reserved as empty pages. For more information, see Configuring the Percentage of Reserved File Segment Pages.

- On platforms that support `fdatasync()` system calls, the `innodb_use_fdatasync` variable, introduced in MySQL 8.0.26, permits using `fdatasync()` instead of `fsync()` for operating system flushes. An `fdatasync()` system call does not flush changes to file metadata unless required for subsequent data retrieval, providing a potential performance benefit.

- As of MySQL 8.0.28, the `tmp_table_size` variable defines the maximum size of any individual in-memory internal temporary table created by the TempTable storage engine. An appropriate size limit prevents individual queries from consuming an inordinate amount global TempTable resources. See Internal Temporary Table Storage Engine.

- From MySQL 8.0.28, the `innodb_open_files` variable, which defines the number of files `InnoDB` can have open at one time, can be set at runtime using a `SELECT`

`innodb_set_open_files_limit(N)` statement. The statement executes a stored procedure that sets the new limit.

To prevent non-LRU manged files from consuming the entire `innodb_open_files` limit, non-LRU managed files are limited to 90 percent of the `innodb_open_files` limit, which reserves 10 percent of the `innodb_open_files` limit for LRU managed files.

The `innodb_open_files` limit includes temporary tablespace files, which were not counted toward the limit previously.

- From MySQL 8.0.28, `InnoDB` supports `ALTER TABLE ... RENAME COLUMN` operations using `ALGORITHM=INSTANT`.

  For more information about this and other DDL operations that support `ALGORITHM=INSTANT`, see Section 15.12.1, “Online DDL Operations”.

- From MySQL 8.0.29, `InnoDB` supports `ALTER TABLE ... DROP COLUMN` operations using `ALGORITHM=INSTANT`.

  Prior to MySQL 8.0.29, an instantly added column could only be added as the last column of the table. From MySQL 8.0.29, an instantly added column can be added to any position in the table.

  Instantly added or dropped columns create a new version of the affected row. Up to 64 row versions are permitted. A new `TOTAL_ROW_VERSIONS` column was added to the Information Schema `INNODB_TABLES` table to track the number of row versions.

  For more information about DDL operations that support `ALGORITHM=INSTANT`, see Section 15.12.1, “Online DDL Operations”.

- From MySQL 8.0.30, the `innodb_doublewrite` system variable supports `DETECT_ONLY` and `DETECT_AND_RECOVER` settings. With the `DETECT_ONLY` setting, database page content is not written to the doublewrite buffer, and recovery does not use the doublewrite buffer to fix incomplete page writes. This lightweight setting is intended for detecting incomplete page writes only. The `DETECT_AND_RECOVER` setting is equivalent to the existing `ON` setting. For more information, see Section 15.6.4, “Doublewrite Buffer”.

- From MySQL 8.0.30, `InnoDB` supports dynamic configuration of redo log capacity. The `innodb_redo_log_capacity` system variable can be set at runtime to increase or decrease the total amount of disk space occupied by redo log files.

  With this change, the number of redo log files and their default location has also changed. From MySQL 8.0.30, `InnoDB` maintains 32 redo log files in the `#innodb_redo` directory in the data directory. Previously, `InnoDB` created two redo log files in the data directory by default, and the number and size of redo log files were controlled by the `innodb_log_files_in_group` and `innodb_log_file_size` variables. These two variables are now deprecated.

  When the `innodb_redo_log_capacity` setting is defined, `innodb_log_files_in_group` and `innodb_log_file_size` settings are ignored; otherwise, those settings are used to compute the `innodb_redo_log_capacity` setting (`innodb_log_files_in_group` * `innodb_log_file_size` = `innodb_redo_log_capacity`). If none of those variables are set, redo log capacity is set to the `innodb_redo_log_capacity` default value, which is 104857600 bytes (100MB).

  Several status variables are provided for monitoring the redo log and redo log resizing operations.

  For more information, see Section 15.6.5, “Redo Log”.

- With MySQL 8.0.31, there are two new status variables for monitoring online buffer pool resizing operations. The `Innodb_buffer_pool_resize_status_code` status variable reports a status code indicating the stage of an online buffer pool resizing operation. The

`Innodb_buffer_pool_resize_status_progress` status variable reports a percentage value indicating the progress of each stage.

For more information, see Section 15.8.3.1, “Configuring InnoDB Buffer Pool Size”.

- **Character set support.** The default character set has changed from `latin1` to `utf8mb4`. The `utf8mb4` character set has several new collations, including `utf8mb4_ja_0900_as_cs`, the first Japanese language-specific collation available for Unicode in MySQL. For more information, see Section 10.10.1, “Unicode Character Sets”.

- **JSON enhancements.** The following enhancements or additions were made to MySQL's JSON functionality:

  - Added the `->>` (inline path) operator, which is equivalent to calling `JSON_UNQUOTE()` on the result of `JSON_EXTRACT()`.

    This is a refinement of the column path operator `->` introduced in MySQL 5.7; `col->>"$.path"` is equivalent to `JSON_UNQUOTE(col->"$.path")`. The inline path operator can be used wherever you can use `JSON_UNQUOTE(JSON_EXTRACT())`, such `SELECT` column lists, `WHERE` and `HAVING` clauses, and `ORDER BY` and `GROUP BY` clauses. For more information, see the description of the operator, as well as JSON Path Syntax.

  - Added two JSON aggregation functions `JSON_ARRAYAGG()` and `JSON_OBJECTAGG()`. `JSON_ARRAYAGG()` takes a column or expression as its argument, and aggregates the result as a single `JSON` array. The expression can evaluate to any MySQL data type; this does not have to be a `JSON` value. `JSON_OBJECTAGG()` takes two columns or expressions which it interprets as a key and a value; it returns the result as a single `JSON` object. For more information and examples, see Section 12.20, “Aggregate Functions”.

  - Added the JSON utility function `JSON_PRETTY()`, which outputs an existing `JSON` value in an easy-to-read format; each JSON object member or array value is printed on a separate line, and a child object or array is intended 2 spaces with respect to its parent.

    This function also works with a string that can be parsed as a JSON value.

    For more detailed information and examples, see Section 12.18.8, “JSON Utility Functions”.

  - When sorting `JSON` values in a query using `ORDER BY`, each value is now represented by a variable-length part of the sort key, rather than a part of a fixed 1K in size. In many cases this can reduce excessive usage. For example, a scalar `INT` or even `BIGINT` value actually requires very few bytes, so that the remainder of this space (up to 90% or more) was taken up by padding. This change has the following benefits for performance:

    - Sort buffer space is now used more effectively, so that filesorts need not flush to disk as early or often as with fixed-length sort keys. This means that more data can be sorted in memory, avoiding unnecessary disk access.

    - Shorter keys can be compared more quickly than longer ones, providing a noticeable improvement in performance. This is true for sorts performed entirely in memory as well as for sorts that require writing to and reading from disk.

  - Added support in MySQL 8.0.2 for partial, in-place updates of `JSON` column values, which is more efficient than completely removing an existing JSON value and writing a new one in its place, as was done previously when updating any `JSON` column. For this optimization to be applied, the update must be applied using `JSON_SET()`, `JSON_REPLACE()`, or `JSON_REMOVE()`. New elements cannot be added to the JSON document being updated; values within the document

cannot take more space than they did before the update. See Partial Updates of JSON Values, for a detailed discussion of the requirements.

Partial updates of JSON documents can be written to the binary log, taking up less space than logging complete JSON documents. Partial updates are always logged as such when statement-based replication is in use. For this to work with row-based replication, you must first set `binlog_row_value_options=PARTIAL_JSON`; see this variable's description for more information.

- Added the JSON utility functions `JSON_STORAGE_SIZE()` and `JSON_STORAGE_FREE()`. `JSON_STORAGE_SIZE()` returns the storage space in bytes used for the binary representation of a JSON document prior to any partial update (see previous item). `JSON_STORAGE_FREE()` shows the amount of space remaining in a table column of type `JSON` after it has been partially updated using `JSON_SET()` or `JSON_REPLACE()`; this is greater than zero if the binary representation of the new value is less than that of the previous value.

  Each of these functions also accepts a valid string representation of a JSON document. For such a value, `JSON_STORAGE_SIZE()` returns the space used by its binary representation following its conversion to a JSON document. For a variable containing the string representation of a JSON document, `JSON_STORAGE_FREE()` returns zero. Either function produces an error if its (non-null) argument cannot be parsed as a valid JSON document, and `NULL` if the argument is `NULL`.

  For more information and examples, see Section 12.18.8, “JSON Utility Functions”.

  `JSON_STORAGE_SIZE()` and `JSON_STORAGE_FREE()` were implemented in MySQL 8.0.2.

- Added support in MySQL 8.0.2 for ranges such as `$[1 to 5]` in XPath expressions. Also added support in this version for the `last` keyword and relative addressing, such that `$[last]` always selects the last (highest-numbered) element in the array and `$[last-1]` the next to last element. `last` and expressions using it can also be included in range definitions. For example, `$[last-2 to last-1]` returns the last two elements but one from an array. See Searching and Modifying JSON Values, for additional information and examples.

- Added a JSON merge function intended to conform to RFC 7396. `JSON_MERGE_PATCH()`, when used on 2 JSON objects, merges them into a single JSON object that has as members a union of the following sets:

  - Each member of the first object for which there is no member with the same key in the second object.
  - Each member of the second object for which there is no member having the same key in the first object, and whose value is not the JSON `null` literal.
  - Each member having a key that exists in both objects, and whose value in the second object is not the JSON `null` literal.

  As part of this work, the `JSON_MERGE()` function has been renamed `JSON_MERGE_PRESERVE()`. `JSON_MERGE()` continues to be recognized as an alias for `JSON_MERGE_PRESERVE()` in MySQL 8.0, but is now deprecated and is subject to removal in a future version of MySQL.

  For more information and examples, see Section 12.18.4, “Functions That Modify JSON Values”.

- Implemented “last duplicate key wins” normalization of duplicate keys, consistent with RFC 7159 and most JavaScript parsers. An example of this behavior is shown here, where only the rightmost member having the key `x` is preserved:

```
mysql> SELECT JSON_OBJECT('x', '32', 'y', '[true, false]',
     >                    'x', '"abc"', 'x', '100') AS Result;
+----------------------------------+
| Result                           |
+----------------------------------+
```

```
| {"x": "100", "y": "[true, false]"} |
+------------------------------------+
1 row in set (0.00 sec)
```

Values inserted into MySQL `JSON` columns are also normalized in this way, as shown in this example:

```
mysql> CREATE TABLE t1 (c1 JSON);

mysql> INSERT INTO t1 VALUES ('{"x": 17, "x": "red", "x": [3, 5, 7]}');

mysql> SELECT c1 FROM t1;
+------------------+
| c1               |
+------------------+
| {"x": [3, 5, 7]} |
+------------------+
```

This is an incompatible change from previous versions of MySQL, where a “first duplicate key wins” algorithm was used in such cases.

See Normalization, Merging, and Autowrapping of JSON Values, for more information and examples.

- Added the `JSON_TABLE()` function in MySQL 8.0.4. This function accepts JSON data and returns it as a relational table having the specified columns.

  This function has the syntax `JSON_TABLE(expr, path COLUMNS column_list) [AS] alias)`, where *`expr`* is an expression that returns JSON data, *`path`* is a JSON path applied to the source, and *`column_list`* is a list of column definitions. An example is shown here:

```
mysql> SELECT *
    -> FROM
    ->   JSON_TABLE(
    ->     '[{"a":3,"b":"0"},{"a":"3","b":"1"},{"a":2,"b":1},{"a":0},{"b":[1,2]}]',
    ->     "$[*]" COLUMNS(
    ->       rowid FOR ORDINALITY,
    ->
    ->       xa INT EXISTS PATH "$.a",
    ->       xb INT EXISTS PATH "$.b",
    ->
    ->       sa VARCHAR(100) PATH "$.a",
    ->       sb VARCHAR(100) PATH "$.b",
    ->
    ->       ja JSON PATH "$.a",
    ->       jb JSON PATH "$.b"
    ->     )
    ->   ) AS  jt1;
+-------+------+------+------+------+------+--------+
| rowid | xa   | xb   | sa   | sb   | ja   | jb     |
+-------+------+------+------+------+------+--------+
|     1 |    1 |    1 | 3    | 0    | 3    | "0"    |
|     2 |    1 |    1 | 3    | 1    | "3"  | "1"    |
|     3 |    1 |    1 | 2    | 1    | 2    | 1      |
|     4 |    1 |    0 | 0    | NULL | 0    | NULL   |
|     5 |    0 |    1 | NULL | NULL | NULL | [1, 2] |
+-------+------+------+------+------+------+--------+
```

  The JSON source expression can be any expression that yields a valid JSON document, including a JSON literal, a table column, or a function call that returns JSON such as `JSON_EXTRACT(t1, data, '$.post.comments')`. For more information, see Section 12.18.6, “JSON Table Functions”.

- **Data type support.** MySQL now supports use of expressions as default values in data type specifications. This includes the use of expressions as default values for the `BLOB`, `TEXT`, `GEOMETRY`, and `JSON` data types, which previously could not be assigned default values at all. For details, see Section 11.6, “Data Type Default Values”.

- **Optimizer.** These optimizer enhancements were added:

  - MySQL now supports invisible indexes. An invisible index is not used by the optimizer at all, but is otherwise maintained normally. Indexes are visible by default. Invisible indexes make it possible to test the effect of removing an index on query performance, without making a destructive change that must be undone should the index turn out to be required. See Section 8.3.12, “Invisible Indexes”.

  - MySQL now supports descending indexes: `DESC` in an index definition is no longer ignored but causes storage of key values in descending order. Previously, indexes could be scanned in reverse order but at a performance penalty. A descending index can be scanned in forward order, which is more efficient. Descending indexes also make it possible for the optimizer to use multiple-column indexes when the most efficient scan order mixes ascending order for some columns and descending order for others. See Section 8.3.13, “Descending Indexes”.

  - MySQL now supports creation of functional index key parts that index expression values rather than column values. Functional key parts enable indexing of values that cannot be indexed otherwise, such as `JSON` values. For details, see Section 13.1.15, “CREATE INDEX Statement”.

  - In MySQL 8.0.14 and later, trivial `WHERE` conditions arising from constant literal expressions are removed during preparation, rather than later on during optimization. Removal of the condition earlier in the process makes it possible to simplify joins for queries with outer joins having trivial conditions, such as this one:

    ```
    SELECT * FROM t1 LEFT JOIN t2 ON condition_1 WHERE condition_2 OR 0 = 1
    ```

    The optimizer now sees during preparation that 0 = 1 is always false, making `OR 0 = 1` redundant, and removes it, leaving this:

    ```
    SELECT * FROM t1 LEFT JOIN t2 ON condition_1 where condition_2
    ```

    Now the optimizer can rewrite the query as an inner join, like this:

    ```
    SELECT * FROM t1 LEFT JOIN t2 WHERE condition_1 AND condition_2
    ```

    For more information, see Section 8.2.1.9, “Outer Join Optimization”.

  - In MySQL 8.0.16 and later, MySQL can use constant folding at optimization time to handle comparisons between a column and a constant value where the constant is out of range or on a range boundary with respect to the type of the column, rather than doing so for each row at execution time. For example, given a table `t` with a `TINYINT UNSIGNED` column `c`, the optimizer can rewrite a condition such as `WHERE c < 256` to `WHERE 1` (and optimize the condition away altogether), or `WHERE c >= 255` to `WHERE c = 255`.

    See Section 8.2.1.14, “Constant-Folding Optimization”, for more information.

  - Beginning with MySQL 8.0.16, the semijoin optimizations used with `IN` subqueries can now be applied to `EXISTS` subqueries as well. In addition, the optimizer now decorrelates trivially-correlated equality predicates in the `WHERE` condition attached to the subquery, so that they can be treated similarly to expressions in `IN` subqueries; this applies to both `EXISTS` and `IN` subqueries.

    For more information, see Section 8.2.2.1, “Optimizing IN and EXISTS Subquery Predicates with Semijoin Transformations”.

  - As of MySQL 8.0.17, the server rewrites any incomplete SQL predicates (that is, predicates having the form `WHERE value`, in which `value` is a column name or constant expression and no comparison operator is used) internally as `WHERE value <> 0` during the contextualization

phase, so that the query resolver, query optimizer, and query executor need work only with complete predicates.

One visible effect of this change is that, for Boolean values, `EXPLAIN` output now shows `true` and `false`, rather than `1` and `0`.

Another effect of this change is that evaluation of a JSON value in an SQL boolean context performs an implicit comparison against JSON integer 0. Consider the table created and populated as shown here:

```
mysql> CREATE TABLE test (id INT, col JSON);

mysql> INSERT INTO test VALUES (1, '{"val":true}'), (2, '{"val":false}');
```

Previously, the server attempted to convert an extracted `true` or `false` value to an SQL boolean when comparing it in an SQL boolean context, as shown by the following query using `IS TRUE`:

```
mysql> SELECT id, col, col->"$.val" FROM test WHERE col->"$.val" IS TRUE;
+------+--------------+--------------+
| id   | col          | col->"$.val" |
+------+--------------+--------------+
|    1 | {"val": true} | true         |
+------+--------------+--------------+
```

In MySQL 8.0.17 and later, the implicit comparison of the extracted value with JSON integer 0 leads to a different result:

```
mysql> SELECT id, col, col->"$.val" FROM test WHERE col->"$.val" IS TRUE;
+------+---------------+--------------+
| id   | col           | col->"$.val" |
+------+---------------+--------------+
|    1 | {"val": true}  | true         |
|    2 | {"val": false} | false        |
+------+---------------+--------------+
```

Beginning with MySQL 8.0.21, you can use `JSON_VALUE()` on the extracted value to perform type conversion prior to performing the test, as shown here:

```
mysql> SELECT id, col, col->"$.val" FROM test
    ->     WHERE JSON_VALUE(col, "$.val" RETURNING UNSIGNED) IS TRUE;
+------+--------------+--------------+
| id   | col          | col->"$.val" |
+------+--------------+--------------+
|    1 | {"val": true} | true         |
+------+--------------+--------------+
```

Also beginning with MySQL 8.0.21, the server provides the warning `Evaluating a JSON value in SQL boolean context does an implicit comparison against JSON integer 0; if this is not what you want, consider converting JSON to an SQL numeric type with JSON_VALUE RETURNING` when comparing extracted values in an SQL boolean context in this manner.

- In MySQL 8.0.17 and later a `WHERE` condition having `NOT IN (subquery)` or `NOT EXISTS (subquery)` is transformed internally into an antijoin. (An antijoin returns all rows from the table for which there is no row in the table to which it is joined matching the join condition.) This removes the subquery which can result in faster query execution since the subquery's tables are now handled on the top level.

  This is similar to, and reuses, the existing `IS NULL` (`Not exists`) optimization for outer joins; see EXPLAIN Extra Information.

- Beginning with MySQL 8.0.21, a single-table `UPDATE` or `DELETE` statement can now in many cases make use of a semijoin transformation or subquery materialization. This applies to statements of the forms shown here:

  - `UPDATE t1 SET t1.a=value WHERE t1.a IN (SELECT t2.a FROM t2)`
  - `DELETE FROM t1 WHERE t1.a IN (SELECT t2.a FROM t2)`

  This can be done for a single-table `UPDATE` or `DELETE` meeting the following conditions:

  - The `UPDATE` or `DELETE` statement uses a subquery having a `[NOT] IN` or `[NOT] EXISTS` predicate.
  - The statement has no `ORDER BY` clause, and has no `LIMIT` clause.

    (The multi-table versions of `UPDATE` and `DELETE` do not support `ORDER BY` or `LIMIT`.)
  - The target table does not support read-before-write removal (relevant only for `NDB` tables).
  - Semijoin or subquery materialization is allowed, based on any hints contained in the subquery and the value of `optimizer_switch`.

  When the semijoin optimization is used for an eligible single-table `DELETE` or `UPDATE`, this is visible in the optimizer trace: for a multi-table statement there is a `join_optimization` object in the trace, while there is none for a single-table statement. The conversion is also visible in the output of `EXPLAIN FORMAT=TREE` or `EXPLAIN ANALYZE`; a single-table statement shows `<not executable by iterator executor>`, while a multi-table statement reports a full plan.

  Also beginning with MySQL 8.0.21, semi-consistent reads are supported by multi-table `UPDATE` statements using `InnoDB` tables, for transaction isolation levels weaker than `REPEATABLE READ`.

- **Improved hash join performance.** MySQL 8.0.23 reimplements the hash table used for hash joins, resulting in several improvements in hash join performance. This work includes a fix for an issue (Bug #31516149, Bug #99933) whereby only roughly 2/3 of the memory allocated for the join buffer (`join_buffer_size`) could actually be used by a hash join.

  The new hash table is generally faster than the old one, and uses less memory for alignment, keys/values, and in scenarios where there are many equal keys. In addition, the server can now free old memory when the size of the hash table increases.

- **Common table expressions.** MySQL now supports common table expressions, both nonrecursive and recursive. Common table expressions enable use of named temporary result sets, implemented by permitting a `WITH` clause preceding `SELECT` statements and certain other statements. For more information, see Section 13.2.20, “WITH (Common Table Expressions)”.

  As of MySQL 8.0.19, the recursive `SELECT` part of a recursive common table expression (CTE) supports a `LIMIT` clause. `LIMIT` with `OFFSET` is also supported. See Recursive Common Table Expressions, for more information.

- **Window functions.** MySQL now supports window functions that, for each row from a query, perform a calculation using rows related to that row. These include functions such as `RANK()`, `LAG()`, and `NTILE()`. In addition, several existing aggregate functions now can be used as window functions (for example, `SUM()` and `AVG()`). For more information, see Section 12.21, “Window Functions”.

- **Lateral derived tables.** A derived table now may be preceded by the `LATERAL` keyword to specify that it is permitted to refer to (depend on) columns of preceding tables in the same `FROM` clause. Lateral derived tables make possible certain SQL operations that cannot be done with nonlateral derived tables or that require less-efficient workarounds. See Section 13.2.15.9, “Lateral Derived Tables”.

- **Aliases in single-table DELETE statements.** In MySQL 8.0.16 and later, single-table `DELETE` statements support the use of table aliases.

- **Regular expression support.** Previously, MySQL used the Henry Spencer regular expression library to support regular expression operators (`REGEXP`, `RLIKE`). Regular expression support has been reimplemented using International Components for Unicode (ICU), which provides full Unicode support and is multibyte safe. The `REGEXP_LIKE()` function performs regular expression matching in the manner of the `REGEXP` and `RLIKE` operators, which now are synonyms for that function. In addition, the `REGEXP_INSTR()`, `REGEXP_REPLACE()`, and `REGEXP_SUBSTR()` functions are available to find match positions and perform substring substitution and extraction, respectively. The `regexp_stack_limit` and `regexp_time_limit` system variables provide control over resource consumption by the match engine. For more information, see Section 12.8.2, “Regular Expressions”. For information about ways in which applications that use regular expressions may be affected by the implementation change, see Regular Expression Compatibility Considerations.

- **Internal temporary tables.** The `TempTable` storage engine replaces the `MEMORY` storage engine as the default engine for in-memory internal temporary tables. The `TempTable` storage engine provides efficient storage for `VARCHAR` and `VARBINARY` columns. The `internal_tmp_mem_storage_engine` session variable defines the storage engine for in-memory internal temporary tables. Permitted values are `TempTable` (the default) and `MEMORY`. The `temptable_max_ram` variable defines the maximum amount of memory that the `TempTable` storage engine can use before data is stored to disk.

- **Logging.** These enhancements were added to improve logging:

  - Error logging was rewritten to use the MySQL component architecture. Traditional error logging is implemented using built-in components, and logging using the system log is implemented as a loadable component. In addition, a loadable JSON log writer is available. For more information, see Section 5.4.2, “The Error Log”.

  - From MySQL 8.0.30, error log components can be loaded implicitly at startup before the `InnoDB` storage engine is available. This new method of loading error log components loads and enables the components defined by the `log_error_services` variable.

    Previously, error log components had to be installed first using `INSTALL COMPONENT` and could only be loaded after `InnoDB` was fully available, as the list of components to load was read from the `mysql.components` table, which is an `InnoDB` table.

    Implicit loading of error log components has these advantages:

    - Log components are loaded earlier in the startup sequence, making logged information available sooner.

    - It helps avoid loss of buffered log information should a failure occur during startup.

    - Loading log components using `INSTALL COMPONENT` is not required, simplifying error log configuration.

    The explicit method of loading log components using `INSTALL COMPONENT` remains supported for backward compatibility.

    For more information, see Section 5.4.2.1, “Error Log Configuration”.

- **Backup lock.** A new type of backup lock permits DML during an online backup while preventing operations that could result in an inconsistent snapshot. The new backup lock is supported by `LOCK INSTANCE FOR BACKUP` and `UNLOCK INSTANCE` syntax. The `BACKUP_ADMIN` privilege is required to use these statements.

- **Replication.** The following enhancements have been made to MySQL Replication:

  - MySQL Replication now supports binary logging of partial updates to JSON documents using a compact binary format, saving space in the log over logging complete JSON documents. Such compact logging is done automatically when statement-based logging is in use, and can be enabled by setting the new `binlog_row_value_options` system variable to `PARTIAL_JSON`. For more information, see Partial Updates of JSON Values, as well as the description of `binlog_row_value_options`.

- **Connection management.** MySQL Server now permits a TCP/IP port to be configured specifically for administrative connections. This provides an alternative to the single administrative connection that is permitted on the network interfaces used for ordinary connections even when `max_connections` connections are already established. See Section 5.1.12.1, “Connection Interfaces”.

  MySQL now provides more control over the use of compression to minimize the number of bytes sent over connections to the server. Previously, a given connection was either uncompressed or used the `zlib` compression algorithm. Now, it is also possible to use the `zstd` algorithm, and to select a compression level for `zstd` connections. The permitted compression algorithms can be configured on the server side, as well as on the connection-origination side for connections by client programs and by servers participating in source/replica replication or Group Replication. For more information, see Section 4.2.8, “Connection Compression Control”.

- **Configuration.** The maximum permitted length of host names throughout MySQL has been raised to 255 ASCII characters, up from the previous limit of 60 characters. This applies to, for example, host name-related columns in the data dictionary, `mysql` system schema, Performance Schema, `INFORMATION_SCHEMA`, and `sys` schema; the `MASTER_HOST` value for the `CHANGE MASTER TO` statement; the `Host` column in `SHOW PROCESSLIST` statement output; host names in account names (such as used in account-management statements and in `DEFINER` attributes); and host name-related command options and system variables.

  Caveats:

  - The increase in permitted host name length can affect tables with indexes on host name columns. For example, tables in the `mysql` system schema that index host names now have an explicit `ROW_FORMAT` attribute of `DYNAMIC` to accommodate longer index values.

  - Some file name-valued configuration settings might be constructed based on the server host name. The permitted values are constrained by the underlying operating system, which may not permit file names long enough to include 255-character host names. This affects the `general_log_file`, `log_error`, `pid_file`, `relay_log`, and `slow_query_log_file` system variables and corresponding options. If host name-based values are too long for the OS, explicit shorter values must be provided.

  - Although the server now supports 255-character host names, connections to the server established using the `--ssl-mode=VERIFY_IDENTITY` option are constrained by maximum host name length supported by OpenSSL. Host name matches pertain to two fields of SSL certificates, which have maximum lengths as follows: Common Name: maximum length 64; Subject Alternative Name: maximum length as per RFC#1034.

- **Plugins.** Previously, MySQL plugins could be written in C or C++. MySQL header files used by plugins now contain C++ code, which means that plugins must be written in C++, not C.

- **C API.** The MySQL C API now supports asynchronous functions for nonblocking communication with the MySQL server. Each function is the asynchronous counterpart to an existing synchronous function. The synchronous functions block if reads from or writes to the server connection must wait. The asynchronous functions enable an application to check whether work on the server connection is ready to proceed. If not, the application can perform other work before checking again later. See C API Asynchronous Interface.

- **Additional target types for casts.** The functions `CAST()` and `CONVERT()` now support conversions to types `DOUBLE`, `FLOAT`, and `REAL`. Added in MySQL 8.0.17. See Section 12.11, "Cast Functions and Operators".

- **JSON schema validation.** MySQL 8.0.17 adds two functions `JSON_SCHEMA_VALID()` and `JSON_SCHEMA_VALIDATION_REPORT()` for validating JSON documents again JSON schemas. `JSON_SCHEMA_VALID()` returns TRUE (1) if the document validates against the schema and FALSE (0) if it does not. `JSON_SCHEMA_VALIDATION_REPORT()` returns a JSON document containing detailed information about the results of the validation. The following statements apply to both of these functions:

  - The schema must conform to Draft 4 of the JSON Schema specification.
  - `required` attributes are supported.
  - External resources and the `$ref` keyword are not supported.
  - Regular expression patterns are supported; invalid patterns are silently ignored.

  See Section 12.18.7, "JSON Schema Validation Functions", for more information and examples.

- **Multi-valued indexes.** Beginning with MySQL 8.0.17, `InnoDB` supports the creation of a multi-valued index, which is a secondary index defined on a `JSON` column that stores an array of values and which can have multiple index records for a single data record. Such an index uses a key part definition such as `CAST(data->'$.zipcode' AS UNSIGNED ARRAY)`. A multi-valued index is used automatically by the MySQL optimizer for suitable queries, as can be viewed in the output of `EXPLAIN`.

  As part of this work, MySQL adds a new function `JSON_OVERLAPS()` and a new `MEMBER OF()` operator for working with `JSON` documents, additionally extending the `CAST()` function with a new `ARRAY` keyword, as described in the following list:

  - `JSON_OVERLAPS()` compares two `JSON` documents. If they contain any key-value pairs or array elements in common, the function returns TRUE (1); otherwise it returns FALSE (0). If both values are scalars, the function performs a simple test for equality. If one argument is a JSON array and the other is a scalar, the scalar is treated as an array element. Thus, `JSON_OVERLAPS()` acts as a complement to `JSON_CONTAINS()`.
  - `MEMBER OF()` tests whether the first operand (a scalar or JSON document) is a member of the JSON array passed as the second operand, returning TRUE (1) if it is, and FALSE (0) if it is not. No type conversion of the operand is performed.
  - `CAST(expression AS type ARRAY)` permits creation of a functional index by casting the JSON array found in a JSON document at *`json_path`* to an SQL array. Type specifiers are limited to those already supported by `CAST()`, with the exception of `BINARY` (not supported). This usage of `CAST()` (and the `ARRAY` keyword) is supported only by `InnoDB`, and only for the creation of a multi-valued index.

  For detailed information about multi-valued indexes, including examples, see Multi-Valued Indexes. Section 12.18.3, "Functions That Search JSON Values", provides information about `JSON_OVERLAPS()` and `MEMBER OF()`, along with examples of use.

- **Hintable time_zone.** As of MySQL 8.0.17, the `time_zone` session variable is hintable using `SET_VAR`.

- **Redo Log Archiving.** As of MySQL 8.0.17, `InnoDB` supports redo log archiving. Backup utilities that copy redo log records may sometimes fail to keep pace with redo log generation while a backup operation is in progress, resulting in lost redo log records due to those records being overwritten. The redo log archiving feature addresses this issue by sequentially writing redo log records to an archive file. Backup utilities can copy redo log records from the archive file as necessary, thereby avoiding the potential loss of data. For more information, see Redo Log Archiving.

- **The Clone Plugin.** As of MySQL 8.0.17, MySQL provides a clone plugin that permits cloning `InnoDB` data locally or from a remote MySQL server instance. A local cloning operation stores cloned data on the same server or node where the MySQL instance runs. A remote cloning operation transfers cloned data over the network from a donor MySQL server instance to the recipient server or node where the cloning operation was initiated.

  The clone plugin supports replication. In addition to cloning data, a cloning operation extracts and transfers replication coordinates from the donor and applies them on the recipient, which enables using the clone plugin for provisioning Group Replication members and replicas. Using the clone plugin for provisioning is considerably faster and more efficient than replicating a large number of transactions. Group Replication members can also be configured to use the clone plugin as an alternative method of recovery, so that members automatically choose the most efficient way to retrieve group data from seed members.

  For more information, see Section 5.6.7, “The Clone Plugin”, and Section 18.5.4.2, “Cloning for Distributed Recovery”.

  As of MySQL 8.0.27, concurrent DDL operations on the donor MySQL Server instance are permitted while a cloning operation is in progress. Previously, a backup lock was held during the cloning operation, preventing concurrent DDL on the donor. To revert to the previous behavior of blocking concurrent DDL on the donor during a clone operation, enable the `clone_block_ddl` variable. See Section 5.6.7.4, “Cloning and Concurrent DDL”.

  As of MySQL 8.0.29, the `clone_delay_after_data_drop` variable permits specifying a delay period immediately after removing existing data on the recipient MySQL Server instance at the start of a remote cloning operation. The delay is intended to provide enough time for the file system on the recipient host to free space before data is cloned from the donor MySQL Server instance. Certain file systems free space asynchronously in a background process. On these file systems, cloning data too soon after dropping existing data can result in clone operation failures due to insufficient space. The maximum delay period is 3600 seconds (1 hour). The default setting is 0 (no delay).

- **Hash Join Optimization.** Beginning with MySQL 8.0.18, a hash join is used whenever each pair of tables in a join includes at least one equi-join condition, and no indexes apply to any join condition. A hash join does not require indexes, although it can be used with indexes applying to single-table predicates only. A hash join is more efficient in most cases than the block-nested loop algorithm. Joins such as those shown here can be optimized in this manner:

  ```
  SELECT *
      FROM t1
      JOIN t2
          ON t1.c1=t2.c1;

  SELECT *
      FROM t1
      JOIN t2
          ON (t1.c1 = t2.c1 AND t1.c2 < t2.c2)
      JOIN t3
          ON (t2.c1 = t3.c1)
  ```

  Hash joins can also be used for Cartesian products—that is, when no join condition is specified.

  You can see when the hash join optimization is being used for a particular query using `EXPLAIN FORMAT=TREE` or `EXPLAIN ANALYZE`. (In MySQL 8.0.20 and later, you can also use `EXPLAIN`, omitting `FORMAT=TREE`.)

  The amount of memory available to a hash join is limited by the value of `join_buffer_size`. A hash join that requires more than this much memory is executed on disk; the number of disk files that can be used by an on-disk hash join is limited by `open_files_limit`.

  As of MySQL 8.0.19, the `hash_join` optimizer switch which was introduced in MySQL 8.0.18 no longer supported (hash_join=on still appears as part of the value of optimizer_switch, but setting it no longer has any effect). The `HASH_JOIN` and `NO_HASH_JOIN` optimizer hints are also no longer

supported. The switch and the hint are both now deprecated; expect them to be removed in a future MySQL release. In MySQL 8.0.18 and later, hash joins can be disabled using the `NO_BNL` optimizer switch.

In MySQL 8.0.20 and later, block nested loop is no longer used in the MySQL server, and a hash join is employed any time a block nested loop would have been used previously, even when the query contains no equi-join conditions. This applies to inner non-equijoins, semijoins, antijoins, left outer joins, and right outer joins. The `block_nested_loop` flag for the `optimizer_switch` system variable as well as the `BNL` and `NO_BNL` optimizer hints are still supported, but henceforth control use of hash joins only. In addition, both inner and outer joins (including semijoins and antijoins) can now employ batched key access (BKA), which allocates join buffer memory incrementally so that individual queries need not use up large amounts of resources that they do not actually require for resolution. BKA for inner joins only is supported starting with MySQL 8.0.18.

MySQL 8.0.20 also replaces the executor used in previous versions of MySQL with the iterator executor. This work includes replacement of the old index subquery engines that governed queries of the form `WHERE value IN (SELECT column FROM table WHERE ...)` for those `IN` queries which have not been optimized as semijoins, as well as queries materialized in the same form, which formerly depended on the old executor.

For more information and examples, see Section 8.2.1.4, “Hash Join Optimization”. See also Batched Key Access Joins.

- **EXPLAIN ANALYZE Statement.** A new form of the `EXPLAIN` statement, `EXPLAIN ANALYZE`, is implemented in MySQL 8.0.18, providing expanded information about the execution of `SELECT` statements in `TREE` format for each iterator used in processing the query, and making it possible to compare estimated cost with the actual cost of the query. This information includes startup cost, total cost, number of rows returned by this iterator, and the number of loops executed.

  In MySQL 8.0.21 and later, this statement also supports a `FORMAT=TREE` specifier. `TREE` is the only supported format.

  See Obtaining Information with EXPLAIN ANALYZE, for more information.

- **Query cast injection.** In version 8.0.18 and later, MySQL injects cast operations into the query item tree inside expressions and conditions in which the data type of the argument and the expected data type do not match. This has no effect on query results or speed of execution, but makes the query as executed equivalent to one which is compliant with the SQL standard while maintaining backwards compatibility with previous releases of MySQL.

  Such implicit casts are now performed between temporal types (`DATE`, `DATETIME`, `TIMESTAMP`, `TIME`) and numeric types (`SMALLINT`, `TINYINT`, `MEDIUMINT`, `INT`/`INTEGER`, `BIGINT`; `DECIMAL`/`NUMERIC`; `FLOAT`, `DOUBLE`, `REAL`; `BIT`) whenever they are compared using any of the standard numeric comparison operators (`=`, `>=`, `>`, `<`, `<=`, `<>`/`!=`, or `<=>`). In this case, any value that is not already a `DOUBLE` is cast as one. Cast injection is also now performed for comparisons between `DATE` or `TIME` values and `DATETIME` values, where the arguments are cast whenever necessary as `DATETIME`.

  Beginning with MySQL 8.0.21, such casts are also performed when comparing string types with other types. String types that are cast include `CHAR`, `VARCHAR`, `BINARY`, `VARBINARY`, `BLOB`, `TEXT`, `ENUM`, and `SET`. When comparing a value of a string type with a numeric type or `YEAR`, the string cast is to `DOUBLE`; if the type of the other argument is not `FLOAT`, `DOUBLE`, or `REAL`, it is also cast to `DOUBLE`. When comparing a string type to a `DATETIME` or `TIMESTAMP` value, the string is cast is to `DATETIME`; when comparing a string type with `DATE`, the string is cast to `DATE`.

  It is possible to see when casts are injected into a given query by viewing the output of `EXPLAIN ANALYZE`, `EXPLAIN FORMAT=JSON`, or, as shown here, `EXPLAIN FORMAT=TREE`:

```
mysql> CREATE TABLE d (dt DATETIME, d DATE, t TIME);
Query OK, 0 rows affected (0.62 sec)
```

```
mysql> CREATE TABLE n (i INT, d DECIMAL, f FLOAT, dc DECIMAL);
Query OK, 0 rows affected (0.51 sec)

mysql> CREATE TABLE s (c CHAR(25), vc VARCHAR(25),
    ->     bn BINARY(50), vb VARBINARY(50), b BLOB, t TEXT,
    ->     e ENUM('a', 'b', 'c'), se SET('x' ,'y', 'z'));
Query OK, 0 rows affected (0.50 sec)

mysql> EXPLAIN FORMAT=TREE SELECT * from d JOIN n ON d.dt = n.i\G
*************************** 1. row ***************************
EXPLAIN: -> Inner hash join (cast(d.dt as double) = cast(n.i as double))
(cost=0.70 rows=1)
    -> Table scan on n  (cost=0.35 rows=1)
    -> Hash
        -> Table scan on d  (cost=0.35 rows=1)

mysql> EXPLAIN FORMAT=TREE SELECT * from s JOIN d ON d.dt = s.c\G
*************************** 1. row ***************************
EXPLAIN: -> Inner hash join (d.dt = cast(s.c as datetime(6)))  (cost=0.72 rows=1)
    -> Table scan on d  (cost=0.37 rows=1)
    -> Hash
        -> Table scan on s  (cost=0.35 rows=1)

1 row in set (0.01 sec)

mysql> EXPLAIN FORMAT=TREE SELECT * from n JOIN s ON n.d = s.c\G
*************************** 1. row ***************************
EXPLAIN: -> Inner hash join (cast(n.d as double) = cast(s.c as double))  (cost=0.70 rows=1)
    -> Table scan on s  (cost=0.35 rows=1)
    -> Hash
        -> Table scan on n  (cost=0.35 rows=1)

1 row in set (0.00 sec)
```

Such casts can also be seen by executing `EXPLAIN [FORMAT=TRADITIONAL]`, in which case it is also necessary to issue `SHOW WARNINGS` after executing the `EXPLAIN` statement.

- **Time zone support for TIMESTAMP and DATETIME.** As of MySQL 8.0.19, the server accepts a time zone offset with inserted datetime (`TIMESTAMP` and `DATETIME`) values. This offset uses the same format as that employed when setting the `time_zone` system variable, except that a leading zero is required when the hours portion of the offset is less than 10, and `'-00:00'` is not allowed. Examples of datetime literals that include time zone offsets are `'2019-12-11 10:40:30-05:00'`, `'2003-04-14 03:30:00+10:00'`, and `'2020-01-01 15:35:45+05:30'`.

  Time zone offsets are not displayed when selecting datetime values.

  Datetime literals incorporating time zone offsets can be used as prepared statement parameter values.

  As part of this work, the value used to set the `time_zone` system variable is now also restricted to the range `-13:59` to `+14:00`, inclusive. (It remains possible to assign name values to `time_zone` such as `'EST'`, `'Posix/Australia/Brisbane'`, and `'Europe/Stockholm'` to this variable, provided that the MySQL time zone tables are loaded; see Populating the Time Zone Tables).

  For more information and examples, see Section 5.1.15, “MySQL Server Time Zone Support”, as well as Section 11.2.2, “The DATE, DATETIME, and TIMESTAMP Types”.

- **Precise information for JSON schema CHECK constraint failures.** When using `JSON_SCHEMA_VALID()` to specify a `CHECK` constraint, MySQL 8.0.19 and later provides precise information about the reasons for failures of such constraints.

  For examples and more information, see JSON_SCHEMA_VALID() and CHECK constraints. See also Section 13.1.20.6, “CHECK Constraints”.

- **Row and column aliases with ON DUPLICATE KEY UPDATE.** Beginning with MySQL 8.0.19, it is possible to reference the row to be inserted, and, optionally, its columns, using aliases. Consider the following `INSERT` statement on a table `t` having columns `a` and `b`:

  ```
  INSERT INTO t SET a=9,b=5
      ON DUPLICATE KEY UPDATE a=VALUES(a)+VALUES(b);
  ```

  Using the alias `new` for the new row, and, in some cases, the aliases `m` and `n` for this row's columns, the `INSERT` statement can be rewritten in many different ways, some examples of which are shown here:

  ```
  INSERT INTO t SET a=9,b=5 AS new
      ON DUPLICATE KEY UPDATE a=new.a+new.b;

  INSERT INTO t VALUES(9,5) AS new
      ON DUPLICATE KEY UPDATE a=new.a+new.b;

  INSERT INTO t SET a=9,b=5 AS new(m,n)
      ON DUPLICATE KEY UPDATE a=m+n;

  INSERT INTO t VALUES(9,5) AS new(m,n)
      ON DUPLICATE KEY UPDATE a=m+n;
  ```

  For more information and examples, see Section 13.2.7.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”.

- **SQL standard explicit table clause and table value constructor.** Added table value constructors and explicit table clauses according to the SQL standard. These are implemented in MySQL 8.0.19, respectively, as the `TABLE` statement and the `VALUES` statement.

  The `TABLE` statement has the format `TABLE` *`table_name`*, and is equivalent to `SELECT * FROM` *`table_name`*. It supports `ORDER BY` and `LIMIT` clauses ( the latter with optional `OFFSET`), but does not allow for the selection of individual table columns. `TABLE` can be used anywhere that you would employ the equivalent `SELECT` statement; this includes joins, unions, `INSERT ... SELECT`, `REPLACE`, `CREATE TABLE ... SELECT` statements, and subqueries. For example:

  - `TABLE t1 UNION TABLE t2` is equivalent to `SELECT * FROM t1 UNION SELECT * FROM t2`
  - `CREATE TABLE t2 TABLE t1` is equivalent to `CREATE TABLE t2 SELECT * FROM t1`
  - `SELECT a FROM t1 WHERE b > ANY (TABLE t2)` is equivalent to `SELECT a FROM t1 WHERE b > ANY (SELECT * FROM t2)`.

  `VALUES` can be used to supply a table value to an `INSERT`, `REPLACE`, or `SELECT` statement, and consists of the `VALUES` keyword followed by a series of row constructors (`ROW()`) separated by commas. For example, the statement `INSERT INTO t1 VALUES ROW(1,2,3), ROW(4,5,6), ROW(7,8,9)` provides an SQL-compliant equivalent to the MySQL-specific `INSERT INTO t1 VALUES (1,2,3), (4,5,6), (7,8,9)`. You can also select from a `VALUES` table value

constructor just as you would a table, bearing in mind that you must supply a table alias when doing so, and use this `SELECT` just as you would any other; this includes joins, unions, and subqueries.

For more information about `TABLE` and `VALUES`, and for examples of their use, see the following sections of this documentation:

- Section 13.2.16, “TABLE Statement”
- Section 13.2.19, “VALUES Statement”
- Section 13.1.20.4, “CREATE TABLE ... SELECT Statement”
- Section 13.2.7.1, “INSERT ... SELECT Statement”
- Section 13.2.13.2, “JOIN Clause”
- Section 13.2.15, “Subqueries”
- Section 13.2.18, “UNION Clause”

- **Optimizer hints for FORCE INDEX, IGNORE INDEX.** MySQL 8.0 introduces index-level optimizer hints which serve as analogs to the traditional index hints as described in Section 8.9.4, “Index Hints”. The new hints are listed here, along with their `FORCE INDEX` or `IGNORE INDEX` equivalents:

  - `GROUP_INDEX`: Equivalent to `FORCE INDEX FOR GROUP BY`

    `NO_GROUP_INDEX`: Equivalent to `IGNORE INDEX FOR GROUP BY`

  - `JOIN_INDEX`: Equivalent to `FORCE INDEX FOR JOIN`

    `NO_JOIN_INDEX`: Equivalent to `IGNORE INDEX FOR JOIN`

  - `ORDER_INDEX`: Equivalent to `FORCE INDEX FOR ORDER BY`

    `NO_ORDER_INDEX`: Equivalent to `IGNORE INDEX FOR ORDER BY`

  - `INDEX`: Same as `GROUP_INDEX` plus `JOIN_INDEX` plus `ORDER_INDEX`; equivalent to `FORCE INDEX` with no modifier

    `NO_INDEX`: Same as `NO_GROUP_INDEX` plus `NO_JOIN_INDEX` plus `NO_ORDER_INDEX`; equivalent to `IGNORE INDEX` with no modifier

  For example, the following two queries are equivalent:

```
SELECT a FROM t1 FORCE INDEX (i_a) FOR JOIN WHERE a=1 AND b=2;

SELECT /*+ JOIN_INDEX(t1 i_a) */ a FROM t1 WHERE a=1 AND b=2;
```

  The optimizer hints listed previously follow the same basic rules for syntax and usage as existing index-level optimizer hints.

  These optimizer hints are intended to replace `FORCE INDEX` and `IGNORE INDEX`, which we plan to deprecate in a future MySQL release, and subsequently to remove from MySQL. They do not implement a single exact equivalent for `USE INDEX`; instead, you can employ one or more of `NO_INDEX`, `NO_JOIN_INDEX`, `NO_GROUP_INDEX`, or `NO_ORDER_INDEX` to achieve the same effect.

  For further information and examples of use, see Index-Level Optimizer Hints.

- **JSON_VALUE() function.** MySQL 8.0.21 implements a new function `JSON_VALUE()` intended to simplify indexing of `JSON` columns. In its most basic form, it takes as arguments a JSON document and a JSON path pointing to a single value in that document, as well as (optionally) allowing you to

specify a return type with the `RETURNING` keyword. `JSON_VALUE(json_doc, path RETURNING type)` is equivalent to this:

```
CAST(
    JSON_UNQUOTE( JSON_EXTRACT(json_doc, path) )
    AS type
);
```

You can also specify `ON EMPTY`, `ON ERROR`, or both clauses, similar to those employed with `JSON_TABLE()`.

You can use `JSON_VALUE()` to create an index on an expression on a `JSON` column like this:

```
CREATE TABLE t1(
    j JSON,
    INDEX i1 ( (JSON_VALUE(j, '$.id' RETURNING UNSIGNED)) )
);

INSERT INTO t1 VALUES ROW('{"id": "123", "name": "shoes", "price": "49.95"}');
```

A query using this expression, such as that shown here, can make use of the index:

```
SELECT name, price FROM t1
    WHERE JSON_VALUE(j, '$.id' RETURNING UNSIGNED) = 123;
```

In many cases, this is simpler than creating a generated column from the `JSON` column and then creating an index on the generated column.

For more information and examples, see the description of `JSON_VALUE()`.

- **User comments and user attributes.** MySQL 8.0.21 introduces the ability to set user comments and user attributes when creating or updating user accounts. A user comment consists of arbitrary text passed as the argument to a `COMMENT` clause used with a `CREATE USER` or `ALTER USER` statement. A user attribute consists of data in the form of a JSON object passed as the argument to an `ATTRIBUTE` clause used with either of these two statements. The attribute can contain any valid key-value pairs in JSON object notation. Only one of `COMMENT` or `ATTRIBUTE` can be used in a single `CREATE USER` or `ALTER USER` statement.

  User comments and user attributes are stored together internally as a JSON object, the comment text as the value of an element having `comment` as its key. This information can be retrieved from the `ATTRIBUTE` column of the Information Schema `USER_ATTRIBUTES` table; since it is in JSON format, you can use MySQL's JSON function and operators to parse its contents (see Section 12.18, “JSON Functions”). Successive changes to the user attribute are merged with its current value as when using the `JSON_MERGE_PATCH()` function.

  Example:

```
mysql> CREATE USER 'mary'@'localhost' COMMENT 'This is Mary Smith\'s account';
Query OK, 0 rows affected (0.33 sec)

mysql> ALTER USER 'mary'@'localhost'
    -≫     ATTRIBUTE '{"fname":"Mary", "lname":"Smith"}';
Query OK, 0 rows affected (0.14 sec)

mysql> ALTER USER 'mary'@'localhost'
    -≫     ATTRIBUTE '{"email":"mary.smith@example.com"}';
Query OK, 0 rows affected (0.12 sec)

mysql> SELECT
    ->    USER,
    ->    HOST,
    ->    ATTRIBUTE->>"$.fname" AS 'First Name',
    ->    ATTRIBUTE->>"$.lname" AS 'Last Name',
    ->    ATTRIBUTE->>"$.email" AS 'Email',
    ->    ATTRIBUTE->>"$.comment" AS 'Comment'
    -> FROM INFORMATION_SCHEMA.USER_ATTRIBUTES
```

```
    -> WHERE USER='mary' AND HOST='localhost'\G
*************************** 1. row ***************************
      USER: mary
      HOST: localhost
First Name: Mary
 Last Name: Smith
     Email: mary.smith@example.com
   Comment: This is Mary Smith's account
1 row in set (0.00 sec)
```

For more information and examples, see Section 13.7.1.3, “CREATE USER Statement”, Section 13.7.1.1, “ALTER USER Statement”, and Section 26.3.46, “The INFORMATION_SCHEMA USER_ATTRIBUTES Table”.

- **New optimizer_switch flags.** MySQL 8.0.21 adds two new flags for the `optimizer_switch` system variable, as described in the following list:

  - `prefer_ordering_index` flag

    By default, MySQL attempts to use an ordered index for any `ORDER BY` or `GROUP BY` query that has a `LIMIT` clause, whenever the optimizer determines that this would result in faster execution. Because it is possible in some cases that choosing a different optimization for such queries actually performs better, it is now possible to disable this optimization by setting the `prefer_ordering_index` flag to `off`.

    The default value for this flag is `on`.

  - `subquery_to_derived` flag

    When this flag is set to `on`, the optimizer transforms eligible scalar subqueries into joins on derived tables. For example, the query `SELECT * FROM t1 WHERE t1.a > (SELECT COUNT(a) FROM t2)` is rewritten as `SELECT t1.a FROM t1 JOIN ( SELECT COUNT(t2.a) AS c FROM t2 ) AS d WHERE t1.a > d.c`.

    This optimization can be applied to a subquery which is part of a `SELECT`, `WHERE`, `JOIN`, or `HAVING` clause; contains one or more aggregate functions but no `GROUP BY` clause; is not correlated; and does not use any nondeterministic functions.

    The optimization can also be applied to a table subquery which is the argument to `IN`, `NOT IN`, `EXISTS`, or `NOT EXISTS`, and which does not contain a `GROUP BY`. For example, the query `SELECT * FROM t1 WHERE t1.b < 0 OR t1.a IN (SELECT t2.a + 1 FROM t2)` is rewritten as `SELECT a, b FROM t1 LEFT JOIN (SELECT DISTINCT 1 AS e1, t2.a AS e2 FROM t2) d ON t1.a + 1 = d.e2 WHERE t1.b < 0 OR d.e1 IS NOT NULL`.

    Starting with MySQL 8.0.24, this optimization can also be applied to a correlated scalar subquery by applying an extra grouping to it, and then an outer join on the lifted predicate. For example, a query such as `SELECT * FROM t1 WHERE (SELECT a FROM t2 WHERE t2.a=t1.a) > 0` can be rewritten as `SELECT t1.* FROM t1 LEFT OUTER JOIN (SELECT a, COUNT(*) AS ct FROM t2 GROUP BY a) AS derived ON t1.a = derived.a WHERE derived.a > 0`. MySQL performs a cardinality check to make sure that the subquery does not return more than one row (`ER_SUBQUERY_NO_1_ROW`). See Section 13.2.15.7, “Correlated Subqueries”, for more information.

    This optimization is normally disabled, since it does not yield a noticeable performance benefit in most cases; the flag is set to `off` by default.

  For more information, see Section 8.9.2, “Switchable Optimizations”. See also Section 8.2.1.19, “LIMIT Query Optimization”, Section 8.2.2.1, “Optimizing IN and EXISTS Subquery Predicates with Semijoin Transformations”, and Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”.

- **XML enhancements.** As of MySQL 8.0.21, the `LOAD XML` statement now supports `CDATA` sections in the XML to be imported.

- **Casting to the YEAR type now supported.** Beginning with MySQL 8.0.22, the server allows casting to `YEAR`. Both the `CAST()` and `CONVERT()` functions support single-digit, two-digit, and four-digit `YEAR` values. For one-digit and two-digit values, the allowed range is 0-99. Four-digit values must be in the range 1901-2155. `YEAR` can also be used as the return type for the `JSON_VALUE()` function; this function supports four-digit years only.

  String, time-and-date, and floating-point values can all be cast to `YEAR`. Casting of `GEOMETRY` values to `YEAR` is not supported.

  For more information, including conversion rules, see the description of the `CONVERT()` function.

- **Retrieval of TIMESTAMP values as UTC.** MySQL 8.0.22 and later supports conversion of a `TIMESTAMP` column value from the system time zone to a UTC `DATETIME` on retrieval, using `CAST(value AT TIME ZONE specifier AS DATETIME)`, where the specifier is one of `[INTERVAL] '+00:00'` or `'UTC'`. The precision of the `DATETIME` value returned by the cast can be specified up to 6 decimal places, if desired. The `ARRAY` keyword is not supported with this construct.

  `TIMESTAMP` values inserted into a table using a timezone offset are also supported. Use of `AT TIME ZONE` is not supported for `CONVERT()` or any other MySQL function or construct.

  For further information and examples, see the description of the `CAST()` function.

- **Dump file output synchronization.** MySQL 8.0.22 and later supports periodic synchronization when writing to files by `SELECT INTO DUMPFILE` and `SELECT INTO OUTFILE` statements. This can be enabled by setting the `select_into_disk_sync` system variable to `ON`; the size of the write buffer is determined by the value set for `select_into_buffer_size`; the default is 131072 ($2^{17}$) bytes.

  In addition, an optional delay following synchronization to disk can be set using `select_into_disk_sync_delay`; the default is no delay (0 milliseconds).

  For more information, see the descriptions of the variables referenced previously in this item.

- **Single preparation of statements.** As of MySQL 8.0.22, a prepared statement is prepared a single time, rather than once each time it is executed. This is done when executing `PREPARE`. This is also true for any statement inside a stored procedure; the statement is prepared once, when the stored procedure is first executed.

  One result of this change is that the fashion in which dynamic parameters used in prepared statements are resolved is also changed in the ways listed here:

  - A prepared statement parameter is assigned a data type when the statement is prepared; the type persists for each subsequent execution of the statement (unless the statement is reprepared; see following).

    Using a different data type for a given parameter or user variable within a prepared statement for executions of the statement subsequent to the first execution may cause the statement to be

reprepared; for this reason, it is advisable to use the same data type for a given parameter when re-executing a prepared statement.

- The following constructs employing window functions are no longer accepted, in order to align with the SQL standard:
  - `NTILE(NULL)`
  - `NTH_VALUE(expr, NULL)`
  - `LEAD(expr, nn)` and `LAG(expr, nn)`, where *nn* is a negative number

  This facilitates greater compliance with the SQL standard. See the individual function descriptions for further details.
- A user variable referenced within a prepared statement now has its data type determined when the statement is prepared; the type persists for each subsequent execution of the statement.
- A user variable referenced by a statement occurring within a stored procedure now has its data type determined the first time the statement is executed; the type persists for any subsequent invocation of the containing stored procedure.
- When executing a prepared statement of the form `SELECT expr1, expr2, ... FROM table ORDER BY ?`, passing an integer value *N* for the parameter no longer causes ordering of the results by the $N^{th}$ expression in the select list; the results are no longer ordered, as is expected with `ORDER BY constant`.

Preparing a statement used as a prepared statement or within a stored procedure only once enhances the performance of the statement, since it negates the added cost of repeated preparation. Doing so also avoids possible multiple rollbacks of preparation structures, which has been the source of numerous issues in MySQL.

For more information, see Section 13.5.1, “PREPARE Statement”.

- **RIGHT JOIN as LEFT JOIN handling.** As of MySQL 8.0.22, the server handles all instances of `RIGHT JOIN` internally as `LEFT JOIN`, eliminating a number of special cases in which a complete conversion was not performed at parse time.
- **Derived condition pushdown optimization.** MySQL 8.0.22 (and later) implements derived condition pushdown for queries having materialized derived tables. For a query such as `SELECT * FROM (SELECT i, j FROM t1) AS dt WHERE i > constant`, it is now possible in many cases to push the outer `WHERE` condition down to the derived table, in this case resulting in `SELECT * FROM (SELECT i, j FROM t1 WHERE i > constant) AS dt`.

  Previously, if the derived table was materialized and not merged, MySQL materialized the entire table, then qualified the rows with the `WHERE` condition. Moving the `WHERE` condition into the subquery using the derived condition pushdown optimization can often reduce the number of rows must be processed, which can decrease the time needed to execute the query.

  An outer `WHERE` condition can be pushed down directly to a materialized derived table when the derived table does not use any aggregate or window functions. When the derived table has a `GROUP BY` and does not use any window functions, the outer `WHERE` condition can be pushed down to the derived table as a `HAVING` condition. The `WHERE` condition can also be pushed down when the derived table uses a window function and the outer `WHERE` references columns used in the window function's `PARTITION` clause.

  Derived condition pushdown is enabled by default, as indicated by the `optimizer_switch` system variable's `derived_condition_pushdown` flag. The flag, added in MySQL 8.0.22, is set to `on` by default; to disable the optimization for a specific query, you can use the `NO_DERIVED_CONDITION_PUSHDOWN` optimizer hint (also added in MySQL 8.0.22). If the

optimization is disabled due to `derived_condition_pushdown` being set to `off`, you can enable it for a given query using `DERIVED_CONDITION_PUSHDOWN`.

The derived condition pushdown optimization cannot be employed for a derived table that contains a `LIMIT` clause. Prior to MySQL 8.0.29, the optimization also cannot be used when the query contains `UNION`. In MySQL 8.0.29 and later, conditions can be pushed down to both query blocks of a union in most cases; see Section 8.2.2.5, “Derived Condition Pushdown Optimization”, for more information.

In addition, a condition that itself uses a subquery cannot be pushed down, and a `WHERE` condition cannot be pushed down to a derived table that is also an inner table of an outer join. For additional information and examples, see Section 8.2.2.5, “Derived Condition Pushdown Optimization”.

- **Non-locking reads on MySQL grant tables.** As of MySQL 8.0.22, to permit concurrent DML and DDL operations on MySQL grant tables, read operations that previously acquired row locks on MySQL grant tables are executed as non-locking reads.

  The operations that are now performed as non-locking reads on MySQL grant tables include:

  - `SELECT` statements and other read-only statements that read data from grant tables through join lists and subqueries, including `SELECT ... FOR SHARE` statements, using any transaction isolation level.
  - DML operations that read data from grant tables (through join lists or subqueries) but do not modify them, using any transaction isolation level.

  For additional information, see Grant Table Concurrency.

- **64-bit support for FROM_UNIXTIME(), UNIX_TIMESTAMP(), CONVERT_TZ().** As of MySQL 8.0.28, the functions `FROM_UNIXTIME()`, `UNIX_TIMESTAMP()`, and `CONVERT_TZ()` handle 64-bit values on platforms that support them. This includes 64-bit versions of Linux, MacOS, and Windows.

  On compatible platforms, `UNIX_TIMESTAMP()` now handles values up to `'3001-01-18 23:59:59.999999'` UTC, and `FROM_UNIXTIME()` can convert values up to 32536771199.999999 seconds since the Unix Epoch; `CONVERT_TZ()` now accepts values that do not exceed `'3001-01-18 23:59:59.999999'` UTC following conversion.

  The behavior of these functions on 32-bit platforms is unaffected by these changes. The behavior of the `TIMESTAMP` type is also not affected (on any platform); for working with datetimes after `'2038-01-19 03:14:07.999999'`, UTC, use the `DATETIME` type instead.

  For more information, see the descriptions of the individual functions just discussed, in Section 12.7, “Date and Time Functions”.

- **Resource allocation control.** Beginning with MySQL 8.0.28, you can see the amount of memory used for queries issued by all regular users by checking the `Global_connection_memory` status variable. (This total does not include resources used by system users such as MySQL root. It is also exclusive of any memory taken by the `InnoDB` buffer pool.)

  To enable updates of `Global_connection_memory`, it is necessary to set `global_connection_memory_tracking = 1`; this is `0` (off) by default. You can control how often `Global_connection_memory` is updated by setting `connection_memory_chunk_size`.

  It is also possible to set memory usage limits for normal users on the session or global level, or both, by setting either or both of the system variables listed here:

  - `connection_memory_limit`: Amount of memory allocated for each connection. Whenever this limit is exceeded for any user, new queries from this user are rejected.
  - `global_connection_memory_limit`: Amount of memory allocated for all connections. Whenever this limit is exceeded, new queries from any regular user are rejected.

These limits do not apply to system processes or administrative accounts.

See the descriptions of the referenced variables for more information.

- **Detached XA transactions.** MySQL 8.0.29 adds support for XA transactions which, once prepared, are no longer connected to the originating connection. This means that they can be committed or rolled back by another connection, and that the current session can immediately begin another transaction.

  A system variable `xa_detach_on_prepare` controls whether XA transaction are detached; the default is `ON`, which causes all XA transactions to be detached. Use of temporary tables is disallowed for XA transactions when this is in effect.

  For more information, see Section 13.3.8.2, “XA Transaction States”.

- **Automatic binary log purge control.** MySQL 8.0.29 adds the `binlog_expire_logs_auto_purge` system variable, which provides a single interface for enabling and disabling automatic purging of the binary logs. This is enabled (`ON`) by default; to disable automatic purging of the binary log files, set this variable to `OFF`.

  `binlog_expire_logs_auto_purge` must be `ON` in order for automatic purging of the binary log files to proceed; the value of this variable takes precedence over that of any other server option or variable, including (but not exclusive to) `binlog_expire_logs_seconds`.

  The setting for `binlog_expire_logs_auto_purge` has no effect on `PURGE BINARY LOGS`.

- **Conditional routine and trigger creation statements.** Beginning with MySQL 8.0.29, the following statements support an `IF NOT EXISTS` option:

  - `CREATE FUNCTION`
  - `CREATE PROCEDURE`
  - `CREATE TRIGGER`

  For `CREATE FUNCTION` when creating a stored function and `CREATE PROCEDURE`, this option prevents an error from occurring if there is already a routine having the same name. For `CREATE FUNCTION` when used to create a loadable function, the option prevents an error if there already exists a loadable function having that name. For `CREATE TRIGGER`, the option prevents an error from occurring if there already exists in the same schema and on the same table a trigger having the same name.

  This enhancement aligns the syntax of these statements more closely with that of `CREATE DATABASE`, `CREATE TABLE`, `CREATE USER`, and `CREATE EVENT` (all of which already support `IF NOT EXISTS`), and acts to complement the `IF EXISTS` option supported by `DROP PROCEDURE`, `DROP FUNCTION`, and `DROP TRIGGER` statements.

  For more information, see the descriptions of the indicated SQL statements, as well as Function Name Resolution. See also Section 17.5.1.7, “Replication of CREATE TABLE ... SELECT Statements”.

- **Included FIDO library upgrade.** MySQL 8.0.30 upgrades the included `fido2` library (used with the `authentication_fido` plugin) from version 1.5.0 to version 1.8.0.

  See Section 6.4.1.11, “FIDO Pluggable Authentication”, for more information.

- **Character sets: Language-specific collations.** Previously, when more than one language had the exact same collation definition, MySQL implemented collations for only one of the languages, which meant that some languages were covered only by `utf8mb4` Unicode 9.0 collations specific to other languages. MySQL 8.0.30 (and later) fixes such issues by providing language-specific

collations for those languages that were previously covered only by language-specific collations for other languages. Languages covered by the new collations are listed here:

- Norwegian (Nynorsk)

  and

  Norwegian (Bokmål)
- Serbian (Latin characters)
- Bosnian (Latin characters)
- Bulgarian
- Galician
- Mongolian (Cyrillic characters)

MySQL provides `*_as_cs` and `*_ai_ci` collations for each of the languages just listed.

For more information, see Language-Specific Collations.

- **IF EXISTS and IGNORE UNKNOWN USER options for REVOKE.** MySQL 8.0.30 implements two new options for `REVOKE` which can be used to determine whether a statement yields an error or a warning when a user, role, or privilege specified in the statement cannot be found, or cannot be assigned. Very basic syntax showing the placement of these new options is provided here:

```
REVOKE [IF EXISTS] privilege_or_role
    ON object
    FROM user_or_role [IGNORE UNKNOWN USER]
```

`IF EXISTS` causes an unsuccessful `REVOKE` statement to raise a warning instead of an error, as long as the named target user or role actually exists, despite any references in the statement to any roles or privileges which cannot be found.

`IGNORE UNKNOWN USER` causes an unsuccessful *`REVOKE`* to raise a warning rather than an error when the target user or role named in the statement cannot be found.

For further information and examples, see Section 13.7.1.8, “REVOKE Statement”.

- **Generated invisible primary keys.** Beginning with MySQL 8.0.30, it is possible to run a replication source server such that a generated invisible primary key (GIPK) is added to any `InnoDB` table that is created without an explicit primary key. The generated key column definition added to such a table is equivalent to what is shown here:

```
my_row_id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT INVISIBLE PRIMARY KEY
```

GIPK mode is not enabled by default. To enable it, set the `sql_generate_invisible_primary_key` server system variable to `ON`.

Generated invisible primary keys are normally visible in the output of statements such as `SHOW CREATE TABLE` and `SHOW INDEX`, as well as in MySQL Information Schema tables such as the `COLUMNS` and `STATISTICS` tables. You can cause them to be hidden in such cases instead, by setting `show_gipk_in_create_table_and_information_schema` to `OFF`.

As part of this work, a new `--skip-generated-invisible-primary-key` option is added to `mysqldump` and `mysqlpump` to exclude generated invisible primary keys, columns, and column values from their output.

**GIPKs and replication between tables with or without primary keys.** In MySQL Replication, a replica effectively ignores any setting for `sql_generate_invisible_primary_key` on the

source, such that it has no effect on replicated tables. MySQL 8.0.32 and later makes it possible for the replica to add a generated invisible primary key to any `InnoDB` table that otherwise, as replicated, has no primary key. You can do this by invoking `CHANGE REPLICATION SOURCE TO ... REQUIRE_TABLE_PRIMARY_KEY_CHECK = GENERATE` on the replica.

`REQUIRE_TABLE_PRIMARY_KEY_CHECK = GENERATE` is not compatible with MySQL Group Replication.

For further information, see Section 13.1.20.11, “Generated Invisible Primary Keys”.

- **Crash-safe XA transactions.** Previously, XA transactions were not fully resilient to an unexpected halt with respect to the binary log, and if this occurred while the server was executing `XA PREPARE`, `XA COMMIT`, or `XA ROLLBACK`, the server was not guaranteed to be recoverable to the correct state, possibly leaving the binary log with extra XA transactions that had not been applied, or missing one or more XA transactions that had been applied. Beginning with MySQL 8.0.30, this is no longer an issue, and a server that drops out of a replication topology for whatever reason can always be brought back to a consistent XA transaction state when rejoining.

  *Known issue*: When the same transaction XID is used to execute XA transactions sequentially and a break occurs during the execution of `XA COMMIT ... ONE PHASE`, using this same XID, after this transaction has been prepared in the storage engine, it may not be possible any longer to synchronize the state between the binary log and the storage engine.

  For more information, see Section 13.3.8.3, “Restrictions on XA Transactions”.

- **Nesting with UNION.** Beginning with MySQL 8.0.31, bodies of parenthesized query expressions can be nested up to 63 levels deep in combination with `UNION`. Such queries were previously rejected with error `ER_NOT_SUPPORTED_YET`, but are now allowed. `EXPLAIN` output for such a query is shown here:

```
mysql> EXPLAIN FORMAT=TREE (
    ->   (SELECT a, b, c FROM t ORDER BY a LIMIT 3) ORDER BY b LIMIT 2
    -> ) ORDER BY c LIMIT 1\G
*************************** 1. row ***************************
EXPLAIN: -> Limit: 1 row(s)  (cost=5.55..5.55 rows=1)
    -> Sort: c, limit input to 1 row(s) per chunk  (cost=2.50 rows=0)
        -> Table scan on <result temporary>  (cost=2.50 rows=0)
            -> Temporary table  (cost=5.55..5.55 rows=1)
                -> Limit: 2 row(s)  (cost=2.95..2.95 rows=1)
                    -> Sort: b, limit input to 2 row(s) per chunk  (cost=2.50 rows=0)
                        -> Table scan on <result temporary>  (cost=2.50 rows=0)
                            -> Temporary table  (cost=2.95..2.95 rows=1)
                                -> Limit: 3 row(s)  (cost=0.35 rows=1)
                                    -> Sort: t.a, limit input to 3 row(s) per chunk  (cost=0.35 rows=1)
                                        -> Table scan on t  (cost=0.35 rows=1)

1 row in set (0.00 sec)
```

  MySQL follows SQL standard semantics when collapsing bodies of parenthesized query expressions, so that a higher outer limit cannot override an inner lower one. For example, `(SELECT ... LIMIT 5) LIMIT 10` can return no more than five rows.

  The 63-level limit is imposed only after the MySQL Optimizer's parser has performed any simplifications or merges which it can.

  For more information, see Section 13.2.11, “Parenthesized Query Expressions”.

- **Disabling query rewrites.** Previously, when using the `Rewriter` plugin, all queries were subject to being rewritten, regardless of user. This could be problematic in certain cases, such as when administering the system, or when applying statements originating from a replication source or a dump file created by `mysqldump` or another MySQL program. MySQL 8.0.31 provides a solution to

such issues by implementing a new user privilege `SKIP_QUERY_REWRITE`; statements issued by a user having this privilege are ignored by `Rewriter` and not rewritten.

MySQL 8.0.31 also adds a new server system variable `rewriter_enabled_for_threads_without_privilege_checks`. When set to `OFF`, rewritable statements issued by threads for which `PRIVILEGE_CHECKS_USER` is `NULL` (such as replication applier threads) are not rewritten by the `Rewriter` plugin. The default is `ON`, which means such statements are rewritten.

For more information, see Section 5.6.4, “The Rewriter Query Rewrite Plugin”.

- **Replication filtering of XA statements.** Previously, the statements `XA START`, `XA END`, `XA COMMIT`, and `XA ROLLBACK` were filtered by the default database whenever using `--replicate-do-db` or `--replicate-ignore-db`, which could lead to missed transactions. As of MySQL 8.0.31, these statements are not filtered in such cases, regardless of the value of `binlog_format`.

- **Replication filtering and privilege checks.** Beginning with MySQL 8.0.31, when replication filtering is in use, a replica no longer raises replication errors related to privilege checks or `require_row_format` validation for events which are filtered out, making it possible to filter out any transactions that fail validation.

  Because privilege checks on filtered rows can no longer cause replication to stop, a replica can now accept only the portion of a database to which a given user has been granted access; this is true as long as as long as updates to this part of the database are replicated only in row-based format.

  This capability may also be of use when migrating to MySQL Database Service from an on-premise or cloud service which uses tables for administration or other purposes to which the inbound replication user does not have access.

  For more information, see Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”, as well as Section 17.5.1.29, “Replica Errors During Replication”.

- **INTERSECT and EXCEPT table operators.** MySQL 8.0.31 adds support for the SQL `INTERSECT` and `EXCEPT` table operators. Where *`a`* and *`b`* represent result sets of queries, these operators behave as follows:

  - *`a`* `INTERSECT` *`b`* includes only rows appearing in both result sets *`a`* and *`b`*.

  - *`a`* `EXCEPT` *`b`* returns only those rows from result set *`a`* which do *not* also appear in *`b`*.

  `INTERSECT DISTINCT`, `INTERSECT ALL`, `EXCEPT DISTINCT`, and `EXCEPT ALL` are all supported; `DISTINCT` is the default for both `INTERSECT` and `EXCEPT` (this is the same as for `UNION`).

  For more information and examples, see Section 13.2.8, “INTERSECT Clause”, and Section 13.2.4, “EXCEPT Clause”.

- **User-defined histograms.** Beginning with MySQL 8.0.31, it is possible to set the histogram of a column to a user-specified JSON value. This can be done using the following SQL syntax:

```
ANALYZE TABLE tbl_name
  UPDATE HISTOGRAM ON col_name
  USING DATA 'json_data'
```

  This statement creates or overwrites a histogram for column *`col_name`* of table *`tbl_name`* using the histogram's JSON representation *`json_data`*. After executing this statement, you can verify that the histogram was created or updated by querying the Information Schema `COLUMN_STATISTICS` table, like this:

```
SELECT HISTOGRAM FROM INFORMATION_SCHEMA.COLUMN_STATISTICS
  WHERE TABLE_NAME='tbl_name'
```

```
AND COLUMN_NAME='col_name';
```

The column value returned should be the same *`json_data`* used in the previous `ANALYZE TABLE` statement.

This can be of use in cases where values deemed important are missed by the histogram sampling process. When this happens, you may want to modify the histogram or set your own histogram based on the complete data set. In addition, sampling a large user data set and building a histogram from it are resource-heavy operations which can impact user queries. With this enhancement, histogram generation can be moved off the (primary) server and performed on a replica instead; the generated histograms can then be assigned to the proper table columns on the source server.

For more information and examples, see Histogram Statistics Analysis.

- **Default EXPLAIN output format.** MySQL 8.0.32 adds a system variable `explain_format` which determines the format of the output from an `EXPLAIN` statement used to obtain a query execution plan in the absence of any `FORMAT` option. For example, if the value of `explain_format` is `TREE`, then the output from any such `EXPLAIN` uses the tree-like format, just as if the statement had specified `FORMAT=TREE`.

  This behavior is overridden by the value set in a `FORMAT` option. Suppose that *`explain_format`* is set to *`TREE`*; even so, `EXPLAIN FORMAT=JSON` *`stmt`* displays the result using the JSON output format.

  For more information and examples, see the description of the `explain_format` system variable, as well as Obtaining Execution Plan Information. There are also implications for the behavior of `EXPLAIN ANALYZE`; see Obtaining Information with EXPLAIN ANALYZE.

## Features Deprecated in MySQL 8.0

The following features are deprecated in MySQL 8.0 and may be removed in a future series. Where alternatives are shown, applications should be updated to use them.

For applications that use features deprecated in MySQL 8.0 that have been removed in a higher MySQL series, statements may fail when replicated from a MySQL 8.0 source to a higher-series replica, or may have different effects on source and replica. To avoid such problems, applications that use features deprecated in 8.0 should be revised to avoid them and use alternatives when possible.

- The `utf8mb3` character set is deprecated. Please use `utf8mb4` instead.
- The following character sets are deprecated:
  - `ucs2` (see Section 10.9.4, “The ucs2 Character Set (UCS-2 Unicode Encoding)”)
  - `macroman` and `macce` (see Section 10.10.2, “West European Character Sets”, and Section 10.10.3, “Central European Character Sets”)
  - `dec` (see Section 10.10.2, “West European Character Sets”)
  - `hp8` (see Section 10.10.2, “West European Character Sets”)

  In MySQL 8.0.28 and later, any of these character sets or their collations produces a deprecation warning when used in either of the following ways:

  - When starting the MySQL server with `--character-set-server` or `--collation-server`
  - When specified in any SQL statement, including but not limited to `CREATE TABLE`, `CREATE DATABASE`, `SET NAMES`, and `ALTER TABLE`

  You should use `utf8mb4` instead any of the character sets listed previously.

User-defined collations are deprecated. Beginning with MySQL 8.0.33, either of the following causes a warning to be written to the log:

- Use of `COLLATE` in any SQL statement together with the name of a user-defined collation
- Using the name of a user-defined collation for the value of `collation_server`, `collation_database`, or `collation_connection`.

You should expect support for user-defined collations to be removed in a future version of MySQL.

- Because `caching_sha2_password` is the default authentication plugin in MySQL 8.0 and provides a superset of the capabilities of the `sha256_password` authentication plugin, `sha256_password` is deprecated; expect it to be removed in a future version of MySQL. MySQL accounts that authenticate using `sha256_password` should be migrated to use `caching_sha2_password` instead.
- The `validate_password` plugin has been reimplemented to use the component infrastructure. The plugin form of `validate_password` is still available but is now deprecated; expect it to be removed in a future version of MySQL. MySQL installations that use the plugin should make the transition to using the component instead. See Section 6.4.3.3, “Transitioning to the Password Validation Component”.
- The `ENGINE` clause for the `ALTER TABLESPACE` and `DROP TABLESPACE` statements is deprecated.
- The `PAD_CHAR_TO_FULL_LENGTH` SQL mode is deprecated.
- `AUTO_INCREMENT` support is deprecated for columns of type `FLOAT` and `DOUBLE` (and any synonyms). Consider removing the `AUTO_INCREMENT` attribute from such columns, or convert them to an integer type.
- The `UNSIGNED` attribute is deprecated for columns of type `FLOAT`, `DOUBLE`, and `DECIMAL` (and any synonyms). Consider using a simple `CHECK` constraint instead for such columns.
- `FLOAT(M,D)` and `DOUBLE(M,D)` syntax to specify the number of digits for columns of type `FLOAT` and `DOUBLE` (and any synonyms) is a nonstandard MySQL extension. This syntax is deprecated.
- The `ZEROFILL` attribute is deprecated for numeric data types, as is the display width attribute for integer data types. Consider using an alternative means of producing the effect of these attributes. For example, applications could use the `LPAD()` function to zero-pad numbers up to the desired width, or they could store the formatted numbers in `CHAR` columns.
- For string data types, the `BINARY` attribute is a nonstandard MySQL extension that is shorthand for specifying the binary (`_bin`) collation of the column character set (or of the table default character set if no column character set is specified). In MySQL 8.0, this nonstandard use of `BINARY` is ambiguous because the `utf8mb4` character set has multiple `_bin` collations, so the `BINARY` attribute is deprecated; expect support for it to be removed in a future version of MySQL. Applications should be adjusted to use an explicit `_bin` collation instead.

  The use of `BINARY` to specify a data type or character set remains unchanged.
- Previous versions of MySQL supported the nonstandard shorthand expressions `ASCII` and `UNICODE`, respectively, for `CHARACTER SET latin1` and `CHARACTER SET ucs2`. `ASCII` and `UNICODE` are deprecated (MySQL 8.0.28 and later) and now produce a warning. Use `CHARACTER SET` instead, in both cases.

- The nonstandard C-style `&&`, `||`, and `!` operators that are synonyms for the standard SQL `AND`, `OR`, and `NOT` operators, respectively, are deprecated. Applications that use the nonstandard operators should be adjusted to use the standard operators.

  **Note**

  Use of `||` is deprecated unless the `PIPES_AS_CONCAT` SQL mode is enabled. In that case, `||` signifies the SQL-standard string concatenation operator).

- The `JSON_MERGE()` function is deprecated. Use `JSON_MERGE_PRESERVE()` instead.

- The `SQL_CALC_FOUND_ROWS` query modifier and accompanying `FOUND_ROWS()` function are deprecated. See the `FOUND_ROWS()` description for information about an alternative strategy.

- Support for `TABLESPACE = innodb_file_per_table` and `TABLESPACE = innodb_temporary` clauses with `CREATE TEMPORARY TABLE` is deprecated as of MySQL 8.0.13.

- For `SELECT` statements, use of an `INTO` clause after `FROM` but not at the end of the `SELECT` is deprecated as of MySQL 8.0.20. It is preferred to place the `INTO` at the end of the statement.

  For `UNION` statements, these two variants containing `INTO` are deprecated as of MySQL 8.0.20:

  - In the trailing query block of a query expression, use of `INTO` before `FROM`.
  - In a parenthesized trailing block of a query expression, use of `INTO`, regardless of its position relative to `FROM`.

  See Section 13.2.13.1, “SELECT ... INTO Statement”, and Section 13.2.18, “UNION Clause”.

- `FLUSH HOSTS` is deprecated as of MySQL 8.0.23. Instead, truncate the Performance Schema `host_cache` table:

  ```
  TRUNCATE TABLE performance_schema.host_cache;
  ```

  The `TRUNCATE TABLE` operation requires the `DROP` privilege for the table.

- The `mysql_upgrade` client is deprecated because its capabilities for upgrading the system tables in the `mysql` system schema and objects in other schemas have been moved into the MySQL server. See Section 2.10.3, “What the MySQL Upgrade Process Upgrades”.

- The `--no-dd-upgrade` server option is deprecated. It is superseded by the `--upgrade` option, which provides finer control over data dictionary and server upgrade behavior.

- The `mysql_upgrade_info` file, which is created data directory and used to store the MySQL version number, is deprecated; expect it to be removed in a future version of MySQL.

- The `relay_log_info_file` system variable and `--master-info-file` option are deprecated. Previously, these were used to specify the name of the relay log info log and source info log when `relay_log_info_repository=FILE` and `master_info_repository=FILE` were set, but those settings have been deprecated. The use of files for the relay log info log and source info log has been superseded by crash-safe replica tables, which are the default in MySQL 8.0.

- The `max_length_for_sort_data` system variable is now deprecated due to optimizer changes that make it obsolete and of no effect.

- These legacy parameters for compression of connections to the server are deprecated:
  The `--compress` client command-line option; the `MYSQL_OPT_COMPRESS` option for the `mysql_options()` C API function; the `slave_compressed_protocol` system variable. For information about parameters to use instead, see Section 4.2.8, “Connection Compression Control”.

- Use of the `MYSQL_PWD` environment variable to specify a MySQL password is deprecated.

- Use of `VALUES()` to access new row values in `INSERT ... ON DUPLICATE KEY UPDATE` is deprecated as of MySQL 8.0.20. Use aliases for the new row and columns, instead.
- Because specifying `ON ERROR` before `ON EMPTY` when invoking `JSON_TABLE()` is counter to the SQL standard, this syntax is now deprecated in MySQL. Beginning with MySQL 8.0.20, the server prints a warning whenever you attempt to do so. When specifying both of these clauses in a single `JSON_TABLE()` invocation, make sure that `ON EMPTY` is used first.
- Columns with index prefixes have never been supported as part of a table's partitioning key; previously, these were allowed when creating, altering, or upgrading partitioned tables but were excluded by the table's partitioning function, and no warning that this had occurred was issued by the server. This permissive behavior is now deprecated, and subject to removal in a future version of MySQL in which using any such columns in the partitioning key causes the `CREATE TABLE` or `ALTER TABLE` statement in they occur to be rejected.

  As of MySQL 8.0.21, whenever columns using index prefixes are specified as part of the partitioning key, a warning is generated for each such column. Whenever a `CREATE TABLE` or `ALTER TABLE` statement is rejected because all columns in the proposed partitioning key would have index prefixes, the resulting error now provides the exact reason for the rejection. In either instance, this includes cases in which the columns used in the partitioning function are defined implicitly as those in the table's primary key by employing an empty `PARTITION BY KEY()` clause.

  For more information and examples, see Column index prefixes not supported for key partitioning.
- The InnoDB memcached plugin is deprecated as of MySQL 8.0.22; expect support for it to be removed in a future version of MySQL.
- The `temptable_use_mmap` variable is deprecated as of MySQL 8.0.26; expect support for it to be removed in a future version of MySQL.
- The `BINARY` operator is deprecated as of MySQL 8.0.27, and you should expect its removal in a future version of MySQL. Use of `BINARY` now causes a warning. Use `CAST(... AS BINARY)` instead.
- The `default_authentication_plugin` variable is deprecated as of MySQL 8.0.27; expect support for it to be removed in a future version of MySQL.

  The `default_authentication_plugin` variable is still used in MySQL 8.0.27, but in conjunction with and at a lower precedence than the new `authentication_policy` system variable, which is introduced in MySQL 8.0.27 with the multifactor authentication feature. For details, see The Default Authentication Plugin.
- The `--abort-slave-event-count` and `--disconnect-slave-event-count` server options, used by the MySQL test suite and not normally used in production, are deprecated as of MySQL 8.0.29; expect both options to be removed in a future version of MySQL.
- The `myisam_repair_threads` system variable and `myisamchk --parallel-recover` option are deprecated as of MySQL 8.0.29; expect support for both to be removed in a future release of MySQL.

  From MySQL 8.0.29, values other than 1 (the default) for `myisam_repair_threads` produce a warning.
- Previously, MySQL accepted `DATE`, `TIME`, `DATETIME`, and `TIMESTAMP` literals containing an arbitrary number of (arbitrary) delimiter characters, as well as `DATETIME` and `TIMESTAMP` literals with an arbitrary number of whitespace characters before, after, and between the date and time parts. As of MySQL 8.0.29, the server raises a deprecation warning whenever the literal value contains any of the following:
  - One or more nonstandard delimiter characters
  - Excess delimiter characters

- Whitespace other than the space character (' ', `0x20`)
- Excess space characters

One deprecation warning is issued per temporal value, even if there are multiple issues with it. This warning is not promoted to an error in strict mode, so that performing an `INSERT` of such a value still succeeds when strict mode is in effect.

You should expect the nonstandard behavior to be removed in a future version of MySQL, and take steps now to insure that your applications do not depend on it.

See String and Numeric Literals in Date and Time Context, for more information and examples.

- The `replica_parallel_type` system variable and its associated server option `--replica-parallel-type` are deprecated as of MySQL 8.0.29. Beginning with this release, reading or setting this value raises a deprecation warning; expect it to be removed in a future version of MySQL.
- The `--skip-host-cache` server option is deprecated beginning with MySQL 8.0.30; expect its removal in a future MySQL release. Use the `host_cache_size` system variable instead.
- Beginning with MySQL 8.0.30, setting the `replica_parallel_workers` system variable (or the equivalent server option) to 0 is deprecated, and elicits a warning. When you want a replica to use single threading, use `replica_parallel_workers=1` instead, which produces the same result, but with no warning.
- The `--old-style-user-limits` option, intended for backwards compatibility with very old (pre-5.0.3) releases, is deprecated as of MySQL 8.0.30; using it now raises a warning. You should expect this option to be removed in a future release of MySQL.
- Prior to MySQL 8.0.30, the `ST_TRANSFORM()` function did not support Cartesian Spatial Reference Systems. In MySQL 8.0.30 and later, this function provides support for the Popular Visualisation Pseudo Mercator (EPSG 1024) projection method, used for WGS 84 Pseudo-Mercator (SRID 3857). MySQL 8.0.32 and later supports all Cartesian SRSs, except for EPSG 1042, EPSG 1043, EPSG 9816, and EPSG 9826.
- MySQL 8.0.31 adds the read-only `build_id` system variable for Linux systems, where a 160-bit `SHA1` signature is generated at compile time; the value of `build_id` is that of the generated value converted to a hexadecimal string, providing a unique identifier for the build.

  `build_id` is written to the server log each time MySQL starts.

  If you build MySQL from source, you can observe that this value changes each time you recompile the server. See Section 2.8, “Installing MySQL from Source”, for more information.

  This variable is not supported on platforms other than Linux.
- The `innodb_log_files_in_group` and `innodb_log_file_size` variables are deprecated as of MySQL 8.0.30. These variables are superseded by the `innodb_redo_log_capacity` variable. For more information, see Section 15.6.5, “Redo Log”.
- As of MySQL 8.0.32, the use of “FULL” as an unquoted identifier is deprecated, due to the fact that it is a reserved keyword in the SQL standard. This means that a statement such as `CREATE TABLE full (c1 INT, c2 INT)` now raises a warning (`ER_WARN_DEPRECATED_TO_BE_REMOVED_IDENT_FULL`). To prevent this from happening, change the name or, as shown here, encase it in backticks (`):

  ```
  CREATE TABLE `full` (c1 INT, c2 INT);
  ```

  For more information, see Section 9.3, “Keywords and Reserved Words”.
- Beginning with MySQL 8.0.32, the use of the dollar sign (`$`) as the leading character of an unquoted identifier is deprecated and produces a warning. Such usage is subject to removal in a future release

of MySQL. This includes identifiers used as names of databases, tables, views, columns, or stored programs, as well as aliases for any of these. The dollar sign may still be used as the first character of a quoted identifier. See Section 9.2, “Schema Object Names”, for more information.

## Features Removed in MySQL 8.0

The following items are obsolete and have been removed in MySQL 8.0. Where alternatives are shown, applications should be updated to use them.

For MySQL 5.7 applications that use features removed in MySQL 8.0, statements may fail when replicated from a MySQL 5.7 source to a MySQL 8.0 replica, or may have different effects on source and replica. To avoid such problems, applications that use features removed in MySQL 8.0 should be revised to avoid them and use alternatives when possible.

- The `innodb_locks_unsafe_for_binlog` system variable was removed. The `READ COMMITTED` isolation level provides similar functionality.

- The `information_schema_stats` variable, introduced in MySQL 8.0.0, was removed and replaced by `information_schema_stats_expiry` in MySQL 8.0.3.

  `information_schema_stats_expiry` defines an expiration setting for cached `INFORMATION_SCHEMA` table statistics. For more information, see Section 8.2.3, “Optimizing INFORMATION_SCHEMA Queries”.

- Code related to obsolete `InnoDB` system tables was removed in MySQL 8.0.3. `INFORMATION_SCHEMA` views based on `InnoDB` system tables were replaced by internal system views on data dictionary tables. Affected `InnoDB` `INFORMATION_SCHEMA` views were renamed:

  **Table 1.1 Renamed InnoDB Information Schema Views**

| Old Name | New Name |
|---|---|
| `INNODB_SYS_COLUMNS` | `INNODB_COLUMNS` |
| `INNODB_SYS_DATAFILES` | `INNODB_DATAFILES` |
| `INNODB_SYS_FIELDS` | `INNODB_FIELDS` |
| `INNODB_SYS_FOREIGN` | `INNODB_FOREIGN` |
| `INNODB_SYS_FOREIGN_COLS` | `INNODB_FOREIGN_COLS` |
| `INNODB_SYS_INDEXES` | `INNODB_INDEXES` |
| `INNODB_SYS_TABLES` | `INNODB_TABLES` |
| `INNODB_SYS_TABLESPACES` | `INNODB_TABLESPACES` |
| `INNODB_SYS_TABLESTATS` | `INNODB_TABLESTATS` |
| `INNODB_SYS_VIRTUAL` | `INNODB_VIRTUAL` |

  After upgrading to MySQL 8.0.3 or later, update any scripts that reference previous `InnoDB` `INFORMATION_SCHEMA` view names.

- The following features related to account management are removed:

  - Using `GRANT` to create users. Instead, use `CREATE USER`. Following this practice makes the `NO_AUTO_CREATE_USER` SQL mode immaterial for `GRANT` statements, so it too is removed, and an error now is written to the server log when the presence of this value for the `sql_mode` option in the options file prevents `mysqld` from starting.

  - Using `GRANT` to modify account properties other than privilege assignments. This includes authentication, SSL, and resource-limit properties. Instead, establish such properties at account-creation time with `CREATE USER` or modify them afterward with `ALTER USER`.

  - `IDENTIFIED BY PASSWORD 'auth_string'` syntax for `CREATE USER` and `GRANT`. Instead, use `IDENTIFIED WITH auth_plugin AS 'auth_string'` for `CREATE USER` and `ALTER USER`, where the `'auth_string'` value is in a format compatible with the named plugin.

    Additionally, because `IDENTIFIED BY PASSWORD` syntax was removed, the `log_builtin_as_identified_by_password` system variable is superfluous and was removed.

  - The `PASSWORD()` function. Additionally, `PASSWORD()` removal means that `SET PASSWORD ... = PASSWORD('auth_string')` syntax is no longer available.

  - The `old_passwords` system variable.

- The query cache was removed. Removal includes these items:

  - The `FLUSH QUERY CACHE` and `RESET QUERY CACHE` statements.

  - These system variables: `query_cache_limit`, `query_cache_min_res_unit`, `query_cache_size`, `query_cache_type`, `query_cache_wlock_invalidate`.

  - These status variables: `Qcache_free_blocks`, `Qcache_free_memory`, `Qcache_hits`, `Qcache_inserts`, `Qcache_lowmem_prunes`, `Qcache_not_cached`, `Qcache_queries_in_cache`, `Qcache_total_blocks`.

  - These thread states: `checking privileges on cached query`, `checking query cache for query`, `invalidating query cache entries`, `sending cached result to client`, `storing result in query cache`, `Waiting for query cache lock`.

  - The `SQL_CACHE SELECT` modifier.

  These deprecated query cache items remain deprecated, but have no effect; expect them to be removed in a future MySQL release:

  - The `SQL_NO_CACHE SELECT` modifier.

  - The `ndb_cache_check_time` system variable.

  The `have_query_cache` system variable remains deprecated, and always has a value of `NO`; expect it to be removed in a future MySQL release.

- The data dictionary provides information about database objects, so the server no longer checks directory names in the data directory to find databases. Consequently, the `--ignore-db-dir` option and `ignore_db_dirs` system variables are extraneous and are removed.

- The DDL log, also known as the metadata log, has been removed. Beginning with MySQL 8.0.3, this functionality is handled by the data dictionary `innodb_ddl_log` table. See Viewing DDL Logs.

- The `tx_isolation` and `tx_read_only` system variables have been removed. Use `transaction_isolation` and `transaction_read_only` instead.

- The `sync_frm` system variable has been removed because `.frm` files have become obsolete.

- The `secure_auth` system variable and `--secure-auth` client option have been removed. The `MYSQL_SECURE_AUTH` option for the `mysql_options()` C API function was removed.

- The `multi_range_count` system variable is removed.

- The `log_warnings` system variable and `--log-warnings` server option have been removed. Use the `log_error_verbosity` system variable instead.

- The global scope for the `sql_log_bin` system variable was removed. `sql_log_bin` has session scope only, and applications that rely on accessing `@@GLOBAL.sql_log_bin` should be adjusted.

- The `metadata_locks_cache_size` and `metadata_locks_hash_instances` system variables are removed.
- The unused `date_format`, `datetime_format`, `time_format`, and `max_tmp_tables` system variables are removed.
- These deprecated compatibility SQL modes are removed: `DB2`, `MAXDB`, `MSSQL`, `MYSQL323`, `MYSQL40`, `ORACLE`, `POSTGRESQL`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS`, `NO_TABLE_OPTIONS`. They can no longer be assigned to the `sql_mode` system variable or used as permitted values for the `mysqldump --compatible` option.

  Removal of `MAXDB` means that the `TIMESTAMP` data type for `CREATE TABLE` or `ALTER TABLE` is treated as `TIMESTAMP`, and is no longer treated as `DATETIME`.
- The deprecated `ASC` or `DESC` qualifiers for `GROUP BY` clauses are removed. Queries that previously relied on `GROUP BY` sorting may produce results that differ from previous MySQL versions. To produce a given sort order, provide an `ORDER BY` clause.
- The `EXTENDED` and `PARTITIONS` keywords for the `EXPLAIN` statement have been removed. These keywords are unnecessary because their effect is always enabled.
- These encryption-related items are removed:
  - The `ENCODE()` and `DECODE()` functions.
  - The `ENCRYPT()` function.
  - The `DES_ENCRYPT()`, and `DES_DECRYPT()` functions, the `--des-key-file` option, the `have_crypt` system variable, the `DES_KEY_FILE` option for the `FLUSH` statement, and the `HAVE_CRYPT` `CMake` option.

  In place of the removed encryption functions: For `ENCRYPT()`, consider using `SHA2()` instead for one-way hashing. For the others, consider using `AES_ENCRYPT()` and `AES_DECRYPT()` instead.
- In MySQL 5.7, several spatial functions available under multiple names were deprecated to move in the direction of making the spatial function namespace more consistent, the goal being that each spatial function name begin with `ST_` if it performs an exact operation, or with `MBR` if it performs an operation based on minimum bounding rectangles. In MySQL 8.0, the deprecated functions are removed to leave only the corresponding `ST_` and `MBR` functions:
  - These functions are removed in favor of the `MBR` names: `Contains()`, `Disjoint()`, `Equals()`, `Intersects()`, `Overlaps()`, `Within()`.
  - These functions are removed in favor of the `ST_` names: `Area()`, `AsBinary()`, `AsText()`, `AsWKB()`, `AsWKT()`, `Buffer()`, `Centroid()`, `ConvexHull()`, `Crosses()`, `Dimension()`, `Distance()`, `EndPoint()`, `Envelope()`, `ExteriorRing()`, `GeomCollFromText()`, `GeomCollFromWKB()`, `GeomFromText()`, `GeomFromWKB()`, `GeometryCollectionFromText()`, `GeometryCollectionFromWKB()`, `GeometryFromText()`, `GeometryFromWKB()`, `GeometryN()`, `GeometryType()`, `InteriorRingN()`, `IsClosed()`, `IsEmpty()`, `IsSimple()`, `LineFromText()`, `LineFromWKB()`, `LineStringFromText()`, `LineStringFromWKB()`, `MLineFromText()`, `MLineFromWKB()`, `MPointFromText()`, `MPointFromWKB()`, `MPolyFromText()`, `MPolyFromWKB()`, `MultiLineStringFromText()`, `MultiLineStringFromWKB()`, `MultiPointFromText()`, `MultiPointFromWKB()`, `MultiPolygonFromText()`, `MultiPolygonFromWKB()`, `NumGeometries()`, `NumInteriorRings()`, `NumPoints()`, `PointFromText()`, `PointFromWKB()`, `PointN()`, `PolyFromText()`, `PolyFromWKB()`, `PolygonFromText()`, `PolygonFromWKB()`, `SRID()`, `StartPoint()`, `Touches()`, `X()`, `Y()`.
  - `GLength()` is removed in favor of `ST_Length()`.

- The functions described in Section 12.17.4, “Functions That Create Geometry Values from WKB Values” previously accepted either WKB strings or geometry arguments. Geometry arguments are no longer permitted and produce an error. See that section for guidelines for migrating queries away from using geometry arguments.

- The parser no longer treats `\N` as a synonym for `NULL` in SQL statements. Use `NULL` instead.

  This change does not affect text file import or export operations performed with `LOAD DATA` or `SELECT ... INTO OUTFILE`, for which `NULL` continues to be represented by `\N`. See Section 13.2.9, “LOAD DATA Statement”.

- `PROCEDURE ANALYSE()` syntax is removed.

- The client-side `--ssl` and `--ssl-verify-server-cert` options have been removed. Use `--ssl-mode=REQUIRED` instead of `--ssl=1` or `--enable-ssl`. Use `--ssl-mode=DISABLED` instead of `--ssl=0`, `--skip-ssl`, or `--disable-ssl`. Use `--ssl-mode=VERIFY_IDENTITY` instead of `--ssl-verify-server-cert` options. (The server-side `--ssl` option is still available, but is deprecated as of MySQL 8.0.26 and subject to removal in a future MySQL version.)

  For the C API, `MYSQL_OPT_SSL_ENFORCE` and `MYSQL_OPT_SSL_VERIFY_SERVER_CERT` options for `mysql_options()` correspond to the client-side `--ssl` and `--ssl-verify-server-cert` options and are removed. Use `MYSQL_OPT_SSL_MODE` with an option value of `SSL_MODE_REQUIRED` or `SSL_MODE_VERIFY_IDENTITY` instead.

- The `--temp-pool` server option was removed.

- The `ignore_builtin_innodb` system variable is removed.

- The server no longer performs conversion of pre-MySQL 5.1 database names containing special characters to 5.1 format with the addition of a `#mysql50#` prefix. Because these conversions are no longer performed, the `--fix-db-names` and `--fix-table-names` options for `mysqlcheck`, the `UPGRADE DATA DIRECTORY NAME` clause for the `ALTER DATABASE` statement, and the `Com_alter_db_upgrade` status variable are removed.

  Upgrades are supported only from one major version to another (for example, 5.0 to 5.1, or 5.1 to 5.5), so there should be little remaining need for conversion of older 5.0 database names to current versions of MySQL. As a workaround, upgrade a MySQL 5.0 installation to MySQL 5.1 before upgrading to a more recent release.

- The `mysql_install_db` program has been removed from MySQL distributions. Data directory initialization should be performed by invoking `mysqld` with the `--initialize` or `--initialize-insecure` option instead. In addition, the `--bootstrap` option for `mysqld` that was used by `mysql_install_db` was removed, and the `INSTALL_SCRIPTDIR` `CMake` option that controlled the installation location for `mysql_install_db` was removed.

- The generic partitioning handler was removed from the MySQL server. In order to support partitioning of a given table, the storage engine used for the table must now provide its own (“native”) partitioning handler. The `--partition` and `--skip-partition` options are removed from the

MySQL Server, and partitioning-related entries are no longer shown in the output of `SHOW PLUGINS` or in the Information Schema `PLUGINS` table.

Two MySQL storage engines currently provide native partitioning support: `InnoDB` and `NDB`. Of these, only `InnoDB` is supported in MySQL 8.0. Any attempt to create partitioned tables in MySQL 8.0 using any other storage engine fails.

**Ramifications for upgrades.** The direct upgrade of a partitioned table using a storage engine other than `InnoDB` (such as `MyISAM`) from MySQL 5.7 (or earlier) to MySQL 8.0 is not supported. There are two options for handling such a table:

- Remove the table's partitioning, using `ALTER TABLE ... REMOVE PARTITIONING`.
- Change the storage engine used for the table to `InnoDB`, with `ALTER TABLE ... ENGINE=INNODB`.

At least one of the two operations just listed must be performed for each partitioned non-`InnoDB` table prior to upgrading the server to MySQL 8.0. Otherwise, such a table cannot be used following the upgrade.

Due to the fact that table creation statements that would result in a partitioned table using a storage engine without partitioning support now fail with an error (`ER_CHECK_NOT_IMPLEMENTED`), you must make sure that any statements in a dump file (such as that written by `mysqldump`) from an older version of MySQL that you wish to import into a MySQL 8.0 server that create partitioned tables do not also specify a storage engine such as `MyISAM` that has no native partitioning handler. You can do this by performing either of the following:

- Remove any references to partitioning from `CREATE TABLE` statements that use a value for the `STORAGE ENGINE` option other than `InnoDB`.
- Specifying the storage engine as `InnoDB`, or allow `InnoDB` to be used as the table's storage engine by default.

For more information, see Section 24.6.2, “Partitioning Limitations Relating to Storage Engines”.

- System and status variable information is no longer maintained in the `INFORMATION_SCHEMA`. These tables are removed: `GLOBAL_VARIABLES`, `SESSION_VARIABLES`, `GLOBAL_STATUS`, `SESSION_STATUS`. Use the corresponding Performance Schema tables instead. See Section 27.12.14, “Performance Schema System Variable Tables”, and Section 27.12.15, “Performance Schema Status Variable Tables”. In addition, the `show_compatibility_56` system variable was removed. It was used in the transition period during which system and status variable information in `INFORMATION_SCHEMA` tables was moved to Performance Schema tables, and is no longer needed. These status variables are removed: `Slave_heartbeat_period`, `Slave_last_heartbeat`, `Slave_received_heartbeats`, `Slave_retried_transactions`, `Slave_running`. The information they provided is available in Performance Schema tables; see Migrating to Performance Schema System and Status Variable Tables.
- The Performance Schema `setup_timers` table was removed, as was the `TICK` row in the `performance_timers` table.

- The `libmysqld` embedded server library is removed, along with:
  - The `mysql_options()` `MYSQL_OPT_GUESS_CONNECTION`, `MYSQL_OPT_USE_EMBEDDED_CONNECTION`, `MYSQL_OPT_USE_REMOTE_CONNECTION`, and `MYSQL_SET_CLIENT_IP` options
  - The `mysql_config` `--libmysqld-libs`, `--embedded-libs`, and `--embedded` options
  - The `CMake` `WITH_EMBEDDED_SERVER`, `WITH_EMBEDDED_SHARED_LIBRARY`, and `INSTALL_SECURE_FILE_PRIV_EMBEDDEDDIR` options
  - The (undocumented) `mysql` `--server-arg` option
  - The `mysqltest` `--embedded-server`, `--server-arg`, and `--server-file` options
  - The `mysqltest_embedded` and `mysql_client_test_embedded` test programs
- The `mysql_plugin` utility was removed. Alternatives include loading plugins at server startup using the `--plugin-load` or `--plugin-load-add` option, or at runtime using the `INSTALL PLUGIN` statement.
- The `resolveip` utility is removed. `nslookup`, `host`, or `dig` can be used instead.
- The `resolve_stack_dump` utility is removed. Stack traces from official MySQL builds are always symbolized, so there is no need to use `resolve_stack_dump`.
- The following server error codes are not used and have been removed. Applications that test specifically for any of these errors should be updated.

```
ER_BINLOG_READ_EVENT_CHECKSUM_FAILURE
ER_BINLOG_ROW_RBR_TO_SBR
ER_BINLOG_ROW_WRONG_TABLE_DEF
ER_CANT_ACTIVATE_LOG
ER_CANT_CHANGE_GTID_NEXT_IN_TRANSACTION
ER_CANT_CREATE_FEDERATED_TABLE
ER_CANT_CREATE_SROUTINE
ER_CANT_DELETE_FILE
ER_CANT_GET_WD
ER_CANT_SET_GTID_PURGED_WHEN_GTID_MODE_IS_OFF
ER_CANT_SET_WD
ER_CANT_WRITE_LOCK_LOG_TABLE
ER_CREATE_DB_WITH_READ_LOCK
ER_CYCLIC_REFERENCE
ER_DB_DROP_DELETE
ER_DELAYED_NOT_SUPPORTED
ER_DIFF_GROUPS_PROC
ER_DISK_FULL
ER_DROP_DB_WITH_READ_LOCK
ER_DROP_USER
ER_DUMP_NOT_IMPLEMENTED
ER_ERROR_DURING_CHECKPOINT
ER_ERROR_ON_CLOSE
ER_EVENTS_DB_ERROR
ER_EVENT_CANNOT_DELETE
ER_EVENT_CANT_ALTER
ER_EVENT_COMPILE_ERROR
ER_EVENT_DATA_TOO_LONG
ER_EVENT_DROP_FAILED
ER_EVENT_MODIFY_QUEUE_ERROR
ER_EVENT_NEITHER_M_EXPR_NOR_M_AT
ER_EVENT_OPEN_TABLE_FAILED
ER_EVENT_STORE_FAILED
ER_EXEC_STMT_WITH_OPEN_CURSOR
ER_FAILED_ROUTINE_BREAK_BINLOG
ER_FLUSH_MASTER_BINLOG_CLOSED
ER_FORM_NOT_FOUND
ER_FOUND_GTID_EVENT_WHEN_GTID_MODE_IS_OFF__UNUSED
ER_FRM_UNKNOWN_TYPE
```

```
ER_GOT_SIGNAL
ER_GRANT_PLUGIN_USER_EXISTS
ER_GTID_MODE_REQUIRES_BINLOG
ER_GTID_NEXT_IS_NOT_IN_GTID_NEXT_LIST
ER_HASHCHK
ER_INDEX_REBUILD
ER_INNODB_NO_FT_USES_PARSER
ER_LIST_OF_FIELDS_ONLY_IN_HASH_ERROR
ER_LOAD_DATA_INVALID_COLUMN_UNUSED
ER_LOGGING_PROHIBIT_CHANGING_OF
ER_MALFORMED_DEFINER
ER_MASTER_KEY_ROTATION_ERROR_BY_SE
ER_NDB_CANT_SWITCH_BINLOG_FORMAT
ER_NEVER_USED
ER_NISAMCHK
ER_NO_CONST_EXPR_IN_RANGE_OR_LIST_ERROR
ER_NO_FILE_MAPPING
ER_NO_GROUP_FOR_PROC
ER_NO_RAID_COMPILED
ER_NO_SUCH_KEY_VALUE
ER_NO_SUCH_PARTITION__UNUSED
ER_OBSOLETE_CANNOT_LOAD_FROM_TABLE
ER_OBSOLETE_COL_COUNT_DOESNT_MATCH_CORRUPTED
ER_ORDER_WITH_PROC
ER_PARTITION_SUBPARTITION_ERROR
ER_PARTITION_SUBPART_MIX_ERROR
ER_PART_STATE_ERROR
ER_PASSWD_LENGTH
ER_QUERY_ON_MASTER
ER_RBR_NOT_AVAILABLE
ER_SKIPPING_LOGGED_TRANSACTION
ER_SLAVE_CHANNEL_DELETE
ER_SLAVE_MULTIPLE_CHANNELS_HOST_PORT
ER_SLAVE_MUST_STOP
ER_SLAVE_WAS_NOT_RUNNING
ER_SLAVE_WAS_RUNNING
ER_SP_GOTO_IN_HNDLR
ER_SP_PROC_TABLE_CORRUPT
ER_SQL_MODE_NO_EFFECT
ER_SR_INVALID_CREATION_CTX
ER_TABLE_NEEDS_UPG_PART
ER_TOO_MUCH_AUTO_TIMESTAMP_COLS
ER_UNEXPECTED_EOF
ER_UNION_TABLES_IN_DIFFERENT_DIR
ER_UNSUPPORTED_BY_REPLICATION_THREAD
ER_UNUSED1
ER_UNUSED2
ER_UNUSED3
ER_UNUSED4
ER_UNUSED5
ER_UNUSED6
ER_VIEW_SELECT_DERIVED_UNUSED
ER_WRONG_MAGIC
ER_WSAS_FAILED
```

- The deprecated `INFORMATION_SCHEMA` `INNODB_LOCKS` and `INNODB_LOCK_WAITS` tables are removed. Use the Performance Schema `data_locks` and `data_lock_waits` tables instead.

  **Note**

  In MySQL 5.7, the `LOCK_TABLE` column in the `INNODB_LOCKS` table and the `locked_table` column in the `sys` schema `innodb_lock_waits` and `x$innodb_lock_waits` views contain combined schema/table name values. In MySQL 8.0, the `data_locks` table and the `sys` schema views contain separate schema name and table name columns. See Section 28.4.3.9, “The innodb_lock_waits and x$innodb_lock_waits Views”.

- `InnoDB` no longer supports compressed temporary tables. When `innodb_strict_mode` is enabled (the default), `CREATE TEMPORARY TABLE` returns an error if `ROW_FORMAT=COMPRESSED` or

`KEY_BLOCK_SIZE` is specified. If `innodb_strict_mode` is disabled, warnings are issued and the temporary table is created using a non-compressed row format.

- `InnoDB` no longer creates `.isl` files (`InnoDB` Symbolic Link files) when creating tablespace data files outside of the MySQL data directory. The `innodb_directories` option now supports locating tablespace files created outside of the data directory.

  With this change, moving a remote tablespace while the server is offline by manually modifying an `.isl` file is no longer supported. Moving remote tablespace files is now supported by the `innodb_directories` option. See Section 15.6.3.6, “Moving Tablespace Files While the Server is Offline”.

- The following `InnoDB` file format variables were removed:

  - `innodb_file_format`
  - `innodb_file_format_check`
  - `innodb_file_format_max`
  - `innodb_large_prefix`

  File format variables were necessary for creating tables compatible with earlier versions of `InnoDB` in MySQL 5.1. Now that MySQL 5.1 has reached the end of its product lifecycle, these options are no longer required.

  The `FILE_FORMAT` column was removed from the `INNODB_TABLES` and `INNODB_TABLESPACES` Information Schema tables.

- The `innodb_support_xa` system variable, which enables support for two-phase commit in XA transactions, was removed. `InnoDB` support for two-phase commit in XA transactions is always enabled.

- Support for DTrace was removed.

- The `JSON_APPEND()` function was removed. Use `JSON_ARRAY_APPEND()` instead.

- Support for placing table partitions in shared `InnoDB` tablespaces was removed in MySQL 8.0.13. Shared tablespaces include the `InnoDB` system tablespace and general tablespaces. For information about identifying partitions in shared tablespaces and moving them to file-per-table tablespaces, see Section 2.10.5, “Preparing Your Installation for Upgrade”.

- Support for setting user variables in statements other than `SET` was deprecated in MySQL 8.0.13. This functionality is subject to removal in MySQL 9.0.

- The `--ndb perror` option was removed. Use the `ndb_perror` utility instead.

- The `innodb_undo_logs` variable was removed. The `innodb_rollback_segments` variables performs the same function and should be used instead.

- The `Innodb_available_undo_logs` status variable was removed. The number of available rollback segments per tablespace may be retrieved using `SHOW VARIABLES LIKE 'innodb_rollback_segments';`

- As of MySQL 8.0.14, the previously deprecated `innodb_undo_tablespaces` variable is no longer configurable. For more information, see Section 15.6.3.4, “Undo Tablespaces”.

- Support for the `ALTER TABLE ... UPGRADE PARTITIONING` statement has been removed.

- As of MySQL 8.0.16, support for the `internal_tmp_disk_storage_engine` system variable has been removed; internal temporary tables on disk now always use the `InnoDB` storage engine. See Storage Engine for On-Disk Internal Temporary Tables,for more information.

- The `DISABLE_SHARED CMake` option was unused and has been removed.
- The `myisam_repair_threads` system variable is removed as of MySQL 8.0.30.

# 1.4 Server and Status Variables and Options Added, Deprecated, or Removed in MySQL 8.0



This section lists server variables, status variables, and options that were added for the first time, have been deprecated, or have been removed in MySQL 8.0.

## Options and Variables Introduced in MySQL 8.0

The following system variables, status variables, and server options have been added in MySQL 8.0.

- `Acl_cache_items_count`: Number of cached privilege objects. Added in MySQL 8.0.0.
- `Audit_log_current_size`: Audit log file current size. Added in MySQL 8.0.11.
- `Audit_log_event_max_drop_size`: Size of largest dropped audited event. Added in MySQL 8.0.11.
- `Audit_log_events`: Number of handled audited events. Added in MySQL 8.0.11.
- `Audit_log_events_filtered`: Number of filtered audited events. Added in MySQL 8.0.11.
- `Audit_log_events_lost`: Number of dropped audited events. Added in MySQL 8.0.11.
- `Audit_log_events_written`: Number of written audited events. Added in MySQL 8.0.11.
- `Audit_log_total_size`: Combined size of written audited events. Added in MySQL 8.0.11.
- `Audit_log_write_waits`: Number of write-delayed audited events. Added in MySQL 8.0.11.
- `Authentication_ldap_sasl_supported_methods`: Supported authentication methods for SASL LDAP authentication. Added in MySQL 8.0.21.
- `Caching_sha2_password_rsa_public_key`: caching_sha2_password authentication plugin RSA public key value. Added in MySQL 8.0.4.
- `Com_alter_resource_group`: Count of ALTER RESOURCE GROUP statements. Added in MySQL 8.0.3.
- `Com_alter_user_default_role`: Count of ALTER USER ... DEFAULT ROLE statements. Added in MySQL 8.0.0.
- `Com_change_replication_source`: Count of CHANGE REPLICATION SOURCE TO and CHANGE MASTER TO statements. Added in MySQL 8.0.23.
- `Com_clone`: Count of CLONE statements. Added in MySQL 8.0.2.
- `Com_create_resource_group`: Count of CREATE RESOURCE GROUP statements. Added in MySQL 8.0.3.
- `Com_create_role`: Count of CREATE ROLE statements. Added in MySQL 8.0.0.
- `Com_drop_resource_group`: Count of DROP RESOURCE GROUP statements. Added in MySQL 8.0.3.

- `Com_drop_role`: Count of DROP ROLE statements. Added in MySQL 8.0.0.
- `Com_grant_roles`: Count of GRANT ROLE statements. Added in MySQL 8.0.0.
- `Com_install_component`: Count of INSTALL COMPONENT statements. Added in MySQL 8.0.0.
- `Com_replica_start`: Count of START REPLICA and START SLAVE statements. Added in MySQL 8.0.22.
- `Com_replica_stop`: Count of STOP REPLICA and STOP SLAVE statements. Added in MySQL 8.0.22.
- `Com_restart`: Count of RESTART statements. Added in MySQL 8.0.4.
- `Com_revoke_roles`: Count of REVOKE ROLES statements. Added in MySQL 8.0.0.
- `Com_set_resource_group`: Count of SET RESOURCE GROUP statements. Added in MySQL 8.0.3.
- `Com_set_role`: Count of SET ROLE statements. Added in MySQL 8.0.0.
- `Com_show_replica_status`: Count of SHOW REPLICA STATUS and SHOW SLAVE STATUS statements. Added in MySQL 8.0.22.
- `Com_show_replicas`: Count of SHOW REPLICAS and SHOW SLAVE HOSTS statements. Added in MySQL 8.0.22.
- `Com_uninstall_component`: Count of UINSTALL COMPONENT statements. Added in MySQL 8.0.0.
- `Compression_algorithm`: Compression algorithm for current connection. Added in MySQL 8.0.18.
- `Compression_level`: Compression level for current connection. Added in MySQL 8.0.18.
- `Connection_control_delay_generated`: How many times server delayed connection request. Added in MySQL 8.0.1.
- `Current_tls_ca`: Current value of ssl_ca system variable. Added in MySQL 8.0.16.
- `Current_tls_capath`: Current value of ssl_capath system variable. Added in MySQL 8.0.16.
- `Current_tls_cert`: Current value of ssl_cert system variable. Added in MySQL 8.0.16.
- `Current_tls_cipher`: Current value of ssl_cipher system variable. Added in MySQL 8.0.16.
- `Current_tls_ciphersuites`: Current value of tsl_ciphersuites system variable. Added in MySQL 8.0.16.
- `Current_tls_crl`: Current value of ssl_crl system variable. Added in MySQL 8.0.16.
- `Current_tls_crlpath`: Current value of ssl_crlpath system variable. Added in MySQL 8.0.16.
- `Current_tls_key`: Current value of ssl_key system variable. Added in MySQL 8.0.16.
- `Current_tls_version`: Current value of tls_version system variable. Added in MySQL 8.0.16.
- `Error_log_buffered_bytes`: Number of bytes used in error_log table. Added in MySQL 8.0.22.
- `Error_log_buffered_events`: Number of events in error_log table. Added in MySQL 8.0.22.
- `Error_log_expired_events`: Number of events discarded from error_log table. Added in MySQL 8.0.22.
- `Error_log_latest_write`: Time of last write to error_log table. Added in MySQL 8.0.22.

- `Firewall_access_denied`: Number of statements rejected by MySQL Enterprise Firewall. Added in MySQL 8.0.11.
- `Firewall_access_granted`: Number of statements accepted by MySQL Enterprise Firewall. Added in MySQL 8.0.11.
- `Firewall_cached_entries`: Number of statements recorded by MySQL Enterprise Firewall. Added in MySQL 8.0.11.
- `Global_connection_memory`: Amount of memory currently used by all user threads. Added in MySQL 8.0.28.
- `Innodb_buffer_pool_resize_status_code`: InnoDB buffer pool resize status code. Added in MySQL 8.0.31.
- `Innodb_buffer_pool_resize_status_progress`: InnoDB buffer pool resize status progress. Added in MySQL 8.0.31.
- `Innodb_redo_log_capacity_resized`: Redo log capacity after the last completed capacity resize operation. Added in MySQL 8.0.30.
- `Innodb_redo_log_checkpoint_lsn`: The redo log checkpoint LSN. Added in MySQL 8.0.30.
- `Innodb_redo_log_current_lsn`: The redo log current LSN. Added in MySQL 8.0.30.
- `Innodb_redo_log_enabled`: InnoDB redo log status. Added in MySQL 8.0.21.
- `Innodb_redo_log_flushed_to_disk_lsn`: The red log flushed-to-disk LSN. Added in MySQL 8.0.30.
- `Innodb_redo_log_logical_size`: The redo log logical size. Added in MySQL 8.0.30.
- `Innodb_redo_log_physical_size`: The redo log physical size. Added in MySQL 8.0.30.
- `Innodb_redo_log_read_only`: Whether the redo log is read-only. Added in MySQL 8.0.30.
- `Innodb_redo_log_resize_status`: The redo log resize status. Added in MySQL 8.0.30.
- `Innodb_redo_log_uuid`: The redo log UUID. Added in MySQL 8.0.30.
- `Innodb_system_rows_deleted`: Number of rows deleted from system schema tables. Added in MySQL 8.0.19.
- `Innodb_system_rows_inserted`: Number of rows inserted into system schema tables. Added in MySQL 8.0.19.
- `Innodb_system_rows_read`: Number of rows read from system schema tables. Added in MySQL 8.0.19.
- `Innodb_undo_tablespaces_active`: Number of active undo tablespaces. Added in MySQL 8.0.14.
- `Innodb_undo_tablespaces_explicit`: Number of user-created undo tablespaces. Added in MySQL 8.0.14.
- `Innodb_undo_tablespaces_implicit`: Number of undo tablespaces created by InnoDB. Added in MySQL 8.0.14.
- `Innodb_undo_tablespaces_total`: Total number of undo tablespaces. Added in MySQL 8.0.14.
- `Mysqlx_bytes_received_compressed_payload`: Number of bytes received as compressed message payloads, measured before decompression. Added in MySQL 8.0.19.
- `Mysqlx_bytes_received_uncompressed_frame`: Number of bytes received as compressed message payloads, measured after decompression. Added in MySQL 8.0.19.

- `Mysqlx_bytes_sent_compressed_payload`: Number of bytes sent as compressed message payloads, measured after compression. Added in MySQL 8.0.19.
- `Mysqlx_bytes_sent_uncompressed_frame`: Number of bytes sent as compressed message payloads, measured before compression. Added in MySQL 8.0.19.
- `Mysqlx_compression_algorithm`: Compression algorithm in use for X Protocol connection for this session. Added in MySQL 8.0.20.
- `Mysqlx_compression_level`: Compression level in use for X Protocol connection for this session. Added in MySQL 8.0.20.
- `Replica_open_temp_tables`: Number of temporary tables that replication SQL thread currently has open. Added in MySQL 8.0.26.
- `Replica_rows_last_search_algorithm_used`: Search algorithm most recently used by this replica to locate rows for row-based replication (index, table, or hash scan). Added in MySQL 8.0.26.
- `Resource_group_supported`: Whether server supports the resource group feature. Added in MySQL 8.0.31.
- `Rpl_semi_sync_replica_status`: Whether semisynchronous replication is operational on replica. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_clients`: Number of semisynchronous replicas. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_net_avg_wait_time`: Average time source has waited for replies from replica. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_net_wait_time`: Total time source has waited for replies from replica. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_net_waits`: Total number of times source waited for replies from replica. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_no_times`: Number of times source turned off semisynchronous replication. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_no_tx`: Number of commits not acknowledged successfully. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_status`: Whether semisynchronous replication is operational on source. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_timefunc_failures`: Number of times source failed when calling time functions. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_tx_avg_wait_time`: Average time source waited for each transaction. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_tx_wait_time`: Total time source waited for transactions. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_tx_waits`: Total number of times source waited for transactions. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_wait_pos_backtraverse`: Total number of times source has waited for event with binary coordinates lower than events waited for previously. Added in MySQL 8.0.26.
- `Rpl_semi_sync_source_wait_sessions`: Number of sessions currently waiting for replica replies. Added in MySQL 8.0.26.

- `Rpl_semi_sync_source_yes_tx`: Number of commits acknowledged successfully. Added in MySQL 8.0.26.
- `Secondary_engine_execution_count`: Number of queries offloaded to a secondary engine.. Added in MySQL 8.0.13.
- `Ssl_session_cache_timeout`: Current SSL session timeout value in cache. Added in MySQL 8.0.29.
- `Telemetry_traces_supported`: Whether server telemetry traces is supported.. Added in MySQL 8.0.33.
- `Tls_library_version`: Runtime version of OpenSSL library in use. Added in MySQL 8.0.30.
- `activate_all_roles_on_login`: Whether to activate all user roles at connect time. Added in MySQL 8.0.2.
- `admin-ssl`: Enable connection encryption. Added in MySQL 8.0.21.
- `admin_address`: IP address to bind to for connections on administrative interface. Added in MySQL 8.0.14.
- `admin_port`: TCP/IP number to use for connections on administrative interface. Added in MySQL 8.0.14.
- `admin_ssl_ca`: File that contains list of trusted SSL Certificate Authorities. Added in MySQL 8.0.21.
- `admin_ssl_capath`: Directory that contains trusted SSL Certificate Authority certificate files. Added in MySQL 8.0.21.
- `admin_ssl_cert`: File that contains X.509 certificate. Added in MySQL 8.0.21.
- `admin_ssl_cipher`: Permissible ciphers for connection encryption. Added in MySQL 8.0.21.
- `admin_ssl_crl`: File that contains certificate revocation lists. Added in MySQL 8.0.21.
- `admin_ssl_crlpath`: Directory that contains certificate revocation list files. Added in MySQL 8.0.21.
- `admin_ssl_key`: File that contains X.509 key. Added in MySQL 8.0.21.
- `admin_tls_ciphersuites`: Permissible TLSv1.3 ciphersuites for encrypted connections. Added in MySQL 8.0.21.
- `admin_tls_version`: Permissible TLS protocols for encrypted connections. Added in MySQL 8.0.21.
- `audit-log`: Whether to activate audit log plugin. Added in MySQL 8.0.11.
- `audit_log_buffer_size`: Size of audit log buffer. Added in MySQL 8.0.11.
- `audit_log_compression`: Audit log file compression method. Added in MySQL 8.0.11.
- `audit_log_connection_policy`: Audit logging policy for connection-related events. Added in MySQL 8.0.11.
- `audit_log_current_session`: Whether to audit current session. Added in MySQL 8.0.11.
- `audit_log_disable`: Whether to disable the audit log. Added in MySQL 8.0.28.
- `audit_log_encryption`: Audit log file encryption method. Added in MySQL 8.0.11.
- `audit_log_exclude_accounts`: Accounts not to audit. Added in MySQL 8.0.11.
- `audit_log_file`: Name of audit log file. Added in MySQL 8.0.11.

- audit_log_filter_id: ID of current audit log filter. Added in MySQL 8.0.11.
- audit_log_flush: Close and reopen audit log file. Added in MySQL 8.0.11.
- audit_log_format: Audit log file format. Added in MySQL 8.0.11.
- audit_log_format_unix_timestamp: Whether to include Unix timestamp in JSON-format audit log. Added in MySQL 8.0.26.
- audit_log_include_accounts: Accounts to audit. Added in MySQL 8.0.11.
- audit_log_max_size: Limit on combined size of JSON audit log files. Added in MySQL 8.0.26.
- audit_log_password_history_keep_days: Number of days to retain archived audit log encryption passwords. Added in MySQL 8.0.17.
- audit_log_policy: Audit logging policy. Added in MySQL 8.0.11.
- audit_log_prune_seconds: The number of seconds after which audit log files become subject to pruning. Added in MySQL 8.0.24.
- audit_log_read_buffer_size: Audit log file read buffer size. Added in MySQL 8.0.11.
- audit_log_rotate_on_size: Close and reopen audit log file at this size. Added in MySQL 8.0.11.
- audit_log_statement_policy: Audit logging policy for statement-related events. Added in MySQL 8.0.11.
- audit_log_strategy: Audit logging strategy. Added in MySQL 8.0.11.
- authentication_fido_rp_id: Relying party ID for FIDO multifactor authentication. Added in MySQL 8.0.27.
- authentication_kerberos_service_key_tab: File containing Kerberos service keys to authenticate TGS ticket. Added in MySQL 8.0.26.
- authentication_kerberos_service_principal: Kerberos service principal name. Added in MySQL 8.0.26.
- authentication_ldap_sasl_auth_method_name: Authentication method name. Added in MySQL 8.0.11.
- authentication_ldap_sasl_bind_base_dn: LDAP server base distinguished name. Added in MySQL 8.0.11.
- authentication_ldap_sasl_bind_root_dn: LDAP server root distinguished name. Added in MySQL 8.0.11.
- authentication_ldap_sasl_bind_root_pwd: LDAP server root bind password. Added in MySQL 8.0.11.
- authentication_ldap_sasl_ca_path: LDAP server certificate authority file name. Added in MySQL 8.0.11.
- authentication_ldap_sasl_group_search_attr: LDAP server group search attribute. Added in MySQL 8.0.11.
- authentication_ldap_sasl_group_search_filter: LDAP custom group search filter. Added in MySQL 8.0.11.
- authentication_ldap_sasl_init_pool_size: LDAP server initial connection pool size. Added in MySQL 8.0.11.
- authentication_ldap_sasl_log_status: LDAP server log level. Added in MySQL 8.0.11.

- `authentication_ldap_sasl_max_pool_size`: LDAP server maximum connection pool size. Added in MySQL 8.0.11.
- `authentication_ldap_sasl_referral`: Whether to enable LDAP search referral. Added in MySQL 8.0.20.
- `authentication_ldap_sasl_server_host`: LDAP server host name or IP address. Added in MySQL 8.0.11.
- `authentication_ldap_sasl_server_port`: LDAP server port number. Added in MySQL 8.0.11.
- `authentication_ldap_sasl_tls`: Whether to use encrypted connections to LDAP server. Added in MySQL 8.0.11.
- `authentication_ldap_sasl_user_search_attr`: LDAP server user search attribute. Added in MySQL 8.0.11.
- `authentication_ldap_simple_auth_method_name`: Authentication method name. Added in MySQL 8.0.11.
- `authentication_ldap_simple_bind_base_dn`: LDAP server base distinguished name. Added in MySQL 8.0.11.
- `authentication_ldap_simple_bind_root_dn`: LDAP server root distinguished name. Added in MySQL 8.0.11.
- `authentication_ldap_simple_bind_root_pwd`: LDAP server root bind password. Added in MySQL 8.0.11.
- `authentication_ldap_simple_ca_path`: LDAP server certificate authority file name. Added in MySQL 8.0.11.
- `authentication_ldap_simple_group_search_attr`: LDAP server group search attribute. Added in MySQL 8.0.11.
- `authentication_ldap_simple_group_search_filter`: LDAP custom group search filter. Added in MySQL 8.0.11.
- `authentication_ldap_simple_init_pool_size`: LDAP server initial connection pool size. Added in MySQL 8.0.11.
- `authentication_ldap_simple_log_status`: LDAP server log level. Added in MySQL 8.0.11.
- `authentication_ldap_simple_max_pool_size`: LDAP server maximum connection pool size. Added in MySQL 8.0.11.
- `authentication_ldap_simple_referral`: Whether to enable LDAP search referral. Added in MySQL 8.0.20.
- `authentication_ldap_simple_server_host`: LDAP server host name or IP address. Added in MySQL 8.0.11.
- `authentication_ldap_simple_server_port`: LDAP server port number. Added in MySQL 8.0.11.
- `authentication_ldap_simple_tls`: Whether to use encrypted connections to LDAP server. Added in MySQL 8.0.11.
- `authentication_ldap_simple_user_search_attr`: LDAP server user search attribute. Added in MySQL 8.0.11.
- `authentication_policy`: Plugins for multifactor authentication.. Added in MySQL 8.0.27.

- `authentication_windows_log_level`: Windows authentication plugin logging level. Added in MySQL 8.0.11.
- `authentication_windows_use_principal_name`: Whether to use Windows authentication plugin principal name. Added in MySQL 8.0.11.
- `binlog_encryption`: Enable encryption for binary log files and relay log files on this server. Added in MySQL 8.0.14.
- `binlog_expire_logs_auto_purge`: Controls automatic purging of binary log files; can be overridden when enabled, by setting both binlog_expire_logs_seconds and expire_logs_days to 0. Added in MySQL 8.0.29.
- `binlog_expire_logs_seconds`: Purge binary logs after this many seconds. Added in MySQL 8.0.1.
- `binlog_rotate_encryption_master_key_at_startup`: Rotate binary log master key at server startup. Added in MySQL 8.0.14.
- `binlog_row_metadata`: Whether to record all or only minimal table related metadata to binary log when using row-based logging. Added in MySQL 8.0.1.
- `binlog_row_value_options`: Enables binary logging of partial JSON updates for row-based replication. Added in MySQL 8.0.3.
- `binlog_transaction_compression`: Enable compression for transaction payloads in binary log files. Added in MySQL 8.0.20.
- `binlog_transaction_compression_level_zstd`: Compression level for transaction payloads in binary log files. Added in MySQL 8.0.20.
- `binlog_transaction_dependency_history_size`: Number of row hashes kept for looking up transaction that last updated some row. Added in MySQL 8.0.1.
- `binlog_transaction_dependency_tracking`: Source of dependency information (commit timestamps or transaction write sets) from which to assess which transactions can be executed in parallel by replica's multithreaded applier. Added in MySQL 8.0.1.
- `build_id`: A unique build ID generated at compile time (Linux only). Added in MySQL 8.0.31.
- `caching_sha2_password_auto_generate_rsa_keys`: Whether to autogenerate RSA key-pair files. Added in MySQL 8.0.4.
- `caching_sha2_password_digest_rounds`: Number of hash rounds for caching_sha2_password authentication plugin. Added in MySQL 8.0.24.
- `caching_sha2_password_private_key_path`: SHA2 authentication plugin private key path name. Added in MySQL 8.0.3.
- `caching_sha2_password_public_key_path`: SHA2 authentication plugin public key path name. Added in MySQL 8.0.3.
- `clone_autotune_concurrency`: Enables dynamic spawning of threads for remote cloning operations. Added in MySQL 8.0.17.
- `clone_block_ddl`: Enables an exclusive backup lock during clone operations. Added in MySQL 8.0.27.
- `clone_buffer_size`: Defines size of intermediate buffer on donor MySQL server instance. Added in MySQL 8.0.17.
- `clone_ddl_timeout`: Number of seconds cloning operation waits for backup lock. Added in MySQL 8.0.17.

- `clone_delay_after_data_drop`: The time delay in seconds before the clone process starts. Added in MySQL 8.0.29.
- `clone_donor_timeout_after_network_failure`: The time allowed to restart a cloning operation after a network failure. Added in MySQL 8.0.24.
- `clone_enable_compression`: Enables compression of data at network layer during cloning. Added in MySQL 8.0.17.
- `clone_max_concurrency`: Maximum number of concurrent threads used to perform cloning operation. Added in MySQL 8.0.17.
- `clone_max_data_bandwidth`: Maximum data transfer rate in MiB per second for remote cloning operation. Added in MySQL 8.0.17.
- `clone_max_network_bandwidth`: Maximum network transfer rate in MiB per second for remote cloning operation. Added in MySQL 8.0.17.
- `clone_ssl_ca`: Specifies path to certificate authority (CA) file. Added in MySQL 8.0.14.
- `clone_ssl_cert`: Specifies path to public key certificate file. Added in MySQL 8.0.14.
- `clone_ssl_key`: Specifies path to private key file. Added in MySQL 8.0.14.
- `clone_valid_donor_list`: Defines donor host addresses for remote cloning operations. Added in MySQL 8.0.17.
- `connection_control_failed_connections_threshold`: Consecutive failed connection attempts before delays occur. Added in MySQL 8.0.1.
- `connection_control_max_connection_delay`: Maximum delay (milliseconds) for server response to failed connection attempts. Added in MySQL 8.0.1.
- `connection_control_min_connection_delay`: Minimum delay (milliseconds) for server response to failed connection attempts. Added in MySQL 8.0.1.
- `connection_memory_chunk_size`: Update Global_connection_memory only when user memory usage changes by this amount or more; 0 disables updating. Added in MySQL 8.0.28.
- `connection_memory_limit`: Maximum amount of memory that can be consumed by any one user connection before additional queries by that user are rejected. Does not apply to system users such as MySQL root. Added in MySQL 8.0.28.
- `create_admin_listener_thread`: Whether to use dedicated listening thread for connections on administrative interface. Added in MySQL 8.0.14.
- `cte_max_recursion_depth`: Common table expression maximum recursion depth. Added in MySQL 8.0.3.
- `ddl-rewriter`: Whether to activate ddl_rewriter plugin. Added in MySQL 8.0.16.
- `default_collation_for_utf8mb4`: Default collation for utf8mb4 character set; for internal use by MySQL Replication only. Added in MySQL 8.0.11.
- `default_table_encryption`: Default schema and tablespace encryption setting. Added in MySQL 8.0.16.
- `dragnet.Status`: Result of most recent assignment to dragnet.log_error_filter_rules. Added in MySQL 8.0.12.
- `dragnet.log_error_filter_rules`: Filter rules for error logging. Added in MySQL 8.0.4.
- `early-plugin-load`: Specify plugins to load before loading mandatory built-in plugins and before storage engine initialization. Added in MySQL 8.0.0.

- `enterprise_encryption.maximum_rsa_key_size`: Maximum size of RSA keys generated by MySQL Enterprise Encryption. Added in MySQL 8.0.30.
- `enterprise_encryption.rsa_support_legacy_padding`: Decrypt and verify legacy MySQL Enterprise Encryption content. Added in MySQL 8.0.30.
- `explain_format`: Determines default output format used by EXPLAIN statements. Added in MySQL 8.0.32.
- `generated_random_password_length`: Maximum length of generated passwords. Added in MySQL 8.0.18.
- `global_connection_memory_limit`: Maximum total amount of memory that can be consumed by all user connections. While Global_connection_memory exceeds this amount, any new queries from regular users are rejected. Does not apply to system users such as MySQL root. Added in MySQL 8.0.28.
- `global_connection_memory_tracking`: Whether or not to calculate global connection memory usage (as shown by Global_connection_memory); default is disabled. Added in MySQL 8.0.28.
- `group_replication_advertise_recovery_endpoints`: Connections offered for distributed recovery. Added in MySQL 8.0.21.
- `group_replication_autorejoin_tries`: Number of tries that member makes to rejoin group automatically. Added in MySQL 8.0.16.
- `group_replication_clone_threshold`: Transaction number gap between donor and recipient above which remote cloning operation is used for state transfer. Added in MySQL 8.0.17.
- `group_replication_communication_debug_options`: Level of debugging messages for Group Replication components. Added in MySQL 8.0.3.
- `group_replication_communication_max_message_size`: Maximum message size for Group Replication communications, larger messages are fragmented. Added in MySQL 8.0.16.
- `group_replication_communication_stack`: Specifies whether the XCom communication stack or the MySQL communication stack is to be used to establish group communication connections between members.. Added in MySQL 8.0.27.
- `group_replication_consistency`: Type of transaction consistency guarantee which group provides. Added in MySQL 8.0.14.
- `group_replication_exit_state_action`: How instance behaves when it leaves group involuntarily. Added in MySQL 8.0.12.
- `group_replication_flow_control_hold_percent`: Percentage of group quota to remain unused. Added in MySQL 8.0.2.
- `group_replication_flow_control_max_quota`: Maximum flow control quota for group. Added in MySQL 8.0.2.
- `group_replication_flow_control_member_quota_percent`: Percentage of quota which member should assume is available for itself when calculating quotas. Added in MySQL 8.0.2.
- `group_replication_flow_control_min_quota`: Lowest flow control quota which can be assigned per member. Added in MySQL 8.0.2.
- `group_replication_flow_control_min_recovery_quota`: Lowest quota which can be assigned per member because another group member is recovering. Added in MySQL 8.0.2.
- `group_replication_flow_control_period`: Defines how many seconds to wait between flow control iterations. Added in MySQL 8.0.2.

- `group_replication_flow_control_release_percent`: How group quota should be released when flow control no longer needs to throttle writer members. Added in MySQL 8.0.2.
- `group_replication_ip_allowlist`: List of hosts permitted to connect to group (MySQL 8.0.22 and later). Added in MySQL 8.0.22.
- `group_replication_member_expel_timeout`: Time between suspected failure of group member and expelling it from group, causing group membership reconfiguration. Added in MySQL 8.0.13.
- `group_replication_member_weight`: Chance of this member being elected as primary. Added in MySQL 8.0.2.
- `group_replication_message_cache_size`: Maximum memory for group communication engine message cache (XCom). Added in MySQL 8.0.16.
- `group_replication_paxos_single_leader`: Use a single consensus leader in single-primary mode. Added in MySQL 8.0.27.
- `group_replication_recovery_compression_algorithms`: Permitted compression algorithms for outgoing recovery connections. Added in MySQL 8.0.18.
- `group_replication_recovery_get_public_key`: Whether to accept preference about fetching public key from donor. Added in MySQL 8.0.4.
- `group_replication_recovery_public_key_path`: To accept public key information. Added in MySQL 8.0.4.
- `group_replication_recovery_tls_ciphersuites`: Permitted cipher suites when TLSv1.3 is used for connection encryption with this instance as client (joining member). Added in MySQL 8.0.19.
- `group_replication_recovery_tls_version`: Permitted TLS protocols for connection encryption as client (joining member). Added in MySQL 8.0.19.
- `group_replication_recovery_zstd_compression_level`: Compression level for recovery connections that use zstd compression. Added in MySQL 8.0.18.
- `group_replication_tls_source`: Source of TLS material for Group Replication. Added in MySQL 8.0.21.
- `group_replication_unreachable_majority_timeout`: How long to wait for network partitions that result in minority to leave group. Added in MySQL 8.0.2.
- `group_replication_view_change_uuid`: UUID for view change event GTIDs. Added in MySQL 8.0.26.
- `histogram_generation_max_mem_size`: Maximum memory for creating histogram statistics. Added in MySQL 8.0.2.
- `immediate_server_version`: MySQL Server release number of server which is immediate replication source. Added in MySQL 8.0.14.
- `information_schema_stats_expiry`: Expiration setting for cached table statistics. Added in MySQL 8.0.3.
- `init_replica`: Statements that are executed when replica connects to source. Added in MySQL 8.0.26.
- `innodb_buffer_pool_debug`: Permits multiple buffer pool instances when buffer pool is less than 1GB in size. Added in MySQL 8.0.0.
- `innodb_buffer_pool_in_core_file`: Controls writing of buffer pool pages to core files. Added in MySQL 8.0.14.

- `innodb_checkpoint_disabled`: Disables checkpoints so that deliberate server exit always initiates recovery. Added in MySQL 8.0.2.
- `innodb_ddl_buffer_size`: The maximum buffer size for DDL operations. Added in MySQL 8.0.27.
- `innodb_ddl_log_crash_reset_debug`: Debug option that resets DDL log crash injection counters. Added in MySQL 8.0.3.
- `innodb_ddl_threads`: The maximum number of parallel threads for index creation. Added in MySQL 8.0.27.
- `innodb_deadlock_detect`: Enables or disables deadlock detection. Added in MySQL 8.0.0.
- `innodb_dedicated_server`: Enables automatic configuration of buffer pool size, log file size, and flush method. Added in MySQL 8.0.3.
- `innodb_directories`: Defines directories to scan at startup for tablespace data files. Added in MySQL 8.0.4.
- `innodb_doublewrite_batch_size`: Number of doublewrite pages to write per batch. Added in MySQL 8.0.20.
- `innodb_doublewrite_dir`: Doublewrite buffer file directory. Added in MySQL 8.0.20.
- `innodb_doublewrite_files`: Number of doublewrite files. Added in MySQL 8.0.20.
- `innodb_doublewrite_pages`: Number of doublewrite pages per thread. Added in MySQL 8.0.20.
- `innodb_extend_and_initialize`: Controls how new tablespace pages are allocated on Linux. Added in MySQL 8.0.22.
- `innodb_fsync_threshold`: Controls how often InnoDB calls fsync when creating new file. Added in MySQL 8.0.13.
- `innodb_idle_flush_pct`: Limits I/0 operations when InnoDB is idle. Added in MySQL 8.0.18.
- `innodb_log_checkpoint_fuzzy_now`: Debug option that forces InnoDB to write fuzzy checkpoint. Added in MySQL 8.0.13.
- `innodb_log_spin_cpu_abs_lwm`: Minimum amount of CPU usage below which user threads no longer spin while waiting for flushed redo. Added in MySQL 8.0.11.
- `innodb_log_spin_cpu_pct_hwm`: Maximum amount of CPU usage above which user threads no longer spin while waiting for flushed redo. Added in MySQL 8.0.11.
- `innodb_log_wait_for_flush_spin_hwm`: Maximum average log flush time beyond which user threads no longer spin while waiting for flushed redo. Added in MySQL 8.0.11.
- `innodb_log_writer_threads`: Enables dedicated log writer threads for writing and flushing redo logs. Added in MySQL 8.0.22.
- `innodb_parallel_read_threads`: Number of threads for parallel index reads. Added in MySQL 8.0.14.
- `innodb_print_ddl_logs`: Whether or not to print DDL logs to error log. Added in MySQL 8.0.3.
- `innodb_redo_log_archive_dirs`: Labeled redo log archive directories. Added in MySQL 8.0.17.
- `innodb_redo_log_capacity`: The size limit for redo log files. Added in MySQL 8.0.30.
- `innodb_redo_log_encrypt`: Controls encryption of redo log data for encrypted tablespaces. Added in MySQL 8.0.1.

- `innodb_scan_directories`: Defines directories to scan for tablespace files during InnoDB recovery. Added in MySQL 8.0.2.
- `innodb_segment_reserve_factor`: The percentage of tablespace file segment pages reserved as empty pages. Added in MySQL 8.0.26.
- `innodb_spin_wait_pause_multiplier`: Multiplier value used to determine number of PAUSE instructions in spin-wait loops. Added in MySQL 8.0.16.
- `innodb_stats_include_delete_marked`: Include delete-marked records when calculating persistent InnoDB statistics. Added in MySQL 8.0.1.
- `innodb_temp_tablespaces_dir`: Session temporary tablespaces path. Added in MySQL 8.0.13.
- `innodb_tmpdir`: Directory location for temporary table files created during online ALTER TABLE operations. Added in MySQL 8.0.0.
- `innodb_undo_log_encrypt`: Controls encryption of undo log data for encrypted tablespaces. Added in MySQL 8.0.1.
- `innodb_use_fdatasync`: Whether InnoDB uses fdatasync() instead of fsync() when flushing data to the operating system. Added in MySQL 8.0.26.
- `innodb_validate_tablespace_paths`: Enables tablespace path validation at startup. Added in MySQL 8.0.21.
- `internal_tmp_mem_storage_engine`: Storage engine to use for internal in-memory temporary tables. Added in MySQL 8.0.2.
- `keyring-migration-destination`: Key migration destination keyring plugin. Added in MySQL 8.0.4.
- `keyring-migration-host`: Host name for connecting to running server for key migration. Added in MySQL 8.0.4.
- `keyring-migration-password`: Password for connecting to running server for key migration. Added in MySQL 8.0.4.
- `keyring-migration-port`: TCP/IP port number for connecting to running server for key migration. Added in MySQL 8.0.4.
- `keyring-migration-socket`: Unix socket file or Windows named pipe for connecting to running server for key migration. Added in MySQL 8.0.4.
- `keyring-migration-source`: Key migration source keyring plugin. Added in MySQL 8.0.4.
- `keyring-migration-to-component`: Keyring migration is from plugin to component. Added in MySQL 8.0.24.
- `keyring-migration-user`: User name for connecting to running server for key migration. Added in MySQL 8.0.4.
- `keyring_aws_cmk_id`: AWS keyring plugin customer master key ID value. Added in MySQL 8.0.11.
- `keyring_aws_conf_file`: AWS keyring plugin configuration file location. Added in MySQL 8.0.11.
- `keyring_aws_data_file`: AWS keyring plugin storage file location. Added in MySQL 8.0.11.
- `keyring_aws_region`: AWS keyring plugin region. Added in MySQL 8.0.11.
- `keyring_encrypted_file_data`: keyring_encrypted_file plugin data file. Added in MySQL 8.0.11.

- keyring_encrypted_file_password: keyring_encrypted_file plugin password. Added in MySQL 8.0.11.
- keyring_hashicorp_auth_path: HashiCorp Vault AppRole authentication path. Added in MySQL 8.0.18.
- keyring_hashicorp_ca_path: Path to keyring_hashicorp CA file. Added in MySQL 8.0.18.
- keyring_hashicorp_caching: Whether to enable keyring_hashicorp caching. Added in MySQL 8.0.18.
- keyring_hashicorp_commit_auth_path: keyring_hashicorp_auth_path value in use. Added in MySQL 8.0.18.
- keyring_hashicorp_commit_ca_path: keyring_hashicorp_ca_path value in use. Added in MySQL 8.0.18.
- keyring_hashicorp_commit_caching: keyring_hashicorp_caching value in use. Added in MySQL 8.0.18.
- keyring_hashicorp_commit_role_id: keyring_hashicorp_role_id value in use. Added in MySQL 8.0.18.
- keyring_hashicorp_commit_server_url: keyring_hashicorp_server_url value in use. Added in MySQL 8.0.18.
- keyring_hashicorp_commit_store_path: keyring_hashicorp_store_path value in use. Added in MySQL 8.0.18.
- keyring_hashicorp_role_id: HashiCorp Vault AppRole authentication role ID. Added in MySQL 8.0.18.
- keyring_hashicorp_secret_id: HashiCorp Vault AppRole authentication secret ID. Added in MySQL 8.0.18.
- keyring_hashicorp_server_url: HashiCorp Vault server URL. Added in MySQL 8.0.18.
- keyring_hashicorp_store_path: HashiCorp Vault store path. Added in MySQL 8.0.18.
- keyring_oci_ca_certificate: CA certificate file for peer authentication. Added in MySQL 8.0.22.
- keyring_oci_compartment: OCI compartment OCID. Added in MySQL 8.0.22.
- keyring_oci_encryption_endpoint: OCI encryption server endpoint. Added in MySQL 8.0.22.
- keyring_oci_key_file: OCI RSA private key file. Added in MySQL 8.0.22.
- keyring_oci_key_fingerprint: OCI RSA private key file fingerprint. Added in MySQL 8.0.22.
- keyring_oci_management_endpoint: OCI management server endpoint. Added in MySQL 8.0.22.
- keyring_oci_master_key: OCI master key OCID. Added in MySQL 8.0.22.
- keyring_oci_secrets_endpoint: OCI secrets server endpoint. Added in MySQL 8.0.22.
- keyring_oci_tenancy: OCI tenancy OCID. Added in MySQL 8.0.22.
- keyring_oci_user: OCI user OCID. Added in MySQL 8.0.22.
- keyring_oci_vaults_endpoint: OCI vaults server endpoint. Added in MySQL 8.0.22.
- keyring_oci_virtual_vault: OCI vault OCID. Added in MySQL 8.0.22.

- `keyring_okv_conf_dir`: Oracle Key Vault keyring plugin configuration directory. Added in MySQL 8.0.11.
- `keyring_operations`: Whether keyring operations are enabled. Added in MySQL 8.0.4.
- `lock_order`: Whether to enable LOCK_ORDER tool at runtime. Added in MySQL 8.0.17.
- `lock_order_debug_loop`: Whether to cause debug assert when LOCK_ORDER tool encounters dependency flagged as loop. Added in MySQL 8.0.17.
- `lock_order_debug_missing_arc`: Whether to cause debug assert when LOCK_ORDER tool encounters undeclared dependency. Added in MySQL 8.0.17.
- `lock_order_debug_missing_key`: Whether to cause debug assert when LOCK_ORDER tool encounters object not properly instrumented with Performance Schema. Added in MySQL 8.0.17.
- `lock_order_debug_missing_unlock`: Whether to cause debug assert when LOCK_ORDER tool encounters lock that is destroyed while still held. Added in MySQL 8.0.17.
- `lock_order_dependencies`: Path to lock_order_dependencies.txt file. Added in MySQL 8.0.17.
- `lock_order_extra_dependencies`: Path to second dependency file. Added in MySQL 8.0.17.
- `lock_order_output_directory`: Directory where LOCK_ORDER tool writes logs. Added in MySQL 8.0.17.
- `lock_order_print_txt`: Whether to perform lock-order graph analysis and print textual report. Added in MySQL 8.0.17.
- `lock_order_trace_loop`: Whether to print log file trace when LOCK_ORDER tool encounters dependency flagged as loop. Added in MySQL 8.0.17.
- `lock_order_trace_missing_arc`: Whether to print log file trace when LOCK_ORDER tool encounters undeclared dependency. Added in MySQL 8.0.17.
- `lock_order_trace_missing_key`: Whether to print log file trace when LOCK_ORDER tool encounters object not properly instrumented with Performance Schema. Added in MySQL 8.0.17.
- `lock_order_trace_missing_unlock`: Whether to print log file trace when LOCK_ORDER tool encounters lock that is destroyed while still held. Added in MySQL 8.0.17.
- `log_error_filter_rules`: Filter rules for error logging. Added in MySQL 8.0.2.
- `log_error_services`: Components to use for error logging. Added in MySQL 8.0.2.
- `log_error_suppression_list`: Warning/information error log messages to suppress. Added in MySQL 8.0.13.
- `log_replica_updates`: Whether replica should log updates performed by its replication SQL thread to its own binary log. Added in MySQL 8.0.26.
- `log_slow_extra`: Whether to write extra information to slow query log file. Added in MySQL 8.0.14.
- `log_slow_replica_statements`: Cause slow statements as executed by replica to be written to slow query log. Added in MySQL 8.0.26.
- `mandatory_roles`: Automatically granted roles for all users. Added in MySQL 8.0.2.
- `mysql_firewall_mode`: Whether MySQL Enterprise Firewall is operational. Added in MySQL 8.0.11.
- `mysql_firewall_trace`: Whether to enable firewall trace. Added in MySQL 8.0.11.
- `mysqlx`: Whether X Plugin is initialized. Added in MySQL 8.0.11.

- `mysqlx_compression_algorithms`: Compression algorithms permitted for X Protocol connections. Added in MySQL 8.0.19.
- `mysqlx_deflate_default_compression_level`: Default compression level for Deflate algorithm on X Protocol connections. Added in MySQL 8.0.20.
- `mysqlx_deflate_max_client_compression_level`: Maximum permitted compression level for Deflate algorithm on X Protocol connections. Added in MySQL 8.0.20.
- `mysqlx_interactive_timeout`: Number of seconds to wait for interactive clients to time out. Added in MySQL 8.0.4.
- `mysqlx_lz4_default_compression_level`: Default compression level for LZ4 algorithm on X Protocol connections. Added in MySQL 8.0.20.
- `mysqlx_lz4_max_client_compression_level`: Maximum permitted compression level for LZ4 algorithm on X Protocol connections. Added in MySQL 8.0.20.
- `mysqlx_read_timeout`: Number of seconds to wait for blocking read operations to complete. Added in MySQL 8.0.4.
- `mysqlx_wait_timeout`: Number of seconds to wait for activity from connection. Added in MySQL 8.0.4.
- `mysqlx_write_timeout`: Number of seconds to wait for blocking write operations to complete. Added in MySQL 8.0.4.
- `mysqlx_zstd_default_compression_level`: Default compression level for zstd algorithm on X Protocol connections. Added in MySQL 8.0.20.
- `mysqlx_zstd_max_client_compression_level`: Maximum permitted compression level for zstd algorithm on X Protocol connections. Added in MySQL 8.0.20.
- `named_pipe_full_access_group`: Name of Windows group granted full access to named pipe. Added in MySQL 8.0.14.
- `no-dd-upgrade`: Prevent automatic upgrade of data dictionary tables at startup. Added in MySQL 8.0.4.
- `no-monitor`: Do not fork monitor process required for RESTART. Added in MySQL 8.0.12.
- `original_commit_timestamp`: Time when transaction was committed on original source. Added in MySQL 8.0.1.
- `original_server_version`: MySQL Server release number of server on which transaction was originally committed. Added in MySQL 8.0.14.
- `partial_revokes`: Whether partial revocation is enabled. Added in MySQL 8.0.16.
- `password_history`: Number of password changes required before password reuse. Added in MySQL 8.0.3.
- `password_require_current`: Whether password changes require current password verification. Added in MySQL 8.0.13.
- `password_reuse_interval`: Number of days elapsed required before password reuse. Added in MySQL 8.0.3.
- `performance-schema-consumer-events-statements-cpu`: Configure statement CPU-usage consumer. Added in MySQL 8.0.28.
- `performance_schema_max_digest_sample_age`: Query resample age in seconds. Added in MySQL 8.0.3.

- performance_schema_show_processlist: Select SHOW PROCESSLIST implementation. Added in MySQL 8.0.22.
- persist_only_admin_x509_subject: SSL certificate X.509 Subject that enables persisting persist-restricted system variables. Added in MySQL 8.0.14.
- persist_sensitive_variables_in_plaintext: Whether the server is permitted to store the values of sensitive system variables in an unencrypted format. Added in MySQL 8.0.29.
- persisted_globals_load: Whether to load persisted configuration settings. Added in MySQL 8.0.0.
- print_identified_with_as_hex: For SHOW CREATE USER, print hash values containing unprintable characters in hex. Added in MySQL 8.0.17.
- protocol_compression_algorithms: Permitted compression algorithms for incoming connections. Added in MySQL 8.0.18.
- pseudo_replica_mode: For internal server use. Added in MySQL 8.0.26.
- regexp_stack_limit: Regular expression match stack size limit. Added in MySQL 8.0.4.
- regexp_time_limit: Regular expression match timeout. Added in MySQL 8.0.4.
- replica_checkpoint_group: Maximum number of transactions processed by multithreaded replica before checkpoint operation is called to update progress status. Not supported by NDB Cluster. Added in MySQL 8.0.26.
- replica_checkpoint_period: Update progress status of multithreaded replica and flush relay log info to disk after this number of milliseconds. Not supported by NDB Cluster. Added in MySQL 8.0.26.
- replica_compressed_protocol: Use compression of source/replica protocol. Added in MySQL 8.0.26.
- replica_exec_mode: Allows for switching replication thread between IDEMPOTENT mode (key and some other errors suppressed) and STRICT mode; STRICT mode is default, except for NDB Cluster, where IDEMPOTENT is always used. Added in MySQL 8.0.26.
- replica_load_tmpdir: Location where replica should put its temporary files when replicating LOAD DATA statements. Added in MySQL 8.0.26.
- replica_max_allowed_packet: Maximum size, in bytes, of packet that can be sent from replication source server to replica; overrides max_allowed_packet. Added in MySQL 8.0.26.
- replica_net_timeout: Number of seconds to wait for more data from source/replica connection before aborting read. Added in MySQL 8.0.26.
- replica_parallel_type: Tells replica to use timestamp information (LOGICAL_CLOCK) or database partitioning (DATABASE) to parallelize transactions. Added in MySQL 8.0.26.
- replica_parallel_workers: Number of applier threads for executing replication transactions in parallel; 0 or 1 disables replica multithreading. NDB Cluster: see documentation. Added in MySQL 8.0.26.
- replica_pending_jobs_size_max: Maximum size of replica worker queues holding events not yet applied. Added in MySQL 8.0.26.
- replica_preserve_commit_order: Ensures that all commits by replica workers happen in same order as on source to maintain consistency when using parallel applier threads. Added in MySQL 8.0.26.
- replica_skip_errors: Tells replication thread to continue replication when query returns error from provided list. Added in MySQL 8.0.26.

- `replica_sql_verify_checksum`: Cause replica to examine checksums when reading from relay log. Added in MySQL 8.0.26.
- `replica_transaction_retries`: Number of times replication SQL thread retries transaction in case it failed with deadlock or elapsed lock wait timeout, before giving up and stopping. Added in MySQL 8.0.26.
- `replica_type_conversions`: Controls type conversion mode on replica. Value is list of zero or more elements from this list: ALL_LOSSY, ALL_NON_LOSSY. Set to empty string to disallow type conversions between source and replica. Added in MySQL 8.0.26.
- `replication_optimize_for_static_plugin_config`: Shared locks for semisynchronous replication. Added in MySQL 8.0.23.
- `replication_sender_observe_commit_only`: Limited callbacks for semisynchronous replication. Added in MySQL 8.0.23.
- `require_row_format`: For internal server use. Added in MySQL 8.0.19.
- `resultset_metadata`: Whether server returns result set metadata. Added in MySQL 8.0.3.
- `rewriter_enabled_for_threads_without_privilege_checks`: If this is set to OFF, rewrites are skipped for replication threads which execute with privilege checks disabled (PRIVILEGE_CHECKS_USER is NULL). Added in MySQL 8.0.31.
- `rpl_read_size`: Set minimum amount of data in bytes which is read from binary log files and relay log files. Added in MySQL 8.0.11.
- `rpl_semi_sync_replica_enabled`: Whether semisynchronous replication is enabled on replica. Added in MySQL 8.0.26.
- `rpl_semi_sync_replica_trace_level`: Semisynchronous replication debug trace level on replica. Added in MySQL 8.0.26.
- `rpl_semi_sync_source_enabled`: Whether semisynchronous replication is enabled on source. Added in MySQL 8.0.26.
- `rpl_semi_sync_source_timeout`: Number of milliseconds to wait for replica acknowledgment. Added in MySQL 8.0.26.
- `rpl_semi_sync_source_trace_level`: Semisynchronous replication debug trace level on source. Added in MySQL 8.0.26.
- `rpl_semi_sync_source_wait_for_replica_count`: Number of replica acknowledgments source must receive per transaction before proceeding. Added in MySQL 8.0.26.
- `rpl_semi_sync_source_wait_no_replica`: Whether source waits for timeout even with no replicas. Added in MySQL 8.0.26.
- `rpl_semi_sync_source_wait_point`: Wait point for replica transaction receipt acknowledgment. Added in MySQL 8.0.26.
- `rpl_stop_replica_timeout`: Number of seconds that STOP REPLICA waits before timing out. Added in MySQL 8.0.26.
- `secondary_engine_cost_threshold`: Optimizer cost threshold for query offload to a secondary engine.. Added in MySQL 8.0.16.
- `select_into_buffer_size`: Size of buffer used for OUTFILE or DUMPFILE export file; overrides read_buffer_size. Added in MySQL 8.0.22.
- `select_into_disk_sync`: Synchronize data with storage device after flushing buffer for OUTFILE or DUMPFILE export file; OFF disables synchronization and is default value. Added in MySQL 8.0.22.

- `select_into_disk_sync_delay`: When select_into_sync_disk = ON, sets delay in milliseconds after each synchronization of OUTFILE or DUMPFILE export file buffer, no effect otherwise. Added in MySQL 8.0.22.
- `show-replica-auth-info`: Show user name and password in SHOW REPLICAS on this source. Added in MySQL 8.0.26.
- `show_create_table_skip_secondary_engine`: Whether to exclude the SECONDARY ENGINE clause from SHOW CREATE TABLE output. Added in MySQL 8.0.18.
- `show_create_table_verbosity`: Whether to display ROW_FORMAT in SHOW CREATE TABLE even if it has default value. Added in MySQL 8.0.11.
- `show_gipk_in_create_table_and_information_schema`: Whether generated invisible primary keys are displayed in SHOW statements and INFORMATION_SCHEMA tables. Added in MySQL 8.0.30.
- `skip-replica-start`: If set, replication is not autostarted when replica server starts. Added in MySQL 8.0.26.
- `source_verify_checksum`: Cause source to examine checksums when reading from binary log. Added in MySQL 8.0.26.
- `sql_generate_invisible_primary_key`: Whether to generate invisible primary keys for any InnoDB tables which were created on this server and which have no explicit PKs. Added in MySQL 8.0.30.
- `sql_replica_skip_counter`: Number of events from source that replica should skip. Not compatible with GTID replication. Added in MySQL 8.0.26.
- `sql_require_primary_key`: Whether tables must have primary key. Added in MySQL 8.0.13.
- `ssl_fips_mode`: Whether to enable FIPS mode on server side. Added in MySQL 8.0.11.
- `ssl_session_cache_mode`: Whether to enable session ticket generation by server. Added in MySQL 8.0.29.
- `ssl_session_cache_timeout`: SSL Session timeout value in seconds. Added in MySQL 8.0.29.
- `sync_source_info`: Synchronize source information after every #th event. Added in MySQL 8.0.26.
- `syseventlog.facility`: Facility for syslog messages. Added in MySQL 8.0.13.
- `syseventlog.include_pid`: Whether to include server PID in syslog messages. Added in MySQL 8.0.13.
- `syseventlog.tag`: Tag for server identifier in syslog messages. Added in MySQL 8.0.13.
- `table_encryption_privilege_check`: Enables TABLE_ENCRYPTION_ADMIN privilege check. Added in MySQL 8.0.16.
- `temptable_max_mmap`: The maximum amount of memory the TempTable storage engine can allocate from memory-mapped temporary files. Added in MySQL 8.0.23.
- `temptable_max_ram`: Defines maximum amount of memory that can occupied by TempTable storage engine before data is stored on disk. Added in MySQL 8.0.2.
- `temptable_use_mmap`: Defines whether TempTable storage engine allocates memory-mapped files when the temptable_max_ram threshold is reached. Added in MySQL 8.0.16.
- `terminology_use_previous`: Use terminology before specified version where changes are incompatible. Added in MySQL 8.0.26.
- `thread_pool_algorithm`: Thread pool algorithm. Added in MySQL 8.0.11.

- `thread_pool_dedicated_listeners`: Dedicates a listener thread in each thread group to listen for network events. Added in MySQL 8.0.23.
- `thread_pool_high_priority_connection`: Whether current session is high priority. Added in MySQL 8.0.11.
- `thread_pool_max_active_query_threads`: Maximum permissible number of active query threads per group. Added in MySQL 8.0.19.
- `thread_pool_max_transactions_limit`: Maximum number of transactions permitted during thread pool operation. Added in MySQL 8.0.23.
- `thread_pool_max_unused_threads`: Maximum permissible number of unused threads. Added in MySQL 8.0.11.
- `thread_pool_prio_kickup_timer`: How long before statement is moved to high-priority execution. Added in MySQL 8.0.11.
- `thread_pool_query_threads_per_group`: Maximum number of query threads for a thread group. Added in MySQL 8.0.31.
- `thread_pool_size`: Number of thread groups in thread pool. Added in MySQL 8.0.11.
- `thread_pool_stall_limit`: How long before statement is defined as stalled. Added in MySQL 8.0.11.
- `thread_pool_transaction_delay`: Delay period before thread pool executes a new transaction. Added in MySQL 8.0.31.
- `tls_ciphersuites`: Permissible TLSv1.3 ciphersuites for encrypted connections. Added in MySQL 8.0.16.
- `upgrade`: Control automatic upgrade at startup. Added in MySQL 8.0.16.
- `use_secondary_engine`: Whether to execute queries using a secondary engine. Added in MySQL 8.0.13.
- `validate-config`: Validate server configuration. Added in MySQL 8.0.16.
- `validate_password.check_user_name`: Whether to check passwords against user name. Added in MySQL 8.0.4.
- `validate_password.dictionary_file`: validate_password dictionary file. Added in MySQL 8.0.4.
- `validate_password.dictionary_file_last_parsed`: When dictionary file was last parsed. Added in MySQL 8.0.4.
- `validate_password.dictionary_file_words_count`: Number of words in dictionary file. Added in MySQL 8.0.4.
- `validate_password.length`: validate_password required password length. Added in MySQL 8.0.4.
- `validate_password.mixed_case_count`: validate_password required number of uppercase/lowercase characters. Added in MySQL 8.0.4.
- `validate_password.number_count`: validate_password required number of digit characters. Added in MySQL 8.0.4.
- `validate_password.policy`: validate_password password policy. Added in MySQL 8.0.4.
- `validate_password.special_char_count`: validate_password required number of special characters. Added in MySQL 8.0.4.

- `version_compile_zlib`: Version of compiled-in zlib library. Added in MySQL 8.0.11.
- `windowing_use_high_precision`: Whether to compute window functions to high precision. Added in MySQL 8.0.2.

## Options and Variables Deprecated in MySQL 8.0

The following system variables, status variables, and options have been deprecated in MySQL 8.0.

- `Compression`: Whether client connection uses compression in client/server protocol. Deprecated in MySQL 8.0.18.
- `Slave_open_temp_tables`: Number of temporary tables that replication SQL thread currently has open. Deprecated in MySQL 8.0.26.
- `Slave_rows_last_search_algorithm_used`: Search algorithm most recently used by this replica to locate rows for row-based replication (index, table, or hash scan). Deprecated in MySQL 8.0.26.
- `abort-slave-event-count`: Option used by mysql-test for debugging and testing of replication. Deprecated in MySQL 8.0.29.
- `admin-ssl`: Enable connection encryption. Deprecated in MySQL 8.0.26.
- `default_authentication_plugin`: Default authentication plugin. Deprecated in MySQL 8.0.27.
- `disconnect-slave-event-count`: Option used by mysql-test for debugging and testing of replication. Deprecated in MySQL 8.0.29.
- `expire_logs_days`: Purge binary logs after this many days. Deprecated in MySQL 8.0.3.
- `group_replication_ip_whitelist`: List of hosts permitted to connect to group. Deprecated in MySQL 8.0.22.
- `group_replication_recovery_complete_at`: Recovery policies when handling cached transactions after state transfer. Deprecated in MySQL 8.0.34.
- `have_openssl`: Whether mysqld supports SSL connections. Deprecated in MySQL 8.0.26.
- `have_ssl`: Whether mysqld supports SSL connections. Deprecated in MySQL 8.0.26.
- `init_slave`: Statements that are executed when replica connects to source. Deprecated in MySQL 8.0.26.
- `innodb_log_file_size`: Size of each log file in log group. Deprecated in MySQL 8.0.30.
- `innodb_log_files_in_group`: Number of InnoDB log files in log group. Deprecated in MySQL 8.0.30.
- `innodb_undo_tablespaces`: Number of tablespace files that rollback segments are divided between. Deprecated in MySQL 8.0.4.
- `log_bin_use_v1_row_events`: Whether server is using version 1 binary log row events. Deprecated in MySQL 8.0.18.
- `log_slave_updates`: Whether replica should log updates performed by its replication SQL thread to its own binary log. Deprecated in MySQL 8.0.26.
- `log_slow_slave_statements`: Cause slow statements as executed by replica to be written to slow query log. Deprecated in MySQL 8.0.26.
- `log_syslog`: Whether to write error log to syslog. Deprecated in MySQL 8.0.2.
- `master-info-file`: Location and name of file that remembers source and where I/O replication thread is in source's binary log. Deprecated in MySQL 8.0.18.

- `master_info_repository`: Whether to write connection metadata repository, containing source information and replication I/O thread location in source's binary log, to file or table. Deprecated in MySQL 8.0.23.
- `master_verify_checksum`: Cause source to examine checksums when reading from binary log. Deprecated in MySQL 8.0.26.
- `max_length_for_sort_data`: Max number of bytes in sorted records. Deprecated in MySQL 8.0.20.
- `myisam_repair_threads`: Number of threads to use when repairing MyISAM tables. 1 disables parallel repair. Deprecated in MySQL 8.0.29.
- `no-dd-upgrade`: Prevent automatic upgrade of data dictionary tables at startup. Deprecated in MySQL 8.0.16.
- `old-style-user-limits`: Enable old-style user limits (before 5.0.3, user resources were counted per each user+host vs. per account). Deprecated in MySQL 8.0.30.
- `pseudo_slave_mode`: For internal server use. Deprecated in MySQL 8.0.26.
- `query_prealloc_size`: Persistent buffer for query parsing and execution. Deprecated in MySQL 8.0.29.
- `relay_log_info_file`: File name for applier metadata repository in which replica records information about relay logs. Deprecated in MySQL 8.0.18.
- `relay_log_info_repository`: Whether to write location of replication SQL thread in relay logs to file or table. Deprecated in MySQL 8.0.23.
- `replica_parallel_type`: Tells replica to use timestamp information (LOGICAL_CLOCK) or database partitioning (DATABASE) to parallelize transactions. Deprecated in MySQL 8.0.29.
- `rpl_stop_slave_timeout`: Number of seconds that STOP REPLICA or STOP SLAVE waits before timing out. Deprecated in MySQL 8.0.26.
- `show-slave-auth-info`: Show user name and password in SHOW REPLICAS and SHOW SLAVE HOSTS on this source. Deprecated in MySQL 8.0.26.
- `skip-host-cache`: Do not cache host names. Deprecated in MySQL 8.0.30.
- `skip-slave-start`: If set, replication is not autostarted when replica server starts. Deprecated in MySQL 8.0.26.
- `slave-skip-errors`: Tells replication thread to continue replication when query returns error from provided list. Deprecated in MySQL 8.0.26.
- `slave_checkpoint_group`: Maximum number of transactions processed by multithreaded replica before checkpoint operation is called to update progress status. Not supported by NDB Cluster. Deprecated in MySQL 8.0.26.
- `slave_checkpoint_period`: Update progress status of multithreaded replica and flush relay log info to disk after this number of milliseconds. Not supported by NDB Cluster. Deprecated in MySQL 8.0.26.
- `slave_compressed_protocol`: Use compression of source/replica protocol. Deprecated in MySQL 8.0.18.
- `slave_load_tmpdir`: Location where replica should put its temporary files when replicating LOAD DATA statements. Deprecated in MySQL 8.0.26.
- `slave_max_allowed_packet`: Maximum size, in bytes, of packet that can be sent from replication source server to replica; overrides max_allowed_packet. Deprecated in MySQL 8.0.26.

- `slave_net_timeout`: Number of seconds to wait for more data from source/replica connection before aborting read. Deprecated in MySQL 8.0.26.
- `slave_parallel_type`: Tells replica to use timestamp information (LOGICAL_CLOCK) or database partioning (DATABASE) to parallelize transactions. Deprecated in MySQL 8.0.26.
- `slave_parallel_workers`: Number of applier threads for executing replication transactions in parallel; 0 or 1 disables replica multithreading. NDB Cluster: see documentation. Deprecated in MySQL 8.0.26.
- `slave_pending_jobs_size_max`: Maximum size of replica worker queues holding events not yet applied. Deprecated in MySQL 8.0.26.
- `slave_preserve_commit_order`: Ensures that all commits by replica workers happen in same order as on source to maintain consistency when using parallel applier threads. Deprecated in MySQL 8.0.26.
- `slave_rows_search_algorithms`: Determines search algorithms used for replica update batching. Any 2 or 3 from this list: INDEX_SEARCH, TABLE_SCAN, HASH_SCAN. Deprecated in MySQL 8.0.18.
- `slave_sql_verify_checksum`: Cause replica to examine checksums when reading from relay log. Deprecated in MySQL 8.0.26.
- `slave_transaction_retries`: Number of times replication SQL thread retries transaction in case it failed with deadlock or elapsed lock wait timeout, before giving up and stopping. Deprecated in MySQL 8.0.26.
- `slave_type_conversions`: Controls type conversion mode on replica. Value is list of zero or more elements from this list: ALL_LOSSY, ALL_NON_LOSSY. Set to empty string to disallow type conversions between source and replica. Deprecated in MySQL 8.0.26.
- `sql_slave_skip_counter`: Number of events from source that replica should skip. Not compatible with GTID replication. Deprecated in MySQL 8.0.26.
- `ssl`: Enable connection encryption. Deprecated in MySQL 8.0.26.
- `symbolic-links`: Permit symbolic links for MyISAM tables. Deprecated in MySQL 8.0.2.
- `sync_master_info`: Synchronize source information after every #th event. Deprecated in MySQL 8.0.26.
- `temptable_use_mmap`: Defines whether TempTable storage engine allocates memory-mapped files when the temptable_max_ram threshold is reached. Deprecated in MySQL 8.0.26.
- `transaction_prealloc_size`: Persistent buffer for transactions to be stored in binary log. Deprecated in MySQL 8.0.29.
- `transaction_write_set_extraction`: Defines algorithm used to hash writes extracted during transaction. Deprecated in MySQL 8.0.26.

## Options and Variables Removed in MySQL 8.0

The following system variables, status variables, and options have been removed in MySQL 8.0.

- `Com_alter_db_upgrade`: Count of ALTER DATABASE ... UPGRADE DATA DIRECTORY NAME statements. Removed in MySQL 8.0.0.
- `Innodb_available_undo_logs`: Total number of InnoDB rollback segments; different from innodb_rollback_segments, which displays number of active rollback segments. Removed in MySQL 8.0.2.
- `Qcache_free_blocks`: Number of free memory blocks in query cache. Removed in MySQL 8.0.3.

- `Qcache_free_memory`: Amount of free memory for query cache. Removed in MySQL 8.0.3.
- `Qcache_hits`: Number of query cache hits. Removed in MySQL 8.0.3.
- `Qcache_inserts`: Number of query cache inserts. Removed in MySQL 8.0.3.
- `Qcache_lowmem_prunes`: Number of queries which were deleted from query cache due to lack of free memory in cache. Removed in MySQL 8.0.3.
- `Qcache_not_cached`: Number of noncached queries (not cacheable, or not cached due to query_cache_type setting). Removed in MySQL 8.0.3.
- `Qcache_queries_in_cache`: Number of queries registered in query cache. Removed in MySQL 8.0.3.
- `Qcache_total_blocks`: Total number of blocks in query cache. Removed in MySQL 8.0.3.
- `Slave_heartbeat_period`: Replica's replication heartbeat interval, in seconds. Removed in MySQL 8.0.1.
- `Slave_last_heartbeat`: Shows when latest heartbeat signal was received, in TIMESTAMP format. Removed in MySQL 8.0.1.
- `Slave_received_heartbeats`: Number of heartbeats received by replica since previous reset. Removed in MySQL 8.0.1.
- `Slave_retried_transactions`: Total number of times since startup that replication SQL thread has retried transactions. Removed in MySQL 8.0.1.
- `Slave_running`: State of this server as replica (replication I/O thread status). Removed in MySQL 8.0.1.
- `bootstrap`: Used by mysql installation scripts. Removed in MySQL 8.0.0.
- `date_format`: DATE format (unused). Removed in MySQL 8.0.3.
- `datetime_format`: DATETIME/TIMESTAMP format (unused). Removed in MySQL 8.0.3.
- `des-key-file`: Load keys for des_encrypt() and des_encrypt from given file. Removed in MySQL 8.0.3.
- `group_replication_allow_local_disjoint_gtids_join`: Allow current server to join group even if it has transactions not present in group. Removed in MySQL 8.0.4.
- `have_crypt`: Availability of crypt() system call. Removed in MySQL 8.0.3.
- `ignore-db-dir`: Treat directory as nondatabase directory. Removed in MySQL 8.0.0.
- `ignore_builtin_innodb`: Ignore built-in InnoDB. Removed in MySQL 8.0.3.
- `ignore_db_dirs`: Directories treated as nondatabase directories. Removed in MySQL 8.0.0.
- `innodb_checksums`: Enable InnoDB checksums validation. Removed in MySQL 8.0.0.
- `innodb_disable_resize_buffer_pool_debug`: Disables resizing of InnoDB buffer pool. Removed in MySQL 8.0.0.
- `innodb_file_format`: Format for new InnoDB tables. Removed in MySQL 8.0.0.
- `innodb_file_format_check`: Whether InnoDB performs file format compatibility checking. Removed in MySQL 8.0.0.
- `innodb_file_format_max`: File format tag in shared tablespace. Removed in MySQL 8.0.0.
- `innodb_large_prefix`: Enables longer keys for column prefix indexes. Removed in MySQL 8.0.0.

- `innodb_locks_unsafe_for_binlog`: Force InnoDB not to use next-key locking. Instead use only row-level locking. Removed in MySQL 8.0.0.
- `innodb_scan_directories`: Defines directories to scan for tablespace files during InnoDB recovery. Removed in MySQL 8.0.4.
- `innodb_stats_sample_pages`: Number of index pages to sample for index distribution statistics. Removed in MySQL 8.0.0.
- `innodb_support_xa`: Enable InnoDB support for XA two-phase commit. Removed in MySQL 8.0.0.
- `innodb_undo_logs`: Number of undo logs (rollback segments) used by InnoDB; alias for innodb_rollback_segments. Removed in MySQL 8.0.2.
- `internal_tmp_disk_storage_engine`: Storage engine for internal temporary tables. Removed in MySQL 8.0.16.
- `log-warnings`: Write some noncritical warnings to log file. Removed in MySQL 8.0.3.
- `log_builtin_as_identified_by_password`: Whether to log CREATE/ALTER USER, GRANT in backward-compatible fashion. Removed in MySQL 8.0.11.
- `log_error_filter_rules`: Filter rules for error logging. Removed in MySQL 8.0.4.
- `log_syslog`: Whether to write error log to syslog. Removed in MySQL 8.0.13.
- `log_syslog_facility`: Facility for syslog messages. Removed in MySQL 8.0.13.
- `log_syslog_include_pid`: Whether to include server PID in syslog messages. Removed in MySQL 8.0.13.
- `log_syslog_tag`: Tag for server identifier in syslog messages. Removed in MySQL 8.0.13.
- `max_tmp_tables`: Unused. Removed in MySQL 8.0.3.
- `metadata_locks_cache_size`: Size of metadata locks cache. Removed in MySQL 8.0.13.
- `metadata_locks_hash_instances`: Number of metadata lock hashes. Removed in MySQL 8.0.13.
- `multi_range_count`: Maximum number of ranges to send to table handler at once during range selects. Removed in MySQL 8.0.3.
- `myisam_repair_threads`: Number of threads to use when repairing MyISAM tables. 1 disables parallel repair. Removed in MySQL 8.0.30.
- `old_passwords`: Selects password hashing method for PASSWORD(). Removed in MySQL 8.0.11.
- `partition`: Enable (or disable) partitioning support. Removed in MySQL 8.0.0.
- `query_cache_limit`: Do not cache results that are bigger than this. Removed in MySQL 8.0.3.
- `query_cache_min_res_unit`: Minimal size of unit in which space for results is allocated (last unit is trimmed after writing all result data). Removed in MySQL 8.0.3.
- `query_cache_size`: Memory allocated to store results from old queries. Removed in MySQL 8.0.3.
- `query_cache_type`: Query cache type. Removed in MySQL 8.0.3.
- `query_cache_wlock_invalidate`: Invalidate queries in query cache on LOCK for write. Removed in MySQL 8.0.3.
- `secure_auth`: Disallow authentication for accounts that have old (pre-4.1) passwords. Removed in MySQL 8.0.3.

- `show_compatibility_56`: Compatibility for SHOW STATUS/VARIABLES. Removed in MySQL 8.0.1.
- `skip-partition`: Do not enable user-defined partitioning. Removed in MySQL 8.0.0.
- `sync_frm`: Sync .frm to disk on create. Enabled by default. Removed in MySQL 8.0.0.
- `temp-pool`: Using this option causes most temporary files created to use small set of names, rather than unique name for each new file. Removed in MySQL 8.0.1.
- `time_format`: TIME format (unused). Removed in MySQL 8.0.3.
- `tx_isolation`: Default transaction isolation level. Removed in MySQL 8.0.3.
- `tx_read_only`: Default transaction access mode. Removed in MySQL 8.0.3.

## 1.5 How to Report Bugs or Problems

Before posting a bug report about a problem, please try to verify that it is a bug and that it has not been reported already:

- Start by searching the MySQL online manual at https://dev.mysql.com/doc/. We try to keep the manual up to date by updating it frequently with solutions to newly found problems. In addition, the release notes accompanying the manual can be particularly useful since it is quite possible that a newer version contains a solution to your problem. The release notes are available at the location just given for the manual.
- If you get a parse error for an SQL statement, please check your syntax closely. If you cannot find something wrong with it, it is extremely likely that your current version of MySQL Server doesn't support the syntax you are using. If you are using the current version and the manual doesn't cover the syntax that you are using, MySQL Server doesn't support your statement.

  If the manual covers the syntax you are using, but you have an older version of MySQL Server, you should check the MySQL change history to see when the syntax was implemented. In this case, you have the option of upgrading to a newer version of MySQL Server.
- For solutions to some common problems, see Section B.3, “Problems and Common Errors”.
- Search the bugs database at http://bugs.mysql.com/ to see whether the bug has been reported and fixed.
- You can also use http://www.mysql.com/search/ to search all the Web pages (including the manual) that are located at the MySQL website.

If you cannot find an answer in the manual, the bugs database, or the mailing list archives, check with your local MySQL expert. If you still cannot find an answer to your question, please use the following guidelines for reporting the bug.

The normal way to report bugs is to visit 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 can enter new reports.

Bugs posted in the bugs database at http://bugs.mysql.com/ that are corrected for a given release are noted in the release notes.

If you find a 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/.

To discuss problems with other users, you can use the MySQL Community Slack.

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

the bug in the next release. This section helps you write your report correctly so that you do not waste your time doing things that may not help us much or at all. Please read this section carefully and make sure that all the information described here is included in your report.

Preferably, you should test the problem using the latest production or development version of MySQL Server before posting. Anyone should be able to repeat the bug by just using `mysql test < script_file` on your test case or by running the shell or Perl script that you include in the bug report. Any bug that we are able to repeat has a high chance of being fixed in the next MySQL release.

It is most helpful when a good description of the problem is included in the bug report. That is, give a good example of everything you did that led to the problem and describe, in exact detail, the problem itself. The best reports are those that include a full example showing how to reproduce the bug or problem. See Section 5.9, “Debugging MySQL”.

Remember that it is possible for us to respond to a report 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 to follow is that 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 the MySQL distribution that you use, and (b) not fully describing the platform on which the MySQL server is installed (including the platform type and version number). These are highly relevant pieces of information, and in 99 cases out of 100, the bug report is useless without them. Very often we get questions like, “Why doesn't this work for me?” Then we find that the feature requested wasn't implemented in that MySQL version, or that a bug described in a report has been fixed in newer MySQL versions. Errors often are 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 you compiled MySQL from source, remember also to provide information about your compiler if it is related to the problem. Often people find bugs in compilers and think the problem is MySQL-related. Most compilers are under development all the time and become better version by version. To determine whether your problem depends on your compiler, we need to know what compiler you used. Note that every compiling problem should be regarded as a bug and reported accordingly.

If a program produces an error message, it is very important to include the message in your report. If we try to search for something from the archives, it is better that the error message reported exactly matches the one that the program produces. (Even the lettercase should be observed.) It is best to copy and paste the entire error message into your report. You should never try to reproduce the message from memory.

If you have a problem with Connector/ODBC (MyODBC), please try to generate a trace file and send it with your report. See How to Report Connector/ODBC Problems or Bugs.

If your report includes long query output lines from test cases that you run with the `mysql` command-line tool, you can make the output more readable by using the `--vertical` option or the `\G` statement terminator. The `EXPLAIN SELECT` example later in this section demonstrates the use of `\G`.

Please include the following information in your report:

- The version number of the MySQL distribution you are using (for example, MySQL 5.7.10). You can find out which version you are running by executing `mysqladmin version`. The `mysqladmin` program can be found in the `bin` directory under your MySQL installation directory.
- The manufacturer and model of the machine on which you experience the problem.
- The operating system name and version. If you work with Windows, you can usually get the name and version number by double-clicking your My Computer icon and pulling down the “Help/About Windows” menu. For most Unix-like operating systems, you can get this information by executing the command `uname -a`.

- Sometimes the amount of memory (real and virtual) is relevant. If in doubt, include these values.
- The contents of the `docs/INFO_BIN` file from your MySQL installation. This file contains information about how MySQL was configured and compiled.
- If you are using a source distribution of the MySQL software, include the name and version number of the compiler that you used. If you have a binary distribution, include the distribution name.
- If the problem occurs during compilation, include the exact error messages and also a few lines of context around the offending code in the file where the error occurs.
- If `mysqld` died, you should also report the statement that caused `mysqld` to unexpectedly exit. You can usually get this information by running `mysqld` with query logging enabled, and then looking in the log after `mysqld` exits. See Section 5.9, “Debugging MySQL”.
- If a database table is related to the problem, include the output from the `SHOW CREATE TABLE db_name.tbl_name` statement in the bug report. This is a very easy way to get the definition of any table in a database. The information helps us create a situation matching the one that you have experienced.
- The SQL mode in effect when the problem occurred can be significant, so please report the value of the `sql_mode` system variable. For stored procedure, stored function, and trigger objects, the relevant `sql_mode` value is the one in effect when the object was created. For a stored procedure or function, the `SHOW CREATE PROCEDURE` or `SHOW CREATE FUNCTION` statement shows the relevant SQL mode, or you can query `INFORMATION_SCHEMA` for the information:

  ```
  SELECT ROUTINE_SCHEMA, ROUTINE_NAME, SQL_MODE
  FROM INFORMATION_SCHEMA.ROUTINES;
  ```

  For triggers, you can use this statement:

  ```
  SELECT EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE, TRIGGER_NAME, SQL_MODE
  FROM INFORMATION_SCHEMA.TRIGGERS;
  ```

- For performance-related bugs or problems with `SELECT` statements, you should always include the output of `EXPLAIN SELECT ...`, and at least the number of rows that the `SELECT` statement produces. You should also include the output from `SHOW CREATE TABLE tbl_name` for each table that is involved. The more information you provide about your situation, the more likely it is that someone can help you.

  The following is an example of a very good bug report. The statements are run using the `mysql` command-line tool. Note the use of the `\G` statement terminator for statements that would otherwise provide very long output lines that are difficult to read.

  ```
  mysql> SHOW VARIABLES;
  mysql> SHOW COLUMNS FROM ...\G
         <output from SHOW COLUMNS>
  mysql> EXPLAIN SELECT ...\G
         <output from EXPLAIN>
  mysql> FLUSH STATUS;
  mysql> SELECT ...;
         <A short version of the output from SELECT,
         including the time taken to run the query>
  mysql> SHOW STATUS;
         <output from SHOW STATUS>
  ```

- If a bug or problem occurs while running `mysqld`, try to provide an input script that reproduces the anomaly. This script should include any necessary source files. The more closely the script can reproduce your situation, the better. If you can make a reproducible test case, you should upload it to be attached to the bug report.

  If you cannot provide a script, you should at least include the output from `mysqladmin variables extended-status processlist` in your report to provide some information on how your system is performing.

- If you cannot produce a test case with only a few rows, or if the test table is too big to be included in the bug report (more than 10 rows), you should dump your tables using `mysqldump` and create a `README` file that describes your problem. Create a compressed archive of your files using `tar` and `gzip` or `zip`. After you initiate a bug report for our bugs database at http://bugs.mysql.com/, click the Files tab in the bug report for instructions on uploading the archive to the bugs database.

- If you believe that the MySQL server produces a strange result from a statement, include not only the result, but also your opinion of what the result should be, and an explanation describing the basis for your opinion.

- When you provide an example of the problem, it is better to use the table names, variable names, and so forth that exist in your actual situation than to come up with new names. The problem could be related to the name of a table or variable. These cases are rare, perhaps, but it is better to be safe than sorry. After all, it should be easier for you to provide an example that uses your actual situation, and it is by all means better for us. If you have data that you do not want to be visible to others in the bug report, you can upload it using the Files tab as previously described. If the information is really top secret and you do not want to show it even to us, go ahead and provide an example using other names, but please regard this as the last choice.

- Include all the options given to the relevant programs, if possible. For example, indicate the options that you use when you start the `mysqld` server, as well as the options that you use to run any MySQL client programs. The options to programs such as `mysqld` and `mysql`, and to the `configure` script, are often key to resolving problems and are very relevant. It is never a bad idea to include them. If your problem involves a program written in a language such as Perl or PHP, please include the language processor's version number, as well as the version for any modules that the program uses. For example, if you have a Perl script that uses the `DBI` and `DBD::mysql` modules, include the version numbers for Perl, `DBI`, and `DBD::mysql`.

- If your question is related to the privilege system, please include the output of `mysqladmin reload`, and all the error messages you get when trying to connect. When you test your privileges, you should execute `mysqladmin reload version` and try to connect with the program that gives you trouble.

- If you have a patch for a bug, do include it. But do not assume that the patch is all we need, or that we can use it, if you do not provide some necessary information such as test cases showing the bug that your patch fixes. We might find problems with your patch or we might not understand it at all. If so, we cannot use it.

  If we cannot verify the exact purpose of the patch, we will not use it. Test cases help us here. Show that the patch handles all the situations that may occur. If we find a borderline case (even a rare one) where the patch will not work, it may be useless.

- Guesses about what the bug is, why it occurs, or what it depends on are usually wrong. Even the MySQL team cannot guess such things without first using a debugger to determine the real cause of a bug.

- Indicate in your bug report that you have checked the reference manual and mail archive so that others know you have tried to solve the problem yourself.

- If your data appears corrupt or you get errors when you access a particular table, first check your tables with `CHECK TABLE`. If that statement reports any errors:

  - The `InnoDB` crash recovery mechanism handles cleanup when the server is restarted after being killed, so in typical operation there is no need to “repair” tables. If you encounter an error with `InnoDB` tables, restart the server and see whether the problem persists, or whether the error affected only cached data in memory. If data is corrupted on disk, consider restarting with the `innodb_force_recovery` option enabled so that you can dump the affected tables.

  - For non-transactional tables, try to repair them with `REPAIR TABLE` or with `myisamchk`. See Chapter 5, *MySQL Server Administration*.

If you are running Windows, please verify the value of `lower_case_table_names` using the `SHOW VARIABLES LIKE 'lower_case_table_names'` statement. This variable affects how the server handles lettercase of database and table names. Its effect for a given value should be as described in Section 9.2.3, “Identifier Case Sensitivity”.

- If you often get corrupted tables, you should try to find out when and why this happens. In this case, the error log in the MySQL data directory may contain some information about what happened. (This is the file with the `.err` suffix in the name.) See Section 5.4.2, “The Error Log”. Please include any relevant information from this file in your bug report. Normally `mysqld` should *never* corrupt a table if nothing killed it in the middle of an update. If you can find the cause of `mysqld` dying, it is much easier for us to provide you with a fix for the problem. See Section B.3.1, “How to Determine What Is Causing a Problem”.
- If possible, download and install the most recent version of MySQL Server and check whether it solves your problem. All versions of the MySQL software are thoroughly tested and should work without problems. We believe in making everything as backward-compatible as possible, and you should be able to switch MySQL versions without difficulty. See Section 2.1.2, “Which MySQL Version and Distribution to Install”.

# 1.6 MySQL Standards Compliance

This section describes how MySQL relates to the ANSI/ISO SQL standards. MySQL Server has many extensions to the SQL standard, and here you can find out what they are and how to use them. You can also find information about functionality missing from MySQL Server, and how to work around some of the differences.

The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992. “SQL:1999”, “SQL:2003”, “SQL:2008”, and “SQL:2011” refer to the versions of the standard released in the corresponding years, with the last being the most recent version. We use the phrase “the SQL standard” or “standard SQL” to mean the current version of the SQL Standard at any time.

One of our main goals with the product is to continue to work toward compliance with the SQL standard, but without sacrificing speed or reliability. We are not afraid to add extensions to SQL or support for non-SQL features if this greatly increases the usability of MySQL Server for a large segment of our user base. The `HANDLER` interface is an example of this strategy. See Section 13.2.5, “HANDLER Statement”.

We continue to support transactional and nontransactional databases to satisfy both mission-critical 24/7 usage and heavy Web or logging usage.

MySQL Server was originally designed to work with medium-sized databases (10-100 million rows, or about 100MB per table) on small computer systems. Today MySQL Server handles terabyte-sized databases.

We are not targeting real-time support, although MySQL replication capabilities offer significant functionality.

MySQL supports ODBC levels 0 to 3.51.

MySQL supports high-availability database clustering using the `NDBCLUSTER` storage engine. See Chapter 23, *MySQL NDB Cluster 8.0*.

We implement XML functionality which supports most of the W3C XPath standard. See Section 12.12, “XML Functions”.

MySQL supports a native JSON data type as defined by RFC 7159, and based on the ECMAScript standard (ECMA-262). See Section 11.5, “The JSON Data Type”. MySQL also implements a subset

of the SQL/JSON functions specified by a pre-publication draft of the SQL:2016 standard; see Section 12.18, “JSON Functions”, for more information.

## Selecting SQL Modes

The MySQL server can operate in different SQL modes, and can apply these modes differently for different clients, depending on the value of the `sql_mode` system variable. DBAs can set the global SQL mode to match site server operating requirements, and each application can set its session SQL mode to its own requirements.

Modes affect the SQL syntax MySQL supports and the data validation checks it performs. This makes it easier to use MySQL in different environments and to use MySQL together with other database servers.

For more information on setting the SQL mode, see Section 5.1.11, “Server SQL Modes”.

## Running MySQL in ANSI Mode

To run MySQL Server in ANSI mode, start `mysqld` with the `--ansi` option. Running the server in ANSI mode is the same as starting it with the following options:

```
--transaction-isolation=SERIALIZABLE --sql-mode=ANSI
```

To achieve the same effect at runtime, execute these two statements:

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET GLOBAL sql_mode = 'ANSI';
```

You can see that setting the `sql_mode` system variable to `'ANSI'` enables all SQL mode options that are relevant for ANSI mode as follows:

```
mysql> SET GLOBAL sql_mode='ANSI';
mysql> SELECT @@GLOBAL.sql_mode;
        -> 'REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ANSI'
```

Running the server in ANSI mode with `--ansi` is not quite the same as setting the SQL mode to `'ANSI'` because the `--ansi` option also sets the transaction isolation level.

See Section 5.1.7, “Server Command Options”.

## 1.6.1 MySQL Extensions to Standard SQL

MySQL Server supports some extensions that you are not likely to find in other SQL DBMSs. Be warned that if you use them, your code is most likely not portable to other SQL servers. In some cases, you can write code that includes MySQL extensions, but is still portable, by using comments of the following form:

```
/*! MySQL-specific code */
```

In this case, MySQL Server parses and executes the code within the comment as it would any other SQL statement, but other SQL servers should ignore the extensions. For example, MySQL Server recognizes the `STRAIGHT_JOIN` keyword in the following statement, but other servers should not:

```
SELECT /*! STRAIGHT_JOIN */ col1 FROM table1,table2 WHERE ...
```

If you add a version number after the `!` character, the syntax within the comment is executed only if the MySQL version is greater than or equal to the specified version number. The `KEY_BLOCK_SIZE` clause in the following comment is executed only by servers from MySQL 5.1.10 or higher:

```
CREATE TABLE t1(a INT, KEY (a)) /*!50110 KEY_BLOCK_SIZE=1024 */;
```

The following descriptions list MySQL extensions, organized by category.

- Organization of data on disk

  MySQL Server maps each database to a directory under the MySQL data directory, and maps tables within a database to file names in the database directory. Consequently, database and table names are case-sensitive in MySQL Server on operating systems that have case-sensitive file names (such as most Unix systems). See Section 9.2.3, “Identifier Case Sensitivity”.

- General language syntax

  - By default, strings can be enclosed by `"` as well as `'`. If the `ANSI_QUOTES` SQL mode is enabled, strings can be enclosed only by `'` and the server interprets strings enclosed by `"` as identifiers.
  - `\` is the escape character in strings.
  - In SQL statements, you can access tables from different databases with the *`db_name.tbl_name`* syntax. Some SQL servers provide the same functionality but call this `User space`. MySQL Server doesn't support tablespaces such as used in statements like this: `CREATE TABLE ralph.my_table ... IN my_tablespace`.

- SQL statement syntax

  - The `ANALYZE TABLE`, `CHECK TABLE`, `OPTIMIZE TABLE`, and `REPAIR TABLE` statements.
  - The `CREATE DATABASE`, `DROP DATABASE`, and `ALTER DATABASE` statements. See Section 13.1.12, “CREATE DATABASE Statement”, Section 13.1.24, “DROP DATABASE Statement”, and Section 13.1.2, “ALTER DATABASE Statement”.
  - The `DO` statement.
  - `EXPLAIN SELECT` to obtain a description of how tables are processed by the query optimizer.
  - The `FLUSH` and `RESET` statements.
  - The `SET` statement. See Section 13.7.6.1, “SET Syntax for Variable Assignment”.
  - The `SHOW` statement. See Section 13.7.7, “SHOW Statements”. The information produced by many of the MySQL-specific `SHOW` statements can be obtained in more standard fashion by using `SELECT` to query `INFORMATION_SCHEMA`. See Chapter 26, *INFORMATION_SCHEMA Tables*.
  - Use of `LOAD DATA`. In many cases, this syntax is compatible with Oracle `LOAD DATA`. See Section 13.2.9, “LOAD DATA Statement”.
  - Use of `RENAME TABLE`. See Section 13.1.36, “RENAME TABLE Statement”.
  - Use of `REPLACE` instead of `DELETE` plus `INSERT`. See Section 13.2.12, “REPLACE Statement”.
  - Use of `CHANGE` *`col_name`*, `DROP` *`col_name`*, or `DROP INDEX`, `IGNORE` or `RENAME` in `ALTER TABLE` statements. Use of multiple `ADD`, `ALTER`, `DROP`, or `CHANGE` clauses in an `ALTER TABLE` statement. See Section 13.1.9, “ALTER TABLE Statement”.
  - Use of index names, indexes on a prefix of a column, and use of `INDEX` or `KEY` in `CREATE TABLE` statements. See Section 13.1.20, “CREATE TABLE Statement”.
  - Use of `TEMPORARY` or `IF NOT EXISTS` with `CREATE TABLE`.
  - Use of `IF EXISTS` with `DROP TABLE` and `DROP DATABASE`.
  - The capability of dropping multiple tables with a single `DROP TABLE` statement.
  - The `ORDER BY` and `LIMIT` clauses of the `UPDATE` and `DELETE` statements.
  - `INSERT INTO` *`tbl_name`* `SET` *`col_name`* `= ...` syntax.

- The `DELAYED` clause of the `INSERT` and `REPLACE` statements.
- The `LOW_PRIORITY` clause of the `INSERT`, `REPLACE`, `DELETE`, and `UPDATE` statements.
- Use of `INTO OUTFILE` or `INTO DUMPFILE` in `SELECT` statements. See Section 13.2.13, “SELECT Statement”.
- Options such as `STRAIGHT_JOIN` or `SQL_SMALL_RESULT` in `SELECT` statements.
- You don't need to name all selected columns in the `GROUP BY` clause. This gives better performance for some very specific, but quite normal queries. See Section 12.20, “Aggregate Functions”.
- You can specify `ASC` and `DESC` with `GROUP BY`, not just with `ORDER BY`.
- The ability to set variables in a statement with the `:=` assignment operator. See Section 9.4, “User-Defined Variables”.

- Data types
  - The `MEDIUMINT`, `SET`, and `ENUM` data types, and the various `BLOB` and `TEXT` data types.
  - The `AUTO_INCREMENT`, `BINARY`, `NULL`, `UNSIGNED`, and `ZEROFILL` data type attributes.
- Functions and operators
  - To make it easier for users who migrate from other SQL environments, MySQL Server supports aliases for many functions. For example, all string functions support both standard SQL syntax and ODBC syntax.
  - MySQL Server understands the `||` and `&&` operators to mean logical OR and AND, as in the C programming language. In MySQL Server, `||` and `OR` are synonyms, as are `&&` and `AND`. Because of this nice syntax, MySQL Server doesn't support the standard SQL `||` operator for string concatenation; use `CONCAT()` instead. Because `CONCAT()` takes any number of arguments, it is easy to convert use of the `||` operator to MySQL Server.
  - Use of `COUNT(DISTINCT value_list)` where *`value_list`* has more than one element.
  - String comparisons are case-insensitive by default, with sort ordering determined by the collation of the current character set, which is `utf8mb4` by default. To perform case-sensitive comparisons instead, you should declare your columns with the `BINARY` attribute or use the `BINARY` cast, which causes comparisons to be done using the underlying character code values rather than a lexical ordering.
  - The `%` operator is a synonym for `MOD()`. That is, *`N % M`* is equivalent to `MOD(N,M)`. `%` is supported for C programmers and for compatibility with PostgreSQL.
  - The `=`, `<>`, `<=`, `<`, `>=`, `>`, `<<`, `>>`, `<=>`, `AND`, `OR`, or `LIKE` operators may be used in expressions in the output column list (to the left of the `FROM`) in `SELECT` statements. For example:

    ```
    mysql> SELECT col1=1 AND col2=2 FROM my_table;
    ```

  - The `LAST_INSERT_ID()` function returns the most recent `AUTO_INCREMENT` value. See Section 12.16, “Information Functions”.
  - `LIKE` is permitted on numeric values.
  - The `REGEXP` and `NOT REGEXP` extended regular expression operators.
  - `CONCAT()` or `CHAR()` with one argument or more than two arguments. (In MySQL Server, these functions can take a variable number of arguments.)

- The `BIT_COUNT()`, `CASE`, `ELT()`, `FROM_DAYS()`, `FORMAT()`, `IF()`, `MD5()`, `PERIOD_ADD()`, `PERIOD_DIFF()`, `TO_DAYS()`, and `WEEKDAY()` functions.
- Use of `TRIM()` to trim substrings. Standard SQL supports removal of single characters only.
- The `GROUP BY` functions `STD()`, `BIT_OR()`, `BIT_AND()`, `BIT_XOR()`, and `GROUP_CONCAT()`. See Section 12.20, “Aggregate Functions”.

## 1.6.2 MySQL Differences from Standard SQL

We try to make MySQL Server follow the ANSI SQL standard and the ODBC SQL standard, but MySQL Server performs operations differently in some cases:

- There are several differences between the MySQL and standard SQL privilege systems. For example, in MySQL, privileges for a table are not automatically revoked when you delete a table. You must explicitly issue a `REVOKE` statement to revoke privileges for a table. For more information, see Section 13.7.1.8, “REVOKE Statement”.
- The `CAST()` function does not support cast to `REAL` or `BIGINT`. See Section 12.11, “Cast Functions and Operators”.

### 1.6.2.1 SELECT INTO TABLE Differences

MySQL Server doesn't support the `SELECT ... INTO TABLE` Sybase SQL extension. Instead, MySQL Server supports the `INSERT INTO ... SELECT` standard SQL syntax, which is basically the same thing. See Section 13.2.7.1, “INSERT ... SELECT Statement”. For example:

```
INSERT INTO tbl_temp2 (fld_id)
    SELECT tbl_temp1.fld_order_id
    FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

Alternatively, you can use `SELECT ... INTO OUTFILE` or `CREATE TABLE ... SELECT`.

You can use `SELECT ... INTO` with user-defined variables. The same syntax can also be used inside stored routines using cursors and local variables. See Section 13.2.13.1, “SELECT ... INTO Statement”.

### 1.6.2.2 UPDATE Differences

If you access a column from the table to be updated in an expression, `UPDATE` uses the current value of the column. The second assignment in the following statement sets `col2` to the current (updated) `col1` value, not the original `col1` value. The result is that `col1` and `col2` have the same value. This behavior differs from standard SQL.

```
UPDATE t1 SET col1 = col1 + 1, col2 = col1;
```

### 1.6.2.3 FOREIGN KEY Constraint Differences

The MySQL implementation of foreign key constraints differs from the SQL standard in the following key respects:

- If there are several rows in the parent table with the same referenced key value, `InnoDB` performs a foreign key check as if the other parent rows with the same key value do not exist. For example, if you define a `RESTRICT` type constraint, and there is a child row with several parent rows, `InnoDB` does not permit the deletion of any of the parent rows.
- If `ON UPDATE CASCADE` or `ON UPDATE SET NULL` recurses to update the *same table* it has previously updated during the same cascade, it acts like `RESTRICT`. This means that you cannot use self-referential `ON UPDATE CASCADE` or `ON UPDATE SET NULL` operations. This is to prevent infinite loops resulting from cascaded updates. A self-referential `ON DELETE SET NULL`, on the

other hand, is possible, as is a self-referential `ON DELETE CASCADE`. Cascading operations may not be nested more than 15 levels deep.

- In an SQL statement that inserts, deletes, or updates many rows, foreign key constraints (like unique constraints) are checked row-by-row. When performing foreign key checks, `InnoDB` sets shared row-level locks on child or parent records that it must examine. MySQL checks foreign key constraints immediately; the check is not deferred to transaction commit. According to the SQL standard, the default behavior should be deferred checking. That is, constraints are only checked after the *entire SQL statement* has been processed. This means that it is not possible to delete a row that refers to itself using a foreign key.

- No storage engine, including `InnoDB`, recognizes or enforces the `MATCH` clause used in referential-integrity constraint definitions. Use of an explicit `MATCH` clause does not have the specified effect, and it causes `ON DELETE` and `ON UPDATE` clauses to be ignored. Specifying the `MATCH` should be avoided.

  The `MATCH` clause in the SQL standard controls how `NULL` values in a composite (multiple-column) foreign key are handled when comparing to a primary key in the referenced table. MySQL essentially implements the semantics defined by `MATCH SIMPLE`, which permits a foreign key to be all or partially `NULL`. In that case, a (child table) row containing such a foreign key can be inserted even though it does not match any row in the referenced (parent) table. (It is possible to implement other semantics using triggers.)

- MySQL requires that the referenced columns be indexed for performance reasons. However, MySQL does not enforce a requirement that the referenced columns be `UNIQUE` or be declared `NOT NULL`.

  A `FOREIGN KEY` constraint that references a non-`UNIQUE` key is not standard SQL but rather an `InnoDB` extension. The `NDB` storage engine, on the other hand, requires an explicit unique key (or primary key) on any column referenced as a foreign key.

  The handling of foreign key references to nonunique keys or keys that contain `NULL` values is not well defined for operations such as `UPDATE` or `DELETE CASCADE`. You are advised to use foreign keys that reference only `UNIQUE` (including `PRIMARY`) and `NOT NULL` keys.

- For storage engines that do not support foreign keys (such as `MyISAM`), MySQL Server parses and ignores foreign key specifications.

- MySQL parses but ignores “inline `REFERENCES` specifications” (as defined in the SQL standard) where the references are defined as part of the column specification. MySQL accepts `REFERENCES` clauses only when specified as part of a separate `FOREIGN KEY` specification.

  Defining a column to use a `REFERENCES` *`tbl_name(col_name)`* clause has no actual effect and *serves only as a memo or comment to you that the column which you are currently defining is intended to refer to a column in another table*. It is important to realize when using this syntax that:

  - MySQL does not perform any sort of check to make sure that *`col_name`* actually exists in *`tbl_name`* (or even that *`tbl_name`* itself exists).

  - MySQL does not perform any sort of action on *`tbl_name`* such as deleting rows in response to actions taken on rows in the table which you are defining; in other words, this syntax induces no `ON DELETE` or `ON UPDATE` behavior whatsoever. (Although you can write an `ON DELETE` or `ON UPDATE` clause as part of the `REFERENCES` clause, it is also ignored.)

  - This syntax creates a *column*; it does **not** create any sort of index or key.

  You can use a column so created as a join column, as shown here:

```
CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
    PRIMARY KEY (id)
);

INSERT INTO person VALUES (NULL, 'Antonio Paz');

SELECT @last := LAST_INSERT_ID();

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', @last),
(NULL, 'dress', 'white', @last),
(NULL, 't-shirt', 'blue', @last);

INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

SELECT @last := LAST_INSERT_ID();

INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', @last),
(NULL, 'polo', 'red', @last),
(NULL, 'dress', 'blue', @last),
(NULL, 't-shirt', 'white', @last);

SELECT * FROM person;
+----+--------------------+
| id | name               |
+----+--------------------+
|  1 | Antonio Paz        |
|  2 | Lilliana Angelovska |
+----+--------------------+

SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+


SELECT s.* FROM person p INNER JOIN shirt s
   ON s.owner = p.id
 WHERE p.name LIKE 'Lilliana%'
   AND s.color <> 'white';

+----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

When used in this fashion, the `REFERENCES` clause is not displayed in the output of `SHOW CREATE TABLE` or `DESCRIBE`:

```
SHOW CREATE TABLE shirt\G
*************************** 1. row ***************************
Table: shirt
Create Table: CREATE TABLE `shirt` (
`id` smallint(5) unsigned NOT NULL auto_increment,
`style` enum('t-shirt','polo','dress') NOT NULL,
```

```
`color` enum('red','blue','orange','white','black') NOT NULL,
`owner` smallint(5) unsigned NOT NULL,
PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
```

For information about foreign key constraints, see Section 13.1.20.5, “FOREIGN KEY Constraints”.

#### 1.6.2.4 '--' as the Start of a Comment

Standard SQL uses the C syntax `/* this is a comment */` for comments, and MySQL Server supports this syntax as well. MySQL also support extensions to this syntax that enable MySQL-specific SQL to be embedded in the comment, as described in Section 9.7, “Comments”.

Standard SQL uses “`--`” as a start-comment sequence. MySQL Server uses `#` as the start comment character. MySQL Server also supports a variant of the `--` comment style. That is, the `--` start-comment sequence must be followed by a space (or by a control character such as a newline). The space is required to prevent problems with automatically generated SQL queries that use constructs such as the following, where we automatically insert the value of the payment for `payment`:

```
UPDATE account SET credit=credit-payment
```

Consider about what happens if `payment` has a negative value such as `-1`:

```
UPDATE account SET credit=credit--1
```

`credit--1` is a valid expression in SQL, but `--` is interpreted as the start of a comment, part of the expression is discarded. The result is a statement that has a completely different meaning than intended:

```
UPDATE account SET credit=credit
```

The statement produces no change in value at all. This illustrates that permitting comments to start with `--` can have serious consequences.

Using our implementation requires a space following the `--` for it to be recognized as a start-comment sequence in MySQL Server. Therefore, `credit--1` is safe to use.

### 1.6.3 How MySQL Deals with Constraints

MySQL enables you to work both with transactional tables that permit rollback and with nontransactional tables that do not. Because of this, constraint handling is a bit different in MySQL than in other DBMSs. We must handle the case when you have inserted or updated a lot of rows in a nontransactional table for which changes cannot be rolled back when an error occurs.

The basic philosophy is that MySQL Server tries to produce an error for anything that it can detect while parsing a statement to be executed, and tries to recover from any errors that occur while executing the statement. We do this in most cases, but not yet for all.

The options MySQL has when an error occurs are to stop the statement in the middle or to recover as well as possible from the problem and continue. By default, the server follows the latter course. This means, for example, that the server may coerce invalid values to the closest valid values.

Several SQL mode options are available to provide greater control over handling of bad data values and whether to continue statement execution or abort when errors occur. Using these options, you can configure MySQL Server to act in a more traditional fashion that is like other DBMSs that reject improper input. The SQL mode can be set globally at server startup to affect all clients. Individual clients can set the SQL mode at runtime, which enables each client to select the behavior most appropriate for its requirements. See Section 5.1.11, “Server SQL Modes”.

The following sections describe how MySQL Server handles different types of constraints.

#### 1.6.3.1 PRIMARY KEY and UNIQUE Index Constraints

Normally, errors occur for data-change statements (such as `INSERT` or `UPDATE`) that would violate primary-key, unique-key, or foreign-key constraints. If you are using a transactional storage engine such as `InnoDB`, MySQL automatically rolls back the statement. If you are using a nontransactional storage engine, MySQL stops processing the statement at the row for which the error occurred and leaves any remaining rows unprocessed.

MySQL supports an `IGNORE` keyword for `INSERT`, `UPDATE`, and so forth. If you use it, MySQL ignores primary-key or unique-key violations and continues processing with the next row. See the section for the statement that you are using (Section 13.2.7, “INSERT Statement”, Section 13.2.17, “UPDATE Statement”, and so forth).

You can get information about the number of rows actually inserted or updated with the `mysql_info()` C API function. You can also use the `SHOW WARNINGS` statement. See mysql_info(), and Section 13.7.7.42, “SHOW WARNINGS Statement”.

`InnoDB` and `NDB` tables support foreign keys. See Section 1.6.3.2, “FOREIGN KEY Constraints”.

### 1.6.3.2 FOREIGN KEY Constraints

Foreign keys let you cross-reference related data across tables, and foreign key constraints help keep this spread-out data consistent.

MySQL supports `ON UPDATE` and `ON DELETE` foreign key references in `CREATE TABLE` and `ALTER TABLE` statements. The available referential actions are `RESTRICT`, `CASCADE`, `SET NULL`, and `NO ACTION` (the default).

`SET DEFAULT` is also supported by the MySQL Server but is currently rejected as invalid by `InnoDB`. Since MySQL does not support deferred constraint checking, `NO ACTION` is treated as `RESTRICT`. For the exact syntax supported by MySQL for foreign keys, see Section 13.1.20.5, “FOREIGN KEY Constraints”.

`MATCH FULL`, `MATCH PARTIAL`, and `MATCH SIMPLE` are allowed, but their use should be avoided, as they cause the MySQL Server to ignore any `ON DELETE` or `ON UPDATE` clause used in the same statement. `MATCH` options do not have any other effect in MySQL, which in effect enforces `MATCH SIMPLE` semantics full-time.

MySQL requires that foreign key columns be indexed; if you create a table with a foreign key constraint but no index on a given column, an index is created.

You can obtain information about foreign keys from the Information Schema `KEY_COLUMN_USAGE` table. An example of a query against this table is shown here:

```
mysql> SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME
     > FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
     > WHERE REFERENCED_TABLE_SCHEMA IS NOT NULL;
+--------------+---------------+-------------+-----------------+
| TABLE_SCHEMA | TABLE_NAME    | COLUMN_NAME | CONSTRAINT_NAME |
+--------------+---------------+-------------+-----------------+
| fk1          | myuser        | myuser_id   | f               |
| fk1          | product_order | customer_id | f2              |
| fk1          | product_order | product_id  | f1              |
+--------------+---------------+-------------+-----------------+
3 rows in set (0.01 sec)
```

Information about foreign keys on `InnoDB` tables can also be found in the `INNODB_FOREIGN` and `INNODB_FOREIGN_COLS` tables, in the `INFORMATION_SCHEMA` database.

`InnoDB` and `NDB` tables support foreign keys.

### 1.6.3.3 Enforced Constraints on Invalid Data

By default, MySQL 8.0 rejects invalid or improper data values and aborts the statement in which they occur. It is possible to alter this behavior to be more forgiving of invalid values, such that the server

coerces them to valid ones for data entry, by disabling strict SQL mode (see Section 5.1.11, “Server SQL Modes”), but this is not recommended.

Older versions of MySQL employed the forgiving behavior by default; for a description of this behavior, see Constraints on Invalid Data.

### 1.6.3.4 ENUM and SET Constraints

`ENUM` and `SET` columns provide an efficient way to define columns that can contain only a given set of values. See Section 11.3.5, “The ENUM Type”, and Section 11.3.6, “The SET Type”.

Unless strict mode is disabled (not recommended, but see Section 5.1.11, “Server SQL Modes”), the definition of a `ENUM` or `SET` column acts as a constraint on values entered into the column. An error occurs for values that do not satisfy these conditions:

- An `ENUM` value must be one of those listed in the column definition, or the internal numeric equivalent thereof. The value cannot be the error value (that is, 0 or the empty string). For a column defined as `ENUM('a','b','c')`, values such as `''`, `'d'`, or `'ax'` are invalid and are rejected.
- A `SET` value must be the empty string or a value consisting only of the values listed in the column definition separated by commas. For a column defined as `SET('a','b','c')`, values such as `'d'` or `'a,b,c,d'` are invalid and are rejected.

Errors for invalid values can be suppressed in strict mode if you use `INSERT IGNORE` or `UPDATE IGNORE`. In this case, a warning is generated rather than an error. For `ENUM`, the value is inserted as the error member (`0`). For `SET`, the value is inserted as given except that any invalid substrings are deleted. For example, `'a,x,b,y'` results in a value of `'a,b'`.

## 1.7 Credits

The following sections list developers, contributors, and supporters that have helped to make MySQL what it is today.

### 1.7.1 Contributors to MySQL

Although Oracle Corporation and/or its affiliates own all copyrights in the `MySQL server` and the `MySQL manual`, we wish to recognize those who have made contributions of one kind or another to the `MySQL distribution`. Contributors are listed here, in somewhat random order:

- Gianmassimo Vigazzola `<qwerg@mbox.vol.it>` or `<qwerg@tin.it>`

  The initial port to Win32/NT.

- Per Eric Olsson

  For constructive criticism and real testing of the dynamic record format.

- Irena Pancirov `<irena@mail.yacc.it>`

  Win32 port with Borland compiler. `mysqlshutdown.exe` and `mysqlwatch.exe`.

- David J. Hughes

  For the effort to make a shareware SQL database. At TcX, the predecessor of MySQL AB, we started with `mSQL`, but found that it couldn't satisfy our purposes so instead we wrote an SQL interface to our application builder Unireg. `mysqladmin` and `mysql` client are programs that were largely influenced by their `mSQL` counterparts. We have put a lot of effort into making the MySQL syntax a superset of `mSQL`. Many of the API's ideas are borrowed from `mSQL` to make it easy to port free `mSQL` programs to the MySQL API. The MySQL software doesn't contain any code from `mSQL`. Two files in the distribution (`client/insert_test.c` and `client/select_test.c`) are based

on the corresponding (noncopyrighted) files in the `mSQL` distribution, but are modified as examples showing the changes necessary to convert code from `mSQL` to MySQL Server. (`mSQL` is copyrighted David J. Hughes.)

- Patrick Lynch

  For helping us acquire http://www.mysql.com/.

- Fred Lindberg

  For setting up qmail to handle the MySQL mailing list and for the incredible help we got in managing the MySQL mailing lists.

- Igor Romanenko `<igor@frog.kiev.ua>`

  `mysqldump` (previously `msqldump`, but ported and enhanced by Monty).

- Yuri Dario

  For keeping up and extending the MySQL OS/2 port.

- Tim Bunce

  Author of `mysqlhotcopy`.

- Zarko Mocnik `<zarko.mocnik@dem.si>`

  Sorting for Slovenian language.

- "TAMITO" `<tommy@valley.ne.jp>`

  The `_MB` character set macros and the ujis and sjis character sets.

- Joshua Chamas `<joshua@chamas.com>`

  Base for concurrent insert, extended date syntax, debugging on NT, and answering on the MySQL mailing list.

- Yves Carlier `<Yves.Carlier@rug.ac.be>`

  `mysqlaccess`, a program to show the access rights for a user.

- Rhys Jones `<rhys@wales.com>` (And GWE Technologies Limited)

  For one of the early JDBC drivers.

- Dr Xiaokun Kelvin ZHU `<X.Zhu@brad.ac.uk>`

  Further development of one of the early JDBC drivers and other MySQL-related Java tools.

- James Cooper `<pixel@organic.com>`

  For setting up a searchable mailing list archive at his site.

- Rick Mehalick `<Rick_Mehalick@i-o.com>`

  For `xmysql`, a graphical X client for MySQL Server.

- Doug Sisk `<sisk@wix.com>`

  For providing RPM packages of MySQL for Red Hat Linux.

- Diemand Alexander V. `<axeld@vial.ethz.ch>`

For providing RPM packages of MySQL for Red Hat Linux-Alpha.

- Antoni Pamies Olive <toni@readysoft.es>

  For providing RPM versions of a lot of MySQL clients for Intel and SPARC.

- Jay Bloodworth <jay@pathways.sde.state.sc.us>

  For providing RPM versions for MySQL 3.21.

- David Sacerdote <davids@secnet.com>

  Ideas for secure checking of DNS host names.

- Wei-Jou Chen <jou@nematic.ieo.nctu.edu.tw>

  Some support for Chinese(BIG5) characters.

- Wei He <hewei@mail.ied.ac.cn>

  A lot of functionality for the Chinese(GBK) character set.

- Jan Pazdziora <adelton@fi.muni.cz>

  Czech sorting order.

- Zeev Suraski <bourbon@netvision.net.il>

  `FROM_UNIXTIME()` time formatting, `ENCRYPT()` functions, and `bison` advisor. Active mailing list member.

- Luuk de Boer <luuk@wxs.nl>

  Ported (and extended) the benchmark suite to `DBI`/`DBD`. Have been of great help with `crash-me` and running benchmarks. Some new date functions. The `mysql_setpermission` script.

- Alexis Mikhailov <root@medinf.chuvashia.su>

  Loadable functions; `CREATE FUNCTION` and `DROP FUNCTION`.

- Andreas F. Bobak <bobak@relog.ch>

  The `AGGREGATE` extension to loadable functions.

- Ross Wakelin <R.Wakelin@march.co.uk>

  Help to set up InstallShield for MySQL-Win32.

- Jethro Wright III <jetman@li.net>

  The `libmysql.dll` library.

- James Pereria <jpereira@iafrica.com>

  Mysqlmanager, a Win32 GUI tool for administering MySQL Servers.

- Curt Sampson <cjs@portal.ca>

  Porting of MIT-pthreads to NetBSD/Alpha and NetBSD 1.3/i386.

- Martin Ramsch <m.ramsch@computer.org>

  Examples in the MySQL Tutorial.

- Steve Harvey

  For making `mysqlaccess` more secure.

- Konark IA-64 Centre of Persistent Systems Private Limited

  Help with the Win64 port of the MySQL server.

- Albert Chin-A-Young.

  Configure updates for Tru64, large file support and better TCP wrappers support.

- John Birrell

  Emulation of `pthread_mutex()` for OS/2.

- Benjamin Pflugmann

  Extended `MERGE` tables to handle `INSERTS`. Active member on the MySQL mailing lists.

- Jocelyn Fournier

  Excellent spotting and reporting innumerable bugs (especially in the MySQL 4.1 subquery code).

- Marc Liyanage

  Maintaining the OS X packages and providing invaluable feedback on how to create OS X packages.

- Robert Rutherford

  Providing invaluable information and feedback about the QNX port.

- Previous developers of NDB Cluster

  Lots of people were involved in various ways summer students, master thesis students, employees. In total more than 100 people so too many to mention here. Notable name is Ataullah Dabaghi who up until 1999 contributed around a third of the code base. A special thanks also to developers of the AXE system which provided much of the architectural foundations for NDB Cluster with blocks, signals and crash tracing functionality. Also credit should be given to those who believed in the ideas enough to allocate of their budgets for its development from 1992 to present time.

- Google Inc.

  We wish to recognize Google Inc. for contributions to the MySQL distribution: Mark Callaghan's SMP Performance patches and other patches.

Other contributors, bugfinders, and testers: James H. Thompson, Maurizio Menghini, Wojciech Tryc, Luca Berra, Zarko Mocnik, Wim Bonis, Elmar Haneke, `<jehamby@lightside>`, `<psmith@BayNetworks.com>`, `<duane@connect.com.au>`, Ted Deppner `<ted@psyber.com>`, Mike Simons, Jaakko Hyvatti.

And lots of bug report/patches from the folks on the mailing list.

A big tribute goes to those that help us answer questions on the MySQL mailing lists:

- Daniel Koch `<dkoch@amcity.com>`

  Irix setup.

- Luuk de Boer `<luuk@wxs.nl>`

  Benchmark questions.

- Tim Sailer `<tps@users.buoy.com>`

`DBD::mysql` questions.

- Boyd Lynn Gerber `<gerberb@zenez.com>`

  SCO-related questions.

- Richard Mehalick `<RM186061@shellus.com>`

  `xmysql`-related questions and basic installation questions.

- Zeev Suraski `<bourbon@netvision.net.il>`

  Apache module configuration questions (log & auth), PHP-related questions, SQL syntax-related questions and other general questions.

- Francesc Guasch `<frankie@citel.upc.es>`

  General questions.

- Jonathan J Smith `<jsmith@wtp.net>`

  Questions pertaining to OS-specifics with Linux, SQL syntax, and other things that might need some work.

- David Sklar `<sklar@student.net>`

  Using MySQL from PHP and Perl.

- Alistair MacDonald `<A.MacDonald@uel.ac.uk>`

  Is flexible and can handle Linux and perhaps HP-UX.

- John Lyon `<jlyon@imag.net>`

  Questions about installing MySQL on Linux systems, using either `.rpm` files or compiling from source.

- Lorvid Ltd. `<lorvid@WOLFENET.com>`

  Simple billing/license/support/copyright issues.

- Patrick Sherrill `<patrick@coconet.com>`

  ODBC and VisualC++ interface questions.

- Randy Harmon `<rjharmon@uptimecomputers.com>`

  `DBD`, Linux, some SQL syntax questions.

## 1.7.2 Documenters and translators

The following people have helped us with writing the MySQL documentation and translating the documentation or error messages in MySQL.

- Kim Aldale

  Helped to rewrite Monty's and David's early attempts at English into English.

- Michael J. Miller Jr. `<mke@terrapin.turbolift.com>`

  For the first MySQL manual. And a lot of spelling/language fixes for the FAQ (that turned into the MySQL manual a long time ago).

- Yan Cailin

  First translator of the MySQL Reference Manual into simplified Chinese in early 2000 on which the Big5 and HK coded versions were based.

- Jay Flaherty <fty@mediapulse.com>

  Big parts of the Perl DBI/DBD section in the manual.

- Paul Southworth <pauls@etext.org>, Ray Loyzaga <yar@cs.su.oz.au>

  Proof-reading of the Reference Manual.

- Therrien Gilbert <gilbert@ican.net>, Jean-Marc Pouyot <jmp@scalaire.fr>

  French error messages.

- Petr Snajdr, <snajdr@pvt.net>

  Czech error messages.

- Jaroslaw Lewandowski <jotel@itnet.com.pl>

  Polish error messages.

- Miguel Angel Fernandez Roiz

  Spanish error messages.

- Roy-Magne Mo <rmo@www.hivolda.no>

  Norwegian error messages and testing of MySQL 3.21.xx.

- Timur I. Bakeyev <root@timur.tatarstan.ru>

  Russian error messages.

- <brenno@dewinter.com> & Filippo Grassilli <phil@hyppo.com>

  Italian error messages.

- Dirk Munzinger <dirk@trinity.saar.de>

  German error messages.

- Billik Stefan <billik@sun.uniag.sk>

  Slovak error messages.

- Stefan Saroiu <tzoompy@cs.washington.edu>

  Romanian error messages.

- Peter Feher

  Hungarian error messages.

- Roberto M. Serqueira

  Portuguese error messages.

- Carsten H. Pedersen

  Danish error messages.

- Arjen Lentz

  Dutch error messages, completing earlier partial translation (also work on consistency and spelling).

## 1.7.3 Packages that support MySQL

The following is a list of creators/maintainers of some of the most important API/packages/applications that a lot of people use with MySQL.

We cannot list every possible package here because the list would then be way to hard to maintain. For other packages, please refer to the software portal at http://solutions.mysql.com/software/.

- Tim Bunce, Alligator Descartes

  For the `DBD` (Perl) interface.

- Andreas Koenig `<a.koenig@mind.de>`

  For the Perl interface for MySQL Server.

- Jochen Wiedmann `<wiedmann@neckar-alb.de>`

  For maintaining the Perl `DBD::mysql` module.

- Eugene Chan `<eugene@acenet.com.sg>`

  For porting PHP for MySQL Server.

- Georg Richter

  MySQL 4.1 testing and bug hunting. New PHP 5.0 `mysqli` extension (API) for use with MySQL 4.1 and up.

- Giovanni Maruzzelli `<maruzz@matrice.it>`

  For porting iODBC (Unix ODBC).

- Xavier Leroy `<Xavier.Leroy@inria.fr>`

  The author of LinuxThreads (used by the MySQL Server on Linux).

## 1.7.4 Tools that were used to create MySQL

The following is a list of some of the tools we have used to create MySQL. We use this to express our thanks to those that has created them as without these we could not have made MySQL what it is today.

- Free Software Foundation

  From whom we got an excellent compiler (`gcc`), an excellent debugger (`gdb` and the `libc` library (from which we have borrowed `strto.c` to get some code working in Linux).

- Free Software Foundation & The XEmacs development team

  For a really great editor/environment.

- Julian Seward

  Author of `valgrind`, an excellent memory checker tool that has helped us find a lot of otherwise hard to find bugs in MySQL.

- Dorothea Lütkehaus and Andreas Zeller

For DDD (The Data Display Debugger) which is an excellent graphical front end to gdb).

## 1.7.5 Supporters of MySQL

Although Oracle Corporation and/or its affiliates own all copyrights in the MySQL server and the MySQL manual, we wish to recognize the following companies, which helped us finance the development of the MySQL server, such as by paying us for developing a new feature or giving us hardware for development of the MySQL server.

- VA Linux / Andover.net

  Funded replication.

- NuSphere

  Editing of the MySQL manual.

- Stork Design studio

  The MySQL website in use between 1998-2000.

- Intel

  Contributed to development on Windows and Linux platforms.

- Compaq

  Contributed to Development on Linux/Alpha.

- SWSoft

  Development on the embedded mysqld version.

- FutureQuest

  The --skip-show-database option.

# Chapter 2 Installing and Upgrading MySQL

## Table of Contents

This chapter describes how to obtain and install MySQL. A summary of the procedure follows and later sections provide the details. If you plan to upgrade an existing version of MySQL to a newer version rather than install MySQL for the first time, see Section 2.10, “Upgrading MySQL”, for information about upgrade procedures and about issues that you should consider before upgrading.

If you are interested in migrating to MySQL from another database system, see Section A.8, “MySQL 8.0 FAQ: Migration”, which contains answers to some common questions concerning migration issues.

Installation of MySQL generally follows the steps outlined here:

1. **Determine whether MySQL runs and is supported on your platform.**

   Please note that not all platforms are equally suitable for running MySQL, and that not all platforms on which MySQL is known to run are officially supported by Oracle Corporation. For information about those platforms that are officially supported, see https://www.mysql.com/support/supportedplatforms/database.html on the MySQL website.

2. **Choose which distribution to install.**

   Several versions of MySQL are available, and most are available in several distribution formats. You can choose from pre-packaged distributions containing binary (precompiled) programs or source code. When in doubt, use a binary distribution. Oracle also provides access to the MySQL source code for those who want to see recent developments and test new code. To determine which version and type of distribution you should use, see Section 2.1.2, “Which MySQL Version and Distribution to Install”.

3. **Download the distribution that you want to install.**

   For instructions, see Section 2.1.3, “How to Get MySQL”. To verify the integrity of the distribution, use the instructions in Section 2.1.4, “Verifying Package Integrity Using MD5 Checksums or GnuPG”.

4. **Install the distribution.**

   To install MySQL from a binary distribution, use the instructions in Section 2.2, “Installing MySQL on Unix/Linux Using Generic Binaries”. Alternatively, use the Secure Deployment Guide, which provides procedures for deploying a generic binary distribution of MySQL Enterprise Edition Server with features for managing the security of your MySQL installation.

To install MySQL from a source distribution or from the current development source tree, use the instructions in Section 2.8, “Installing MySQL from Source”.

5. **Perform any necessary postinstallation setup.**

   After installing MySQL, see Section 2.9, “Postinstallation Setup and Testing” for information about making sure the MySQL server is working properly. Also refer to the information provided in Section 2.9.4, “Securing the Initial MySQL Account”. This section describes how to secure the initial MySQL `root` user account, *which has no password* until you assign one. The section applies whether you install MySQL using a binary or source distribution.

6. If you want to run the MySQL benchmark scripts, Perl support for MySQL must be available. See Section 2.12, “Perl Installation Notes”.

Instructions for installing MySQL on different platforms and environments is available on a platform by platform basis:

- **Unix, Linux**

  For instructions on installing MySQL on most Linux and Unix platforms using a generic binary (for example, a `.tar.gz` package), see Section 2.2, “Installing MySQL on Unix/Linux Using Generic Binaries”.

  For information on building MySQL entirely from the source code distributions or the source code repositories, see Section 2.8, “Installing MySQL from Source”

  For specific platform help on installation, configuration, and building from source see the corresponding platform section:

  - Linux, including notes on distribution specific methods, see Section 2.5, “Installing MySQL on Linux”.
  - IBM AIX, see Section 2.7, “Installing MySQL on Solaris”.

- **Microsoft Windows**

  For instructions on installing MySQL on Microsoft Windows, using either the MySQL Installer or Zipped binary, see Section 2.3, “Installing MySQL on Microsoft Windows”.

  For details and instructions on building MySQL from source code using Microsoft Visual Studio, see Section 2.8, “Installing MySQL from Source”.

- **macOS**

  For installation on macOS, including using both the binary package and native PKG formats, see Section 2.4, “Installing MySQL on macOS”.

  For information on making use of an macOS Launch Daemon to automatically start and stop MySQL, see Section 2.4.3, “Installing and Using the MySQL Launch Daemon”.

  For information on the MySQL Preference Pane, see Section 2.4.4, “Installing and Using the MySQL Preference Pane”.

# 2.1 General Installation Guidance

The immediately following sections contain the information necessary to choose, download, and verify your distribution. The instructions in later sections of the chapter describe how to install the distribution that you choose. For binary distributions, see the instructions at Section 2.2, “Installing MySQL on Unix/Linux Using Generic Binaries” or the corresponding section for your platform if available. To build MySQL from source, use the instructions in Section 2.8, “Installing MySQL from Source”.

## 2.1.1 Supported Platforms

MySQL platform support evolves over time; please refer to https://www.mysql.com/support/supportedplatforms/database.html for the latest updates.

## 2.1.2 Which MySQL Version and Distribution to Install

When preparing to install MySQL, decide which version and distribution format (binary or source) to use.

First, decide whether to install a development release or a General Availability (GA) release. Development releases have the newest features, but are not recommended for production use. GA releases, also called production or stable releases, are meant for production use. We recommend using the most recent GA release.

The naming scheme in MySQL 8.0 uses release names that consist of three numbers and an optional suffix (for example, **mysql-8.0.1-dmr**). The numbers within the release name are interpreted as follows:

- The first number (**8**) is the major version number.
- The second number (**0**) is the minor version number. Taken together, the major and minor numbers constitute the release series number. The series number describes the stable feature set.
- The third number (**1**) is the version number within the release series. This is incremented for each new bugfix release. In most cases, the most recent version within a series is the best choice.

Release names can also include a suffix to indicate the stability level of the release. Releases within a series progress through a set of suffixes to indicate how the stability level improves. The possible suffixes are:

- **dmr** indicates a development milestone release (DMR). MySQL development uses a milestone model, in which each milestone introduces a small subset of thoroughly tested features. From one milestone to the next, feature interfaces may change or features may even be removed, based on feedback provided by community members who try these early releases. Features within milestone releases may be considered to be of pre-production quality.
- **rc** indicates a Release Candidate (RC). Release candidates are believed to be stable, having passed all of MySQL's internal testing. New features may still be introduced in RC releases, but the focus shifts to fixing bugs to stabilize features introduced earlier within the series.
- Absence of a suffix indicates a General Availability (GA) or Production release. GA releases are stable, having successfully passed through the earlier release stages, and are believed to be reliable, free of serious bugs, and suitable for use in production systems.

Development within a series begins with DMR releases, followed by RC releases, and finally reaches GA status releases.

After choosing which MySQL version to install, decide which distribution format to install for your operating system. For most use cases, a binary distribution is the right choice. Binary distributions are available in native format for many platforms, such as RPM packages for Linux or DMG packages for macOS. Distributions are also available in more generic formats such as Zip archives or compressed `tar` files. On Windows, you can use the MySQL Installer to install a binary distribution.

Under some circumstances, it may be preferable to install MySQL from a source distribution:

- You want to install MySQL at some explicit location. The standard binary distributions are ready to run at any installation location, but you might require even more flexibility to place MySQL components where you want.

- You want to configure `mysqld` with features that might not be included in the standard binary distributions. Here is a list of the most common extra options used to ensure feature availability:
  - `-DWITH_LIBWRAP=1` for TCP wrappers support.
  - `-DWITH_ZLIB={system|bundled}` for features that depend on compression
  - `-DWITH_DEBUG=1` for debugging support

  For additional information, see Section 2.8.7, “MySQL Source-Configuration Options”.
- You want to configure `mysqld` without some features that are included in the standard binary distributions.
- You want to read or modify the C and C++ code that makes up MySQL. For this purpose, obtain a source distribution.
- Source distributions contain more tests and examples than binary distributions.

## 2.1.3 How to Get MySQL

Check our downloads page at https://dev.mysql.com/downloads/ for information about the current version of MySQL and for downloading instructions.

For RPM-based Linux platforms that use Yum as their package management system, MySQL can be installed using the MySQL Yum Repository. See Section 2.5.1, “Installing MySQL on Linux Using the MySQL Yum Repository” for details.

For Debian-based Linux platforms, MySQL can be installed using the MySQL APT Repository. See Section 2.5.2, “Installing MySQL on Linux Using the MySQL APT Repository” for details.

For SUSE Linux Enterprise Server (SLES) platforms, MySQL can be installed using the MySQL SLES Repository. See Section 2.5.3, “Installing MySQL on Linux Using the MySQL SLES Repository” for details.

To obtain the latest development source, see Section 2.8.5, “Installing MySQL Using a Development Source Tree”.

## 2.1.4 Verifying Package Integrity Using MD5 Checksums or GnuPG

After downloading the MySQL package that suits your needs and before attempting to install it, make sure that it is intact and has not been tampered with. There are three means of integrity checking:

- MD5 checksums
- Cryptographic signatures using `GnuPG`, the GNU Privacy Guard
- For RPM packages, the built-in RPM integrity verification mechanism

The following sections describe how to use these methods.

If you notice that the MD5 checksum or GPG signatures do not match, first try to download the respective package one more time, perhaps from another mirror site.

### 2.1.4.1 Verifying the MD5 Checksum

After you have downloaded a MySQL package, you should make sure that its MD5 checksum matches the one provided on the MySQL download pages. Each package has an individual checksum that you can verify against the package that you downloaded. The correct MD5 checksum is listed on the

downloads page for each MySQL product; you should compare it against the MD5 checksum of the file (product) that you download.

Each operating system and setup offers its own version of tools for checking the MD5 checksum. Typically the command is named `md5sum`, or it may be named `md5`, and some operating systems do not ship it at all. On Linux, it is part of the **GNU Text Utilities** package, which is available for a wide range of platforms. You can also download the source code from http://www.gnu.org/software/textutils/. If you have OpenSSL installed, you can use the command `openssl md5 package_name` instead. A Windows implementation of the `md5` command line utility is available from http://www.fourmilab.ch/md5/. `winMd5Sum` is a graphical MD5 checking tool that can be obtained from http://www.nullriver.com/index/products/winmd5sum. Our Microsoft Windows examples assume the name `md5.exe`.

Linux and Microsoft Windows examples:

```
$> md5sum mysql-standard-8.0.32-linux-i686.tar.gz
aaab65abbec64d5e907dcd41b8699945  mysql-standard-8.0.32-linux-i686.tar.gz
```

```
$> md5.exe mysql-installer-community-8.0.32.msi
aaab65abbec64d5e907dcd41b8699945  mysql-installer-community-8.0.32.msi
```

You should verify that the resulting checksum (the string of hexadecimal digits) matches the one displayed on the download page immediately below the respective package.

**Note**

Make sure to verify the checksum of the *archive file* (for example, the `.zip`, `.tar.gz`, or `.msi` file) and not of the files that are contained inside of the archive. In other words, verify the file before extracting its contents.

### 2.1.4.2 Signature Checking Using GnuPG

Another method of verifying the integrity and authenticity of a package is to use cryptographic signatures. This is more reliable than using MD5 checksums, but requires more work.

We sign MySQL downloadable packages with `GnuPG` (GNU Privacy Guard). `GnuPG` is an Open Source alternative to the well-known Pretty Good Privacy (`PGP`) by Phil Zimmermann. Most Linux distributions ship with `GnuPG` installed by default. Otherwise, see http://www.gnupg.org/ for more information about `GnuPG` and how to obtain and install it.

To verify the signature for a specific package, you first need to obtain a copy of our public GPG build key, which you can download from http://pgp.mit.edu/. The key that you want to obtain is named `mysql-build@oss.oracle.com`. The keyID for MySQL 8.0.28 packages and higher is `3A79BD29`. After obtaining this key, you should compare it with the key shown following, before using it verify MySQL packages. Alternatively, you can copy and paste the key directly from the text below.

**Note**

The following public GPG build key is for MySQL 8.0.28 packages and higher. For the public GPG build key for earlier MySQL release packages (keyID `5072E1F5`), see Section 2.1.4.5, “GPG Public Build Key for Archived Packages”.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: SKS 1.1.6
Comment: Hostname: pgp.mit.edu

mQINBGG4urcBEACrbsRa7tSSyxSfFkB+KXSbNM9rxYqoB78u107skReefq4/+Y72TpDvlDZL
mdv/lK0IpLa3bnvsM9IE1trNLrfi+JES62kaQ6hePPgn2RqxyIirt2seSi3Z3n3jlEg+mSdh
AvW+b+hFnqxo+TY0U+RBwDi4oO0YzHefkYPSmNPdlxRPQBMv4GPTNfxERx6XvVSPcL1+jQ4R
2cQFBryNhidBFIkoCOszjWhm+WnbURsLheBp757lqEyrpCufz77zlq2gEi+wtPHItfqsx3rz
xSRqatztMGYZpNUHNBJkr13npZtGW+kdN/xu980QLZxN+bZ88pNoOuzD6dKcpMJ0LkdUmTx5
z9ewiFiFbUDzZ7PECOm2g3veJrwr79CXDLE1+39Hr8rDM2kDhSr9tAlPTnHVDcaYIGgSNIBc
```

```
YfLmt91133klHQHBIdWCNVtWJjq5YcLQJ9TxG9GQzgABPrm6NDd1t9j7w1L7uwBvMB1wgpir
RTPVfnUSCd+025PEF+wTcBhfnzLtFj5xD7mNsmDmeHkF/sDfNOfAzTE1v2wq0ndYU60xbL6/
yl/Nipyr7WiQjCG0m3WfkjjVDTfs7/DXUqHFDOu4WMF9v+oqwpJXmAeGhQTWZC/QhWtrjrNJ
AgwKpp263gDSdW70ekhRzsok1HJwX1SfxHJYCMFs2aH6ppzNsQARAQABtDZNeVNRTCBSZWxl
YXNlIEVuZ2luZWVyaW5nIDxteXNxbC1idWlsZEBvc3Mub3JhY2xlLmNvbT6JAlQEEwEIAD4W
IQSFm+jXxYb1OEMLGcJGe5QtOnm9KQUCYbi6twIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgID
AQIeAQIXgAAKCRBGe5QtOnm9KUewD/992sS31WLGoUQ6NoL7qOB4CErkqXtMzpJAKKg2jtBG
G3rKE1/0VAg1D8AwEK4LcCO407wohnH0hNiUbeDck5x20pgS5SplQpuXX1K9vPzHeL/WNTb9
8S3H2Mzj4o9obED6Ey52tTupttMF8pC9TJ93LxbJlCHIKKwCA1cXud3GycRN72eqSqZfJGds
aeWLmFmHf6oee27d8XLoNjbyAxna/4jdWoTqmp8oT3bgv/TBco23NzqUSVPi+7ljS1hHvcJu
oJYqaztGrAEf/lWIGdfl/kLEh8IYx8OBNUojh9mzCDlwbs83CBqoUdlzLNDdwmzu34Aw7xK1
4RAVinGFCpo/7EWoX6weyB/zqevUIIE89UABTeFoGih/hx2jdQV/NQNthWTW0jH0hmPnajBV
AJPYwAuO82rx2pnZCxDATMn0elOkTue3PCmzHBF/GT6c65aQC4aojj0+Veh787QllQ9FrWbw
nTz+4fNzU/MBZtyLZ4JnsiWUs9eJ2V1g/A+RiIKu357Qgy1ytLqlgYiWfzHFlYjdtbPYKjDa
ScnvtY8VO2Rktm7XiV4zKFKiaWp+vuVYpR0/7Adgnlj5Jt9lQQGOr+Z2VYx8SvBcC+by3XAt
YkRHtX5u4MLlVS3gcoWfDiWwCpvqdK21EsXjQJxRr3dbSn0HaVj4FJZX0QQ7WZm6WLkCDQRh
uLq3ARAA6RYjqfC0YcLGKvHhoBnsX29vy9Wn1y2JYpEnPUIB8X0VOyz5/ALv4Hqtl4THkH+m
mMuhtndoq2BkCCk508jWBvKS1S+Bd2esB45BDDmIhuX3ozu9Xza4i1FsPnLkQ0uMZJv30ls2
pXFmskhYyzmo6aOmH2536LdtPSlXtywfNV1HEr69V/AHbrEzfoQkJ/qvPzELBOjfjwtDPDeP
iVgW9LhktzVzn/BjO7XlJxw4PGcxJG6VApsXmM3t2fPN9eIHDUq8ocbHdJ4en8/bJDXZd9eb
QoILUuCg46hE3p6nTXfnPwSRnIRnsgCzeAz4rxDR4/Gv1Xpzv5wqpL21XQi3nvZKlcv7J1IR
VdphK66De9GpVQVTqC102gqJUErdjGmxmyCA1OOORqEPfKTrXz5YUGsWwpH+4xCuNQP0qmre
Rw3ghrH8potIr0iOVXFic5vJfBTgtcuEB6E6ulAN+3jqBGTaBML0jxgj3Z5VC5HKVbpg2DbB
/wMrLwFHNAbzV5hj2Os5Zmva0ySP1YHB26pAW8dwB38GBaQvfZq3ezM4cRAo/iJ/GsVE98dZ
EBO+Ml+0KYj+ZG+vyxzo20sweun7ZKT+9qZM90f6cQ3zqX6IfXZHHmQJBNv73mcZWNhDQOHs
4wBoq+FGQWNqLU9xaZxdXw80r1viDAwOy13EUtcVbTkAEQEAAYkCPAQYAQgAJhYhBIWb6NfF
hvU4QwsZwkZ7lC06eb0pBQJhuLq3AhsMBQkDwmcAAAoJEEZ7lC06eb0pSi8P/iy+dNnxrtiE
Nn9vkkA7AmZ8RsvPXYVeDCDSsL7UfhbS77r2L1qTa2aB3gAZUDIOXln51lSxMeeLtOequLME
V2Xi5km70rdtnja5SmWfc9fyExunXnsOhg6UG872At5CGEZU0c2Nt/hlGtOR3xbt3O/Uwl+d
ErQPA4BUbW5K1T7OC6oPvtlKfF4bGZFloHgt2yE9YSNWZsTPe6XJSapemHZLPOxJLnhs3VBi
rWE31QS0bRl5AzlO/fg7ia65vQGMOCOTLpgChTbcZHtozeFqva4IeEgE4xN+6r8WtgSYeGGD
RmeMEVjPM9dzQObf+SvGd58u2z9f2agPK1H32c69RLoA0mHRe7Wkv4izeJUc5tumUY0e8Ojd
enZZjT3hjLh6tM+mrp2oWnQIoed4LxUw1dhMOj0rYXv6laLGJ1FsW5eSke7ohBLcfBBTKnMC
BohROHy2E63Wggfsdn3UYzfqZ8cfbXetkXuLS/OM3MXbiNjg+ElYzjgWrkayu7yLakZx+mx6
sHPIJYm2hzkniMG29d5mGl7ZT9emP9b+CfqGUxoXJkjs0gnDl44bwGJ0dmIBu3ajVAaHODXy
Y/zdDMGjskfEYbNXCAY2FRZSE58tgTvPKD++Kd2KGplMU2EIFT7JYfKhHAB5DGMkx92HUMid
sTSKHe+QnnnoFmu4gnmDU31i
=Xqbo
-----END PGP PUBLIC KEY BLOCK-----
```

To import the build key into your personal public GPG keyring, use `gpg --import`. For example, if you have saved the key in a file named `mysql_pubkey.asc`, the import command looks like this:

```
$> gpg --import mysql_pubkey.asc
gpg: key 3A79BD29: public key "MySQL Release Engineering
<mysql-build@oss.oracle.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

You can also download the key from the public keyserver using the public key id, `3A79BD29`:

```
$> gpg --recv-keys 3A79BD29
gpg: requesting key 3A79BD29 from hkp server keys.gnupg.net
gpg: key 3A79BD29: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
1 new user ID
gpg: key 3A79BD29: "MySQL Release Engineering <mysql-build@oss.oracle.com>"
53 new signatures
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:           new user IDs: 1
gpg:         new signatures: 53
```

If you want to import the key into your RPM configuration to validate RPM install packages, you should be able to import the key directly:

```
$> rpm --import mysql_pubkey.asc
```

If you experience problems or require RPM specific information, see Section 2.1.4.4, “Signature Checking Using RPM”.

After you have downloaded and imported the public build key, download your desired MySQL package and the corresponding signature, which also is available from the download page. The signature file has the same name as the distribution file with an `.asc` extension, as shown by the examples in the following table.

**Table 2.1 MySQL Package and Signature Files for Source files**

| File Type | File Name |
| --- | --- |
| Distribution file | `mysql-standard-8.0.32-linux-i686.tar.gz` |
| Signature file | `mysql-standard-8.0.32-linux-i686.tar.gz.asc` |

Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file:

```
$> gpg --verify package_name.asc
```

If the downloaded package is valid, you should see a `Good signature` message similar to this:

```
$> gpg --verify mysql-standard-8.0.32-linux-i686.tar.gz.asc
gpg: Signature made Tue 01 Feb 2011 02:38:30 AM CST using DSA key ID 3A79BD29
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
```

The `Good signature` message indicates that the file signature is valid, when compared to the signature listed on our site. But you might also see warnings, like so:

```
$> gpg --verify mysql-standard-8.0.32-linux-i686.tar.gz.asc
gpg: Signature made Wed 23 Jan 2013 02:25:45 AM PST using DSA key ID 3A79BD29
gpg: checking the trustdb
gpg: no ultimately trusted keys found
gpg: Good signature from "MySQL Release Engineering <mysql-build@oss.oracle.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5
```

That is normal, as they depend on your setup and configuration. Here are explanations for these warnings:

- *gpg: no ultimately trusted keys found*: This means that the specific key is not "ultimately trusted" by you or your web of trust, which is okay for the purposes of verifying file signatures.
- *WARNING: This key is not certified with a trusted signature! There is no indication that the signature belongs to the owner.*: This refers to your level of trust in your belief that you possess our real public key. This is a personal decision. Ideally, a MySQL developer would hand you the key in person, but more commonly, you downloaded it. Was the download tampered with? Probably not, but this decision is up to you. Setting up a web of trust is one method for trusting them.

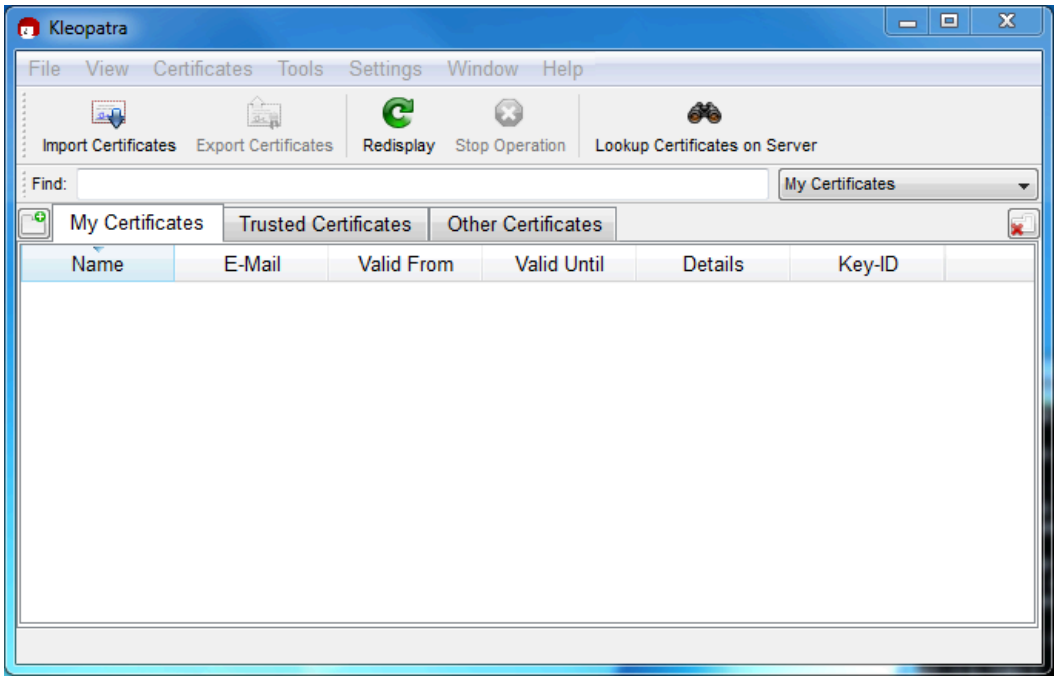
See the GPG documentation for more information on how to work with public keys.

### 2.1.4.3 Signature Checking Using Gpg4win for Windows

The Section 2.1.4.2, “Signature Checking Using GnuPG” section describes how to verify MySQL downloads using GPG. That guide also applies to Microsoft Windows, but another option is to use a GUI tool like Gpg4win. You may use a different tool but our examples are based on Gpg4win, and utilize its bundled `Kleopatra` GUI.

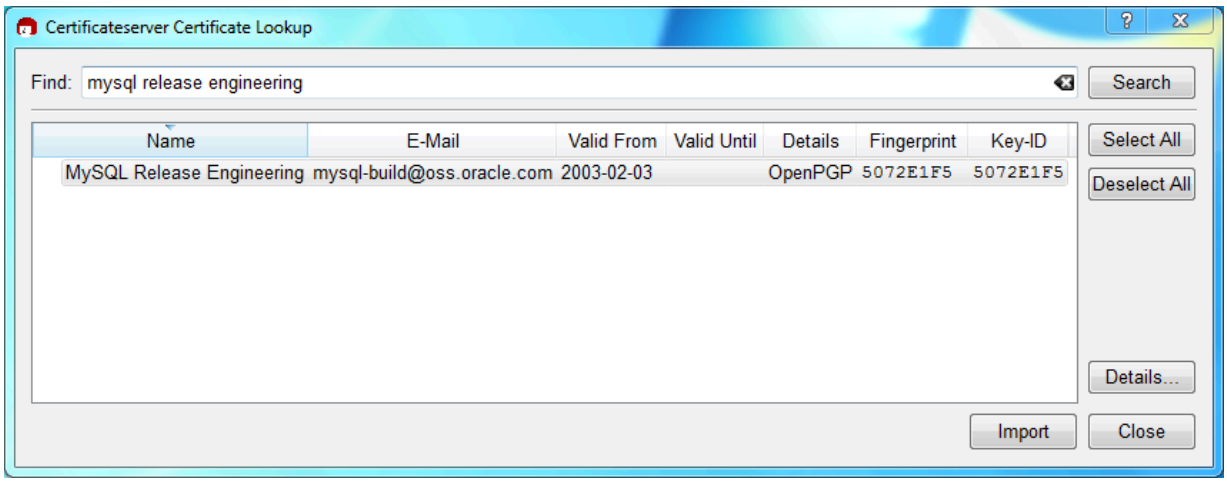
Download and install Gpg4win, and then load Kleopatra. The dialog should look similar to:

**Figure 2.1 Kleopatra: Initial Screen**




Next, add the MySQL Release Engineering certificate. Do this by clicking **File**, **Lookup Certificates on Server**. Type "Mysql Release Engineering" into the search box and press **Search**.

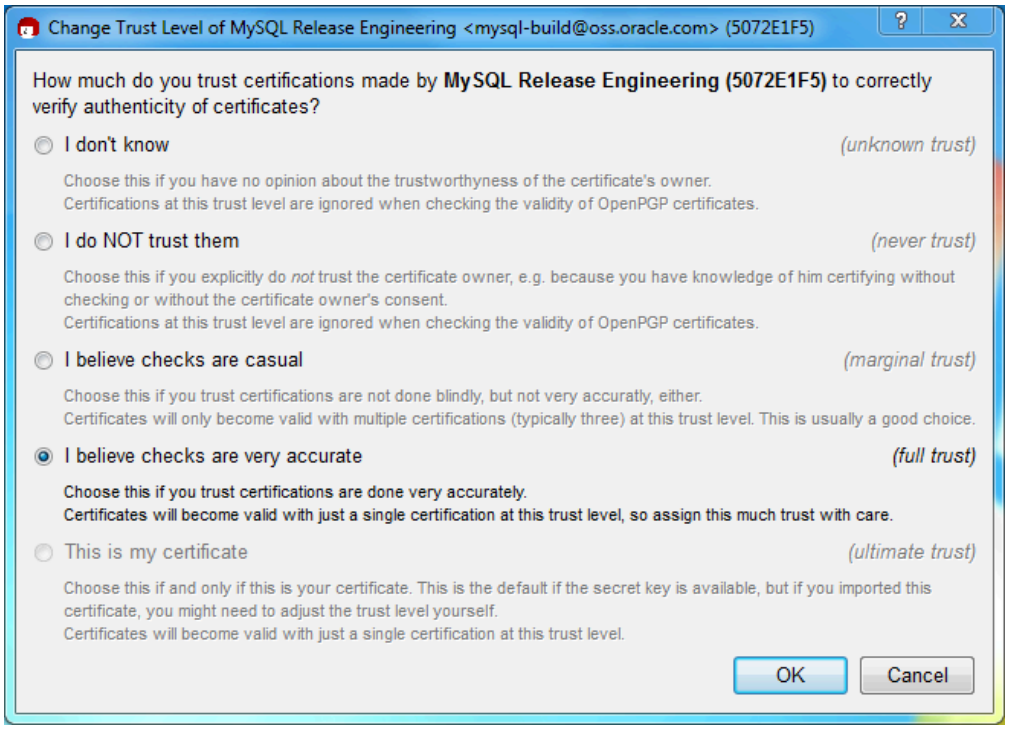
**Figure 2.2 Kleopatra: Lookup Certificates on Server Wizard: Finding a Certificate**




Select the "MySQL Release Engineering" certificate. The Fingerprint and Key-ID must be "3A79BD29" for MySQL 8.0.28 and higher or "5072E1F5" for MySQL 8.0.27 and earlier, or choose **Details...** to confirm the certificate is valid. Now, import it by clicking **Import**. When the import dialog is displayed, choose **Okay**, and this certificate should now be listed under the **Imported Certificates** tab.

Next, configure the trust level for our certificate. Select our certificate, then from the main menu select **Certificates**, **Change Owner Trust...**. We suggest choosing **I believe checks are very accurate** for our certificate, as otherwise you might not be able to verify our signature. Select **I believe checks are very accurate** to enable "full trust" and then press **OK**.

**Figure 2.3 Kleopatra: Change Trust level for MySQL Release Engineering**



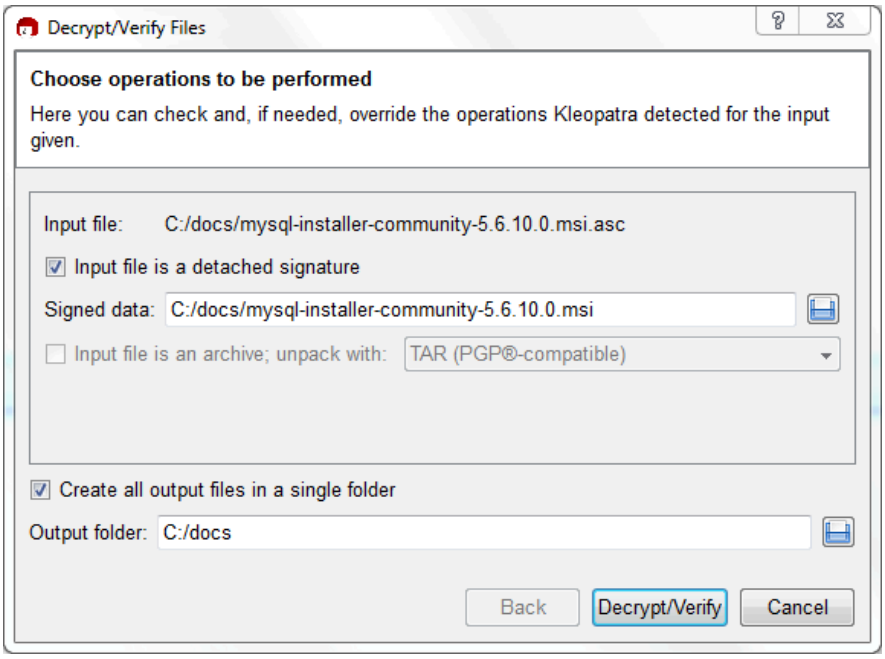

Next, verify the downloaded MySQL package file. This requires files for both the packaged file, and the signature. The signature file must have the same name as the packaged file but with an appended `.asc` extension, as shown by the example in the following table. The signature is linked to on the downloads page for each MySQL product. You must create the `.asc` file with this signature.

**Table 2.2 MySQL Package and Signature Files for MySQL Installer for Microsoft Windows**

| File Type | File Name |
| --- | --- |
| Distribution file | `mysql-installer-community-8.0.32.msi` |
| Signature file | `mysql-installer-community-8.0.32.msi.asc` |

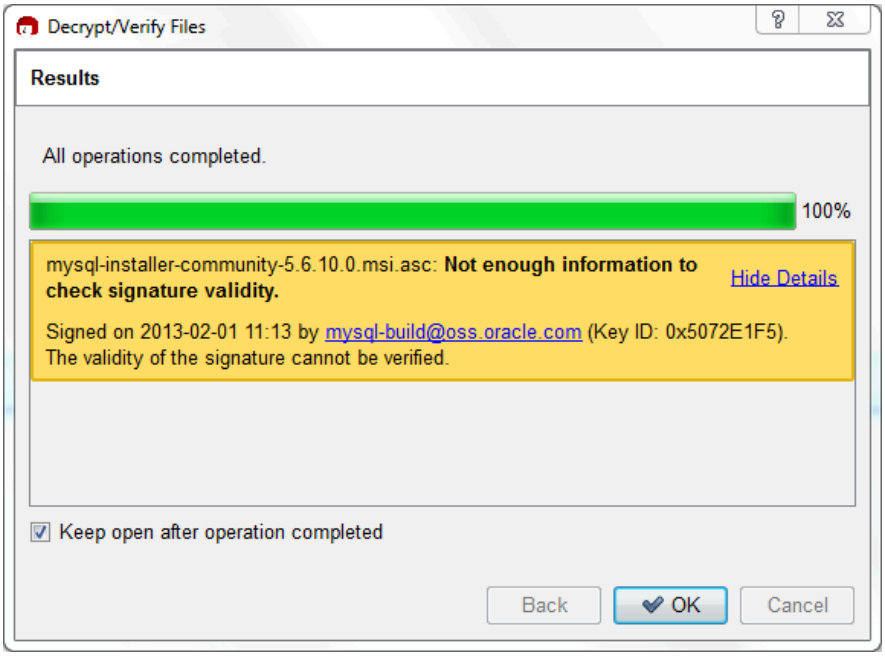
Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file. Either drag and drop the signature (`.asc`) file into Kleopatra, or load the dialog from **File**, **Decrypt/Verify Files...**, and then choose either the `.msi` or `.asc` file.

**Figure 2.4 Kleopatra: The Decrypt and Verify Files Dialog**



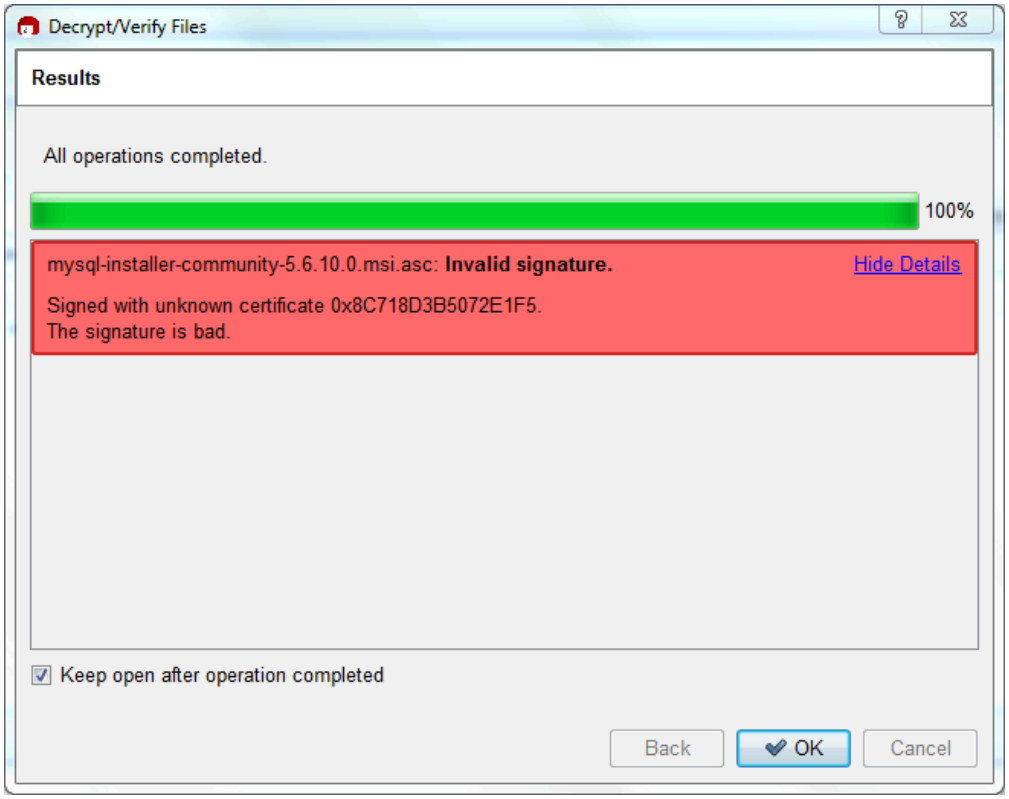

Click **Decrypt/Verify** to check the file. The two most common results look like the following figure; although the yellow warning may look problematic, the following means that the file check passed with success. You may now run this installer.

**Figure 2.5 Kleopatra: the Decrypt and Verify Results Dialog: All operations completed**




Seeing a red `The signature is bad` error means the file is invalid. Do not execute the MSI file if you see this error.

**Figure 2.6 Kleopatra: the Decrypt and Verify Results Dialog: Bad**




The Section 2.1.4.2, “Signature Checking Using GnuPG”, section explains why you do not see a green `Good signature` result.

### 2.1.4.4 Signature Checking Using RPM

For RPM packages, there is no separate signature. RPM packages have a built-in GPG signature and MD5 checksum. You can verify a package by running the following command:

```
$> rpm --checksig package_name.rpm
```

Example:

```
$> rpm --checksig mysql-community-server-8.0.32-1.el8.x86_64.rpm
mysql-community-server-8.0.32-1.el8.x86_64.rpm: digests signatures OK
```

**Note**

If you are using RPM 4.1 and it complains about `(GPG) NOT OK (MISSING KEYS: GPG#3a79bd29)`, even though you have imported the MySQL public build key into your own GPG keyring, you need to import the key into the RPM keyring first. RPM 4.1 no longer uses your personal GPG keyring (or GPG itself). Rather, RPM maintains a separate keyring because it is a system-wide application and a user's GPG public keyring is a user-specific file. To import the MySQL public key into the RPM keyring, first obtain the key, then use `rpm --import` to import the key. For example:

```
$> gpg --export -a 3a79bd29 > 3a79bd29.asc
$> rpm --import 3a79bd29.asc
```

Alternatively, `rpm` also supports loading the key directly from a URL:

```
$> rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
```

You can also obtain the MySQL public key from this manual page: Section 2.1.4.2, “Signature Checking Using GnuPG”.

### 2.1.4.5 GPG Public Build Key for Archived Packages

The following GPG public build key (keyID 5072E1F5) can be used to verify the authenticity and integrity of MySQL 8.0.27 packages and earlier. For signature checking instructions, see Section 2.1.4.2, “Signature Checking Using GnuPG”.

#### GPG Public Build Key for MySQL 8.0.27 Packages and Earlier

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: SKS 1.1.6
Comment: Hostname: pgp.mit.edu

mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3RODjQRey
CITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZfw2vOUgCmYv2hW0h
yDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3BqOxRznNCRCRxAuAuVztHRcE
AJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNWhxwR9pRWVArNYJdDRT+rf2RUe3vpquKN
QU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLVK2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sB
gACyP/Vb7hiPwxh6rDZ7ITnEkYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgt
obZX9qnrAXw+uNDIQJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAn
WqcyefeprV0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q2TXlTUUwgUmVs
ZWFzZSBFbmdpbmVlcmluZyA8bXlzcWwtYnVpbGRAb3NzLm9yYWNsZS5jb20+iEYEEBECAAYF
AlldBJ4ACgkQvcMmpx2w8a2MYQCgga9wXfwOe/52xg0RTkhsbDQhvdAAn30njwoLBhKdDBxk
hVmwZQvzdYYNiGYEExECACYCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAUCTnc+KgUJE/sC
FQAKCRCMcY07UHLh9SbMAJ4l1+qBz2BZNSGCZwwA6YbhGPC7FwCgp8z5TzIw4YQuL5NGJ/sy
0oSazqmIZgQTEQIAJgUCTnc9dgIbIwUJEPPzpwYLCQgHAwIEFQIIAwQWAgMBAh4BAheAAAoJ
EIxxjTtQcuH1Ut4AoIKjhdf70899d+7JFq3LD7zeeyI0AJ9Z+YyE1HZSnzYi73brScilbIV6
sYhpBBMRAgApAhsjBgsJCAcDAgQVAggDBBYCAwECHgECF4ACGQEFAlGUkToFCRU3IaoACgkQ
jHGNO1By4fWLQACfV6wP8ppZqMz2Z/gPZbPP7sDHE7EAn2kDDatXTZIR9pMgcnN0cff1tsX6
iGkEExECACkCGyMGCwkIBwMCBBUCCAMEFgIDAQIeAQIXgAIZAQUCUwHUZgUJGmbLywAKCRCM
cY07UHLh9V+DAKCjS1gGwgVI/eut+5L+l2v3ybl+ZgCcD7ZoA341HtoroV3U6xRD09fUgeqI
bAQTEQIALAIbIwIeAQIXgAIZAQYLCQgHAwIGFQoJCAIDBRYCAwEABQJYpXsIBQkeKT7NAAoJ
EIxxjTtQcuH1wrMAnRGuZVbriMR077KTGAVhJF2uKJiPAJ9rCpXYFve2IdxST2i7w8nygefV
a4hsBBMRAgAsAhsjAh4BAheAAhkBBgsJCAcDAgYVCgkIAgMFFgIDAQAFAlinBSAFCR4qyRQA
CgkQjHGNO1By4fVXBQCeOqVMlXfAWdq+QqaTAtbZskN3HkYAn1T8LlbIktFREeVlKrQEA7fg
6HrQiGwEExECACwCGyMCHgECF4ACGQEGCwkIBwMCBhUKCQgCAwUWAgMBAAUCXEBY+wUJI87e
5AAKCRCMcY07UHLh9RZPAJ9uvm0zlzfCN+DHxHVaoFLFjdVYTQCfborsC9tmEZYawhhogjeB
kZkorbyJARwEEAECAAYFAlAS6+UACgkQ8aIC+GoXHivrWwf/dtLk/x+NC2VMDlg+vOeM0qgG
1IlhXZfiNsEisvvGaz4m8fSFRGe+1bvvfDoKRhxiGXU48RusjixzvBb6KTMuY6JpOVfz9Dj3
H9spYriHa+i6rYySXZIpOhfLiMnTy7NH2OvYCyNzSS/ciIUACIfH/2NH8zNT5CNF1uPNRs7H
sHzzz7pOlTjtTWiF4cq/Ij6Z6CNrmdj+SiMvjYN9u6sdEKGtoNtpycgD5HGKR+I7Nd/7v56y
haUe4FpuvsNXig86K9tI6MUFS8CUyy7Hj3kVBZOUWVBM053knGdALSygQr50DA3jMGKVl4Zn
Hje2RVWRmFTr5YWoRTMxUSQPMLpBNIkBHAQQAQIABgUCU1B+vQAKCRAohbcD0zcc8dWwCACW
XXWDXIcAWRUw+j3ph8dr9u3SItljn3wBc7clpclKWPuLvTz7lGgzlVB0s8hH4xgkSA+zLzl6
u56mpUzskFl7f1I3Ac9GGpM40M5vmmR9hwlD1HdZtGfbD+wkjlqgitNLoRcGdRf/+U7x09Gh
SS7Bf339sunIX6sMgXSC4L32D3zDjF5icGdb0kj+3lCrRmp853dGyA3ff9yUiBkxcKNawpi7
Vz3D2ddUpOF3BP+8NKPg4P2+srKgkFbd4HidcISQCt3rY4vaTkEkLKg0nNA6U4r0YgOa7wIT
SsxFlntMMzaRg53QtK0+YkH0KuZR3GY8B7pi+tlgycyVR7mIFo7riQEcBBABAgAGBQJcSESc
AAoJENwpi/UwTWr2X/YH/0JLr/qBW7cDIx9admk5+vjPoUl6U6SGzCkIlfK24j90kU0oJxDn
FVwc9tcxGtxK8n6AEc5G0FQzjuXeYQ1SAHXquZ9CeGjidmsrRLVKXwOIcFZPBmfS9JBzdHa9
W1b99NWHOehWWnyIITVZ1KeBLbI7uoyXkvZgVp0REd37XWGgYEhT0JwAXnk4obH6djY3T/Hf
D70piuvFU7w84IRAqevUcaDppU/1QluDiOnViq6MAki85Z+uoM6ojUZtwmqXDSYIPzRHctfx
Vdv3HS423RUvcfpMUGG94r7tTOSXhHS9rcs6lzLnKl84J0xzI5bWS/Fw+5h40Gpd4HTR/kiE
Xu2JARwEEAEIAAYFAlaBV3QACgkQRm7hv+CThQqT0wf9Ge3sRxw+NIkLkKsHYBTktjYOyv49
48ja5s9awR0bzapKOMaluEgfwtKD8/NCgYeIVYyaZlYmS1FP51yAtuzdvZXAI0DAITyM4d1S
RCESjCCiZ028eIEcoeM/j+UXrwo4+I7/abFhiSakzsFZ/eQHnsMnkJOLf8kug3vMXjSoiz+n
T14++fBK2mCVtu1Sftc877X8R7xUfOKYAGibnY+RAi7E2JVTMtWfdtJaqt3l5y6ouTrLOM9d
3ZeEMdYL1PCmXrwZ4+u7oTNC26yLSbpL+weAReqH8jGsVlUmWWMXvkm+ixmrnN66WvSLqQ6K
P5jWnowV9+KEhNnWBOaT4Iu8rYkBIgQQAQIADAUCTndBLgUDABJ1AAAKCRCXELibyletfAnx
B/9t79Q72ap+hzawzKHAyk3j990FbB8uQDXYVdAM5Ay/Af0eyYSOd9SBgpexyFlGL4O4dd7U
/uXwbZpAu5uEGxB/16Mq9EVPO5YxCR0ir7oqi6XG/qh+QJy/d3XG07ZbudvnLFylUE+tF8YU
Z5sm9lrnwPKYI2DIa0BToA7Pi95q82Yjb4YgNCxjrr61gO9n4LHDN1i74cNX0easl9zp14zS
acGftJGOrPEk+ChNCGKFNq/qr9Hn/ank29D8fzg6BLoaOix8ZzZ25QPMI/+SF4xEp/O7IoI4
dA+0m4iPz76B+ke0RTsgNRfVKjdz2fQ92l4G9yWwNulGcI3FBZTiYGi3iQEiBBABAgAMBQJO
iLYZBQMAEnUAAAoJEJcQuJvKV618tkAH/2hGrH40L3xRAP/CXEJHK3O+L8y4+duBBQ8scRqn
XS28SLfdL8f/ENH+1wah9jhyMC+jmyRldd5ar3cC/s8AJRvOSDRfR5KvagvrDLrrF+i/vYDB
K5f6JQrryq0poupEuK0zTbLxo1FX+CAq+3tQy8aY6+znItpiWhvK8ZoULYKV+Q063YyVWdBk
KadgELA6S08aQTGK7bJkyJ9xgbFBykcpUUbn0p4XZwzZ3jFgzwcmqRIYZbfTosVVLJ5HAb7B
u22AukPlsz9PZvd8X8nfmtoJIwtl5qtFOrxrKA+X5czswzZ5H3jprDqOY6yA0EStu+8h1CPo
u50BmP7yKZxdXYqJASIEEAECAAwFAk6Z2dEFAwASdQAACgkQlxC4m8pXrXwC8ggAgQXVkn5H
LtY50oXmh5D/KdphSKDM33Z9b/3MHzK5CWeCQUkaJ1gxtyLW1HWyLOIhUkW6xHdmieoA8Yr9
JS1r1jopYuGZztzlScQeSWr8190xnZZVIjKReVy2rDSxtv7PV5wR3gby72PmKWUw7UHfqtBr
```

```
JgA+h5ctfx1jhXIUtUZpDTStZAFgVmunDXoBNZtYYk/ffY1J8KTjNmrqRcRbTurSy3dgGAAA
Z01DIR5kJrh3ikFFJfrXz0qODoYOchxqI4Xoc7o8uv19GUuvk5sKBT4b2ASF+JXAMRX0T7v8
Gralhn3CGGQGpZDN2ldM1Mzbi5oSETTUQ87nN4I7bXirqYkBIgQQAQIADAUCTqumAQUDABJ1
AAAKCRCXELibyletfMCHB/9/0733PXrdjkVlUjF7HKpdD8xy324oe5cRWdEVhsDj11AsPhLv
c37M3uCf2MV5BwGjjDypVRX3hT+1r9VsuR201ETKmU8zhdjxgTlZ931t/KDerU9sSJWOT33m
wEX7b5Oj31hgqy2Bc+qOUfSNR8TIOZ7E6P6GynxFzreS+QjHfpUFrg41FgV58YCEoMyKAvZg
CFzVSQa2QZO4uaUIbAhXqW+INkPdEl/nfvlUWdoe/t5d/BDELAT4HEbcJRGuN/GNrExOYw/I
AbauEOnmhNQS+oNg1uSjlTFg6atKO8XgXNfCp6sSVclSRTNKHSmntHEcH/WULEOzsPUXWGWA
VC40iQEiBBABAgAMBQJOvNkcBQMAEnUAAAoJEJcQuJvKV618xSkH/izTt1ERQsgGcDUPqqvd
8exAk1mpsC7IOW+AYYtbOjIQOz7UkwUWVpr4R4sijXfzoZTYNqaYMLbencgHv25CEl4PZnVN
xWDhwDrhJ8X8Idxrlyh5FKt0CK53NT9yAsa1cg/85oVqZeB0zECGWgsVtIc8JmTJvTSmFVrz
7F4hUOsrUcHJmw0hfL9JIrxTbpLY9VnajXh9a8psnUCBrw3oO5Zj8Pw/aaLdEBuK5mB/OSYo
vmJ0f/BIp+cUp1OAnOyx0JzWNkQZWTmsVhxY6skBEd4/+2ydv9TEoESw207t7c3Z7+stWcTK
RUg7TrqHPvFkr9U0FKnHeTeqPhc8rjUgfLaJASIEEAECAAwFAk7Oo7wFAwASdQAACgkQlxC4
m8pXrXza3Af/QjONcvE3jme8h8SMLvlr6L1lIuWpHyWwcvgakRJwUojRrSVPghUAhjZEob4w
CzZ4ebRR8q7AazmOW5Fn1GoqtzrWxjRdBX3/vOdj0NvXqCFfTgmOSc4qz98+Lzuu8qQH9DEl
ZLyptv96tGZb5w82NtHFMU9LkkjAVYcDXqJ4USm90CApXqd+8lVOrWuM8NycgD0Ik3ZKZQXH
1DHdJFzohNtqbWGMWdjqwKHoBSHEsjZ/WarXEf0+oTLjZSbrymtGpPInsijHWD9QMOR55RwC
DtPW+JPPu5elLdaurjPOjjI6lol8sNHekjmDZmRI0ZMyjprJITg4AG3yLU9zU+boCYkBIgQQ
AQIADAUCTvI8VgUDABJ1AAAKCRCXELibyletfNeIB/0Wtd7SWBw8z61g5YwuG/mBcmLZVQFo
vGnJFeb+QlybEicqrUYJ3fIPj8Usc27dlwLP+6SU8BtldYjQ7p7CrQtaxG2SWYmNaJ50f6Eb
JpO/3lWSWiNEgF3ycFonoz3yuWMwEdMXBa+NAVV/gUtElBmoeW+NwKSrYN30FYmkZe+v+Ckq
SYwlg0r9+19lFwKFvfk0jX1ZGk6GP27zTw49yopW9kFw/AUZXlwQHOYAL3gnslwPz5LwiTyJ
QkxAYYvdByZk4GjOi+HzqGPspNIQEeUteXzfbPz0fWEt64tudegYu/fN5QVLGS/WHfkuFkuo
gwNBFcu5TPEYcwGkuE/IZZEniQEiBBABAgAMBQJPBAkXBQMAEnUAAAoJEJcQuJvKV618AG8H
/0LLr1yM6osbLAhOzcKmD//jSOZ2FIBXGKUmK8/onlu5sUcMmVVPHnjUO/mHiFMbYFC655Di
rVUKlIZb6sdx2E/K+ZkZHPWvF1BAaHUO/QGh3Zzc8lVJg9KtFLAJkmQkc61VEF2MriaRlvlo
VPNr5Oiv2THOPgVxdV3goBL6EdAdgdwCvy23Z44vOp0QVNQt4aJKg2f49XO/N1+Gd2mEr7wX
aN9DZQq5zTU7uTRif3FlXHQ4bp8TWBK3Mu/sLlqZYtF3z7GH4w3QbwyA2CWkGgTGwQwyU8Fh
JQdrqXGl0w0y6JusjJWdwT1fxA6Eia3wrSw2f8R1u6V0k0ZhsMu3s7iJASIEEAECAAwFAk8V
1NwFAwASdQAACgkQlxC4m8pXrXzijAf7Bn+4ul7NedLGKB4fWyKDvZARcys13kNUcIl2KDdu
j4rliaY3vXT+bnP7rdcpQRal3r+SdqM5uByROHNZ+014rVJIVAY+ahhk/0RmdJTsv791JSkT
FuPzjYbkthqCsLIwa2XFHLBYSZuLvZMpL8k4rSMuI529XL48etlK7QNNVDtwmHUGY+xvPvPP
GOZwjmX7sHsrtEdkerjmcMughpvANpyPsFe8ErQCOrPhDIkZBSNcLur7zwj6m0+85eUTmcj8
1uIIk4wjp39tY3UrBisLzR9m4VrOd9AVw/JRoPDJFq6f4reQSOLbBd5yr7IyYtQSnTVMqxR4
4vnQcPqEcfTtb4kBIgQQAQIADAUCTzltCwUDABJ1AAAKCRCXELibyletfAo9CACWRtSxOvue
Sr6Fo6TSMqlodYRtEwQYysEjcXsT5EM7pX/zLgm2fTgRgNzwaBkwFqH6Y6B4g2rfLyNExhXm
NW1le/YxZgVRyMyRUEp6qGL+kYSOZR2Z23cOU+/dn58xMxGYChwj3zWJj+Cjw9U+D/6etHpw
UrbHGc5HxNpyKQkEV5J+SQ5GDW0POONi/UHlkgSSmmV6mXlqEkEGrtyliFN1jpiTRLPQnzAR
198tJo3GtG5YutGFbNlTun1sXN9v/s4dzbV0mcHvAq/lW+2AT6OJDD204pp/mFxKBFi4XqF6
74HbmBzlS7zyWjjT2ZnujFDqEMKfske/OHSuGZI34qJ3iQEiBBABAgAMBQJPSpCtBQMAEnUA
AAoJEJcQuJvKV618L1QH/ijaCAlgzQIvESk/QZTxQo6Hf7/ObUM3tB7iRjaIK0XWmUodBpOC
3kWWBEIVqJdxW/tbMbP8WebGidHWV4uX6R9GXDI8+egj8BY8LL807gKXkqeOxKax0NSk5vBn
gpix2KVlHtWIm7azB0AiCdcFTCuVElHsIrhMAqtN6idGBVKtXHxW3//z9xiPvcIuryhj8orS
IeJCtLCjji7KF2IUgCyyPJefr/YT7DTOC897E1I01E4dDymNur41NjobAogaxp6PdRNHBDum
y8pfPzLvF3OY4Cv+SEa/EHmCOTHTamKaN6Jry/rpofqtueiMkwCi81RLgQd0ee6W/iui8Lwp
/2KJASIEEAECAAwFAk9V2xoFAwASdQAACgkQlxC4m8pXrXy9UQgAsVc8HNwA7VKdBqsEvPJg
xVlm6Y+9JcqdQcA77qSMClc8n6oVF1RpI2yFnFUpj1mvJuW7iiX98tRO3QKWJIMjEPovgZcS
bhVhgKXiU87dtWwmcYhMsXBAYczbsSaNWhOIPwKHuQ+rYRevd0xGDOOl3P7pocZJR850tM9e
58O9bzdsRYZpFW5MkrD7Aity5GpD65xYmAkbBwTjN4eNlp0nHVdSbVf4Fsjve6JC6yzKOGFB
VU1TtAR2uPK6xxpn8ffzCNTA1vKXEM8Hgjyq4LWSdDTBIevuAqkz4T2eGJLXimhGpTXy7vz+
wnYxQ9edADrnfcgLbfz8s/wmCoH4GJAFNIkBIgQQAQIADAUCT2eDdwUDABJ1AAAKCRCXELib
yletfFBEB/9RmWSSkUmPWib2EhHPuBL6Xti9NopLOmj5MFzHcLtqoommKvpOUwr1xv0cZMej
ZenU3cW1AvvY287oJwmkFRFu9LJviLSGub9hxtQLhjd5qNaGRFLeJV8Y0Vtz+se2FWLPSvpj
mWFdfXppWQO/kIgVZoXcGJQrQWcetmLLgU9pxRcLASO/e5/wynFXmgSajxWzWHhMvehvJTOq
siYWsQxgT/XaWQTyJHkpYJoXx4XKXnocvc8+X3QkxAFfOHCwWhYI+7CN8znDqxYuX//PKfDG
2Un0JHP1za8rponwNG7c58Eo3WKIRw0TKeSwOc1cSufnFcrPenmlh2p70EvNRAINiQEiBBAB
AgAMBQJPeKdGBQMAEnUAAAoJEJcQuJvKV618YwoIAMn3uqSB4Ge1D61m0pIXJfOcC6BhCZvM
mV3xTp4ZJCdCQzjRV3rZRkt0DwyOVYpLzLgDgvbRwjXjOzm0ob1DvYHFA7DnGTGUsBLDX/xZ
5gRvDtkD6w8b/+r2/eQiSu7ey/riYwB6dm3GzKR7FEbIK6bEuPOUBwvV2tYkZRgTYqXq7NBL
uNv7c80GWhC/PqdvdhFn4KAvL0PjVIgr5+mdXyviKqG7uvguYBDtDUMX1qgZpi+fb7EsbJYf
EkBR63jGQw04unqT1EXWds17gj+yp4IHbkJmEJMS8d2NIZMPbIlHmN+haTA73DwNkbVD1ata
qSLiFIGXRyZy87fikLVIljOJASIEEAECAAwFAk+KdAUFAwASdQAACgkQlxC4m8pXrXwIUQgA
mnkFtxXv4kExFK+ShRwBYOglI/a6D3MbDkUHwn3Q8N58pYIqzlONrJ/ZO8zme2rkMT1IZpdu
WgjBrvgWhmWCqWExngC1j0Gv6jI8nlLzjjCkCZYwVzo2cQ8VodCRD5t0lilFU132XNqAk/br
U/dL5L1PZR4dV04kGBYir0xuziWdnNaydl9DguzPRo+p7jy2RTyHD6d+VvL33iojA06WT+74
j+Uls3PnMNj3WixxdNGXaNXWoGApjDAJfHIHeP1/JWlGX7tCeptNZwIgJUUv665ik/QeN2go
2qHMSC4BRBAs4H2aw9Nd9raEb7fZliDmnMjlXsYIerQo7q7kK2PdMYkBIgQQAQIADAUCT5xA
QQUDABJ1AAAKCRCXELibyletfOLsCADHzAnM10PtSWB0qasAr/9ioftqtKyxvfdd/jmxUcOl
RUDjngNd4GtmmL7MS6jTejkGEC5/fxzB9uRXqM3WYLY3QVl+nLi/tHEcotivu2vqv4NGfUvW
CJfnJvEKBjR8sDGTCxxZQoYoAFbGTP1v9t4Rdo7asy37sMFR2kA4/kU1FDxYtFYFwwZCJpNL
hhw0MCI2StI/wIwtA/7TiFCNqHHAKAGeSzKVyKrPdjn8yt7Js2dM6t2NUOwXQ563S4s6JZdR
```

```
lXUV9oYh1v+gFAuD57UHvinn6rdoXxgj3uoBmk9rWqJDNYgNfwtf1BcQXJnea+rMavGQWihx
eV40+BZPx9G6iQEiBBABAgAMBQJPrg39BQMAEnUAAAoJEJcQuJvKV618M4YIAIp9yNCVLGta
URSthhmmgE/sMT5h2Uga6a3mXq8GbGa3/k4SGqv51bC6iLILm2b0K8lu5m6nxqdZ8XNNMmY9
E+yYTjPsST7cI0xUzbAjKews63WlEUrj/lE2NEtvAjoS2gJB+ktxkn/9IHnqwrgOgUofbw6T
hymURI+egyoDdBp91IQD8Uuq9lX+I+C1PPu+NCQyCtcAhQzh+8p7eJeQATEZe2aB1cdUWgqY
evEnYNNK8zv/X3OMYl67YyEgofKoSYKTqEuPHIITmkAfn0qVsBA4/VtLbzGVGyQECmbbA34s
5lbMLrYeERF5DnSKcIa665srQ+pRCfJhz6VQXGsWlyWJASIEEAECAAwFAk+/2VUFAwASdQAA
CgkQlxC4m8pXrXwDOAf+JEUUKLiqO+iqOLV+LvI09lU4ww7YfXcqz4B9yNG0e5VprfS7nQ0P
tMf5dB7rJ6tNqkuHdoCb+w0/31pPEi7BFKXIoSgOz3f5dVKBGo8GBsX+/G/TKSiTenov0PEU
7/DlwvwmsGExmgmsSQgEWTA3y1aVxc9EVC9x0Fi/czcNNlSpj5Qec7Ee9LOyX4snRL1dx30L
lu9h9puZgm8bl5FLemPUv/LdrrLDqG9j4m2dACS3TlN14cwiBAf/NvxX3DEPOYTS6fwvKgLY
nHlOmKRCwlJ6PArpvdyjFUGWeCS7r4KoMCKY5tkvDof3FhggrQWgmzuPltBkTBQ7s4sGCNww
6okBIgQQAQIADAUCT9GlzwUDABJ1AAAKCRCXELibyletfDj1B/9N01u6faG1D5xFZquzM7Hw
EsSJb/Ho9XJRClmdX/Sq+ErOUlSMz2FA9wDQCw6OGq0I3oLLwpdsr9O8+b0P82TodbAPU+ib
OslUWTbLAYUi5NH6WW4pKnubObnKbTAmzlw+rvfUibfVFRBTyd2Muur1g5/kVUvw2qZw4BTg
Tx3rwFuZUJALkwyvT3TUUrArOdKF+nLtVg3bn8EBKPx2GfKcFhASupOg4kHoKd0mF1OVt9Hh
KKuoBhlmDdd6oaEHLK0QcTXHsUxZYViF022ycBWFgFtaoDMGzyUX0l0yFp/RVBT/jPXSBWtG
1ctH+LGsKL4/hwz985CSp3qnCpaRpe3qiQEiBBABAgAMBQJP43EgBQMAEnUAAAoJEJcQuJvK
V618UEEIALr7RNQkNw1qo7E4bUpWJjopiD00IvynA0r5Eo0r83VX5YYlAfuoMzBGg6ffKiCs
drHjEh45aIguu8crQ7p2tLUOOzKYiFFKbZdsT/yliYRu4n28eHdv8VMKGZIA7t0ONIp1YPd2
9pjyVKy4MOo91NfwXM5+tcIzbYL9g+DuhQbYDmy8TVv7KKyY/gqZU1YB6kS49lycQw8WCine
FoeD1fb6aP9u0MFivqn2QCAhjXueKC01M2O0jR0wu7jdojN50Jgeo6U0eIHTj2OQmznh8wYG
MX2o+1ybSTjjHIp3X8ldYx01Sa3AqwKEBclLdg5yIyAjHq2phROd2s/gjqrWt+uJASIEEAEC
AAwFAk/1PVUFAwASdQAACgkQlxC4m8pXrXwn3AgAjWUh31IxsQcXo8pdF7XniUSlqnmKYxT+
UZOP71lxeaV/yjY+gwyZvf8TWT4RlRp5IGg6aNLwLaDB3lcXBGuXAANGUr+kblewviHnCY3Z
+PWiuiusle+ofjbs8tFAr3LN3Abj70dME7GOhLyplP2mXIoAlnMDJ0AyrKx5EeA2jS8zCWCu
ziiOj4ZwUZAesXchpSO9V9Q86YiPtp+ikV0hmYgZpIXRNcHOpxnVyEW/95MFwi4gpG+VoN57
kWBXv6csfaco4BEIu9X/7y4OLbNuvzcinnHa0Pde5RnRlbEPQBBZyst2YZviWTFsbG8K2xok
dotdZDabvrRGMhRzBUwQEokBIgQQAQIADAUCUAZhawUDABJ1AAAKCRCXELibyletfDJUCAC+
68SXrK4aSeJY6W+4cS6xS//7YYIGDqpX4gSlW1tMIKCIWNhHkZqxKnWClnmvgGhw6VsZ2N0k
YdOnIrzEPWL7qplZRiE1GDY85dRXNw0SXaGGi7A8s6J9yZPAApTvpMS/cvlJO+IveFaBRHbI
RRndS3QgZVXq48RH2OlHep3o7c964WTB/41oZPJ7iOKgsDLdpjC1kJRfO9iY0s/3QrjL7nJq
5m14uY16rbqaIoL81C7iyc0UKU9sZGMcPV7H0oOIAy206A3hYSruytOtiC1PnfVZjh14ek2C
g+Uc+4B8LQf5Lpha4xuB9xvp1X5Gt3wiPrMzcH89yOaxhR8490+0iQEiBBABAgAMBQJQGC19
BQMAEnUAAAoJEJcQuJvKV618CbcIAJCXDbUt96B3xGYghOx+cUb+x8zcy9lyNV8QC2xjd9Mr
02LJTQHfJfQ9Td6LfuoRb7nQHOqJK1/lWE28t9tlH7I+i7ujYwA/fWardRzqCulNXrgFEiQK
ZFaDjRYyM0jWG/sA3/Rq2CMBNhBeCcTDuZ8VvRdm0xMPpyavP8D2dM9WBkPHOik4yAIILVkr
hWmr0Up0JhRoelfeyqcN/6ClUgeRMIyBYthA55fk2X5+CerommlpDfJJlFQOv64VSzS68NG8
j9yf66uuL3bB0OdzOMW6Yq/P9wskCDlMbYm/UnHfB5wAuxWpDeAvt/u+vU4xqqEjkUQGp03b
0v1xl79maSuJASIEEgEKAAwFAlWg3HIFgweGH4AACgkQSjPs1SbI/EsPUQf/Z6Htrj7wDWU8
vLYv3Fw23ZuJ8t8U/akSNwbq6UGgwqke+5MKC1fpk90ekzu5Q6N78XUII3Qg8HnfdTU0ihYg
qd3A1QmO6CG2hEz5xoxR1jJziRCbb1J7qEw8N/KzBcTkHB4+ag6bjFY9U4f9xU3TjPIu7F2V
Bk1AX+cmDo8yzPjDnP4ro0Yabbg0Q9xzvaK/7pFRz+vL/u/lxW7iE7n6vXTiaY1XnIt5xAXX
dwfLYmWeAgdc9KXFNlt4lfuqrETtNCHme+JI+B2Tz2gHmMVLHiDV59eLC0uU/uVsOXEd26ib
JC4f3KqY9kxuQm325kNzxnMxiwMPCVzsEh7lsYp+OokBMwQQAQgAHRYhBADTXowDFGilEoOK
6kPAyq+7WPawBQJasiYMAAoJEEPAyq+7WPawox0H/i96nkg1ID61ux+i20cOhVZylNJ770Vv
0zfXddWRN/67SuMVjLLiD/WfnDpw6ow6NM7vfEwbmvo1qeFF7rWWTPLm57uZfTk73un3fbaL
JiDZyrUStQKK/yhGAZmwulOQq7XBm+u8G9UcFi4XQxuoc5I/v/lUgbxXBADlxlfzpkIDwOaB
s23RDiMcWZGcosUkYHXlm8scU0tRANVLQ/PHgttlUl3x2PLzrdQm3YUDKUJ9+ynO2jN2sYwt
laSohj4UbLnq6pI4CXWZR7XWQs+NX7P3R359FDtw7OhyKoVuIkRFZljY0i3wQgwl/Sm2DAg9
3lsZDVc/avEUaOO+VuJuvJ+JATMEEAEIAB0WIQQGFx4znGT7HFjpuwT3iPLIbOWZfAUCXJ7Q
KwAKCRD3iPLIbOWZfGoXB/wN0P3m27fY/6UXTl0Ua3H+24ueUdLipsvR8ZTwEfnwkhLrbggE
0Em7ZuhZkzv7j856gv/tOekYYqWGg1CLalD3y371LAGq1tjY3k/g2RWLxLXNdzgXEyFvaNQA
oQa9aC2Q7FOyEMwVkkXrGa4MML7IBkrtMds9QPKtfipachPf6tQOFc12zHRjXMZi0eRWyQue
0sLLiJZPn7N8bBAJyZ9IJEpkhNrKS+9J5D1Refj++DwBKDh04kQXZFEZZhxcungQW5oMBQgr
uW2hULTLeiEV+C516OnwWJOz6XKJpOJp8PY0bO8pGgToGIYHkoX2x64yoROuZasFDv7sFGX6
7QxyiQEzBBABCAAdFiEEEN0MfMPATUAxIpzAoiiOmODCOrwFAlv/EJIACgkQoiiOmODCOrwg
uAf+IVXpOb2S3UQzWJLSQyWG0wQ51go4IBVpHv6hKUhDFj47YdUbYWO+cgGNBjC7FVz54PUM
PIdxImGHE1NHH+DNR8hvvAi+YpnqqdT3g+OgZ6XoYevret5B2b5fRgN1/HWUjaJ/n5g6SMsC
+3DrmdMu1FEDnKv/1HwQvOQXKt/U2rXE1ILOmVdMavRJEwkrk2SVwbdeass2EInZVsmWL+ot
9dU5hrkmLAl6iHUoK6zF6WaI1oi7UU2kgUF2DNyZG/5AumsNhxE608EAs1zEdN8wibXL48vq
Z4Ue9GvImokdlq/r/4BMUdF1qLEZHBkbaklK1zXxl7uMiW3ZIcqpg5HgwYkBMwQQAQgAHRYh
BBTHGHD/tHbAjAF4NhhrZPEl5/iCBQJZ+o/oAAoJEBhrZPEl5/iCyfMH/3YP3ND8jFqIWkmG
JaITHP9GhAQda73g7BFIrBHeL033tcLtUbEHXvnIZzulo7jiu9oQBjQvgGgIl5AqH1m7lHaD
iAL3VmuUFZ4wys7SODHvSZUW1aPLEdOoLKeiG9J6elu0d/xWZmj86IaHMHrUEm1itMoo0m+U
MwVNLFNZrAjCn82DiS6sS0A52tOlpq/jR4v9AYfMZSnd1MLm/CZaZpzWq6aqm7ef7CDfsUvU
w7VsL3p1s+Jgo6+8RwQ1W2Lgt5ORthvpjPKE1z0qgDpoXTkPOi8M20taD5UZbpByzMZPJXXr
+LBrRbs48IcPVHx8sxHMh1HsQCiXHDGiTNSaJ1qJATMEEAEIAB0WIQQazDqcUxAL9VrKN9zD
LyvJ+reoRgUCW4YZiAAKCRDDLyvJ+reoRptWCACoIgFrvhbr3c1WVq16LJ8UmQLk/6uFFZPN
CiR6ZbvzOd+a3gk1G8AhDEW2zoNhFg9+I7yqUBGqn+B1nDZ6psyu8d5EoRUFTm3PghqEccy5
KixqoPxBTquzkKGbN8PDLUY5KvpTOLLlYZxlHzSHw4roPsU4rxZtxyu98sSW0cm47VPr069p
91p9rCoHY8Fng7r3w28tVfvLuZ1SK4jtykIvw+M/pVBk9rQVCAJ0JjkAHkTOpkHqsVBYhtu7
```

```
mzsXfkQZkeuxdNx6X1fMrbJofzH0GYTT8Knn75Ljhr3hozrsL4Kz4J9gsLHCjkD5XKzLwCFK
R6UhhZZr7uhufbqZIyTLiQEzBBABCAAdFiEELLeCvUfxyJI8qMqHHSPVZ6Jn8NcFAltZjFMA
CgkQHSPVZ6Jn8NfKSggApk065wFrxq2uqkZKfJGw2mdsGeDVjGq9tMKUWeYVxTNxjiYly8Dc
/jrOS3AU6q7X7tAAcmvaXoBfW3xEIXMSH73GeinVG7wnlab6GKPDRKJzXfJ88rF07pX8R1pc
ZH+eikiFsN9bcnEycH82bonS7dzyoo6yg2zBqNtsmWYLDg2hcoTw4UHAPwdX6+n99m3VzOqO
8ThQI9hqpUYGvP5qyYahFf+39HSViof+Kq5KKhvSoiS9NzFzYZ0ZszYt+2jozUpAM6XqtEGu
TMzXHkE+/V4yI3hIsvHNkXKgDrqjwA+UmT1R4/gBoiRhZ8r4mn1gYI08darQmkppf9MEbcDz
U4kBMwQQAQgAHRYhBC1hIxvZohEBMIEUf5vAD7YffmHCBQJcns2XAAoJEJvAD7YffmHCC0UH
/R8c5xY96ntPI2u6hwn5i0BGD/2IdO+VdnBUnyE4k9t2fXKDRtq6LAR2PAD0OehSe4qiR6hw
ldaC8yiyg+zgpZusbCLGxbsBdYEqMwTIeFsa8DyPMANpJ0XLkGGf8oC7+6RuAJvlm6DRlurr
U93/QIG6M2SNsmnPgSZWYV4Y5/G7Xxyj0Fc3gNjjjGGP61CBR01W6rgNPn35sZ9GYCZcGlQA
GGrT8mSVoUhPgPCXKz2dZDzsmDHn7rULB6bXcsHiC/nW/wFBpoVOIFIxND0rb1SYyJzPdPtO
K6S+o+ancZct8ed/4fUJPBGqrBsuFS1SKzvJfPXjHGtZBitqOE7h57SJATMEEAEIAB0WIQQt
9h/1MHY0zPQ0K+NHN096zf0O3AUCXK2H5QAKCRBHN096zf0O3OJtB/wKbQN4IjVNkmWxSaBc
JABRu/WSbNjoTo/auJV6IRUBpwR130izMw239w5suuWx1phjPq3PdglBaKKeQNdeRoiudUjd
hydON1cq2wh9O073wU2GHeZLi48MopUNksrhHfd/XWV//0LcSpERsqIBVIUi+8DHwFvpCzCz
zIRg9lOcQmEtJAFFUtkF9FEeZgO2NPO3fEwkjKDeJYUiB+mD9BliyxhU8apUx/c2zaFGQOCr
MllN/gHztAWDcIadK/tujqRWR4wnJ0+ny/HP+bWd18+YjhcWzUQ8FytG+DA3oylQ1d0w0emt
qfn0zqiFkJQdG0M4qtItJYEYHlYpG2yoQHcCiQEzBBABCAAdFiEERVx3frY8YaOOhcAGjZrN
vi2vIgUFAlnScGAACgkQjZrNvi2vIgW5IQf8DKjeoHF9ChDcb4T01uJJiAUu6lxewSRD7iwD
6MjCsaxgMifTD7Bzvdem4finoOul2YAPtlLfIfVtVRtGG97R/Wvs3yjI9NSzxkDGuuE7/IIi
4dKlcKkvijg7G6A8+MGXaQTw8iOePI/44IyG5yogKjno7L4h0f3WguGzmCRUJcgYm23IsaTh
Pvdq39ARyHAlrk0hXZ+OqsYBrlW7KLyPrbPA3N+/2RkMz6m+T8ZksOrEdF/90nC9Rky4Wbg4
SJqWQNNSMfgT0rQL2Qvne598FKmltrTJuwBtIrSeuL/dbKt+hkLgnRjnmtA5yPaf0gXvMtfU
P9goQMWD+A2BU/bXJokBMwQQAQgAHRYhBFBgHh7ZZZpG0pg7f1ToXvZveJ/LBQJblegpAAoJ
EFToXvZveJ/LS0YH/jpcVprmEGnqlC0mYG2MlRqeK4T8Y6UnHE2zBPc125P4QcQfhgUJ98m4
0B5UkzljreFr9Zebk3pE8r4NBsamlJvi8sGbZONTsX4D3oW9ks0eicKOcTZJgtX5RmSNFh63
+EHbqTneK/NTQIuqRSCOufqCOH6QY1PVsICBlFZUPMfuxRlO7EwHKNIHPVBZNlM7AXxdjCMU
kXvda8V14kActb1w7NWxWxo5q4hkQ2K3FsmbWXvz+YBhJ8FnRjdzWNUoWveggOD6u4H7GuOg
kCyXn1fVnbCyJWsXQT9polJRnIAJMAtykcYVLNS/IS65U+K1cMshcF+Gil9BuGyckbRuNaSJ
ATMEEAEIAB0WIQRh2+o6RdTFb7cSlWG3d+zE2Q5m7gUCWdJutAAKCRC3d+zE2Q5m7rgJB/9k
c+prmrnjsq/Lt6d90LqYoavvIeFkAoDhhWgQeEOAD1wgyHIpS6qoMKgvBlvda2r0bmk1kUL2
xQaiDj36wB5yJHauOnFX+3ZJ6QCYUaeoWtqO2ROHvTiuyUdVKC5NtKaHpM1/lP/jl/1ZRWay
idggH7EnwDMt+9O0xD02n5J29Vp9uPO1GtMVsVSiJCGcOxwNBgNiXX1BpZbN4bRm5F8DAGiN
v4ZI69QZFWbpj8wFVJ/rV4ouvCFPlutVEAuIlKpAj35joXDFJhMvPpnPj84iocGqYPZHKR6j
a90+o8dZw3hXObFowjcxsJuQUTVkPuhzqr6kEu1ampaQ8OGpXCZHiQEzBBABCAAdFiEEZ/mR
TQQxCZjglXUwgzhtKKq2evsFAltbmWkACgkQgzhtKKq2evsdrAgAubfuG1vWX3TTG/VYYrfM
1aS1Roc034ePoJHK5rLT0O/TnnnObw38kJM1juyu4Ebfou+ZAlspiWgHad62R1B29Kys/6uC
qG2Jvbf716da4oLXeLYd9eb+IKVEiSb2yfbsLtLLB0c/kBdcHUp6A1zz0HV8l1HWj1Wx8cFU
MV7aAQoOfnNBbnNWLzNXXLYGHh47/QmjifE5V8r6UJZGsyv/1hP4JHsQ2nqcM8Vfj+K+HEuu
nnxzgWAcQXP/0IhIllVwoWhsJlHW+4kwW02DDopdBfLTzCtzcdOkfBcCg8hsmC4Jpxww5eHm
saY6sIB32keCpikVOGwdGDbRH7+da8knzokBMwQQAQgAHRYhBG4VA/IlW5kLV/VchhLcHkBr
mersBQJaX4N4AAoJEBLcHkBrmersksUH/3M0cypXBnyGIl/yE576MDa0G1xJvciup0ELeyhj
48Y7IAr7XiqDtiPt8tlIiPFF8iaw56vJw5H6UKraOcjZHOH1SwDr5gAWJgMqnqlFX/DxVKif
USt81KX0tHN6t6oMESgm2jRKvcWjh6PvEZlIArxZG4IjrErqWIJjUJR86xzkLyhRVTkUL/Yk
uNl1i013AlaD/0CGuAnjrluUUXypadtNr7/qsBx8dG6B/VMLWToEDEon76b8BzL/Cqr0eRyg
Qz6KWi3hmsK+mE4+2VoDGwuHquM90R0uS9Z+7LUws24mX5QE7fz+AT9F5pthJQzN9BTVgvGc
kpI2sz3PNvzBL5WJATMEEAEIAB0WIQR00X0/mB27LBoNhwQL60sMns+mzQUCWoyYfgAKCRAL
60sMns+mzYgnB/9y+G1B/9tGDC+9pitnVtCL2yCHGpGAg+TKhQsabXzzQfyykTgzCHhvqRQc
XHz5NSgR0Io+kbGMUUqCaen6OlcORVxYIuivZekJOAG+9kiqWRbyTv4aR6zvh8O5wCyEhhyi
ifi65PM7y9lD6i22qTt/JoDnFkP5Ri6Af/fZ9iaIaluQKJCU5xY1Lt/BorGlrGvX5KiZD8xc
AjhJRATZ0CJ21gbxISSxELAfH42KzGAvJw/0hARrMkl/eK0HVDpD47mcmC5h/O/HlwPYi0hn
xB+6/nuwwtRgMDBufNV0StU43njxCYmGI9/I1z5Vs+zhz8ypw/xCr1U7aAPZQdSSsfEViQEz
BBABCAAdFiEEelR8OpStCJs7bhrK1TniJxBsvzsFAlv+8d0ACgkQ1TniJxBsvzsiFwf/a3lt
OuSrFs4M03YVp6LoCM6CwZfvcFl+6B0TAurOiCja9lsNmbusSx0ad7bZy6/kHDXH/eqomXeu
O4hkxxBvGK3gZt7iQsr9vsUSbbJnc1zMyOZKlhdxAOLOskttqtPs6hiJ9kUHFGZe47V3c77G
GMgi/akIU5PkxhK7+/bbAsW0iK60aXCZ5nAbWlzTQLgJnYrlk4b920rzGe8nDTGzGmSjIGnb
YvuD9ZI40DZRWVf1tXqCY643AXFYoOhRxj54uHnMLYhc0I65u2ZGwRiTI0g/en5E8i7WoejA
/sR0+cYs7l1IJwlNRwfqmnJWRGREEHcJ3N52k3X7ayq3qmr3K4kBMwQQAQgAHRYhBJSRYHFB
cqf4Tl2vzE+YN4Ly8sn+BQJae/KHAAoJEE+YN4Ly8sn+5ckH/juc2h7bC4OGmRHcZBLAG2vW
WEMTc8dAr9ZyJYXzR25W1/Cz/JXgJgMjSrE6m9ptycpvWc6IRlrQM/IqG+ywYFPwNp3PYsc0
1N33yC15W7DPRDTtJE+9yUbSY9FeYraV4ghxiBxD1cDwtd7DFNGNRvBDH7yQHmXBW0K8x6yX
Mwl1gj2/MvdFUKmz8Lku94OmrbDOi83cnAjUNbN15Wle7hWAIRALt3P1VusjV/XyzxvcSffb
mt3CgBCyK9CNyEr27CVkhZ8pcabITx9afMd1UTEii90+qzgcJwcR46bJPZBdavMt56kVCeC0
kG44O3OOk+OahKXzw4YspZMO046gYRKJATMEEAEIAB0WIQSm5fcyEkLUw6FcN0ZJlMJhNZ28
bgUCXTJMCQAKCRBJlMJhNZ28bsgCB/96PlBUdsKgnh/RpmPB+piFQf6Og+97L4fxHuQbzKOe
UNCSWNF7saVa5VaPxbV/9jDCTPZI5vBtnJebXtkmLoWFSZaXCYb49SijfvRsRAeX5QSqIRd4
3KMuO7nAvbPVYtMChCO/g1T3riF2icC6pgvmNZWm5Nu4pkLzRmQv8U33BAkL7EYIjZZaC/9h
o4Sh4l/gLNItOxMdsD34sJwBLvEi1pQOa1xNJ4kfQSRD/8ufakE5wfSie/s04w/2Cp7RD9H0
VlD+7FwPO1HQ3XJjONvOzj6uVdwCC5fcmbXbb2bbJ/xe4YVL3xmwWz5m2w+kBSpaZ6VHNocB
8S2OmIIPpr7OiQEzBBABCAAdFiEEp/cyEkLUw6FcN0ZJrn5Z0o967I/htVRVZtQSYFAlqnkGEACgkQ/htV
RVZtQSYV2Af9E7FLIUi8lqOyYyZuX6skkNf5rNSew+7i5NsiNpQzZMdscJh9eJzyLrePLp7q
```

```
9HUOhMF/Fc0SgbDtKSWbfSidXkeaQ2twPj4rP1xxYBc0OY0OX4fNVA5O/pTI9nxIVQCDTljl
/WIY+fnj88lCkaKWoRJITaotjFmYt+gbJMBn3MMYf0VODeIRozV7//NdkzFXKmJ3fsCDGXXF
CVWM1Fn3M91o1fh3FSgKd+0sexUDn5afwWCqjGgiXDsE7fEdwsbnz1rDzWvuqCoZyIh1RXQf
QVbiakpzfvtDytC3Vo6F2KzpZ9d69Adhfn2ydAYxL/Xuvk9pWdEBNF4T+HfS9Z30BokBMwQQ
AQgAHRYhBPJCF6TG7RrucA13q1lkfneVsjZHBQJawgLrAAoJEFlkfneVsjZHgNsIAIaSJ3gF
tBtf0WLxYIo5zhNclXOnfgUUNjGrXHm5NxoI4Eulpx9dQYCJ++whMFbxpZQTgFAUq8q342EZ
raLCWwALZEZmkZjv+FX6bk8sgqZESpUOLJAIqpobKpaawOQ7LS+XWO0SchH1oLFAgDyBeIDZ
N/LiTlIdkJe1xpDQDtgUHawksqMCbIaBe60B5xvm1NkhnrmnM1p+e3LUd4j+XxACdcY5LSqV
zVT4OyD1WkKzk8EAASUI8xysNBEeX9/8/EXaAciECQb3MkYxTQZ4WqCLU0GCGl6Sx2fY5zI6
4Y1j/Sfn3JHikJots8eR1D/UxrXOuG5n9VUY/4tTa0UGPuCJAU4EEAEIADgWIQRLXddYAQl0
69GnwU+qS4a3H5yDGgUCX6xjgBoUgAAAAAANAARyZW1AZ251cGcub3JnYW5uaQAKCRCqS4a3
H5yDGkRfB/9z/5MuAWLwoRLJtnJQzEOW7jsfzYpepL3ocT9tdGcs8jJTH3vh2x4Kp2d0Zaxx
Zs7R8ehZO5XJQ/DWdhH+7cifoeXmAEqDnlKSXZQZY/bG054tM6zes3tFTH3dCrn7LF59fQOG
OaZHgbFRQJO6F++90Mj9WAgeqGxyEhAlFIxFw4Cuul8OZAUIfq7YISnpkg2Tm/Q0SRRDJE4i
/7WJE/HVMB0Rf9KJXuk2BJlRIpQz8Cf+GVZ5aGIlXdM58QknprnollxoTKhrE74rAGHW7nRD
xIxOoP8odiXbLzn//g2m123usqncCKWZONDdVupax3RQ7xsIuFc9Kx4OtjwPQftziQFOBBAB
CAA4FiEE6hBKAqPbygqOC7fUwpbDMFwG9MsFAl8u+m8aFIAAAAAADQAEcmVtQGdudXBnLm9y
Z2FubmkACgkQwpbDMFwG9MsIvggAhRfd2Z5WLR6hGxOHu+A+ysjX6xKjcqshCYr8jRuOflFN
vxugQQoFM5pQr15TyhokaU78aDUoIbLnKcxxmH1l4hXxcRtg/9Y22TidOVN4jjNbc69KvCC4
uANYuAJaI3o5fb1jv8Lx82OiRDMhtRqyTdSGdU5//8X5FXCt+HhhzpSNoNtpxyhsKP0PAWao
zuETqvxy7t0uy0f1OTbZLI5nb52DxjBdZlThnJ2L9RwR2nSGhxjhTFg8LrZWgWNtY5HG+vk9
qbCwaC6ovNJ0G98i0DMrlbyGCbxa4Rv332n1xPfl/EPYWmNPlMu0V3bSCqxVa5u3etA5fw3r
qIm333vgFIkBswQQAQoAHRYhBJTatFFgHAZYHkTw9GcRGDP/RljgBQJa7LubAAoJEGcRGDP/
RljgNu8L/jN8j4HSggpnzJ0+3dFjVg7FUHJF6BZ84tv9huhmyrByaIrEfFf9ARn8OizKgdpC
/wJT1+KXarvsxdnEDlYSat3HS/sEw3BmZjAeTwPi0ShloiSjYgYRbg3irDskqUHML4hhvMx0
x9nZIag2XoSSH7kPEd5jOb8cd7jJeoGg6Z9Z9lMHuyqTGi0T/EbnhjQfVTxWkSkcDvdxbSuW
D96mvZrbRnrMebXKkISb0uVUn3/o11iUo9jXs+Q/03Tb9i0H3eOliP1kcB/kggu9xblIPM+J
VaK5Z+zAVLPKTQJi+sP/ayEux0xZzfbZ96WERnzT4E7Wwv8MvaLbybtID28Oy9YoBBYv7CrC
tyfrHh1t4v2AedRSZcTPKAaQ5NtLAvIdex0kOvvofaGi+7nmgV00vCZFBSXetvBMZkCapW09
vF7wcahaXpF+0Spl9vE2JiesST7uQobCUm1EjxJP0vMDcO1vIfJHlbIhB/f3PE3rXZIzYTdL
s3Kb4OONaUfNy9jYtYkCHAQQAQIABgUCVJqcUgAKCRB3MepTnaVyot2+D/9wAQ+p03lVMpYS
gMWMNLgjq3z7QrN0NYNpxUXAonxECjUzZKSUPGci+fPKxl3ZUenk+ruLgtgJmjmUOR6u1Dov
BpDFzhfqbIpjgtMDrnY5sWqxJ+CH2Rb5okEEDJ5qE9DwIMP5iXbf4xjnBOyPiq3sp983PLvy
8ttidWe9FDf8JuhWLHRJHODQjc6LufcHSWKG9fLmCjL2KSPNl696MwR+N95EKCivLL2PlG8c
f08Xd8lW1S0cJLh/6TEuZtAnVeo0NUOGUXOPPyhTPP/xhfLeKbkxjtm6rg/jBaIjuuQgUyNN
hKnP96/GRWWRHvio6eBPalhUcvImSrCHnqLRpdyMxmK67ZzKZS3YsH0ixozJYE0mNevZ2hEY
wB+O5HllqK22YwvJnCLH2ZZWTu2TCUjGZP8hbo2nSoyENlxZio9Gl/v4ypjdlgwrjnnZvxoM
yOFeuc47AuzP5QjhtlrWv12C4hYi3YLZvkLVFD0CxAE/CDuHk/4eFG4UC4Mor6+BXwVG7NEl
4qQWrAHjLQ2/sHMpsUqY/5X7+StG/78PLP0HP+PIBCDDTa7W0+6kf0EaGVHKW43IIkVNI2Ps
b44tTT+Xhc2mHk44LuzL4Axlywv+CxP9NcKLNFwK4Ck1M8Np6cAKlu+Dw6gjOY1aGHgtdsBQ
cIqZj/+ETD0+9NkDXEoeDIkCHAQSAQIABgUCUliwpAAKCRCiKuTrQynFRXZdD/9vb+69OGSR
t456C6wMLgBl+Ocv9XeaCTiJjLgAL2G6bRH2g2VcNHnU/VMTD2YLVu0eP7ubsirVrmR7nAgL
sQ1mKKWvTI+p5aAvn4sL3x3P8vzmGoDAigZ458yGuVpVsBkSPjJBMAkMDfm9kdWxCanzuKXS
b59lfTg4EtcHPDzoSgABntASgfioVxP2TVPfre282cibeYS+RDlaMTVH25yElrWDuF2U1CVW
SMWY9mskr1+XjPnoO2jz0+jhKB7jyMMfSmJqzgcBNgezFbzX2fPmNnMZzEucVFFHmIhNVmL2
rOwc/s1tSHerG5YIdL3HOJek5xJljzjzFfDrdjmMMl+nO6nO78oePoLNdglQQSqn0yW6gZv8
EIIQ/N1nSi/LEW60z8FFxzoO8TqxMMX9QRLbVE6p+7C0nqolhZf6UEiDIIm+PihF1vPFSV54
+7OoLObCshe2g4pbRGWPhIJ4X3ILBQwFMZbn+cIuY3h3B/UpbZE/YSDgRFu5TLtCfBE/lQKX
7QhJknJhQhJ+Dx+Y8h1Cx61Qr0KP5DmOkHYZfAQtdacgrqEr/qNen4QYRdKp0gTne8AV7svB
8eI/8PkzvUPaHrax0g6ZSbeWbvEw6czm0qUGJX7iMlJSauIJPrbOjvXT7qIsaqZRRiUSWXo+
m+jzK5qdeRhEIUmlJI/tU/RsGokCMwQQAQgAHRYhBEW+vuyVCr0Fzw71w1CgTQw7ZRfyBQJd
hy3eAAoJEFCgTQw7ZRfyRf4P/3Igs5dYm0fhposI5iwBGtN5SsxYTZGte2cZ+dXVcnLwLIZc
Ry1nDu/SFXPUS0lQBj7/Bc2kl8934+pUtte+B5KZI2s/28Gn98C2IjxxU+YZ1X1LbUkx0cPA
jFWjUh/JSfu6Hif2J0NAG3meySnlmpxl6oZeTojeWo1t39PF4N/ay7S2TqIjGSBfxvD1peIU
bnziKsyM5ULbkMdgHssQvyZvrVzQxacRzPK424jXtKR6B2oA0wqMcP4c69UmVKEKIzJNYrn4
Kjs+An8vZvJYAVbiWEyEseTTo3XJePdBNs1xxK2vWLA5PeLkE8bmzHr8iQ3hA0NaY7jSJp3e
GrhWIdXV+nfclrFUPghYr5z+ljCSK5sow+aRiED39qd1Y+0iUAy94cqY3MQ4ayGgnB/+YuSx
B5jNjCBYJetFWWSJXnkbiYRLjU88dflXCrTbhkSuCu3agOjsBJYUyg/c1Z4eCQgpTWB2cjYQ
0ucKOsWt8U6qsl12qwYLr0RfcP2aCwTTnWIxqIN9F6iMafOsG+za8JY+B8PDJxxwWWz8vCvX
ChTYrfiFei8oUqoHYTbw07cxaxkDd2CgXsQMmOcZSoXZZPAe8AhsUibDl+BZs/vLZT7HrXtt
/ggz8LzVCcyQqwmCHurvgjauwjk6IcyZ5CzHFUTYWUjvFqYfAoN15xUZbvPYiQIzBBABCAAd
FiEERsRGITzmkUU5TZu635zONxKwpCkFAlxFLcAACgkQ35zONxKwpClKVw/+PfrtIVHFsOdl
2crWBSo5Hifvx9Vn2nPiNKErygB+tPWDS4UwzVUnpZfXCM7bKJFFPeKbitYxN3BlDmVhZMkc
1DZMAtIPSstO2oX7Tv/C0WOZPlAWkp5m0DPV3iGbGZjwmy5wz8fNtaWyxtcUeaEXY8j151gm
Wfl1LMvgwnFsQ74xobnCpssLgmogXfoLFQNF/VUfRveJ2Ci8raWyAdXFBdAIrejawAx5MMhO
/lEfQ3W3f9bqtJZ5DzLbxQ3Xtqs+RY1ihv1y12lr9vLpgKKGmZ92KDvjv2UXHd7XZ90aPMj7
Rx0MQ1d+5d/tNQ8rLJGuj1I7NqHmLHMz67TvRtPl4aNP7Mss8OHiEKLYq23kGqXN+6cjG3UM
i290uJZaAnTno65Cgsyn7JFKyXDdTOmp3TSoyVsPFq92qgd/jFBf3dJj8c+mZEVXkUFeeUEK
31EMGFCH+oE8un7nu+XWqFyFSw5wn+PGYDXkSd6z/NyIN5DXa326KV+qpUmIWOlcymm7cmZ4
KJQt7zgWCxh2DuWQzRlTjeQd8Iw62V8tIOBokWP9Thes18Qk2GOUeCnvczLdevT4lqr8IzvV
nSwX/LQyxmmz2/dmPhzJ6kA6KQKGOSF6WnV/WuD4kESFKwtABFi6mYQi1F6CynpVw/nu535C
4fFG4d+A5G6sKJx//hjOCgmJAjMEEAEIAB0WIQRGxEYhPOaRRTlNm7rfnM43ErCkKQUCXa6e
```

```
YgAKCRDfnM43ErCkKfNXD/0cTEjvQlgyy3UI3xfhYtRng8fsRXcACjMajnrvYCoRceWwF6D+
Ekvh5hNQqrZsxrD6nozY+iJhkkaQitIj4qw7i4KY03fo613FjeLFXWqf4sfLTANSsRNxawEo
/JxP1JeOToOgYTkikWOkgZWSs/mqvHAxJZrVq/Zhz06OugfOYVGmGZonU7zP12toiwParIZ9
hcZ/byxfNoXEtsQyUHO1Tu8Fdypmk0zYUgZK2kGwXslfOGj5m0M5nfUuVWq5C5mWtOI6ZngT
LPJ32tRW526KIXXZMTc0PzrQqQvTFHEWRLdc3MAOI1gumHzSE9fgIBjvzBUvs665ChAVE7p2
BU6nx1tC4DojuwXWECVMlqLOHKjC5xvmil12QhseV7Da341I0k5TcLRcomkbkv8IhcCI5gO8
1gUq1YwZAMflienJt4zRPVSPyYKa4sfPuIzlPYxXB01lGEpuE5UKJ94ld+BJu04alQJ6jKz2
DUdH/Vg/1L7YJNALV2cHKsis2z9JBaRg/AsFGN139XqoOatJ8yDs+FtSy1t12u1waT33TqJ0
nHZ8nuAfyUmpdG74RC0twbv94EvCebmqVg2lJIxcxaRdU0ZiSDZJNbXjcgVA4gvIRCYbadl9
OTHPTKUYrOZ2hN1LUKVoLmWkpsO4J2D1T5wXgcSH5DfdToMd88RGhkhH7YkCMwQQAQgAHRYh
BH+P4y2Z05oUXOVHZQXCWLGt3v4UBQJhrDYPAAoJEAXCWLGt3v4Uh2oQAMS3sK0MEnTPE+gu
7lLi9rMbD/3O5nlAxBJLX4MzLi2xP1648YV5nq9WMMt6qyp+OVwDXefneYNMgfU2/uu/Wi/o
XTHBJuU36lmFzhRWPj2h/vtfgDIYG2wio0DNJyaUQwLEi6gqPm0AHhKS4td69R+7qyQsbUIa
BFgoytxFzxDb5o2hicEOXa573m4myfAdCx5ucYfq+jlXJW9Wgw7ERnF1v9xQDXiuryXWFRdv
UOOWzVPu9T0gPkcG8NABwqxs28Oc7n9Al9HM2FtDAkD0LIcm/I4ZEhFVqvG6Hj966+FeuICw
OaefFhthOoi3ycO+pkj1IePz/TmnsplTvvZOXH+6XEMPpPRQpvf5IZKJyrvuzoU8vkXYY2h/
gJHi9HiSIIQ/BVEpvp6UjXvIbNP1K31II88qx9EfT/tv434wlZpC6V1FzE2LtxyNcj/+OUvj
9hKOJ7lKOVpsnBbGiWg809s4sCIZ/ifLfWAKOJgxAEk/GcRkkkCqGNx7HA+coteNHqXLa/Lb
2/r8gGn6kH9YhQootJsGhhSsY+6CW5TM5E+FhSRJU7MFHRpA94N7Hn6OFUK2OXtHyRhxE867
R+ChJaZXbtoQJVNv2Rv9yoZrBki3RoQ6/6/fcnR1x2moTMYg7K8AMMv7ZCfaP6AjPOjTVnMV
CpNy1Ao7smOzLAfKbbeXiQIzBBABCAAdFiEEjy2YV7IZJ8NHv36cSrDCiwqTaaEFAmF9XbsA
CgkQSrDCiwqTaaFUGw//WSUO22Csa60I6VN8yJQmf0wCo9sieWDXCdHZ+CB0+gu0I3EMYR2a
gL8lqCd6M79fpP8DiLKOJvn9mhXCsjYjTJQUsuNi5kQ/O9gwarRsr7EjJ7R8u8lpSh9YPlMS
yN6XXfOa4Qy5HOw9idJdb3owKAXSjuRdi/hUExjA8TWliyWrfwiVDQi/aCoLZ4b9p6SfGR3Y
gE8UIZLZtdWgsPJHkvdvntTPi4fwMsadBfa2f+m4Wq2CAU5KSfYsVpKAwSQ1OsdUZUK7g+Ui
jy//ad7eZ+BAc75blHs7ua2iiF8Sc7MC55ZM5ldkv+0lqJ7td5vOCT1LKJg5PKKUC7YTTh9U
PHlERJ/SWcHNES1YhwLvUO2VROlPN9H1QkPnEMBOObpmYkNQyLBfFwioJ3ilptYY0IUX5qBM
5UkwgyqMsdyrL+2ozIYc+/A8KUnZXozOAG9LP8gBE5jBJSIkbqsi9Fumf7Q63++g4ojcYpOZ
F92X6kQMGqBvkvs8UajR5f/n6QH0je4XFPj4l4lVM/PPfZSShNGdOOi4l+KwozICnQ1+fhwh
N0VG4eALSJ6XQEEfJ18PrBRS3sdC7OVEMLevEC8ojSQeZE1lCLe1qAUoEcmgmXjsODaJn2tt
qNYYUxcFOycFnzgWL679C9FVp+DAg9jzDMKsqWo/Lt3IDNF19ZUc93WJAjMEEAEKAB0WIQSC
piWCWP+fBOH/9bx9bbut3FAu7gUCW8ygHQAKCRB9bbut3FAu7mOaD/9QJ1MiyKvw9rYqTvkU
OSDSLu88g6NP5R9ozgGZegInZ/NzT8u5emYccflnLlfvRQZPnT7YIH4+h25CCGQ5HzXUGENx
ndeuG4dm3B10A8hxv+abEM9VYDGqSIvF6z1xObvENOpMgmlmFdDi9O9d6jFFy4Hd6/BWejbU
4M3kfuD39RxaT1OEWfqvTVf4GKiLqM71glNB8WrTqxt2t/Mo2h6UPCF7/wPF/idMAbKEn0ye
b1WDCaZVXxAQETfNo129hPb2qxPGoCWGw24ySpGrM5We4Nd3bbdGItSZ0mATNM1+m9FY9j30
vpePFzzYGZ+23EcpxWU+7jWbjZ42ssCW6kx2/ERLVma7FuneEAqUc3gZr/3ZdZOVMvseg8c0
n66D/NRLgMcpOQK62qJfSrxQj6sJCGRY4dxAfdTZWrcxu8UvvcINezGITOQ0y+Mc5LM1vMOd
srXcaVnuJTfWorOeqnFecnClcOwKNAKBXjE8bSANUBKlrw0RIpye/IilrKGEMaYkP2nnnNZE
GPmumGkejDstWGmnHi5IogN8ibzyywsbNsO+qDdlUFA2bmVhh2uK7M95kyuMH3GnWbz4IiMx
RyUVEyK8yKnEmgOmLG4WiJjksP1jIPf3ztTEVVDJxy1gT3R36lsxd+OabnPOgiz1oFewKaur
aWX1e0E6eBWJ95ufookCMwQQAQoAHRYhBM8z5mfkMwAXdpGlbLdWs0L0i1qEBQJcBMl7AAoJ
ELdWs0L0i1qEmxwP/jDweTwTh1s+7Pp39L6aLB7nuQzdMleTksPGgmtguRBZipbOYOryEozK
9hI3Hq/ymV/loINv6GZhieDoZvxrv9eEKgO2eUE0IletSy7znlhV6MB7PBOc29dbCMf5L4qo
xUG/f+XfHkRZEkjZRWMlitlERlDU5gHAQ3skLuT9bu3aZkGdBgw0U5qjVvGzYxp2LFpNHXlf
TrlN3RZoDbRI+E9BPILqZFIZczp/fxRRNkXyogkrGD+0PANFsjySQKd/rr8/Z4isl3AM8CZ7
s4tMWM4EVJ2OygnrcMuIEJdXVsR0Ln1gJLuQ9HpWehve0d7/cIZkN7a0fqgE7bMvSPyxWL3m
yTA4FwdbrebBr2y7ixlXZ6WtX/rqTvo2HTDFLle0ZwMbbfAtoFX0M0lPtXTLmJAl5w1G8Nj8
bthWdN4KVFyOpqPt7OXc/G1YNLzcyYQXX5e8Uskmg40OH5cQV5OFEG8qpxTg53wANDdxXGzs
NUQe84Qkoyk75nwzVfsi00/OhTZmfIC48esXcs0kTrkSPrFcHktSMoYPmHfV3dTF17ifjz5a
C2SL22R+RokWuzGxxpvEaQAWIyCt6izf1a+CjnXPD2Jw3yDC/Oeg68XYiSrbeFdCRzQbS9YP
ipUFIlHuCiNZeGg3rFL2N2JodXg2LGORJz1RKazT7uAfRr5z7W1FtDtNeVNRTCBQYWNrYWdl
IHNpZ25pbmcga2V5ICh3d3cubXlzcWwuY29tKSA8YnVpbGRAbXlzcWwuY29tPohGBBARAgAG
BQI/rOOvAAoJEK/FI0h4g3QP9pYAoNtSISDDAAU2HafyAYlLD/yUC4hKAJ0czMsBLbo0M/xP
aJ6Ox9Q5Hmw2uIhGBBARAgAGBQI/tEN3AAoJEIWWr6swc05mxsMAnRag9X61Ygu1kbfBiqDk
u4czTd9pAJ4q5W8KZ0+2ujTrEPN55NdWtnXj4YhGBBARAgAGBQJDW7PqAAoJEIvYLm8wuUtc
f3QAnRCyqF0CpMCTdIGc7bDO5I7CIMhTAJ0UTGx0O1d/VwvdDiKWj45N2tNbYIhGBBARAgAG
BQJEgG8nAAoJEAssGHlMQ+b1g3AAn0LFZP1xoiExchVUNyEf91re86gTAKDYbKP3F/FVH7Ng
c8T77xkt8vuUPYhGBBARAgAGBQJFMJ7XAAoJEDiOJeizQZWJMhYAmwXMOYCIotEUwybHTYri
Q3LvzT6hAJ4kqvYk2i44BR2W2os1FPGq7FQgeYhGBBARAgAGBQJFoaNrAAoJELvbtoQbsCq+
m48An2u2Sujvl5k9PEsrIOAxKGZyuC/VAKC1oB7mIN+cG2WMfmVE4ffHYhlP5ohGBBMRAgAG
BQJE8TMmAAoJEPZJxPRgk1MMCnEAoIm2pP0sIcVh9Yo0YYGAqORrTOL3AJwIbcy+e8HMNSoN
V5u51RnrVKie34hMBBARAgAMBQJBgcsBBYMGItmLAAoJEBhZ0B9ne6HsQo0AnA/LCTQ3P5kv
JvDhg1DsfVTFnJxpAJ49WFjg/kIcaN5iP1JfaBAITZI3H4hMBBARAgAMBQJBgcs0BYMGItlY
AAoJEIHC9+viE7aSIiMAnRVTVVAfMXvJhV6D5uHfWeeD046TAJ4kjwP2bHyd6DjCymq+BdED
z63axohMBBARAgAMBQJBgctiBYMGItkqAAoJEGtw7Nldw/RzCaoAmwWM6+Rj1zl4D/PIys5n
W48Hql3hAJ0bLOBthv96g+7oUy9Uj09Uh41lF4hMBBARAgAMBQJB0JMkBYMF1BFoAAoJEH0l
ygrBKafCYlUAoIb1r5D6qMLMPMO1krHk3MNbX5b5AJ4vryx5fw6iJctC5GWJ+Y8ytXab34hM
BBARAgAMBQJCK1u6BYMFeUjSAAoJEOYbpIkV67mr8xMAoJMy+UJC0sqXMPSxh3BUsdcmtFS+
AJ9+Z15LpoOnAidTT/K9iODXGViK6ohMBBIRAgAMBQJAKlk6BYMHektSAAoJEDyhHzSU+vhh
JlwAnA/gOdwOThjO8O+dFtdbpKuImfXJAJ0TL53QKp92EzscZSz49lD2YkoEqohMBBIRAgAM
BQJAPfq6BYMHZqnSAAoJEPLXXGPjnGWcst8AoLQ3MJWqttMNHDblxSyzXhFGhRU8AJ4ukRzf
```

```
NJqElQHQ00ZM2WnCVNzOUIhMBBIRAgAMBQJBDgqEBYMGlpoIAAoJEDnKK/Q9aopf/N0AniE2
fcCKO1wDIwusuGVlC+JvnnWbAKDDoUSEYuNn5qzRbrzWW5zBno/Nb4hMBBIRAgAMBQJCgKU0
BYMFI/9YAAoJEAQNwIV8g5+o4yQAnA9QOFLV5POCddyUMqB/fnctuO9eAJ4sJbLKP/Z3SAiT
pKrNo+XZRxauqIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAn1mxHijft00b
KXvucSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJYiKJ
AAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8JEg2aLos+5zEY
rB/LsohMBBMRAgAMBQI+TU2EBYMJV1cIAAoJEC27dr+t1MkzBQwAoJU+RuTVSn+TI+uWxUpT
82/ds5NkAJ9bnNodffyMMK7GyMiv/TzifiTD+4hMBBMRAgAMBQJB14B2BYMFzSQWAAoJEGbv
28jNgv0+P7wAn13uu8YkhwfNMJJhWdpK2/qM/4AQAJ40drnKW2qJ5EEIJwtxpwapgrzWiYhM
BBMRAgAMBQJCGIEOBYMFjCN+AAoJEHbBAxyiMW6hoO4An0Ith3Kx5/sixbjZR9aEjoePGTNK
AJ94SldLiESaYaJx2lGIlD9bbVoHQYhdBBMRAgAdBQI+PqMMBQkJZgGABQsHCgMEAxUDAgMW
AgECF4AACgkQjHGNO1By4fVxjgCeKVTBNefwxq1A6IbRr9s/Gu8r+AIAniiKdI1lFhOduUKH
AVprO3s8XerMiF0EExECAB0FAkeslLQFCQ0wWKgFCwcKAwQDFQMCAxYCAQIXgAAKCRCMcY07
UHLh9a6SAJ9/PgZQSPNeQ6LvVVzCALEBJOBt7QCffgs+vWP18JutdZc7XiawgAN9vmmIXQQT
EQIAHQUCR6yUzwUJDTBYqAULBwoDBAMVAwIDFgIBAheAAAoJEIxxjTtQcuH1dCoAoLC6RtsD
9K3N7NOxcp3PYOzH2oqzAKCFHn0jSqxk7E8by3sh+Ay8yVv0BYhdBBMRAgAdBQsHCgMEAxUD
AgMWAgECF4AFAkequSEFCQ0ufRUACgkQjHGNO1By4fUdtwCfRNcueXikBMy7tE2BbfwEyTLB
TFAAnifQGbkmcARVS7nqauGhe1ED/vdgiF0EExECAB0FCwcKAwQDFQMCAxYCAQIXgAUCS3Au
ZQUJEPPyWQAKCRCMcY07UHLh9aA+AKCHDkOBKBrGb8tOg9BIub3LFhMvHQCeIOOot1hHHUls
TIXAUrD8+ubIeZaIZQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAABIJEIxx
jTtQcuH1B2VHUEcAAQFxjgCeKVTBNefwxq1A6IbRr9s/Gu8r+AIAniiKdI1lFhOduUKHAVpr
O3s8XerMiGUEExECAB0FAkeslLQFCQ0wWKgFCwcKAwQDFQMCAxYCAQIXgAASCRCMcY07UHLh
9QdlR1BHAAEBrpIAn38+BlBI815Dou9VXMIAsQEk4G3tAJ9+Cz69Y/Xwm611lzteJrCAA32+
aYhlBBMRAgAdBQsHCgMEAxUDAgMWAgECF4AFAktwL8oFCRDz86cAEgdlR1BHAAEBCRCMcY07
UHLh9bDbAJ4mKWARqsvx4TJ8N1hPJF2oTjkeSgCeMVJljxmD+Jd4SscjSvTgFG6Q1WCIbwQw
EQIALwUCTnc9rSgdIGJ1aWxkQG15c3FsLmNvbSB3aWxsIHN0b3Agd29ya2luZyBzb29uAAoJ
EIxxjTtQcuH1tT0An3EMrSjEkUv29OX05JkLiVfQr0DPAJwKtL1ycnLPv15pGMvSzav8JyWN
3Ih7BDARAgA7BQJCdzX1NB0AT29wcy4uLiBzaG91bGQgaGF2ZSBiZWVuIGxvY2FsISBJJ20g
KnNvKiBzdHVwaWQuLi4ACgkQOcor9D1qil/vRwCdFo08f66oKLiuEAqzlf9iDlPozEEAn2Eg
vCYLCCHjfGosrkrU3WK5NFVgiI8EMBECAE8FAkVvAL9IHQBTaG91bGQgaGF2ZSBiZWVuIGEg
bG9jYWwgc2lnbmF0dXJlLCBvciBzb21ldGhpbmcgLSBXVEYgd2FzIEkgdGhpbmtpbmc/AAoJ
EDnKK/Q9aopfoPsAn3BVqKOalJeF0xPSvLR90PsRlnmGAJ44oisY7Tl3NJbPgZal8W32fbqg
bIkBHAQSAQIABgUCS8IiAwAKCRDc9Osew28OLx5CB/91LHRH0qWjPPyIrv3DTQ06x2gljQ1r
Q1MWZNuoeDfRcmgbrZxdiBzf5Mmd36liFiLmDIGLEX8vyT+Q9U/Nf1bRh/AKFkOx9PDSINWY
bE6zCI2PNKjSWFarzr+cQvfQqGX0CEILVcU1HDxZlir1nWpRccCnasMBFp52+koc6PNFjQ13
HpHbM3IcPHaaV8JD3ANyFYS4l0C/S4etDQdX37GruVb9Dcv9XkC5TS2KjDIBsEs89isHrH2+
3ZlxdLsE7LxJ9DWLxbZAND9OiiuThjAGK/pYJb+hyLLuloCg85ZX81/ZLqEOKyl55xuTvCql
tSPmSUObCuWAH+OagBdYSduxiQEiBBABAgAMBQJJKmigBQMAEnUAAAoJEJcQuJvKV618U4wI
AKk/45VnuUf9w1j7fvdzgWdIjT9Lk9dLQAGB13gEVZEVYqtYF5cEZzyxl8c7NUTCTNX3qLId
ul114A4CQQDg5U9bUwwUKaUfGLaz380mtKtM9V9A4fl9H2Gfsdumr8RPDQihfUUqju+d0ycd
mcUScj48Nctx0xhCCWNjOFPERHi9hjRQq7x6RKyFTLjM5ftdInHCo9S+mzyqz9O+iMgX68Mm
+AVgdWSC9L6yGnw6H97GD28oRMGWBTzsmCyqf9I3YutH8mGXRot3QbSJD7/AeZVh1BQwVoJn
CT8Eo1pc/OYZkRRndE1thrX0yjuFwTeOzvqeHlgzEW/FtOCBW7iR0WSJASIEEAECAAwFAkoz
TogFAwASdQAACgkQlxC4m8pXrXwXiAf+Ked6Mgd98YyTyNiLHhllPulboCnKgj430jLzkfgv
7ytVCu1xMfKrRWRw3fA9LC19mzNQX/So/o/ywsk0nUG2sfEs5FiMk+aC957Ic/MDagmXqKap
ZROJbzbZ/KNj9rPCG9kXPGa9sUn6vk39nnv4hri30tNKpM0fMxRhpcoNoCrNl4rs/QTpdRpp
7KBuNaMEtDU7R7OjMDL4qT+BcCmYMIYW4dIV7tmaC0VxtcszZcVCkxSigRMPZHwxSx37GdCx
9/+TqlA4vGL6NQSxZKv+Kqa+WTqBngOl6YGO6FxdiXEliNRpf1mafmz6h8XgYXFGpehjuX1n
60Iz0BffuWbpL4kBIgQQAQIADAUCSkRyCgUDABJ1AAAKCRCXELibyletfPaaB/9FCSmYwz7m
vzOfHZOlEAYeLnCS290XGW89o4FYTbw0PBOulygyqj2TMCK68RCNU2KFs/bXBHeS+dDzitMA
fSaULYi7LJuCCmrDM5SX5aLSj6+TxkDQDR1K1ZE3y6qd4Kx3VeeoN7Wu+oLj/3Jjbbe0uYCQ
+/PniRra9f0Z0neTExZ7CGtVBIsKS1CnKBTR26MZMOom2eTRZwGFUX1PzuW/dbZ4Z0+J6XMd
Tm2td7OYYWPbV3noblkUrxyjtGtO3ip3Oe3zSCWHUFMaaEuXOMw8tN51wy6ybcPVAH0hOiBw
b3iCFJ/20QqaZEno6edYzkqf0pwvrcTmiPb+Vj0fnjBJiQEiBBABAgAMBQJKVj5HBQMAEnUA
AAoJEJcQuJvKV61845AH/R3IkGIGOB/7x3fI0gOkOS0uFljDxysiM8FV06BfXbFpRgFMZxAh
NFUdKCDN98MDkFBd5S5aGkvhAHS7PVwQ8/BIyJaJeUG3AXmrpFV/c9kYn1+YW5OQ9E7tKu5l
5UOj1Y/weNtC04u6Rh/nrp6CvMBhH2nvhSBZ+2kO2auqtFOhuK6+wUHGixt5EK8RAKs3Sf6n
kP2EJUHzy1Q8ec5YDiaV24AVkPFBZMCkpD3Z+seIGrL4zUkV7PPY4zd9g34Oqj8JvtnA4AD/
Z1vBLujLixcQdt9aieOySA9DAVgHbe2wVS4zi5nBURsmD5u96CUOwNK1sOV+ACtdIv/T5qSU
VweJASIEEAECAAwFAkpoCoQFAwASdQAACgkQlxC4m8pXrXysfQf+IJyIPhTphk0kGPQY3v9e
3znW30VahyZxoL6q25eeQWGmVeTFlU4JThUEyzgYGip8i9qBsFPJ9XgOL5bxTGv7/WOK7eX8
e+gXHB3A2QYbrM0GFZKN3BCkbA++HmvJXU58tf+aBCB0ObG+rPn6QUNSPibu4tp65TaPVPSV
HjNTTICxu3sneHB+okJcc5z1ubme8nAytKb6x0JM/keNSXAev2ZN7zG5m+Pqw7/DQ/gCogzG
ML1bulP2rSh8bYpJPC3vAVuHTmxsbhRBg4l7j5KiHf4qMBrVzRy+YiHhwpf2p8JbCGF141+H
UD1VMeGeXnNO/9SO+dC2OGUf8WrV4FIpxIkBIgQQAQIADAUCSnkuCgUDABJ1AAAKCRCXELib
yletfBjrCACDd/zvoveoNlNiUUBazelcGXwaxSvUMSROUQNkxkoMzfA+aFpYFHWEwDfLqndp
oJTIkgkESd5fODJT26oLFekLvx3mpzfGz8l39KzDM1i6+7Mtg7DnA3kvfVIuZBNDwqoTS6hH
KcGa0MJDgzZQqJ9Ke/7T7eY+HzktUBLjzUY2kv5VV8Ji0p6xY27jT73xiDov00ZbBFN+xBtx
2iRmjjgnPtjt/zU5sLiv9fUOA+Pb53gBT+mXMNx2tsg07Kmuz7vfjR5ydoY7guyB3X1vUK9y
AmCW1Gq67eRG934SujZFikO/oZUrwRrQu2jj5v8B7xwtcCFCdpZAIRabD4BTglvPiQEiBBAB
AgAMBQJKjl+9BQMAEnUAAAoJEJcQuJvKV618DTwH/3DzIl1zwr6TTtTfTBH9FSDdhvaUEPKC
bLT3WZWzIHREaLEENcQ85cGoYoBeJXVBIwBczZUpGy4pqFjYcWQ9vKFm2Nt1Nrs+v9tKc+9G
```

```
ECH0Y1a+9GDYqnepcN2O/3HLASCEpXFwQhVe01G+lupGgqYfMgTG9RByTkMzVXB9ER5gijGC
zjTflYAOFUx2eBBLYa3w/ZZpT+nwRmEUaDpfwq06UPrzMZuhol7SGPZUNz4lz4p2NF8Td9bk
hOiJ3+gORRohbq0HdaRdvSDoP/aGsQltfeF5p0KEcpIHx5B05H1twIkOGFTxyx3nTWqauEJy
2a+Wl5ZBl0hB2TqwAE9Z54KJASIEEAECAAwFAkqgEkcFAwASdQAACgkQlxC4m8pXrXwyXwf/
UPzz+D+n19JWivha7laUxuDzMQCKTcEjFCu4QVZ1rqcBFPoz0Tt74/X75QdmxZizqX1E6lbF
EsbVjL2Mt5zZjedS1vbSbrmn4hV4pHZr08dbflZkNX105g8ZlpsqQ7VyUt5YtWCn0tGNn4B5
Eb6WMeqxQteujV3B7AtMH+CD0ja+A2/p0rHIpqScz8aupksBMCrYqhoT+7/qXNEVkjNmcu2N
mHxfv6dL5Xy/0iJjie2umStu8WTfRTpYmnv2gEhbCdb/zhFvG61GgTBJqv9MvBVGRxnJFd4l
NqlucsadD+UM7WjV3v5VuN2r9KD9wocd/s22ELCRA2wKccvR/nWBkIkBIgQQAQIADAUCSqgQ
AAUDABJ1AAAKCRCXELibyletfAT8B/9cPhH8DlHoiv+cK8rAJMomZqVqOyy4BwsRrakycVlg
7/yvMs74anynSoUf0LgsXADQ29Hmrpf+zC5E5/jPGWNK81x2VBVoB8nZkMSAnkZfOw+mWu9I
Aj2NLcsvt9JYNmAq5R7RrirHsDQ2DIYxRgaE/5CVEVry9YQEj18A13/SYyoB4FWpDI4fRfUW
JbUJrYmfg0p+4zL0YS9F11UhsHUu+g1W1c83N54ozI1v0l3HUwVayzII4E/YNrIkpOaO+o8R
z9g6M6jCg3mwn+OfiZVJO++VOiguJF5KzoZIICMxXE3t5hL87Kroi7UkNwm+YHw3ZaLEBm0B
WAXw4DsJZcpViQEiBBABAgAMBQJKuceJBQMAEnUAAAoJEJcQuJvKV6188KEH/24QK2LV1l42
4Wx3T9G4bJFRWWuuEkTpYJw6ss72lqus9t7BsoGaNLMHQzKAlca9wLTqY826q4nv9anEqwWZ
+Di8kE+UAMUq2BFTL0EvOMJ6i1ZyE8cUFVb1+09tpBWJJS7t3z00uMMMznGuHzSm4MgCnGhA
sOgiuHdPWSlnHnqNJa/SB6UVQxtcDOaqQlLIvhd2HVqrOBRtER3td/YgLO6HSxXpXtz8DBa2
NYQYSwAdlqJAPLBnBsLXwbCswuIDMZZv8BJwUNBEJkokOMv5CXxhPrP5kxWvyBvsIhTk8ph2
GIh/ZRVNDAsChbuU1EJBACpwaMrcgwjPtI7/KTgeZVSJASIEEAECAAwFAkreCMYFAwASdQAA
CgkQlxC4m8pXrXyOQQf7BvRm/3PvFCCksyjBW4EVBW7z/Ps/kBK6bIE9Q7f7QlXFIcGGUIpA
rufXWbV+G4a3Z8LFeFJTovNePfquwpFjneUZn1CG+oVS1AfddvYhAsgkLhQqMbaNJIJ1y4D/
H3xvCna/s7Teufud0JLXoLBedFXeB5Cg2KlEoxINqMo+lm/VGJmbykwqoRvxZLDfnbFag5zG
59+OWw4TC8nzlIQYIBn22YiWRk5zsCJA40O+KL1vwBiFDrREhALQc/YBJKYrRX3ZV4U/EeYD
KB0NCBk1W1tXGCee3uhM0S5VFc1j7Pg58ECuntH5xOy+KMNFljiQwvWfbaFTJvCjFQS+OplX
b4kBIgQQAQIADAUCSu86VAUDABJ1AAAKCRCXELibyletfGs8CACteI2BmKs24GF80JeWTOQI
cvHnCdV7hKZOltbNPBbDv6qTt3iX2GVa10iYhI5Eg3Ojt/hKFJTMlfYZyI1peFodGjv7Lk5l
u7zaNBvT1pBCP+eJspi6rGpSuhtMSb4O5jPclRBmbY+w9wctLyZf1zG+slSdw8adcRXQNFqr
vVIZYOmu2S8FunqLfxpjewiFiDPzAzmbWzMoO2PLCYFhwV6Eh2jO33OGbvBmyHNFZBfX5F/+
kiyeT47MEhrfhytJ6ZOdpxtX8HvbvzPZcDLOI80W6rPTG76KW06ZiZrJ81YCa6a7D01y7BYy
W2HoxzYcuumjRkGF4nqK4Mw+wefCp0H/iQEiBBABAgAMBQJLAF3aBQMAEnUAAAoJEJcQuJvK
V618/q0H/ibXDQG2WQmC1LoT4H+ezXjPgDg8aiuz6f4xibTmrO+L4ScMX+zK0KZVwp6Kau28
Nx+gO0oAUW8mNxhd+cl0ZaY+7RIkxEvkooKKsArBmZT+xrE6CgHlAs3D4Mc+14nfD0aZaUbE
iobWvXlYLl27MELLcWyeMlgbeNoucc473JddvmHSRRM5F9Qp28CvWDEXYqhq1laoaho8+cei
pvzyuO3OTwjuAOqhefOHzAvFrRli99ML8xzF1ZOvBct+36SuYxDXyIhkSd7aG9Us0lW6W5Si
JYt4cDyI0JDhbhZN0tzWYKcKMZMxf8w3jW4sfQL0prhHrARqqPiU8OTUH/VNX5CJASIEEAEC
AAwFAksRgasFAwASdQAACgkQlxC4m8pXrXydogf/a31ofmYFMoE3p9SqGt/v28iyO0j9A1Lm
qKwEhJkxff/X/Qa7pafGQ9J90JQkxYKMxydWPspTbDFMccZWkBK132vZp9Q3FHKpnDPDLK2S
25miTReeAAQNgMMFLeyy7ZHi5YsKwLbKxcSo7/m0jlitNYlmt94imFNpg/mHGsy6O+rLeQTA
opuIzP3VwN6ItL5gIFxqWPmf/V0xh/vxTwLqJ66vECD8vyHrHblUzgiXHgyYbZPxAa2SRRd3
4V38phaZ/QsTkss+Sd/QeHChWyU9d6KengWwcr/nDO+K/hhmnO5Oqz02UpwyxrgI6484HQUN
/Smf44VBsSD1DBjaAKjMr4kBIgQQAQIADAUCSyNN1AUDABJ1AAAKCRCXELibyletfCWiB/9c
EZtdFVcsxpE3hJzM6PBPf+1QKuJORve/7MqNEb3TMWFgBxyOfvD7uMpCJyOrqq5AbUQfZfj9
K7qmzWUMuoYceGIlbdmHFBJwtmaF0BiyHaobgY/9RbdCNcbtzrW34feiW9aDZyvCoLHEVkCC
QACSv3FwdYVkkRB5eihvpwJk5tpScdIA12YLqzmVTFdhrZuYvtDdQHjgoLMO8B9s9kok7D2T
SpveVzXXPH68Z3JkVubhHT7cs+n+9PRvcaVJtsX2VTUY5eFVqmGuAUVrvp2aN8cKQ+mVcCQr
VVIhT9o8YB5925MUx2VJml0y0nkBQuMZyzMEOVGkuU/G+pVrRmmAiQEiBBABAgAMBQJLJyaS
BQMAEnUAAAoJEJcQuJvKV618eU0IAKnVh6ymId9C3ZqVyxwTnOB8RMQceJzwCLqk2RT0dPhN
5ZwUcQN7lCp9hymMutC8FdKRK/ESK21vJF2/576Pln4fIeOIbycBAEvqrL14epATj53uBizo
NOTuwb1kximFERuW3MP4XiFUJB0tPws2vR5UU3t6GoQJJwNoIbz9DK2L6X/Qz3Tb9if6bPSK
U6JR1Yn3Hos9ogg21vWCxgMTKUuPAYhmYjSvkqH3BihXi+c17MVvE7W5GJbQHuJo+MgSxu04
4qnvDHZpf4Mzc30XcG1ohjxefNyeiY2bzdI2yCaCtmWOlCW1Sc2oiE0zwO6lD4hY5XmC2Xql
MLsKB5VNXJGJASIEEAECAAwFAks4Ze4FAwASdQAACgkQlxC4m8pXrXyWXggAon2abiNvRzx9
7364Mjx4IlFvM1tVebzNbOkDwZS1ABqTDGgq/ffZA/VZrU+h2eL97cQyGxJEQ5kkm/v1iobE
ZEFMT0pv9WMzfidqzhdKdcpbbxdaErIjD5fBACKdjazAUeH7zce2v+bBN0l9LZoRiXbNugG9
38lkJ2E4ZTYYfvftL/e4RzOgqR9VD/A5MzxfXFbCVharHbeT8OwZy4Oz2UDaDszHsNKoG1WN
pOSf2HTMBPNcsOSY/hIBRWNxnzdYOkWt7laeLNmN1eUEwzk4J7GnlambPIctOdoEUriMSaey
TkLZGejKnwi/PqARyDW1FsReKNHD753ZMViUnAsq2IkBIgQQAQIADAUCS0oyJwUDABJ1AAAK
CRCXELibyletfGodCAC5hjmxwquHSb8ZL0RifIL3j3iU6U7qLK1TQKkTqgELfUzeF9f8NuNR
txLmzNk1T7YI9iji6NAtnuy43v61OMbqlkV8x69qNP36Owv408wXxEt0s5ViZuVOZJAY075c
YRhopgfmhkh4hbkAoKCLajOR0WUEEsDHsqqj8XLJuGRREURy8TJWaB/cotXsgiJf99gt+gIw
In8tyb3+WVIUHWfw2+Drpd3nfcMqgeO54PePJo0BWWjaar+wgC/76Se286IHcYMrml/Adnvx
ZaIKmxZmkTmDMCfMnVjRYSKBGjQ9Uu7dws7SMsbbd34f8Jt9nyuRqMcl4INAXthWY/S3Sdil
iQEiBBABAgAMBQJLW/5mBQMAEnUAAAoJEJcQuJvKV6181L8IAKq3ZOQHzqaOoz5wnvj51YG8
nZoW5RG7HOb3mL1D9b+FTTzaIxsLf7STagPwKtM57rU/7ehHIuO/9QQNQ3Mudw17ZiwD0l5X
7iG8/AflWnc6bXfTz18IplRuqyVc0qQeJZhT7MBpklcS4ZGZHPQdtAh4Aw5YXihrbbq6jV7j
CzUmFz4XcT8CkJHIUGoFR0vTmFqlAt2K1imwGMh2IEamPOJ0wsTbBfZbhmkB03RToEjIipGZ
M+NtKS/NL2RJYWZ+FCCcEMoRgmlVmATWw3natgLWwN4Z6K4rGXONWi/0wyFgxZpmjdHmjcXa
Igz8EroVsLbnaV/8yG7cgK5e6M0Fk1iJASIEEAECAAwFAkttIfgFAwASdQAACgkQlxC4m8pX
rXyR3QgAksvAMfqC+ACUEWSVAlepDFR1xI45UwBa2UeBY7KjOOCiZlkGREvx20IOv1gExyPl
zNxDeqmYsl2mleEoH6QlXaJRd8MxIVfAnjAt8izwU2dfDwflTTWgGQYf8q7qeAv1XC34yNge
0JaTD1C55QpmcO51f2ojMsAi36bBJO4Dr59jhVYiDjQADS/d7FpAznlhH9SGUq6ekYb2jxCS
```

```
rvt0wRtMyk6YGgts4xEHcN0wC9VTobaXo9xvsqhtUK44Gdvptq1cBFX8byzD6fN8nXp+v8qh
tlPYDqb4muqTh2UXXiWMtvPXo7kkZQ8CvI3YbZ10F1IDLt20VJWFZaJYL2fzyokCIgQQAQIA
DAUCQYHLhQWDBiLZBwAKCRCq4+bOZqFEaKgvEACCErnaHGyUYa0wETjj6DLEXsqeOiXad4i9
aBQxnD35GUgcFofC/nCY4XcnCMMEnmdQ9ofUuU3OBJ6BNJIbEusAabgLooebP/3KEaiCIiyh
HYU5jarpZAh+Zopgs3Oc11mQ1tIaS69iJxrGTLodkAsAJAeEUwTPq9fHFFzC1eGBysoyFWg4
bIjz/zClI+qyTbFA5g6tRoiXTo8ko7QhY2AA5UGEg+83Hdb6akC04Z2QRErxKAqrphHzj8Xp
jVOsQAdAi/qVKQeNKROlJ+iq6+YesmcWGfzeb87dGNweVFDJIGA0qY27pTb2lExYjsRFN4Cb
13NfodAbMTOxcAWZ7jAPCxAPlHUG++mHMrhQXEToznBFE4nbnC7vOBNgWdjUgXcpkUCkop4b
17BFpR+k8ZtYLSS8p2LLz4uAeCcSm2/msJxT7rC/FvoH8428oHincqs2ICo9zO/Ud4HmmO0O
+SsZdVKIIjinGyOVWb4OOzkAlnnhEZ3o6hAHcREIsBgPwEYVTj/9ZdC0AO44Nj9cU7awaqgt
rnwwfr/o4V2gl8bLSkltZU27/29HeuOeFGjlFe0YrDd/aRNsxbyb2O28H4sG1CVZmC5uK1iQ
BDiSyA7Q0bbdofCWoQzm5twlpKWnY8Oe0ub9XP5p/sVfck4FceWFHwv+/PC9RzSl33lQ6vM2
wIkCIgQTAQIADAUCQp8KHAWDBQWacAAKCRDYwgoJWiRXzyE+D/9uc7z6fIsalfOYoLN60ajA
bQbI/uRKBFugyZ5RoaItusn9Z2rAtn61WrFhu4uCSJtFN1ny2RERg40f56pTghKrD+YEt+Nz
e6+FKQ5AbGIdFsR/2bUk+ZZRSt83e14Lcb6ii/fJfzkoIox9ltkifQxqY7Tvk4noKu4oLSc8
O1Wsfc/y0B9sYUUCmUfcnq58DEmGie9ovUslmyt5NPnveXxp5UeaRc5Rqt9tK2B4A+7/cqEN
rdZJbAMSunt2+2fkYiRunAFPKPBdJBsY1sxeL/A9aKe0viKEXQdAWqdNZKNCi8rd/oOP99/9
lMbFudAbX6nL2DSb1OG2Z7NWEqgIAzjmpwYYPCKeVz5Q8R+if9/fe5+STY/55OaI33fJ2H3v
+U435VjYqbrerWe36xJItcJeqUzW71fQtXi1CTEl3w2ch7VF5oj/QyjabLnAlHgSlkSi6p7B
y5C2MnbCHlCfPnIinPhFoRcRGPjJe9nFwGs+QblvS/Chzc2WX3s/2SWm4gEUKRX4zsAJ5ocy
fa/vkxCkSxK/erWlCPf/J1T70+i5waXDN/E3enSet/WL7h94pQKpjz8OdGL4JSBHuAVGA+a+
dknqnPF0KMKLhjrgV+L7O84FhbmAP7PXm3xmiMPriXf+el5fZZequQoIagf8rdRHHhRJxQgI
0HNknkaOqs8dtrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWsEN/lxaZoJYc3a6M0
2WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLmRDRiRjd1DTCHqeyX7CHhcghj
/dNRlW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hkAWzE7zaD5cH9J7yv/6xuZVw411x0h4Uq
sTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fmLe11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRs
KTfozBu74F47D8Ilbf5vSYHbuE5p/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnl
LzKUb/F5GwADBQf+Lwqqa8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPD
sDD9MZ1ZaSafanFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGo
TbOWI39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42LmuQT5N
dKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt926s/ymfdf5Hkd
Q1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZWhe70YGNPw1yjWJT1IhM
BBgRAgAMBQI+PqMdBQkJZgGAAAoJEIxxjTtQcuH17p4An3r1QpVC9yhnW2cSAjq+kr72GX0e
AJ4295kl6NxYEuFApmr1+0uUq/SlsYhMBBgRAgAMBQJHrJT8BQkNMFjfAAoJEIxxjTtQcuH1
pc4An0I965H3JY2GTrizp+dCezxbhexaAJ48FhocFYvfhZtgeUWb6aPvgQZHT4hUBBgRAgAM
BQI+PqMdBQkJZgGAABIJEIxxjTtQcuH1B2VHUEcAAQHungCfevVClUL3KGdbZxICOr6SvvYZ
fR4Anjb3mSXo3FgS4UCmavX7S5Sr9KWxiFQEGBECAAwFAk53Pe0FCRP7AbgAEgdlR1BHAAEB
CRCMcY07UHLh9RSbAJsFivb5sESf8vYE5yfD1n9AVa6FEwCgpWAIWbl9p1DcB+L5RCUBw6mG
uck=
=yia9
-----END PGP PUBLIC KEY BLOCK-----
```

## 2.1.5 Installation Layouts

The installation layout differs for different installation types (for example, native packages, binary tarballs, and source tarballs), which can lead to confusion when managing different systems or using different installation sources. The individual layouts are given in the corresponding installation type or platform chapter, as described following. Note that the layout of installations from vendors other than Oracle may differ from these layouts.

- Section 2.3.1, “MySQL Installation Layout on Microsoft Windows”
- Section 2.8.3, “MySQL Layout for Source Installation”
- Table 2.3, “MySQL Installation Layout for Generic Unix/Linux Binary Package”
- Table 2.12, “MySQL Installation Layout for Linux RPM Packages from the MySQL Developer Zone”
- Table 2.7, “MySQL Installation Layout on macOS”

## 2.1.6 Compiler-Specific Build Characteristics

In some cases, the compiler used to build MySQL affects the features available for use. The notes in this section apply for binary distributions provided by Oracle Corporation or that you compile yourself from source.

**`icc` (Intel C++ Compiler) Builds**

A server built with `icc` has these characteristics:

- SSL support is not included.

## 2.2 Installing MySQL on Unix/Linux Using Generic Binaries

Oracle provides a set of binary distributions of MySQL. These include generic binary distributions in the form of compressed `tar` files (files with a `.tar.xz` extension) for a number of platforms, and binaries in platform-specific package formats for selected platforms.

This section covers the installation of MySQL from a compressed `tar` file binary distribution on Unix/Linux platforms. For Linux-generic binary distribution installation instructions with a focus on MySQL security features, refer to the Secure Deployment Guide. For other platform-specific binary package formats, see the other platform-specific sections in this manual. For example, for Windows distributions, see Section 2.3, “Installing MySQL on Microsoft Windows”. See Section 2.1.3, “How to Get MySQL” on how to obtain MySQL in different distribution formats.

MySQL compressed `tar` file binary distributions have names of the form `mysql-VERSION-OS.tar.xz`, where *`VERSION`* is a number (for example, `8.0.32`), and *`OS`* indicates the type of operating system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

There is also a “minimal install” version of the MySQL compressed `tar` file for the Linux generic binary distribution, which has a name of the form `mysql-VERSION-OS-GLIBCVER-ARCH-minimal.tar.xz`. The minimal install distribution excludes debug binaries and is stripped of debug symbols, making it significantly smaller than the regular binary distribution. If you choose to install the minimal install distribution, remember to adjust for the difference in file name format in the instructions that follow.

**Warnings**

- If you have previously installed MySQL using your operating system native package management system, such as Yum or APT, you may experience problems installing using a native binary. Make sure your previous MySQL installation has been removed entirely (using your package management system), and that any additional files, such as old versions of your data files, have also been removed. You should also check for configuration files such as `/etc/my.cnf` or the `/etc/mysql` directory and delete them.

  For information about replacing third-party packages with official MySQL packages, see the related APT guide or Yum guide.

- MySQL has a dependency on the `libaio` library. Data directory initialization and subsequent server startup steps fail if this library is not installed locally. If necessary, install it using the appropriate package manager. For example, on Yum-based systems:

  ```
  $> yum search libaio  # search for info
  $> yum install libaio # install library
  ```

  Or, on APT-based systems:

  ```
  $> apt-cache search libaio # search for info
  $> apt-get install libaio1 # install library
  ```

- **Oracle Linux 8 / Red Hat 8** (EL8): These platforms by default do not install the file `/lib64/libtinfo.so.5`, which is required by the MySQL client `bin/mysql` for packages `mysql-VERSION-el7-x86_64.tar.gz` and `mysql-VERSION-linux-glibc2.12-x86_64.tar.xz`. To work around this issue, install the `ncurses-compat-libs` package:

```
$> yum install ncurses-compat-libs
```

To install a compressed `tar` file binary distribution, unpack it at the installation location you choose (typically `/usr/local/mysql`). This creates the directories shown in the following table.

**Table 2.3 MySQL Installation Layout for Generic Unix/Linux Binary Package**

| Directory | Contents of Directory |
|---|---|
| `bin` | `mysqld` server, client and utility programs |
| `docs` | MySQL manual in Info format |
| `man` | Unix manual pages |
| `include` | Include (header) files |
| `lib` | Libraries |
| `share` | Error messages, dictionary, and SQL for database installation |
| `support-files` | Miscellaneous support files |

Debug versions of the `mysqld` binary are available as `mysqld-debug`. To compile your own debug version of MySQL from a source distribution, use the appropriate configuration options to enable debugging support. See Section 2.8, “Installing MySQL from Source”.

To install and use a MySQL binary distribution, the command sequence looks like this:

```
$> groupadd mysql
$> useradd -r -g mysql -s /bin/false mysql
$> cd /usr/local
$> tar xvf /path/to/mysql-VERSION-OS.tar.xz
$> ln -s full-path-to-mysql-VERSION-OS mysql
$> cd mysql
$> mkdir mysql-files
$> chown mysql:mysql mysql-files
$> chmod 750 mysql-files
$> bin/mysqld --initialize --user=mysql
$> bin/mysql_ssl_rsa_setup
$> bin/mysqld_safe --user=mysql &
# Next command is optional
$> cp support-files/mysql.server /etc/init.d/mysql.server
```

**Note**

This procedure assumes that you have `root` (administrator) access to your system. Alternatively, you can prefix each command using the `sudo` (Linux) or `pfexec` (Solaris) command.

The `mysql-files` directory provides a convenient location to use as the value for the `secure_file_priv` system variable, which limits import and export operations to a specific directory. See Section 5.1.8, “Server System Variables”.

A more detailed version of the preceding description for installing a binary distribution follows.

## Create a mysql User and Group

If your system does not already have a user and group to use for running `mysqld`, you may need to create them. The following commands add the `mysql` group and the `mysql` user. You might want to call the user and group something else instead of `mysql`. If so, substitute the appropriate name in the following instructions. The syntax for `useradd` and `groupadd` may differ slightly on different versions of Unix/Linux, or they may have different names such as `adduser` and `addgroup`.

```
$> groupadd mysql
$> useradd -r -g mysql -s /bin/false mysql
```

**Note**

Because the user is required only for ownership purposes, not login purposes, the `useradd` command uses the `-r` and `-s /bin/false` options to create a user that does not have login permissions to your server host. Omit these options if your `useradd` does not support them.

## Obtain and Unpack the Distribution

Pick the directory under which you want to unpack the distribution and change location into it. The example here unpacks the distribution under `/usr/local`. The instructions, therefore, assume that you have permission to create files and directories in `/usr/local`. If that directory is protected, you must perform the installation as `root`.

```
$> cd /usr/local
```

Obtain a distribution file using the instructions in Section 2.1.3, “How to Get MySQL”. For a given release, binary distributions for all platforms are built from the same MySQL source distribution.

Unpack the distribution, which creates the installation directory. `tar` can uncompress and unpack the distribution if it has `z` option support:

```
$> tar xvf /path/to/mysql-VERSION-OS.tar.xz
```

The `tar` command creates a directory named `mysql-VERSION-OS`.

To install MySQL from a compressed `tar` file binary distribution, your system must have GNU `XZ Utils` to uncompress the distribution and a reasonable `tar` to unpack it.

**Note**

The compression algorithm changed from Gzip to XZ in MySQL Server 8.0.12; and the generic binary's file extension changed from .tar.gz to .tar.xz.

GNU `tar` is known to work. The standard `tar` provided with some operating systems is not able to unpack the long file names in the MySQL distribution. You should download and install GNU `tar`, or if available, use a preinstalled version of GNU tar. Usually this is available as `gnutar`, `gtar`, or as `tar` within a GNU or Free Software directory, such as `/usr/sfw/bin` or `/usr/local/bin`. GNU `tar` is available from http://www.gnu.org/software/tar/.

If your `tar` does not support the `xz` format then use the `xz` command to unpack the distribution and `tar` to unpack it. Replace the preceding `tar` command with the following alternative command to uncompress and extract the distribution:

```
$> xz -dc /path/to/mysql-VERSION-OS.tar.xz | tar x
```

Next, create a symbolic link to the installation directory created by `tar`:

```
$> ln -s full-path-to-mysql-VERSION-OS mysql
```

The `ln` command makes a symbolic link to the installation directory. This enables you to refer more easily to it as `/usr/local/mysql`. To avoid having to type the path name of client programs always when you are working with MySQL, you can add the `/usr/local/mysql/bin` directory to your `PATH` variable:

```
$> export PATH=$PATH:/usr/local/mysql/bin
```

## Perform Postinstallation Setup

The remainder of the installation process involves setting distribution ownership and access permissions, initializing the data directory, starting the MySQL server, and setting up the configuration file. For instructions, see Section 2.9, “Postinstallation Setup and Testing”.

# 2.3 Installing MySQL on Microsoft Windows

**Important**

MySQL 8.0 Server requires the Microsoft Visual C++ 2019 Redistributable Package to run on Windows platforms. Users should make sure the package has been installed on the system before installing the server. The package is available at the Microsoft Download Center. Additionally, MySQL debug binaries require Visual Studio 2019 to be installed.

MySQL is available for Microsoft Windows 64-bit operating systems only. For supported Windows platform information, see https://www.mysql.com/support/supportedplatforms/database.html.

There are different methods to install MySQL on Microsoft Windows.

## MySQL Installer Method

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure a specific version of MySQL Server as follows:

1. Download MySQL Installer from https://dev.mysql.com/downloads/installer/ and execute it.

   **Note**

   Unlike the standard MySQL Installer, the smaller `web-community` version does not bundle any MySQL applications, but downloads only the MySQL products you choose to install.

2. Determine the setup type to use for the initial installation of MySQL products. For example:

   - **Developer Default**: Provides a setup type that includes the selected version of MySQL Server and other MySQL tools related to MySQL development, such as MySQL Workbench.
   - **Server Only**: Provides a setup for the selected version of MySQL Server without other products.
   - **Custom**: Enables you to select any version of MySQL Server and other MySQL products.

3. Install the server instance (and products) and then begin the server configuration by following the onscreen instructions. For more information about each individual step, see MySQL Server Configuration with MySQL Installer.

MySQL is now installed. If you configured MySQL as a service, then Windows automatically starts the MySQL server every time you restart the system. Also, this process installs the MySQL Installer application on the local host, which you can use later to upgrade or reconfigure MySQL server.

**Note**

If you installed MySQL Workbench on your system, consider using it to check your new MySQL server connection. By default, the program automatically start after installing MySQL.

## Additional Installation Information

It is possible to run MySQL as a standard application or as a Windows service. By using a service, you can monitor and control the operation of the server through the standard Windows service management tools. For more information, see Section 2.3.4.8, “Starting MySQL as a Windows Service”.

To accommodate the `RESTART` statement, the MySQL server forks when run as a service or standalone, to enable a monitor process to supervise the server process. In this case, there are two `mysqld` processes. If `RESTART` capability is not required, the server can be started with the `--no-monitor` option. See Section 13.7.8.8, “RESTART Statement”.

Generally, you should install MySQL on Windows using an account that has administrator rights. Otherwise, you may encounter problems with certain operations such as editing the `PATH` environment variable or accessing the `Service Control Manager`. When installed, MySQL does not need to be executed using a user with Administrator privileges.

For a list of limitations on the use of MySQL on the Windows platform, see Section 2.3.7, “Windows Platform Restrictions”.

In addition to the MySQL Server package, you may need or want additional components to use MySQL with your application or development environment. These include, but are not limited to:

- To connect to the MySQL server using ODBC, you must have a Connector/ODBC driver. For more information, including installation and configuration instructions, see MySQL Connector/ODBC Developer Guide.

  **Note**

  MySQL Installer installs and configures Connector/ODBC for you.

- To use MySQL server with .NET applications, you must have the Connector/NET driver. For more information, including installation and configuration instructions, see MySQL Connector/NET Developer Guide.

  **Note**

  MySQL Installer installs and configures MySQL Connector/NET for you.

MySQL distributions for Windows can be downloaded from https://dev.mysql.com/downloads/. See Section 2.1.3, “How to Get MySQL”.

MySQL for Windows is available in several distribution formats, detailed here. Generally speaking, you should use MySQL Installer. It contains more features and MySQL products than the older MSI, is simpler to use than the compressed file, and you need no additional tools to get MySQL up and running. MySQL Installer automatically installs MySQL Server and additional MySQL products, creates an options file, starts the server, and enables you to create default user accounts. For more information on choosing a package, see Section 2.3.2, “Choosing an Installation Package”.

- A MySQL Installer distribution includes MySQL Server and additional MySQL products including MySQL Workbench, and MySQL for Visual Studio. MySQL Installer can also be used to upgrade these products in the future (see https://dev.mysql.com/doc/mysql-compat-matrix/en/).

  For instructions on installing MySQL using MySQL Installer, see Section 2.3.3, “MySQL Installer for Windows”.

- The standard binary distribution (packaged as a compressed file) contains all of the necessary files that you unpack into your chosen location. This package contains all of the files in the full Windows MSI Installer package, but does not include an installation program.

  For instructions on installing MySQL using the compressed file, see Section 2.3.4, “Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive”.

- The source distribution format contains all the code and support files for building the executables using the Visual Studio compiler system.

  For instructions on building MySQL from source on Windows, see Section 2.8, “Installing MySQL from Source”.

## MySQL on Windows Considerations

- **Large Table Support**

If you need tables with a size larger than 4GB, install MySQL on an NTFS or newer file system. Do not forget to use `MAX_ROWS` and `AVG_ROW_LENGTH` when you create tables. See Section 13.1.20, “CREATE TABLE Statement”.

- **MySQL and Virus Checking Software**

  Virus-scanning software such as Norton/Symantec Anti-Virus on directories containing MySQL data and temporary tables can cause issues, both in terms of the performance of MySQL and the virus-scanning software misidentifying the contents of the files as containing spam. This is due to the fingerprinting mechanism used by the virus-scanning software, and the way in which MySQL rapidly updates different files, which may be identified as a potential security risk.

  After installing MySQL Server, it is recommended that you disable virus scanning on the main directory (`datadir`) used to store your MySQL table data. There is usually a system built into the virus-scanning software to enable specific directories to be ignored.

  In addition, by default, MySQL creates temporary files in the standard Windows temporary directory. To prevent the temporary files also being scanned, configure a separate temporary directory for MySQL temporary files and add this directory to the virus scanning exclusion list. To do this, add a configuration option for the `tmpdir` parameter to your `my.ini` configuration file. For more information, see Section 2.3.4.2, “Creating an Option File”.

## 2.3.1 MySQL Installation Layout on Microsoft Windows

For MySQL 8.0 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.0` for installations performed with MySQL Installer. If you use the ZIP archive method to install MySQL, you may prefer to install in `C:\mysql`. However, the layout of the subdirectories remains the same.

All of the files are located within this parent directory, using the structure shown in the following table.

**Table 2.4 Default MySQL Installation Layout for Microsoft Windows**

| Directory | Contents of Directory | Notes |
|---|---|---|
| `bin` | `mysqld` server, client and utility programs | |
| `%PROGRAMDATA%\MySQL\MySQL Server 8.0\` | Log files, databases | The Windows system variable `%PROGRAMDATA%` defaults to `C:\ProgramData`. |
| `docs` | Release documentation | With MySQL Installer, use the `Modify` operation to select this optional folder. |
| `include` | Include (header) files | |
| `lib` | Libraries | |
| `share` | Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation | |

## 2.3.2 Choosing an Installation Package

For MySQL 8.0, there are multiple installation package formats to choose from when installing MySQL on Windows. The package formats described in this section are:

- MySQL Installer
- MySQL noinstall ZIP Archives

- MySQL Docker Images

Program Database (PDB) files (with file name extension `pdb`) provide information for debugging your MySQL installation in the event of a problem. These files are included in ZIP Archive distributions (but not MSI distributions) of MySQL.

## MySQL Installer

This package has a file name similar to `mysql-installer-community-8.0.32.0.msi` or `mysql-installer-commercial-8.0.32.0.msi`, and utilizes MSIs to install MySQL server and other products automatically. MySQL Installer downloads and applies updates to itself, and to each of the installed products. It also configures the installed MySQL server (including a sandbox InnoDB cluster test setup) and MySQL Router. MySQL Installer is recommended for most users.

MySQL Installer can install and manage (add, modify, upgrade, and remove) many other MySQL products, including:

- Applications – MySQL Workbench, MySQL for Visual Studio, MySQL Shell, and MySQL Router (see https://dev.mysql.com/doc/mysql-compat-matrix/en/)
- Connectors – MySQL Connector/C++, MySQL Connector/NET, Connector/ODBC, MySQL Connector/Python, MySQL Connector/J, MySQL Connector/Node.js
- Documentation – MySQL Manual (PDF format), samples and examples

MySQL Installer operates on all MySQL supported versions of Windows (see https://www.mysql.com/support/supportedplatforms/database.html).

**Note**

Because MySQL Installer is not a native component of Microsoft Windows and depends on .NET, it does not work with minimal installation options like the Server Core version of Windows Server.

For instructions on how to install MySQL using MySQL Installer, see Section 2.3.3, “MySQL Installer for Windows”.

## MySQL noinstall ZIP Archives

These packages contain the files found in the complete MySQL Server installation package, with the exception of the GUI. This format does not include an automated installer, and must be manually installed and configured.

The `noinstall` ZIP archives are split into two separate compressed files. The main package is named `mysql-VERSION-winx64.zip`. This contains the components needed to use MySQL on your system. The optional MySQL test suite, MySQL benchmark suite, and debugging binaries/information components (including PDB files) are in a separate compressed file named `mysql-VERSION-winx64-debug-test.zip`.

If you choose to install a `noinstall` ZIP archive, see Section 2.3.4, “Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive”.

## MySQL Docker Images

For information on using the MySQL Docker images provided by Oracle on Windows platform, see Section 2.5.6.3, “Deploying MySQL on Windows and Other Non-Linux Platforms with Docker”.

**Warning**

The MySQL Docker images provided by Oracle are built specifically for Linux platforms. Other platforms are not supported, and users running the MySQL Docker images from Oracle on them are doing so at their own risk.

## 2.3.3 MySQL Installer for Windows

MySQL Installer is a standalone application designed to ease the complexity of installing and configuring MySQL products that run on Microsoft Windows. It is downloaded with and supports the following MySQL products:

- MySQL Servers

  MySQL Installer can install and manage multiple, separate MySQL server instances on the same host at the same time. For example, MySQL Installer can install, configure, and upgrade separate instances of MySQL 5.7 and MySQL 8.0 on the same host. MySQL Installer does not permit server upgrades between major and minor version numbers, but does permit upgrades within a release series (such as 8.0.21 to 8.0.22).

  **Note**

  MySQL Installer cannot install both *Community* and *Commercial* releases of MySQL server on the same host. If you require both releases on the same host, consider using the ZIP archive distribution to install one of the releases.

- MySQL Applications

  MySQL Workbench, MySQL Shell, MySQL Router, and MySQL for Visual Studio.

- MySQL Connectors

  MySQL Connector/NET, MySQL Connector/Python, MySQL Connector/ODBC, MySQL Connector/J, and MySQL Connector/C++. To install MySQL Connector/Node.js, see https://dev.mysql.com/downloads/connector/nodejs/.

### Installation Requirements

MySQL Installer requires Microsoft .NET Framework 4.5.2 or later. If this version is not installed on the host computer, you can download it by visiting the Microsoft website.

To invoke MySQL Installer after a successful installation:

1. Right-click Windows Start, select **Run**, and then click **Browse**. Navigate to `Program Files (x86) > MySQL > MySQL Installer for Windows` to open the program folder.
2. Select one of the following files:
   - `MySQLInstaller.exe` to open the graphical application.
   - `MySQLInstallerConsole.exe` to open the command-line application.
3. Click **Open** and then click **OK** in the Run window. If you are prompted to allow the application to make changes to the device, select `Yes`.

Each time you invoke MySQL Installer, the initialization process looks for the presence of an internet connection and prompts you to enable offline mode if it finds no internet access (and offline mode is disabled). Select `Yes` to run MySQL Installer without internet-connection capabilities. MySQL product availability is limited to only those products currently in the product cache when you enable offline mode. To download MySQL products, click the offline mode **Disable** quick action shown on the dashboard.

An internet connection is required to download a manifest containing metadata for the latest MySQL products that are not part of a full bundle. MySQL Installer attempts to download the manifest when you start the application for the first time and then periodically in configurable intervals (see MySQL Installer options). Alternatively, you can retrieve an updated manifest manually by clicking **Catalog** in the MySQL Installer dashboard.

**Note**

If the first-time or subsequent manifest download is unsuccessful, an error is logged and you may have limited access to MySQL products during your session. MySQL Installer attempts to download the manifest with each startup until the initial manifest structure is updated. For help finding a product, see Locating Products to Install.

### MySQL Installer Community Release

Download software from https://dev.mysql.com/downloads/installer/ to install the Community release of all MySQL products for Windows. Select one of the following MySQL Installer package options:

- *Web*: Contains MySQL Installer and configuration files only. The web package option downloads only the MySQL products you select to install, but it requires an internet connection for each download. The size of this file is approximately 2 MB. The file name has the form `mysql-installer-community-web-VERSION.N.msi` in which *`VERSION`* is the MySQL server version number such as 8.0 and `N` is the package number, which begins at 0.
- *Full or Current Bundle*: Bundles all of the MySQL products for Windows (including the MySQL server). The file size is over 300 MB, and the name has the form `mysql-installer-community-VERSION.N.msi` in which *`VERSION`* is the MySQL Server version number such as 8.0 and `N` is the package number, which begins at 0.

### MySQL Installer Commercial Release

Download software from https://edelivery.oracle.com/ to install the Commercial release (Standard or Enterprise Edition) of MySQL products for Windows. If you are logged in to your My Oracle Support (MOS) account, the Commercial release includes all of the current and previous GA versions available in the Community release, but it excludes development-milestone versions. When you are not logged in, you see only the list of bundled products that you downloaded already.

The Commercial release also includes the following products:

- Workbench SE/EE
- MySQL Enterprise Backup
- MySQL Enterprise Firewall

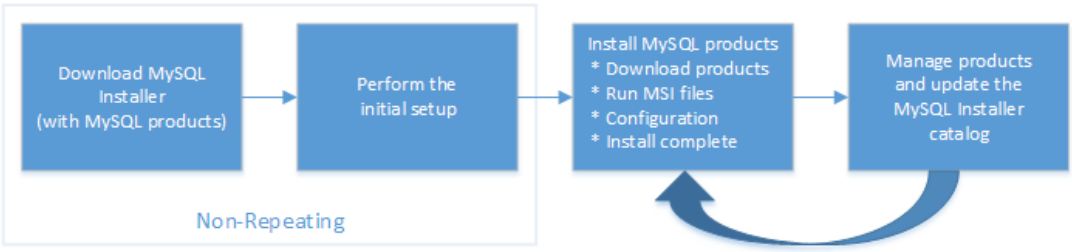
The Commercial release integrates with your MOS account. For knowledge-base content and patches, see My Oracle Support.

### 2.3.3.1 MySQL Installer Initial Setup

- Choosing a Setup Type
- Path Conflicts
- Check Requirements
- MySQL Installer Configuration Files

When you download MySQL Installer for the first time, a setup wizard guides you through the initial installation of MySQL products. As the following figure shows, the initial setup is a one-time activity in the overall process. MySQL Installer detects existing MySQL products installed on the host during its initial setup and adds them to the list of products to be managed.

**Figure 2.7 MySQL Installer Process Overview**




MySQL Installer extracts configuration files (described later) to the hard drive of the host during the initial setup. Although MySQL Installer is a 32-bit application, it can install both 32-bit and 64-bit binaries.

The initial setup adds a link to the Start menu under the **MySQL** folder group. Click **Start**, **MySQL**, and **MySQL Installer - [Community | Commercial]** to open the community or commercial release of the graphical tool.

## Choosing a Setup Type

During the initial setup, you are prompted to select the MySQL products to be installed on the host. One alternative is to use a predetermined setup type that matches your setup requirements. By default, both GA and pre-release products are included in the download and installation with the **Developer Default**, **Client only**, and **Full** setup types. Select the **Only install GA products** option to restrict the product set to include GA products only when using these setup types.

**Note**

Commercial-only MySQL products, such as MySQL Enterprise Backup, are available to select and install if you are using the Commercial version of MySQL Installer (see MySQL Installer Commercial Release).

Choosing one of the following setup types determines the initial installation only and does not limit your ability to install or update MySQL products for Windows later:

- **Developer Default**: Install the following products that compliment application development with MySQL:
  - MySQL Server (Installs the version that you selected when you downloaded MySQL Installer.)
  - MySQL Shell
  - MySQL Router
  - MySQL Workbench
  - MySQL for Visual Studio
  - MySQL Connectors (for .NET, Python, ODBC, Java, and C++)
  - MySQL Documentation
  - MySQL Samples and Examples
- **Server only**: Only install the MySQL server. This setup type installs the general availability (GA) or development release server that you selected when you downloaded MySQL Installer. It uses the default installation and data paths.
- **Client only**: Only install the most recent MySQL applications and MySQL connectors. This setup type is similar to the `Developer Default` type, except that it does not include MySQL server or the client programs typically bundled with the server, such as `mysql` or `mysqladmin`.

- **Full**: Install all available MySQL products.
- **Custom**: The custom setup type enables you to filter and select individual MySQL products from the MySQL Installer catalog.

**Note**

For MySQL Server versions 8.0.20 (and earlier) and MySQL 5.7 (all versions), the account you use to run MySQL Installer may not have adequate permission to install the server data files and this can interrupt the installation because the `ExecSecureObjects` MSI action cannot be executed. To proceed, deselect the **Server data files** feature before attempting to install the server again. For help, see Product Features To Install).

The **Server data files** check box was removed from the feature tree for MySQL Server 8.0.21 (and higher).

Use the `Custom` setup type to install:

- A product or product version that is not available from the usual download locations. The catalog contains all product releases, including the other releases between pre-release (or development) and GA.
- An instance of MySQL server using an alternative installation path, data path, or both. For instructions on how to adjust the paths, see Section 2.3.3.2, “Setting Alternative Server Paths with MySQL Installer”.
- Two or more MySQL server versions on the same host at the same time (for example, 5.7 and 8.0).
- A specific combination of products and features not offered as a predetermine setup type. For example, you can install a single product, such as MySQL Workbench, instead of installing all client applications for Windows.

### Path Conflicts

When the default installation or data folder (required by MySQL server) for a product to be installed already exists on the host, the wizard displays the **Path Conflict** step to identify each conflict and enable you to take action to avoid having files in the existing folder overwritten by the new installation. You see this step in the initial setup only when MySQL Installer detects a conflict.

To resolve the path conflict, do one of the following:

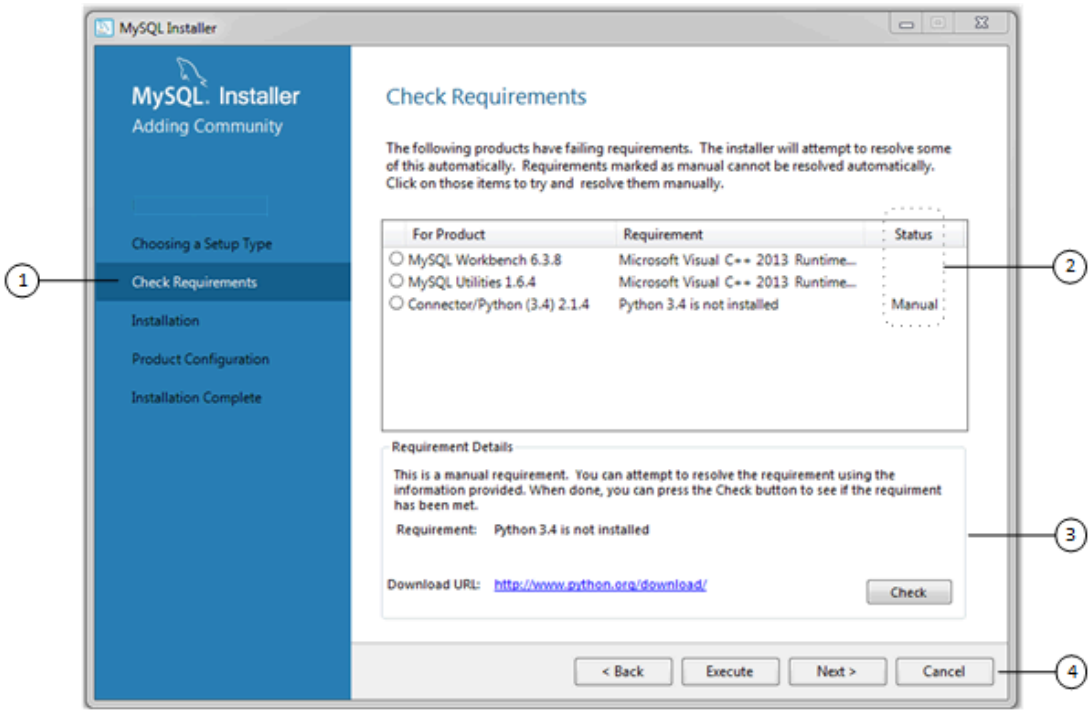
- Select a product from the list to display the conflict options. A warning symbol indicates which path is in conflict. Use the browse button to choose a new path and then click **Next**.
- Click **Back** to choose a different setup type or product version, if applicable. The `Custom` setup type enables you to select individual product versions.
- Click **Next** to ignore the conflict and overwrite files in the existing folder.
- Delete the existing product. Click **Cancel** to stop the initial setup and close MySQL Installer. Open MySQL Installer again from the Start menu and delete the installed product from the host using the Delete operation from the MySQL Installer dashboard.

### Check Requirements

MySQL Installer uses entries in the `package-rules.xml` file to determine whether the prerequisite software for each product is installed on the host. When the requirements check fails, MySQL Installer displays the **Check Requirements** step to help you update the host. Requirements are evaluated

each time you download a new product (or version) for installation. The following figure identifies and describes the key areas of this step.

**Figure 2.8 Check Requirements**




**Description of Check Requirements Elements**

1. Shows the current step in the initial setup. Steps in this list may change slightly depending on the products already installed on the host, the availability of prerequisite software, and the products to be installed on the host.

2. Lists all pending installation requirements by product and indicates the status as follows:

   - A blank space in the **Status** column means that MySQL Installer can attempt to download and install the required software for you.

   - The word *Manual* in the **Status** column means that you must satisfy the requirement manually. Select each product in the list to see its requirement details.

3. Describes the requirement in detail to assist you with each manual resolution. When possible, a download URL is provided. After you download and install the required software, click **Check** to verify that the requirement has been met.

4. Provides the following set operations to proceed:

   - **Back** – Return to the previous step. This action enables you to select a different the setup type.

   - **Execute** – Have MySQL Installer attempt to download and install the required software for all items without a manual status. Manual requirements are resolved by you and verified by clicking **Check**.

   - **Next** – Do not execute the request to apply the requirements automatically and proceed to the installation without including the products that fail the check requirements step.

   - **Cancel** – Stop the installation of MySQL products. Because MySQL Installer is already installed, the initial setup begins again when you open MySQL Installer from the Start menu and click **Add**

from the dashboard. For a description of the available management operations, see Product Catalog.

### MySQL Installer Configuration Files

All MySQL Installer files are located within the `C:\Program Files (x86)` and `C:\ProgramData` folders. The following table describes the files and folders that define MySQL Installer as a standalone application.

**Note**

Installed MySQL products are neither altered nor removed when you update or uninstall MySQL Installer.

**Table 2.5 MySQL Installer Configuration Files**

| File or Folder | Description | Folder Hierarchy |
|---|---|---|
| `MySQL Installer for Windows` | This folder contains all of the files needed to run MySQL Installer and `MySQLInstallerConsole.exe`, a command-line program with similar functionality. | `C:\Program Files (x86)` |
| `Templates` | The `Templates` folder has one file for each version of MySQL server. Template files contain keys and formulas to calculate some values dynamically. | `C:\ProgramData\MySQL \MySQL Installer for Windows\Manifest` |
| `package-rules.xml` | This file contains the prerequisites for every product to be installed. | `C:\ProgramData\MySQL \MySQL Installer for Windows\Manifest` |
| `produts.xml` | The `products` file (or product catalog) contains a list of all products available for download. | `C:\ProgramData\MySQL \MySQL Installer for Windows\Manifest` |
| `Product Cache` | The `Product Cache` folder contains all standalone `.msi` files bundled with the full package or downloaded afterward. | `C:\ProgramData\MySQL \MySQL Installer for Windows` |

## 2.3.3.2 Setting Alternative Server Paths with MySQL Installer

You can change the default installation path, the data path, or both when you install MySQL server. After you have installed the server, the paths cannot be altered without removing and reinstalling the server instance.

**Note**

Starting with MySQL Installer 1.4.39, if you move the data directory of an installed server manually, MySQL Installer identifies the change and can process a reconfiguration operation without errors.

**To change paths for MySQL server**

1. Identify the MySQL server to change and enable the **Advanced Options** link as follows:

   a. Navigate to the **Select Products** page by doing one of the following:

i. If this is an initial setup of MySQL Installer, select the `Custom` setup type and click **Next**.

ii. If MySQL Installer is installed on your computer, click **Add** from the dashboard.

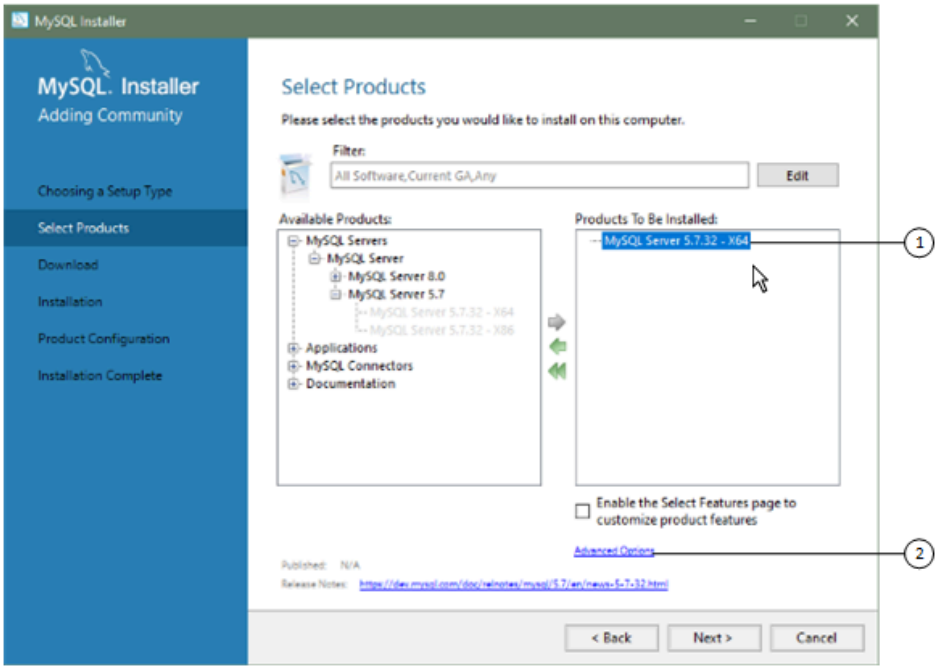
b. Click **Edit** to apply a filter on the product list shown in **Available Products** (see Locating Products to Install).

c. With the server instance selected, use the arrow to move the selected server to the **Products To Be Installed** list.

d. Click the server to select it. When you select the server, the **Advanced Options** link is enabled below the list of products to be installed (see the following figure).

2. Click **Advanced Options** to open a dialog box where you can enter alternative path names. After the path names are validated, click **Next** to continue with the configuration steps.

**Figure 2.9 Change MySQL Server Path**




## 2.3.3.3 Installation Workflows with MySQL Installer

MySQL Installer provides a wizard-like tool to install and configure new MySQL products for Windows. Unlike the initial setup, which runs only once, MySQL Installer invokes the wizard each time you download or install a new product. For first-time installations, the steps of the initial setup proceed directly into the steps of the installation. For assistance with product selection, see Locating Products to Install.

**Note**

Full permissions are granted to the user executing MySQL Installer to all generated files, such as `my.ini`. This does not apply to files and directories for specific products, such as the MySQL server data directory in `%ProgramData%` that is owned by `SYSTEM`.

Products installed and configured on a host follow a general pattern that might require your input during the various steps. If you attempt to install a product that is incompatible with the existing MySQL server version (or a version selected for upgrade), you are alerted about the possible mismatch.

MySQL Installer provides the following sequence of actions that apply to different workflows:

- **Select Products.** If you selected the `Custom` setup type during the initial setup or clicked **Add** from the MySQL Installer dashboard, MySQL Installer includes this action in the sidebar. From this page, you can apply a filter to modify the Available Products list and then select one or more products to move (using arrow keys) to the Products To Be Installed list.

  Select the check box on this page to activate the Select Features action where you can customize the products features after the product is downloaded.

- **Download.** If you installed the full (not web) MySQL Installer package, all `.msi` files were loaded to the `Product Cache` folder during the initial setup and are not downloaded again. Otherwise, click **Execute** to begin the download. The status of each product changes from `Ready to Download`, to `Downloading`, and then to `Downloaded`.

  To retry a single unsuccessful download, click the **Try Again** link.

  To retry all unsuccessful downloads, click **Try All**.

- **Select Features To Install (disabled by default).** After MySQL Installer downloads a product's `.msi` file, you can customize the features if you enabled the optional check box previously during the Select Products action.

  To customize product features after the installation, click **Modify** in the MySQL Installer dashboard.

- **Installation.** The status of each product in the list changes from `Ready to Install`, to `Installing`, and lastly to `Complete`. During the process, click **Show Details** to view the installation actions.

  If you cancel the installation at this point, the products are installed, but the server (if installed) is not yet configured. To restart the server configuration, open MySQL Installer from the Start menu and click **Reconfigure** next to the appropriate server in the dashboard.

- **Product configuration.** This step applies to MySQL Server, MySQL Router, and samples only. The status for each item in the list should indicate `Ready to Configure`. Click **Next** to start the configuration wizard for all items in the list. The configuration options presented during this step are specific to the version of database or router that you selected to install.

  Click **Execute** to begin applying the configuration options or click **Back** (repeatedly) to return to each configuration page.

- **Installation complete.** This step finalizes the installation for products that do not require configuration. It enables you to copy the log to a clipboard and to start certain applications, such as MySQL Workbench and MySQL Shell. Click **Finish** to open the MySQL Installer dashboard.

### MySQL Server Configuration with MySQL Installer

MySQL Installer performs the initial configuration of the MySQL server. For example:

- It creates the configuration file (`my.ini`) that is used to configure the MySQL server. The values written to this file are influenced by choices you make during the installation process. Some definitions are host dependent. For example, query_cache is enabled if the host has fewer than three cores.

  **Note**

  Query cache was deprecated in MySQL 5.7 and removed in MySQL 8.0 (and later).

- By default, a Windows service for the MySQL server is added.

- Provides default installation and data paths for MySQL server. For instructions on how to change the default paths, see Section 2.3.3.2, “Setting Alternative Server Paths with MySQL Installer”.

- It can optionally create MySQL server user accounts with configurable permissions based on general roles, such as DB Administrator, DB Designer, and Backup Admin. It optionally creates a Windows user named `MysqlSys` with limited privileges, which would then run the MySQL Server.

  User accounts may also be added and configured in MySQL Workbench.

- Checking **Show Advanced Options** enables additional **Logging Options** to be set. This includes defining custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log.

During the configuration process, click **Next** to proceed to the next step or **Back** to return to the previous step. Click **Execute** at the final step to apply the server configuration.

The sections that follow describe the server configuration options that apply to MySQL server on Windows. The server version you installed will determine which steps and options you can configure. Configuring MySQL server may include some or all of the steps.

**Type and Networking**

- Server Configuration Type

  Choose the MySQL server configuration type that describes your setup. This setting defines the amount of system resources (memory) to assign to your MySQL server instance.

  - **Development**: A computer that hosts many other applications, and typically this is your personal workstation. This setting configures MySQL to use the least amount of memory.
  - **Server**: Several other applications are expected to run on this computer, such as a web server. The Server setting configures MySQL to use a medium amount of memory.
  - **Dedicated**: A computer that is dedicated to running the MySQL server. Because no other major applications run on this server, this setting configures MySQL to use the majority of available memory.
  - **Manual**

    Prevents MySQL Installer from attempting to optimize the server installation, and instead, sets the default values to the server variables included in the `my.ini` configuration file. With the `Manual` type selected, MySQL Installer uses the default value of 16M for the `tmp_table_size` variable assignment.

- Connectivity

  Connectivity options control how the connection to MySQL is made. Options include:

  - **TCP/IP**: This option is selected by default. You may disable TCP/IP Networking to permit local host connections only. With the TCP/IP connection option selected, you can modify the following items:
    - **Port** for classic MySQL protocol connections. The default value is `3306`.
    - **X Protocol Port** shown when configuring MySQL 8.0 server only. The default value is `33060`
    - **Open Windows Firewall port for network access**, which is selected by default for TCP/IP connections.

    If a port number is in use already, you will see the information icon (  ) next to the default value and **Next** is disabled until you provide a new port number.

  - **Named Pipe**: Enable and define the pipe name, similar to setting the `named_pipe` system variable. The default name is `MySQL`.

When you select **Named Pipe** connectivity, and then proceed to the next step, you are prompted to set the level of access control granted to client software on named-pipe connections. Some clients require only minimum access control for communication, while other clients require full access to the named pipe.

You can set the level of access control based on the Windows user (or users) running the client as follows:

- **Minimum access to all users (RECOMMENDED).** This level is enabled by default because it is the most secure.
- **Full access to members of a local group.** If the minimum-access option is too restrictive for the client software, use this option to reduce the number of users who have full access on the named pipe. The group must be established on Windows before you can select it from the list. Membership in this group should be limited and managed. Windows requires a newly added member to first log out and then log in again to join a local group.
- **Full access to all users (NOT RECOMMENDED).** This option is less secure and should be set only when other safeguards are implemented.

- **Shared Memory**: Enable and define the memory name, similar to setting the `shared_memory` system variable. The default name is `MySQL`.

- Advanced Configuration

  Check **Show Advanced and Logging Options** to set custom logging and advanced options in later steps. The Logging Options step enables you to define custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log. The Advanced Options step enables you to set the unique server ID required when binary logging is enabled in a replication topology.

- MySQL Enterprise Firewall (Enterprise Edition only)

  The **Enable MySQL Enterprise Firewall** check box is deselected by default. Select this option to enable a security list that offers protection against certain types of attacks. Additional post-installation configuration is required (see Section 6.4.7, “MySQL Enterprise Firewall”).

**Important**

There is an issue for MySQL 8.0.19 that prevents the server from starting if MySQL Enterprise Firewall is selected during the server configuration steps. If the server startup operation fails, click **Cancel** to end the configuration process and return to the dashboard. You must uninstall the server.

The workaround is to run MySQL Installer without MySQL Enterprise Firewall selected. (That is, do not select the **Enable MySQL Enterprise Firewall** check box.) Then install MySQL Enterprise Firewall afterward using the instructions for manual installation (see Section 6.4.7.2, “Installing or Uninstalling MySQL Enterprise Firewall”).

### Authentication Method

The **Authentication Method** step is visible only during the installation or upgrade of MySQL 8.0.4 or higher. It introduces a choice between two server-side authentication options. The MySQL user accounts that you create in the next step will use the authentication method that you select in this step.

MySQL 8.0 connectors and community drivers that use `libmysqlclient` 8.0 now support the `mysql_native_password` default authentication plugin. However, if you are unable to update your clients and applications to support this new authentication method, you can configure the MySQL

server to use `mysql_native_password` for legacy authentication. For more information about the implications of this change, see caching_sha2_password as the Preferred Authentication Plugin.

If you are installing or upgrading to MySQL 8.0.4 or higher, select one of the following authentication methods:

- Use Strong Password Encryption for Authentication (RECOMMENDED)

  MySQL 8.0 supports a new authentication based on improved, stronger SHA256-based password methods. It is recommended that all new MySQL server installations use this method going forward.

  **Important**

  The `caching_sha2_password` authentication plugin on the server requires new versions of connectors and clients, which add support for the new MySQL 8.0 default authentication.

- Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

  Using the old MySQL 5.x legacy authentication method should be considered only in the following cases:

  - Applications cannot be updated to use MySQL 8.0 connectors and drivers.
  - Recompilation of an existing application is not feasible.
  - An updated, language-specific connector or driver is not available yet.

**Accounts and Roles**

- Root Account Password

  Assigning a root password is required and you will be asked for it when performing other MySQL Installer operations. Password strength is evaluated when you repeat the password in the box provided. For descriptive information regarding password requirements or status, move your mouse pointer over the information icon (  ) when it appears.

- MySQL User Accounts (Optional)

  Click **Add User** or **Edit User** to create or modify MySQL user accounts with predefined roles. Next, enter the required account credentials:

  - **User Name:** MySQL user names can be up to 32 characters long.
  - **Host:** Select `localhost` for local connections only or `<All Hosts (%)>` when remote connections to the server are required.
  - **Role:** Each predefined role, such as `DB Admin`, is configured with its own set of privileges. For example, the `DB Admin` role has more privileges than the `DB Designer` role. The **Role** drop-down list contains a description of each role.
  - **Password:** Password strength assessment is performed while you type the password. Passwords must be confirmed. MySQL permits a blank or empty password (considered to be insecure).

  **MySQL Installer Commercial Release Only:** MySQL Enterprise Edition for Windows, a commercial product, also supports an authentication method that performs external authentication on Windows. Accounts authenticated by the Windows operating system can access the MySQL server without providing an additional password.

  To create a new MySQL account that uses Windows authentication, enter the user name and then select a value for **Host** and **Role**. Click **Windows** authentication to enable the

`authentication_windows` plugin. In the Windows Security Tokens area, enter a token for each Windows user (or group) who can authenticate with the MySQL user name. MySQL accounts can include security tokens for both local Windows users and Windows users that belong to a domain. Multiple security tokens are separated by the semicolon character (`;`) and use the following format for local and domain accounts:

- Local account

  Enter the simple Windows user name as the security token for each local user or group; for example, **`finley;jeffrey;admin`**.

- Domain account

  Use standard Windows syntax (*`domain\domainuser`*) or MySQL syntax (*`domain\\domainuser`*) to enter Windows domain users and groups.

  For domain accounts, you may need to use the credentials of an administrator within the domain if the account running MySQL Installer lacks the permissions to query the Active Directory. If this is the case, select **Validate Active Directory users with** to activate the domain administrator credentials.

Windows authentication permits you to test all of the security tokens each time you add or modify a token. Click **Test Security Tokens** to validate (or revalidate) each token. Invalid tokens generate a descriptive error message along with a red `X` icon and red token text. When all tokens resolve as valid (green text without an `X` icon), you can click **OK** to save the changes.

**Windows Service**

On the Windows platform, MySQL server can run as a named service managed by the operating system and be configured to start up automatically when Windows starts. Alternatively, you can configure MySQL server to run as an executable program that requires manual configuration.

- **Configure MySQL server as a Windows service** (Selected by default.)

  When the default configuration option is selected, you can also select the following:

  - **Start the MySQL Server at System Startup**

    When selected (default), the service startup type is set to Automatic; otherwise, the startup type is set to Manual.

  - **Run Windows Service as**

    When **Standard System Account** is selected (default), the service logs on as Network Service.

    The **Custom User** option must have privileges to log on to Microsoft Windows as a service. The **Next** button will be disabled until this user is configured with the required privileges.

    A custom user account is configured in Windows by searching for "local security policy" in the Start menu. In the Local Security Policy window, select **Local Policies**, **User Rights Assignment**, and then **Log On As A Service** to open the property dialog. Click **Add User or Group** to add the custom user and then click **OK** in each dialog to save the changes.

- Deselect the Windows Service option.

**Server File Permissions**

Optionally, permissions set on the folders and files located at `C:\ProgramData\MySQL\MySQL Server 8.0\Data` can be managed during the server configuration operation. You have the following options:

- MySQL Installer can configure the folders and files with full control granted exclusively to the user running the Windows service, if applicable, and to the Administrators group.

  All other groups and users are denied access. This is the default option.

- Have MySQL Installer use a configuration option similar to the one just described, but also have MySQL Installer show which users could have full control.

  You are then able to decide if a group or user should be given full control. If not, you can move the qualified members from this list to a second list that restricts all access.

- Have MySQL Installer skip making file-permission changes during the configuration operation.

  If you select this option, you are responsible for securing the `Data` folder and its related files manually after the server configuration finishes.

### Logging Options

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

Advanced configuration options are related to the following MySQL log files:

- Error Log
- General Log
- Slow Query Log
- Bin Log

**Note**

The binary log is enabled by default for MySQL 5.7 and higher.

### Advanced Options

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

The advanced-configuration options include:

- **Server ID**

  Set the unique identifier used in a replication topology. If binary logging is enabled, you must specify a server ID. The default ID value depends on the server version. For more information, see the description of the `server_id` system variable.

- **Table Names Case**

  You can set the following options during the initial and subsequent configuration the server. For the MySQL 8.0 release series, these options apply only to the initial configuration of the server.

  - Lower Case

    Sets the `lower_case_table_names` option value to 1 (default), in which table names are stored in lowercase on disk and comparisons are not case-sensitive.

  - Preserve Given Case

    Sets the `lower_case_table_names` option value to 2, in which table names are stored as given but compared in lowercase.

#### Apply Server Configuration

All configuration settings are applied to the MySQL server when you click **Execute**. Use the **Configuration Steps** tab to follow the progress of each action; the icon for each toggles from white to green (with a check mark) on success. Otherwise, the process stops and displays an error message if an individual action times out. Click the **Log** tab to view the log.

When the installation completes successfully and you click **Finish**, MySQL Installer and the installed MySQL products are added to the Microsoft Windows Start menu under the `MySQL` group. Opening MySQL Installer loads the dashboard where installed MySQL products are listed and other MySQL Installer operations are available.

### MySQL Router Configuration with MySQL Installer

MySQL Installer downloads and installs a suite of tools for developing and managing business-critical applications on Windows. The suite consists of applications, connectors, documentation, and samples.

During the initial setup, choose any predetermined setup type, except `Server only`, to install the latest GA version of the tools. Use the `Custom` setup type to install an individual tool or specific version. If MySQL Installer is installed on the host already, use the **Add** operation to select and install tools from the MySQL Installer dashboard.

#### MySQL Router Configuration

MySQL Installer provides a configuration wizard that can bootstrap an installed instance of MySQL Router 8.0 to direct traffic between MySQL applications and an InnoDB Cluster. When configured, MySQL Router runs as a local Windows service.

**Note**

You are prompted to configure MySQL Router after the initial installation and when you reconfigure an installed router explicitly. In contrast, the upgrade operation does not require or prompt you to configure the upgraded product.

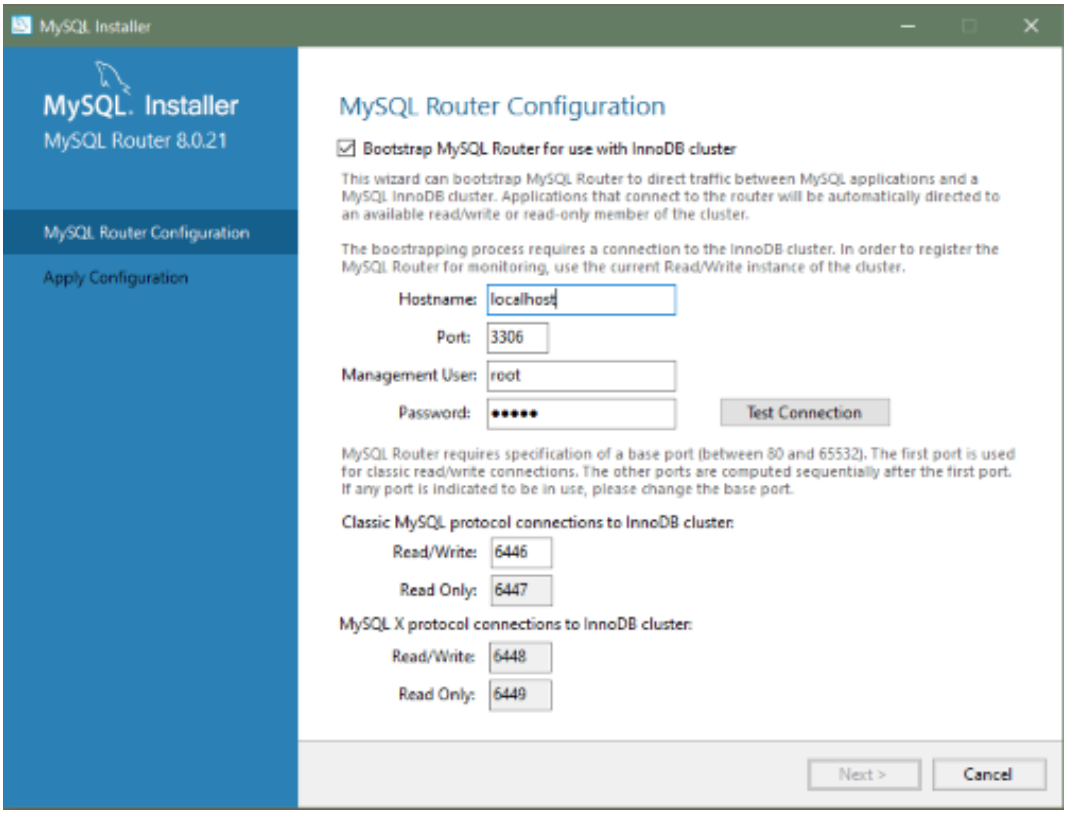
To configure MySQL Router, do the following:

1. Set up InnoDB Cluster.
2. Using MySQL Installer, download and install the MySQL Router application. After the installation finishes, the configuration wizard prompts you for information. Select the **Configure MySQL Router for InnoDB Cluster** check box to begin the configuration and provide the following configuration values:
   - **Hostname:** Host name of the primary (seed) server in the InnoDB Cluster (`localhost` by default).
   - **Port:** The port number of the primary (seed) server in the InnoDB Cluster (`3306` by default).
   - **Management User:** An administrative user with root-level privileges.
   - **Password:** The password for the management user.
   - **Classic MySQL protocol connections to InnoDB Cluster**

     **Read/Write:** Set the first base port number to one that is unused (between 80 and 65532) and the wizard will select the remaining ports for you.

     The figure that follows shows an example of the MySQL Router configuration page, with the first base port number specified as 6446 and the remaining ports set by the wizard to 6447, 6448, and 6449.

**Figure 2.10 MySQL Router Configuration**




3. Click **Next** and then **Execute** to apply the configuration. Click **Finish** to close MySQL Installer or return to the MySQL Installer dashboard.

After configuring MySQL Router, the root account exists in the user table as `root@localhost` (local) only, instead of `root@%` (remote). Regardless of where the router and client are located, even if both are located on the same host as the seed server, any connection that passes through the router is viewed by server as being remote, not local. As a result, a connection made to the server using the local host (see the example that follows), does not authenticate.

```
$> \c root@localhost:6446
```

### 2.3.3.4 MySQL Installer Product Catalog and Dashboard

This section describes the MySQL Installer product catalog, the dashboard, and other actions related to product selection and upgrades.

- Product Catalog
- MySQL Installer Dashboard
- Locating Products to Install
- Upgrading MySQL Server
- Removing MySQL Server
- Upgrading MySQL Installer

#### Product Catalog

The product catalog stores the complete list of released MySQL products for Microsoft Windows that are available to download from MySQL Downloads. By default, and when an Internet connection is

present, MySQL Installer attempts to update the catalog at startup every seven days. You can also update the catalog manually from the dashboard (described later).

An up-to-date catalog performs the following actions:

- Populates the **Available Products** pane of the Select Products page. This step appears when you select:
  - The `Custom` setup type during the initial setup.
  - The **Add** operation from the dashboard.
- Identifies when product updates are available for the installed products listed in the dashboard.

The catalog includes all development releases (Pre-Release), general releases (Current GA), and minor releases (Other Releases). Products in the catalog will vary somewhat, depending on the MySQL Installer release that you download.

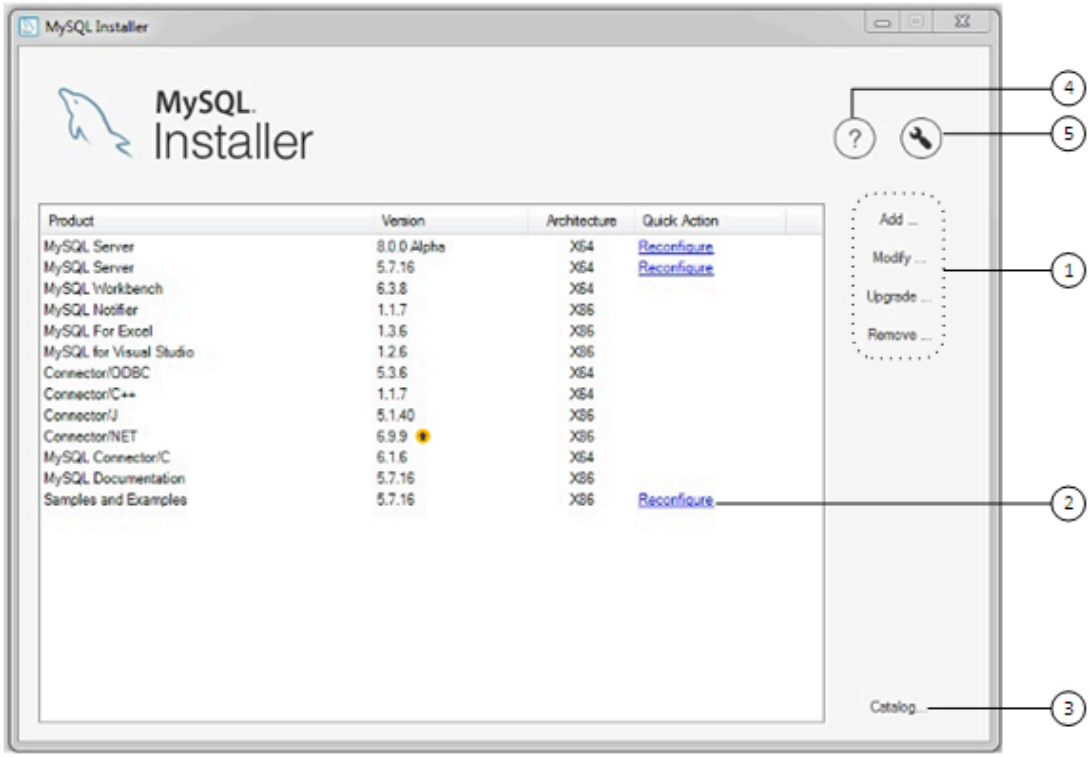
## MySQL Installer Dashboard

The MySQL Installer dashboard is the default view that you see when you start MySQL Installer after the initial setup finishes. If you closed MySQL Installer before the setup was finished, MySQL Installer resumes the initial setup before it displays the dashboard.

**Note**

Products covered under Oracle Lifetime Sustaining Support, if installed, may appear in the dashboard. These products, such as MySQL for Excel and MySQL Notifier, can be modified or removed only.

**Figure 2.11 MySQL Installer Dashboard Elements**




### Description of MySQL Installer Dashboard Elements

1. MySQL Installer dashboard operations provide a variety of actions that apply to installed products or products listed in the catalog. To initiate the following operations, first click the operation link and then select the product or products to manage:

- **Add**: This operation opens the Select Products page. From there you can adjust the filter, select one or more products to download (as needed), and begin the installation. For hints about using the filter, see Locating Products to Install.

  Use the directional arrows to move each product from the **Available Products** column to the **Products To Be Installed** column. To enable the Product Features page where you can customize features, click the related check box (disabled by default).

  **Note**

  For MySQL Server versions 8.0.20 (and earlier) and MySQL 5.7 (all versions), the account you use to run MySQL Installer may not have adequate permission to install the server data files and this can interrupt the installation because the `ExecSecureObjects` MSI action cannot be executed. To proceed, deselect the **Server data files** feature before attempting to install the server again.

  The **Server data files** check box was removed from the feature tree for MySQL Server 8.0.21 (or higher).

- **Modify**: Use this operation to add or remove the features associated with installed products. Features that you can modify vary in complexity by product. When the **Program Shortcut** check box is selected, the product appears in the Start menu under the `MySQL` group.

- **Upgrade**: This operation loads the Select Products to Upgrade page and populates it with all the upgrade candidates. An installed product can have more than one upgrade version and the operation requires a current product catalog. MySQL Installer upgrades all of the selected products in one action. Click **Show Details** to view the actions performed by MySQL Installer.

- **Remove**: This operation opens the Remove Products page and populates it with the MySQL products installed on the host. Select the MySQL products you want to remove (uninstall) and then click **Execute** to begin the removal process. During the operation, an indicator shows the number of steps that are executed as a percentage of all steps.

  To select products to remove, do one of the following:

  - Select the check box for one or more products.
  - Select the **Product** check box to select all products.

2. The **Reconfigure** link in the Quick Action column next to each installed server loads the current configuration values for the server and then cycles through all configuration steps enabling you to change the options and values. You must provide credentials with root privileges to reconfigure these items. Click the **Log** tab to show the output of each configuration step performed by MySQL Installer.

   On completion, MySQL Installer stops the server, applies the configuration changes, and restarts the server for you. For a description of each configuration option, see MySQL Server Configuration with MySQL Installer. Installed `Samples and Examples` associated with a specific MySQL server version can be also be reconfigured to apply new feature settings, if any.

3. The **Catalog** link enables you to download the latest catalog of MySQL products manually and then to integrate those product changes with MySQL Installer. The catalog-download action does not perform an upgrade of the products already installed on the host. Instead, it returns to the

dashboard and adds an arrow icon to the Version column for each installed product that has a newer version. Use the **Upgrade** operation to install the newer product version.

You can also use the **Catalog** link to display the current change history of each product without downloading the new catalog. Select the **Do not update at this time** check box to view the change history only.

4. The MySQL Installer About icon (  ) shows the current version of MySQL Installer and general information about MySQL. The version number is located above the **Back** button.

   **Tip**

   Always include this version number when reporting a problem with MySQL Installer.

   In addition to the About MySQL information (  ), you can also select the following icons from the side panel:

   - License icon (  ) for MySQL Installer.

     This product may include third-party software, used under license. If you are using a Commercial release of MySQL Installer, the icon opens the MySQL Installer 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 Installer, the icon opens the MySQL Installer Community License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

   - Resource links icon (  ) to the latest MySQL product documentation, blogs, webinars, and more.

5. The MySQL Installer Options icon (  ) includes the following tabs:

   - **General**: Enables or disables the Offline mode option. If selected, this option configures MySQL Installer to run without depending on internet-connection capabilities. When running MySQL Installer in offline mode, you see a warning together with a **Disable** quick action on the dashboard. The warning serves to remind you that running MySQL Installer in offline mode prevents you from downloading the latest MySQL products and product catalog updates. Offline mode persists until you disable the option.

     At startup, MySQL Installer determines whether an internet connection is present, and, if not, prompts you to enable offline mode to resume working without a connection.

   - **Product Catalog**: Manages the automatic catalog updates. By default, MySQL Installer checks for catalog updates at startup every seven days. When new products or product versions are available, MySQL Installer adds them to the catalog and then inserts an arrow icon (  ) next to the version number of installed products listed in the dashboard.

     Use the product catalog option to enable or disable automatic updates and to reset the number of days between automatic catalog downloads. At startup, MySQL Installer uses the number of days you set to determine whether a download should be attempted. This action is repeated during next startup if MySQL Installer encounters an error downloading the catalog.

   - **Connectivity Settings**: Several operations performed by MySQL Installer require internet access. This option enables you to use a default value to validate the connection or to use

a different URL, one selected from a list or added by you manually. With the **Manual** option selected, new URLs can be added and all URLs in the list can be moved or deleted. When the **Automatic** option is selected, MySQL Installer attempts to connect to each default URL in the list (in order) until a connection is made. If no connection can be made, it raises an error.

- **Proxy**: MySQL Installer provides multiple proxy modes that enable you to download MySQL products, updates, or even the product catalog in most network environments. The mode are:

  - **No proxy**

    Select this mode to prevent MySQL Installer from looking for system settings. This mode disables any proxy settings.

  - **Automatic**

    Select this mode to have MySQL Installer look for system settings and to use those settings if found, or to use no proxy if nothing is found. This mode is the default.

  - **Manual**

    Select this mode to have MySQL Installer use your authentication details to configuration proxy access to the internet. Specifically:

    - A proxy-server address (`http://address-to-server`) and port number
    - A user name and password for authentication

## Locating Products to Install

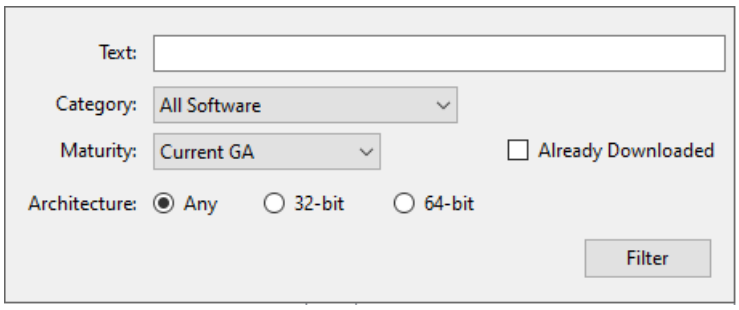
MySQL products in the catalog are listed by category: MySQL Servers, Applications, MySQL Connectors, and Documentation. Only the latest GA versions appear in the **Available Products** pane by default. If you are looking for a pre-release or older version of a product, it may not be visible in the default list.

**Note**

Keep the product catalog up-to-date. Click **Catalog** on the MySQL Installer dashboard to download the latest manifest.

To change the default product list, click **Add** in the dashboard to open the Select Products page, and then click **Edit** to open the dialog box shown in the figure that follows. Modify the settings and then click **Filter**.

**Figure 2.12 Filter Available Products**




Reset one or more of the following fields to modify the list of available products:

- Text: Filter by text.
- Category: All Software (default), MySQL Servers, Applications, MySQL Connectors, or Documentation (for samples and documentation).

- Maturity: Current Bundle (appears initially with the full package only), Pre-Release, Current GA, or Other Releases. If you see a warning, confirm that you have the most recent product manifest by clicking **Catalog** on the MySQL Installer dashboard. If MySQL Installer is unable to download the manifest, the range of products you see is limited to bundled products, standalone product MSIs located in the `Product Cache` folder already, or both.

  **Note**

  The Commercial release of MySQL Installer does not display any MySQL products when you select the Pre-Release maturity filter. Products in development are available from the Community release of MySQL Installer only.

- Already Downloaded (the check box is deselected by default). Permits you to view and manage downloaded products only.
- Architecture: Any (default), 32-bit, or 64-bit.

### Upgrading MySQL Server

Important server upgrade conditions:

- MySQL Installer does not permit server upgrades between major release versions or minor release versions, but does permit upgrades within a release series, such as an upgrade from 5.7.18 to 5.7.19.
- Upgrades between milestone releases (or from a milestone release to a GA release) are not supported. Significant development changes take place in milestone releases and you may encounter compatibility issues or problems starting the server.
- For upgrades to MySQL 8.0.16 server and higher, a check box enables you to skip the upgrade check and process for system tables, while checking and processing data dictionary tables normally. MySQL Installer does not prompt you with the check box when the previous server upgrade was skipped or when the server was configured as a sandbox InnoDB Cluster. This behavior represents a change in how MySQL Server performs an upgrade (see Section 2.10.3, “What the MySQL Upgrade Process Upgrades”) and it alters the sequence of steps that MySQL Installer applies to the configuration process.

  If you select **Skip system tables upgrade check and process. (Not recommended)**, MySQL Installer starts the upgraded server with the `--upgrade=MINIMAL` server option, which upgrades the data dictionary only. If you stop and then restart the server without the `--upgrade=MINIMAL` option, the server upgrades the system tables automatically, if needed.

  The following information appears in the **Log** tab and log file after the upgrade configuration (with system tables skipped) is complete:

```
WARNING: The system tables upgrade was skipped after upgrading MySQL Server. The
server will be started now with the --upgrade=MINIMAL option, but then each
time the server is started it will attempt to upgrade the system tables, unless
you modify the Windows service (command line) to add --upgrade=MINIMAL to bypass
the upgrade.

FOR THE BEST RESULTS: Run mysqld.exe --upgrade=FORCE on the command line to upgrade
the system tables manually.
```

To choose a new server version:

1. Click **Upgrade**. Confirm that the check box next to product name in the **Upgradeable Products** pane has a check mark. Deselect the products that you do not intend to upgrade at this time.

**Note**

For server milestone releases in the same release series, MySQL Installer deselects the server upgrade and displays a warning to indicate that the upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform a logical upgrade manually. You can reselect server upgrade at your own risk. For instructions on how to perform a logical upgrade with a milestone release, see Logical Upgrade.

2. Click a product in the list to highlight it. This action populates the **Upgradeable Versions** pane with the details of each available version for the selected product: version number, published date, and a `Changes` link to open the release notes for that version.

### Removing MySQL Server

To remove a local MySQL server:

1. Determine whether the local data directory should be removed. If you retain the data directory, another server installation can reuse the data. This option is enabled by default (removes the data directory).
2. Click **Execute** to begin uninstalling the local server. Note that all products that you selected to remove are also uninstalled at this time.
3. (Optional) Click the **Log** tab to display the current actions performed by MySQL Installer.

### Upgrading MySQL Installer

MySQL Installer remains installed on your computer, and like other software, MySQL Installer can be upgraded from the previous version. In some cases, other MySQL software may require that you upgrade MySQL Installer for compatibility. This section describes how to identify the current version of MySQL Installer and how to upgrade MySQL Installer manually.

**To locate the installed version of MySQL Installer:**

1. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.
2. Click the MySQL Installer About icon ( ). The version number is located above the **Back** button.

**To initiate an on-demand upgrade of MySQL Installer:**

1. Connect the computer with MySQL Installer installed to the internet.
2. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.
3. Click **Catalog** on the bottom of the dashboard to open the Update Catalog window.
4. Click **Execute** to begin the process. If the installed version of MySQL Installer can be upgraded, you will be prompted to start the upgrade.
5. Click **Next** to review all changes to the catalog and then click **Finish** to return to the dashboard.
6. Verify the (new) installed version of MySQL Installer (see the previous procedure).

## 2.3.3.5 MySQL Installer Console Reference

`MySQLInstallerConsole.exe` provides command-line functionality that is similar to MySQL Installer. This reference includes:

- MySQL Product Names
- Command Syntax

- Command Actions

The console is installed when MySQL Installer is initially executed and then available within the `MySQL Installer for Windows` directory. By default, the directory location is `C:\Program Files (x86)\MySQL\MySQL Installer for Windows`. You must run the console as administrator.

To use the console:

1. Open a command prompt with administrative privileges by selecting **Windows System** from **Start**, then right-click **Command Prompt**, select **More**, and select **Run as administrator**.

2. From the command line, optionally change the directory to where the `MySQLInstallerConsole.exe` command is located. For example, to use the default installation location:

```
cd Program Files (x86)\MySQL\MySQL Installer for Windows
```

3. Type `MySQLInstallerConsole.exe` (or `mysqlinstallerconsole`) followed by a command action to perform a task. For example, to show the console's help:

```
MySQLInstallerConsole.exe --help
```

```
================== Start Initialization ==================
MySQL Installer is running in Community mode

Attempting to update manifest.
Initializing product requirements.
Loading product catalog.
Checking for product packages in the bundle.
Categorizing product catalog.
Finding all installed packages.
Your product catalog was last updated at 23/08/2022 12:41:05 p. m.
Your product catalog has version number 671.
================== End Initialization ==================

The following actions are available:

Configure - Configures one or more of your installed programs.
Help      - Provides list of available command actions.
Install   - Installs and configures one or more available MySQL programs.
List      - Lists all available MySQL products.
Modify    - Modifies the features of installed products.
Remove    - Removes one or more products from your system.
Set       - Configures the general options of MySQL Installer.
Status    - Shows the status of all installed products.
Update    - Updates the current product catalog.
Upgrade   - Upgrades one or more of your installed programs.

The basic syntax for using MySQL Installer command actions. Brackets denote optional entities.
Curly braces denote a list of possible entities.

...
```

## MySQL Product Names

Many of the `MySQLInstallerConsole` command actions accept one or more abbreviated phrases that can match a MySQL product (or products) in the catalog. The current set of valid short phrases for use with commands is shown in the following table.

**Table 2.6 MySQL Product Phrases for use with the MySQLInstallerConsole.exe command**

| Phrase | MySQL Product |
|---|---|
| `server` | MySQL Server |
| `workbench` | MySQL Workbench |
| `shell` | MySQL Shell |
| `visual` | MySQL for Visual Studio |

| Phrase | MySQL Product |
|---|---|
| router | MySQL Router |
| backup | MySQL Enterprise Backup (requires the commercial release) |
| net | MySQL Connector/NET |
| odbc | MySQL Connector/ODBC |
| c++ | MySQL Connector/C++ |
| python | MySQL Connector/Python |
| j | MySQL Connector/J |
| documentation | MySQL Server Documentation |
| samples | MySQL Samples (sakila and world databases) |

### Command Syntax

The MySQLInstallerConsole.exe command can be issued with or without the file extension (.exe) and the command is not case-sensitive.

mysqlinstallerconsole[.exe] [[[--]*action*] [*action_blocks_list*] [*options_list*]]

Description:

*action*
: One of the permitted operational actions. If omitted, the default action is equivalent to the --status action. Using the -- prefix is optional for all actions.

  Possible actions are: [--]configure, [--]help, [--]install, [--]list, [--]modify, [--]remove, [--]set, [--]status, [--]update, and [--]upgrade.

*action_blocks_list*
: A list of blocks in which each represents a different item depending on the selected action. Blocks are separated by commas.

  The --remove and --upgrade actions permit specifying an asterisk character (*) to indicate all products. If the * character is detected at the start of this block, it is assumed all products are to be processed and the remainder of the block is ignored.

  Syntax: *|*action_block*[,*action_block*] [,*action_block*]...

  *action_block*: Contains a product selector followed by an indefinite number of argument blocks that behave differently depending on the selected action (see Command Actions).

*options_list*
: Zero or more options with possible values separated by spaces. See Command Actions to identify the options permitted for the corresponding action.

  Syntax: *option_value_pair*[ *option_value_pair*][ *option_value_pair*]...

  *option_value_pair*: A single option (for example, --silent) or a tuple of a key and a corresponding value with an options prefix. The key-value pair is in the form of --*key*[=*value*].

### Command Actions

MySQLInstallerConsole.exe supports the following command actions:

**Note**

Configuration block (or *arguments_block*) values that contain a colon character (`:`) must be wrapped in quotation marks. For example, `install_dir="C:\MySQL\MySQL Server 8.0"`.

- `[--]configure [product1]:[configuration_argument]=[value], [product2]:[configuration_argument]=[value], [...]`

  Configures one or more MySQL products on your system. Multiple *`configuration_argument`*=*`value`* pairs can be configured for each product.

  Options:

  | | |
  |---|---|
  | `--continue` | Continues processing the next product when an error is caught while processing the action blocks containing arguments for each product. If not specified the whole operation is aborted in case of an error. |
  | `--help` | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |
  | `--show-settings` | Displays the available options for the selected product by passing in the product name after `--show-settings`. |
  | `--silent` | Disables confirmation prompts. |

  Examples:

  ```
  MySQLInstallerConsole --configure --show-settings server
  ```

  ```
  mysqlinstallerconsole.exe --configure server:port=3307
  ```

- `[--]help`

  Displays a help message with usage examples and then exits. Pass in an additional command action to receive help specific to that action.

  Options:

  | | |
  |---|---|
  | `--action=[action]` | Shows the help for a specific action. Same as using the `--help` option with an action.<br><br>Permitted values are: `all`, `configure`, `help` (default), `install`, `list`, `modify`, `remove`, `status`, `update`, `upgrade`, and `set`. |
  | `--help` | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |

  Examples:

  ```
  MySQLInstallerConsole help
  ```

  ```
  MySQLInstallerConsole help --action=install
  ```

- `[--]install [product1]:[features]:[config block]:[config block], [product2]:[config block], [...]`

  Installs one or more MySQL products on your system. If pre-release products are available, both GA and pre-release products are installed when the value of the `--type` option value is `Developer`,

`Client`, or `Full`. Use the `--only_ga_products` option to restrict the product set to GA products only when using these setup types.

Description:

| | |
|---|---|
| `[product]` | Each product can be specified by a product phrase with or without a semicolon-separated version qualifier. Passing in a product keyword alone selects the latest version of the product. If multiple architectures are available for that version of the product, the command returns the first one in the manifest list for interactive confirmation. Alternatively, you can pass in the exact version and architecture (`x86` or `x64`) after the product keyword using the `--silent` option. |
| `[features]` | All features associated with a MySQL product are installed by default. The feature block is a semicolon-separated list of features or an asterisk character (`*`) that selects all features. To remove a feature, use the `modify` command. |
| `[config block]` | One or more configuration blocks can be specified. Each configuration block is a semicolon-separated list of key-value pairs. A block can include either a `config` or `user` type key; `config` is the default type if one is not defined.<br><br>Configuration block values that contain a colon character (`:`) must be wrapped in quotation marks. For example, `installdir="C:\MySQL\MySQL Server 8.0"`. Only one configuration type block can be defined for each product. A user block should be defined for each user to be created during the product installation. |

**Note**

The `user` type key is not supported when a product is being reconfigured.

Options:

| | |
|---|---|
| `--auto-handle-prereqs` | If present, MySQL Installer attempts to download and install some software prerequisites, not currently present. that can be resolved with minimal intervention. If the `--silent` option is not present, you are presented with installation pages for each prerequisite. If the `--auto-handle-prereqs` options is omitted, packages with missing prerequisites are not installed. |
| `--continue` | Continues processing the next product when an error is caught while processing the action blocks containing arguments for each product. If not specified the whole operation is aborted in case of an error. |
| `--help` | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |
| `--mos-password=password` | Sets the My Oracle Support (MOS) user's password for commercial versions of the MySQL Installer. |
| `--mos-user=user_name` | Specifies the My Oracle Support (MOS) user name for access to the commercial version of MySQL Installer. If not present, only the products in the bundle, if any, are available to be installed. |

| | |
|---|---|
| `--only-ga-products` | Restricts the product set to include GA products only. |
| `--setup-type=setup_type` | Installs a predefined set of software. The setup type can be one of the following:<br>• `Developer`: Installs a complete development environment.<br>• `Server`: Installs a single MySQL server<br>• `Client`: Installs client programs and libraries<br>• `Full`: Installs everything<br>• `Custom`: Installs user-selected products. This is the default option.<br>  **Note**<br>Non-custom setup types are valid only when no other MySQL products are installed. |
| `--show-settings` | Displays the available options for the selected product, by passing in the product name after `-showsettings`. |
| `--silent` | Disable confirmation prompts. |

Examples:

```
mysqlinstallerconsole.exe --install j;8.0.29, net;8.0.28 --silent
```

```
MySQLInstallerConsole install server;8.0.30:*:port=3307;server_id=2:type=user;user=foo
```

An example that passes in additional configuration blocks, separated by `^` to fit:

```
MySQLInstallerConsole --install server;8.0.30;x64:*:type=config;open_win_firewall=true; ^
   general_log=true;bin_log=true;server_id=3306;tcp_ip=true;port=3306;root_passwd=pass; ^
   install_dir="C:\MySQL\MySQL Server 8.0":type=user;user_name=foo;password=bar;role=DBManager
```

- `[--]list`

  When this action is used without options, it activates an interactive list from which all of the available MySQL products can be searched. Enter `MySQLInstallerConsole --list` and specify a substring to search.

  Options:

  | | |
  |---|---|
  | `--all` | Lists all available products. If this option is used, all other options are ignored. |
  | `--arch=`*`architecture`* | Lists that contain the specified architecture. Permitted values are: `x86`, `x64`, and `any` (default). This option can be combined with the `--name` and `--version` options. |
  | `--help` | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |
  | `--name=`*`package_name`* | Lists products that contain the specified name (see product phrase), This option can be combined with the `--version` and `--arch` options. |
  | `--version=`*`version`* | Lists products that contain the specified version, such as 8.0 or 5.7. This option can be combined with the `--name` and `--arch` options. |

  Examples:

  ```
  MySQLInstallerConsole --list --name=net --version=8.0
  ```

- `[--]modify [`*`product1`*`:-`*`removelist`*`|+`*`addlist`*`], [`*`product2`*`:-`*`removelist`*`|+`*`addlist`*`] [...]`

  Modifies or displays features of a previously installed MySQL product. To display the features of a product, append the product keyword to the command, for example:

  ```
  MySQLInstallerConsole --modify server
  ```

  Options:

  | | |
  |---|---|
  | `--help` | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |
  | `--silent` | Disable confirmation prompts. |

  Examples:

  ```
  MySQLInstallerConsole --modify server:+documentation
  ```

  ```
  MySQLInstallerConsole modify server:-debug
  ```

- `[--]remove [product1], [product2] [...]`

  Removes one ore more products from your system. An asterisk character (`*`) can be passed in to remove all MySQL products with one command.

  Options:

  | | |
  |---|---|
  | `--continue` | Continue the operation even if an error occurs. |
  | `--help` | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |
  | `--keep-datadir` | Skips the removal of the data directory when removing MySQL Server products. |
  | `--silent` | Disable confirmation prompts. |

  Examples:

  ```
  mysqlinstallerconsole.exe remove *
  ```

  ```
  MySQLInstallerConsole --remove server --continue
  ```

- `[--]set`

  Sets one or more configurable options that affect how the MySQL Installer program connects to the internet and whether the automatic products-catalog updates feature is activated.

  Options:

  | | |
  |---|---|
  | `--catalog-update=bool_value` | Enables (`true`, default) or disables (`false`) the automatic products catalog update. This option requires an active connection to the internet. |
  | `--catalog-update-days=int_value` | Accepts an integer between 1 (default) and 365 to indicate the number of days between checks for a new catalog update when MySQL Installer is started. If `--catalog-update` is `false`, this option is ignored. |
  | `--connection-validation=validation_type` | Sets how MySQL Installer performs the check for an internet connection. Permitted values are `automatic` (default) and `manual`. |
  | `--connection-validation-urls=url_list` | A double-quote enclosed and comma-separated string that defines the list of URLs to use for checking the internet connection when `--connection-validation` is set to |

`manual`. Checks are made in the same order provided. If the first URL fails, the next URL in the list is used and so on.

`--offline-mode=bool_value`
Enables MySQL Installer to run with or without internet capabilities. Valid modes are:

- `True` to enable offline mode (run without an internet connection).
- `False` (default) to disable offline mode (run with an internet connection). Set this mode before downloading the product catalog or any products to install.

`--proxy-mode`
Specifies the proxy mode. Valid modes are:

- `Automatic` to automatically identify the proxy based on the system settings.
- `None` to ensure that no proxy is configured.
- `Manual` to set the proxy details manually (`--proxy-server`, `--proxy-port`, `--proxy-username`, `--proxy-password`).

`--proxy-password`
The password used to authenticate to the proxy server.

`--proxy-port`
The port used for the proxy server.

`--proxy-server`
The URL that point to the proxy server.

`--proxy-username`
The user name used to authenticate to the proxy server.

`--reset-defaults`
Resets the MySQL Installer options associated with the `--set` action to the default values.

Examples:

```
MySQLIntallerConsole.exe set --reset-defaults
```

```
mysqlintallerconsole.exe --set --catalog-update=false
```

```
MySQLIntallerConsole --set --catalog-update-days=3
```

```
mysqlintallerconsole --set --connection-validation=manual
--connection-validation-urls="https://www.bing.com,http://www.google.com"
```

- `[--]status`

Provides a quick overview of the MySQL products that are installed on the system. Information includes product name and version, architecture, date installed, and install location.

Options:

`--help`
Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well.

Examples:

```
MySQLInstallerConsole status
```

- [--]update

  Downloads the latest MySQL product catalog to your system. On success, the catalog is applied the next time either MySQLInstaller or MySQLInstallerConsole.exe is executed.

  MySQL Installer automatically checks for product catalog updates when it is started if *n* days have passed since the last check. Starting with MySQL Installer 1.6.4, the default value is 1 day. Previously, the default value was 7 days.

  Options:

  | | |
  |---|---|
  | --help | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |

  Examples:

  ```
  MySQLInstallerConsole update
  ```

- [--]upgrade [*product1*:*version*], [*product2*:*version*] [...]

  Upgrades one or more products on your system. The following characters are permitted for this action:

  | | |
  |---|---|
  | * | Pass in * to upgrade all products to the latest version, or pass in specific products. |
  | ! | Pass in ! as a version number to upgrade the MySQL product to its latest version. |

  Options:

  | | |
  |---|---|
  | --continue | Continue the operation even if an error occurs. |
  | --help | Shows the options and available arguments for the corresponding action. If present the action is not executed, only the help is shown, so other action-related options are ignored as well. |
  | --mos-password=*password* | Sets the My Oracle Support (MOS) user's password for commercial versions of the MySQL Installer. |
  | --mos-user=*user_name* | Specifies the My Oracle Support (MOS) user name for access to the commercial version of MySQL Installer. If not present, only the products in the bundle, if any, are available to be installed. |
  | --silent | Disable confirmation prompts. |

  Examples:

  ```
  MySQLInstallerConsole upgrade *
  ```

  ```
  MySQLInstallerConsole upgrade workbench:8.0.31
  ```

  ```
  MySQLInstallerConsole upgrade workbench:!
  ```

  ```
  MySQLInstallerConsole --upgrade server;8.0.30:!, j;8.0.29:!
  ```

## 2.3.4 Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive

Users who are installing from the noinstall package can use the instructions in this section to manually install MySQL. The process for installing MySQL from a ZIP Archive package is as follows:

1. Extract the main archive to the desired install directory

   *Optional*: also extract the debug-test archive if you plan to execute the MySQL benchmark and test suite

2. Create an option file
3. Choose a MySQL server type
4. Initialize MySQL
5. Start the MySQL server
6. Secure the default user accounts

This process is described in the sections that follow.

### 2.3.4.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version please refer to Section 2.10.10, “Upgrading MySQL on Windows”, before beginning the upgrade process.
2. Make sure that you are logged in as a user with administrator privileges.
3. Choose an installation location. Traditionally, the MySQL server is installed in `C:\mysql`. If you do not install MySQL at `C:\mysql`, you must specify the path to the install directory during startup or in an option file. See Section 2.3.4.2, “Creating an Option File”.

   **Note**

   The MySQL Installer installs MySQL under `C:\Program Files\MySQL`.

4. Extract the install archive to the chosen installation location using your preferred file-compression tool. Some tools may extract the archive to a folder within your chosen installation location. If this occurs, you can move the contents of the subfolder into the chosen installation location.

### 2.3.4.2 Creating an Option File

If you need to specify startup options when you run the server, you can indicate them on the command line or place them in an option file. For options that are used every time the server starts, you may find it most convenient to use an option file to specify your MySQL configuration. This is particularly true under the following circumstances:

- The installation or data directory locations are different from the default locations (`C:\Program Files\MySQL\MySQL Server 8.0` and `C:\Program Files\MySQL\MySQL Server 8.0\data`).
- You need to tune the server settings, such as memory, cache, or InnoDB configuration information.

When the MySQL server starts on Windows, it looks for option files in several locations, such as the Windows directory, `C:\`, and the MySQL installation directory (for the full list of locations, see Section 4.2.2.2, “Using Option Files”). The Windows directory typically is named something like `C:\WINDOWS`. You can determine its exact location from the value of the `WINDIR` environment variable using the following command:

```
C:\> echo %WINDIR%
```

MySQL looks for options in each location first in the `my.ini` file, and then in the `my.cnf` file. However, to avoid confusion, it is best if you use only one file. If your PC uses a boot loader where `C:` is not the

boot drive, your only option is to use the my.ini file. Whichever option file you use, it must be a plain text file.

**Note**

When using the MySQL Installer to install MySQL Server, it creates the my.ini at the default location, and the user executing MySQL Installer is granted full permissions to this new my.ini file.

In other words, be sure that the MySQL Server user has permission to read the my.ini file.

You can also make use of the example option files included with your MySQL distribution; see Section 5.1.2, "Server Configuration Defaults".

An option file can be created and modified with any text editor, such as Notepad. For example, if MySQL is installed in E:\mysql and the data directory is in E:\mydata\data, you can create an option file containing a [mysqld] section to specify values for the basedir and datadir options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=E:/mydata/data
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=E:\\mysql
# set datadir to the location of your data directory
datadir=E:\\mydata\\data
```

The rules for use of backslash in option file values are given in Section 4.2.2.2, "Using Option Files".

The ZIP archive does not include a data directory. To initialize a MySQL installation by creating the data directory and populating the tables in the mysql system database, initialize MySQL using either --initialize or --initialize-insecure. For additional information, see Section 2.9.1, "Initializing the Data Directory".

If you would like to use a data directory in a different location, you should copy the entire contents of the data directory to the new location. For example, if you want to use E:\mydata as the data directory instead, you must do two things:

1. Move the entire data directory and all of its contents from the default location (for example C:\Program Files\MySQL\MySQL Server 8.0\data) to E:\mydata.
2. Use a --datadir option to specify the new data directory location each time you start the server.

### 2.3.4.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 8.0.

| Binary | Description |
|---|---|
| mysqld | Optimized binary with named-pipe support |
| mysqld-debug | Like mysqld, but compiled with full debugging and automatic memory allocation checking |

All of the preceding binaries are optimized for modern Intel processors, but should work on any Intel i386-class or higher processor.

Each of the servers in a distribution support the same set of storage engines. The SHOW ENGINES statement displays which engines a given server supports.

All Windows MySQL 8.0 servers have support for symbolic linking of database directories.

MySQL supports TCP/IP on all Windows platforms. MySQL servers on Windows also support named pipes, if you start the server with the named_pipe system variable enabled. It is necessary to enable this variable explicitly because some users have experienced problems with shutting down the MySQL server when named pipes were used. The default is to use TCP/IP regardless of platform because named pipes are slower than TCP/IP in many Windows configurations.

### 2.3.4.4 Initializing the Data Directory

If you installed MySQL using the noinstall package, no data directory is included. To initialize the data directory, use the instructions at Section 2.9.1, “Initializing the Data Directory”.

### 2.3.4.5 Starting the Server for the First Time

This section gives a general overview of starting the MySQL server. The following sections provide more specific information for starting the MySQL server from the command line or as a Windows service.

The information here applies primarily if you installed MySQL using the noinstall version, or if you wish to configure and test MySQL manually rather than with the MySQL Installer.

The examples in these sections assume that MySQL is installed under the default location of C:\Program Files\MySQL\MySQL Server 8.0. Adjust the path names shown in the examples if you have MySQL installed in a different location.

Clients have two options. They can use TCP/IP, or they can use a named pipe if the server supports named-pipe connections.

MySQL for Windows also supports shared-memory connections if the server is started with the shared_memory system variable enabled. Clients can connect through shared memory by using the --protocol=MEMORY option.

For information about which server binary to run, see Section 2.3.4.3, “Selecting a MySQL Server Type”.

Testing is best done from a command prompt in a console window (or “DOS window”). In this way you can have the server display status messages in the window where they are easy to see. If something is wrong with your configuration, these messages make it easier for you to identify and fix any problems.

**Note**

The database must be initialized before MySQL can be started. For additional information about the initialization process, see Section 2.9.1, “Initializing the Data Directory”.

To start the server, enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --console
```

You should see messages similar to those following as it starts (the path names and sizes may differ). The ready for connections messages indicate that the server is ready to service client connections.

```
[Server] C:\mysql\bin\mysqld.exe (mysqld 8.0.30) starting as process 21236
[InnoDB] InnoDB initialization has started.
[InnoDB] InnoDB initialization has ended.
[Server] CA certificate ca.pem is self signed.
```

```
[Server] Channel mysql_main configured to support TLS.
Encrypted connections are now supported for this channel.
[Server] X Plugin ready for connections. Bind-address: '::' port: 33060
[Server] C:\mysql\bin\mysqld.exe: ready for connections.
Version: '8.0.30'  socket: ''  port: 3306  MySQL Community Server - GPL.
```

You can now open a new console window in which to run client programs.

If you omit the `--console` option, the server writes diagnostic output to the error log in the data directory (`C:\Program Files\MySQL\MySQL Server 8.0\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

**Note**

The initial `root` account in the MySQL grant tables has no password. After starting the server, you should set up a password for it using the instructions in Section 2.9.4, “Securing the Initial MySQL Account”.

### 2.3.4.6 Starting MySQL from the Windows Command Line

The MySQL server can be started manually from the command line. This can be done on any version of Windows.

To start the `mysqld` server from the command line, you should start a console window (or “DOS window”) and enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
```

The path to `mysqld` may vary depending on the install location of MySQL on your system.

You can stop the MySQL server by executing this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" -u root shutdown
```

**Note**

If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

**Note**

Users in the MySQL grant system are wholly independent from any operating system users under Microsoft Windows.

If `mysqld` doesn't start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. By default, the error log is located in the `C:\Program Files\MySQL\MySQL Server 8.0\data` directory. It is the file with a suffix of `.err`, or may be specified by passing in the `--log-error` option. Alternatively, you can try to start the server with the `--console` option; in this case, the server may display some useful information on the screen to help solve the problem.

The last option is to start `mysqld` with the `--standalone` and `--debug` options. In this case, `mysqld` writes a log file `C:\mysqld.trace` that should contain the reason why `mysqld` doesn't start. See Section 5.9.4, “The DBUG Package”.

Use `mysqld --verbose --help` to display all the options that `mysqld` supports.

### 2.3.4.7 Customizing the PATH for MySQL Tools

**Warning**

You must exercise great care when editing your system PATH by hand; accidental deletion or modification of any portion of the existing PATH value can leave you with a malfunctioning or even unusable system.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL bin directory to your Windows system PATH environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL bin directory (for example, C:\Program Files\MySQL\MySQL Server 8.0\bin)

**Note**

There must be a semicolon separating this path from any values present in this field.

Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. The new PATH value should now be available to any new command shell you open, allowing you to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the mysql client, and all MySQL command-line utilities such as mysqladmin and mysqldump.

You should not add the MySQL bin directory to your Windows PATH if you are running multiple MySQL servers on the same machine.

## 2.3.4.8 Starting MySQL as a Windows Service

On Windows, the recommended way to run MySQL is to install it as a Windows service, so that MySQL starts and stops automatically when Windows starts and stops. A MySQL server installed as a service can also be controlled from the command line using NET commands, or with the graphical Services utility. Generally, to install MySQL as a Windows service you should be logged in using an account that has administrator rights.

The Services utility (the Windows Service Control Manager) can be found in the Windows Control Panel. To avoid conflicts, it is advisable to close the Services utility while performing server installation or removal operations from the command line.

### Installing the service

Before installing MySQL as a Windows service, you should first stop the current server if it is running by using the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin"
         -u root shutdown
```

**Note**

If the MySQL root user account has a password, you need to invoke mysqladmin with the -p option and supply the password when prompted.

This command invokes the MySQL administrative utility mysqladmin to connect to the server and tell it to shut down. The command connects as the MySQL root user, which is the default administrative account in the MySQL grant system.

**Note**

Users in the MySQL grant system are wholly independent from any operating system users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --install
```

The service-installation command does not start the server. Instructions for that are given later in this section.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL bin directory to your Windows system PATH environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL bin directory (for example, C:\Program Files\MySQL\MySQL Server 8.0\bin), and there should be a semicolon separating this path from any values present in this field. Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the mysql client, and all MySQL command-line utilities such as mysqladmin and mysqldump.

  You should not add the MySQL bin directory to your Windows PATH if you are running multiple MySQL servers on the same machine.

**Warning**

You must exercise great care when editing your system PATH by hand; accidental deletion or modification of any portion of the existing PATH value can leave you with a malfunctioning or even unusable system.

The following additional arguments can be used when installing the service:

- You can specify a service name immediately following the --install option. The default service name is MySQL.
- If a service name is given, it can be followed by a single option. By convention, this should be --defaults-file=*file_name* to specify the name of an option file from which the server should read options when it starts.

  The use of a single option other than --defaults-file is possible but discouraged. --defaults-file is more flexible because it enables you to specify multiple startup options for the server by placing them in the named option file.
- You can also specify a --local-service option following the service name. This causes the server to run using the LocalService Windows account that has limited system privileges. If both --defaults-file and --local-service are given following the service name, they can be in any order.

For a MySQL server that is installed as a Windows service, the following rules determine the service name and option files that the server uses:

- If the service-installation command specifies no service name or the default service name (MySQL) following the --install option, the server uses the service name of MySQL and reads options from the [mysqld] group in the standard option files.
- If the service-installation command specifies a service name other than MySQL following the --install option, the server uses that service name. It reads options from the [mysqld] group and the group that has the same name as the service in the standard option files. This enables you to use the [mysqld] group for options that should be used by all MySQL services, and an option group with the service name for use by the server installed with that service name.
- If the service-installation command specifies a --defaults-file option after the service name, the server reads options the same way as described in the previous item, except that it reads options only from the named file and ignores the standard option files.

As a more complex example, consider the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
          --install MySQL --defaults-file=C:\my-opts.cnf
```

Here, the default service name (MySQL) is given after the --install option. If no --defaults-file option had been given, this command would have the effect of causing the server to read the [mysqld] group from the standard option files. However, because the --defaults-file option is present, the server reads options from the [mysqld] option group, and only from the named file.

**Note**

On Windows, if the server is started with the --defaults-file and --install options, --install must be first. Otherwise, mysqld.exe attempts to start the MySQL server.

You can also specify options as Start parameters in the Windows Services utility before you start the MySQL service.

Finally, before trying to start the MySQL service, make sure the user variables %TEMP% and %TMP% (and also %TMPDIR%, if it has ever been set) for the operating system user who is to run the service are pointing to a folder to which the user has write access. The default user for running the MySQL service is LocalSystem, and the default value for its %TEMP% and %TMP% is C:\Windows\Temp, a directory LocalSystem has write access to by default. However, if there are any changes to that default setup (for example, changes to the user who runs the service or to the mentioned user variables, or the --tmpdir option has been used to put the temporary directory somewhere else), the MySQL service might fail to run because write access to the temporary directory has not been granted to the proper user.

### Starting the service

After a MySQL server instance has been installed as a service, Windows starts the service automatically whenever Windows starts. The service also can be started immediately from the Services utility, or by using an sc start *mysqld_service_name* or NET START *mysqld_service_name* command. SC and NET commands are not case-sensitive.

When run as a service, mysqld has no access to a console window, so no messages can be seen there. If mysqld does not start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. The error log is located in the MySQL data directory (for example, C:\Program Files\MySQL\MySQL Server 8.0\data). It is the file with a suffix of .err.

When a MySQL server has been installed as a service, and the service is running, Windows stops the service automatically when Windows shuts down. The server also can be stopped manually

using the `Services` utility, the `sc stop mysqld_service_name` command, the `NET STOP mysqld_service_name` command, or the `mysqladmin shutdown` command.

You also have the choice of installing the server as a manual service if you do not wish for the service to be started automatically during the boot process. To do this, use the `--install-manual` option rather than the `--install` option:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --install-manual
```

#### Removing the service

To remove a server that is installed as a service, first stop it if it is running by executing `SC STOP mysqld_service_name` or `NET STOP mysqld_service_name`. Then use `SC DELETE mysqld_service_name` to remove it:

```
C:\> SC DELETE mysql
```

Alternatively, use the `mysqld --remove` option to remove the service.

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --remove
```

If `mysqld` is not running as a service, you can start it from the command line. For instructions, see Section 2.3.4.6, “Starting MySQL from the Windows Command Line”.

If you encounter difficulties during installation, see Section 2.3.5, “Troubleshooting a Microsoft Windows MySQL Server Installation”.

For more information about stopping or removing a Windows service, see Section 5.8.2.2, “Starting Multiple MySQL Instances as Windows Services”.

### 2.3.4.9 Testing The MySQL Installation

You can test whether the MySQL server is working by executing any of the following commands:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql" test
```

If `mysqld` is slow to respond to TCP/IP connections from client programs, there is probably a problem with your DNS. In this case, start `mysqld` with the `skip_name_resolve` system variable enabled and use only `localhost` and IP addresses in the `Host` column of the MySQL grant tables. (Be sure that an account exists that specifies an IP address or you may not be able to connect.)

You can force a MySQL client to use a named-pipe connection rather than TCP/IP by specifying the `--pipe` or `--protocol=PIPE` option, or by specifying `.` (period) as the host name. Use the `--socket` option to specify the name of the pipe if you do not want to use the default pipe name.

If you have set a password for the `root` account, deleted the anonymous account, or created a new user account, then to connect to the MySQL server you must use the appropriate `-u` and `-p` options with the commands shown previously. See Section 4.2.4, “Connecting to the MySQL Server Using Command Options”.

For more information about `mysqlshow`, see Section 4.5.7, “mysqlshow — Display Database, Table, and Column Information”.

## 2.3.5 Troubleshooting a Microsoft Windows MySQL Server Installation

When installing and running MySQL for the first time, you may encounter certain errors that prevent the MySQL server from starting. This section helps you diagnose and correct some of these errors.

Your first resource when troubleshooting server issues is the error log. The MySQL server uses the error log to record information relevant to the error that prevents the server from starting. The error log is located in the data directory specified in your `my.ini` file. The default data directory location is `C:\Program Files\MySQL\MySQL Server 8.0\data`, or `C:\ProgramData\Mysql` on Windows

7 and Windows Server 2008. The `C:\ProgramData` directory is hidden by default. You need to change your folder options to see the directory and contents. For more information on the error log and understanding the content, see Section 5.4.2, “The Error Log”.

For information regarding possible errors, also consult the console messages displayed when the MySQL service is starting. Use the `SC START mysqld_service_name` or `NET START mysqld_service_name` command from the command line after installing `mysqld` as a service to see any error messages regarding the starting of the MySQL server as a service. See Section 2.3.4.8, “Starting MySQL as a Windows Service”.

The following examples show other common error messages you might encounter when installing MySQL and starting the server for the first time:

- If the MySQL server cannot find the `mysql` privileges database or other critical files, it displays these messages:

  ```
  System error 1067 has occurred.
  Fatal error: Can't open and lock privilege tables:
  Table 'mysql.user' doesn't exist
  ```

  These messages often occur when the MySQL base or data directories are installed in different locations than the default locations (`C:\Program Files\MySQL\MySQL Server 8.0` and `C:\Program Files\MySQL\MySQL Server 8.0\data`, respectively).

  This situation can occur when MySQL is upgraded and installed to a new location, but the configuration file is not updated to reflect the new location. In addition, old and new configuration files might conflict. Be sure to delete or rename any old configuration files when upgrading MySQL.

  If you have installed MySQL to a directory other than `C:\Program Files\MySQL\MySQL Server 8.0`, ensure that the MySQL server is aware of this through the use of a configuration (`my.ini`) file. Put the `my.ini` file in your Windows directory, typically `C:\WINDOWS`. To determine its exact location from the value of the `WINDIR` environment variable, issue the following command from the command prompt:

  ```
  C:\> echo %WINDIR%
  ```

  You can create or modify an option file with any text editor, such as Notepad. For example, if MySQL is installed in `E:\mysql` and the data directory is `D:\MySQLdata`, you can create the option file and set up a `[mysqld]` section to specify values for the `basedir` and `datadir` options:

  ```
  [mysqld]
  # set basedir to your installation path
  basedir=E:/mysql
  # set datadir to the location of your data directory
  datadir=D:/MySQLdata
  ```

  Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

  ```
  [mysqld]
  # set basedir to your installation path
  basedir=C:\\Program Files\\MySQL\\MySQL Server 8.0
  # set datadir to the location of your data directory
  datadir=D:\\MySQLdata
  ```

  The rules for use of backslash in option file values are given in Section 4.2.2.2, “Using Option Files”.

  If you change the `datadir` value in your MySQL configuration file, you must move the contents of the existing MySQL data directory before restarting the MySQL server.

  See Section 2.3.4.2, “Creating an Option File”.

- If you reinstall or upgrade MySQL without first stopping and removing the existing MySQL service and install MySQL using the MySQL Installer, you might see this error:

```
Error: Cannot create Windows service for MySql. Error: 0
```

This occurs when the Configuration Wizard tries to install the service and finds an existing service with the same name.

One solution to this problem is to choose a service name other than `mysql` when using the configuration wizard. This enables the new service to be installed correctly, but leaves the outdated service in place. Although this is harmless, it is best to remove old services that are no longer in use.

To permanently remove the old `mysql` service, execute the following command as a user with administrative privileges, on the command line:

```
C:\> SC DELETE mysql
[SC] DeleteService SUCCESS
```

If the SC utility is not available for your version of Windows, download the `delsrv` utility from http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp and use the `delsrv mysql` syntax.

## 2.3.6 Windows Postinstallation Procedures

GUI tools exist that perform most of the tasks described in this section, including:

- MySQL Installer: Used to install and upgrade MySQL products.
- MySQL Workbench: Manages the MySQL server and edits SQL statements.

If necessary, initialize the data directory and create the MySQL grant tables. Windows installation operations performed by MySQL Installer initialize the data directory automatically. For installation from a ZIP Archive package, initialize the data directory as described at Section 2.9.1, “Initializing the Data Directory”.

Regarding passwords, if you installed MySQL using the MySQL Installer, you may have already assigned a password to the initial `root` account. (See Section 2.3.3, “MySQL Installer for Windows”.) Otherwise, use the password-assignment procedure given in Section 2.9.4, “Securing the Initial MySQL Account”.

Before assigning a password, you might want to try running some client programs to make sure that you can connect to the server and that it is operating properly. Make sure that the server is running (see Section 2.3.4.5, “Starting the Server for the First Time”). You can also set up a MySQL service that runs automatically when Windows starts (see Section 2.3.4.8, “Starting MySQL as a Windows Service”).

These instructions assume that your current location is the MySQL installation directory and that it has a `bin` subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

If you installed MySQL using MySQL Installer (see Section 2.3.3, “MySQL Installer for Windows”), the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.0`:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 8.0"
```

A common installation location for installation from a ZIP archive is `C:\mysql`:

```
C:\> cd C:\mysql
```

Alternatively, add the `bin` directory to your `PATH` environment variable setting. That enables your command interpreter to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See Section 2.3.4.7, “Customizing the PATH for MySQL Tools”.