Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

To explicitly specify that there is no password and that the client program should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, the client program prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that the client program should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but the client program does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use. The default port number is 3306.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  This option explicitly specifies which transport protocol to use for connecting to the server. It is useful when other connection parameters normally result in use of a protocol other than the one you want. For example, connections on Unix to `localhost` are made using a Unix socket file by default:

  ```
  mysql --host=localhost
  ```

  To force TCP/IP transport to be used instead, specify a `--protocol` option:

```
mysql --host=localhost --protocol=TCP
```

The following table shows the permissible `--protocol` option values and indicates the applicable platforms for each value. The values are not case-sensitive.

`--protocol` Value	Transport Protocol Used	Applicable Platforms
`TCP`	TCP/IP transport to local or remote server	All
`SOCKET`	Unix socket-file transport to local server	Unix and Unix-like systems
`PIPE`	Named-pipe transport to local server	Windows
`MEMORY`	Shared-memory transport to local server	Windows

See also Section 4.2.7, “Connection Transport Protocols”

- `--shared-memory-base-name=`*`name`*

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--socket=`*`path`*, `-S` *`path`*

  On Unix, the name of the Unix socket file to use for connections made using a named pipe to a local server. The default Unix socket file name is `/tmp/mysql.sock`.

  On Windows, the name of the named pipe to use for connections to a local server. The default Windows pipe name is `MySQL`. The pipe name is not case-sensitive.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--user=`*`user_name`*, `-u` *`user_name`*

  The user name of the MySQL account to use for connecting to the server. The default user name is `ODBC` on Windows or your Unix login name on Unix.

Command Options for Encrypted Connections

This section describes options for client programs that specify whether to use encrypted connections to the server, the names of certificate and key files, and other parameters related to encrypted-connection support. For examples of suggested use and how to check whether a connection is encrypted, see Section 6.3.1, “Configuring MySQL to Use Encrypted Connections”.

Note

These options have an effect only for connections that use a transport protocol subject to encryption; that is, TCP/IP and Unix socket-file connections. See Section 4.2.7, “Connection Transport Protocols”

For information about using encrypted connections from the MySQL C API, see Support for Encrypted Connections.

Table 4.4 Connection-Encryption Option Summary

Option Name	Description	Introduced
--get-server-public-key	Request RSA public key from server	
--server-public-key-path	Path name to file containing RSA public key	
--ssl-ca	File that contains list of trusted SSL Certificate Authorities	
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files	
--ssl-cert	File that contains X.509 certificate	
--ssl-cipher	Permissible ciphers for connection encryption	
--ssl-crl	File that contains certificate revocation lists	
--ssl-crlpath	Directory that contains certificate revocation-list files	
--ssl-fips-mode	Whether to enable FIPS mode on client side	
--ssl-key	File that contains X.509 key	
--ssl-mode	Desired security state of connection to server	
--ssl-session-data	File that contains SSL session data	8.0.29
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16
--tls-version	Permissible TLS protocols for encrypted connections	

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=`*`file_name`* is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--server-public-key-path=`*`file_name`*

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is

ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

This option is available only if MySQL was built using OpenSSL.

For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--ssl-ca=file_name`

  The path name of the Certificate Authority (CA) certificate file in PEM format. The file contains a list of trusted SSL Certificate Authorities.

  To tell the client not to authenticate the server certificate when establishing an encrypted connection to the server, specify neither `--ssl-ca` nor `--ssl-capath`. The server still verifies the client according to any applicable requirements established for the client account, and it still uses any `ssl_ca` or `ssl_capath` system variable values specified on the server side.

  To specify the CA file for the server, set the `ssl_ca` system variable.

- `--ssl-capath=dir_name`

  The path name of the directory that contains trusted SSL certificate authority (CA) certificate files in PEM format.

  To tell the client not to authenticate the server certificate when establishing an encrypted connection to the server, specify neither `--ssl-ca` nor `--ssl-capath`. The server still verifies the client according to any applicable requirements established for the client account, and it still uses any `ssl_ca` or `ssl_capath` system variable values specified on the server side.

  To specify the CA directory for the server, set the `ssl_capath` system variable.

- `--ssl-cert=file_name`

  The path name of the client SSL public key certificate file in PEM format.

  To specify the server SSL public key certificate file, set the `ssl_cert` system variable.

  

  Note

  Chained SSL certificate support was added in v8.0.30; previously only the first certificate was read.

- `--ssl-cipher=cipher_list`

  The list of permissible encryption ciphers for connections that use TLS protocols up through TLSv1.2. If no cipher in the list is supported, encrypted connections that use these TLS protocols do not work.

  For greatest portability, `cipher_list` should be a list of one or more cipher names, separated by colons. Examples:

```
--ssl-cipher=AES128-SHA
```

```
--ssl-cipher=DHE-RSA-AES128-GCM-SHA256:AES128-SHA
```

OpenSSL supports the syntax for specifying ciphers described in the OpenSSL documentation at https://www.openssl.org/docs/manmaster/man1/ciphers.html.

For information about which encryption ciphers MySQL supports, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

To specify the encryption ciphers for the server, set the `ssl_cipher` system variable.

- `--ssl-crl=file_name`

  The path name of the file containing certificate revocation lists in PEM format.

  If neither `--ssl-crl` nor `--ssl-crlpath` is given, no CRL checks are performed, even if the CA path contains certificate revocation lists.

  To specify the revocation-list file for the server, set the `ssl_crl` system variable.

- `--ssl-crlpath=dir_name`

  The path name of the directory that contains certificate revocation-list files in PEM format.

  If neither `--ssl-crl` nor `--ssl-crlpath` is given, no CRL checks are performed, even if the CA path contains certificate revocation lists.

  To specify the revocation-list directory for the server, set the `ssl_crlpath` system variable.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permissible:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permissible value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

  To specify the FIPS mode for the server, set the `ssl_fips_mode` system variable.

- `--ssl-key=file_name`

  The path name of the client SSL private key file in PEM format. For better security, use a certificate with an RSA key size of at least 2048 bits.

  If the key file is protected by a passphrase, the client program prompts the user for the passphrase. The password must be given interactively; it cannot be stored in a file. If the passphrase is incorrect, the program continues as if it could not read the key.

  To specify the server SSL private key file, set the `ssl_key` system variable.

- `--ssl-mode=mode`

This option specifies the desired security state of the connection to the server. These mode values are permissible, in order of increasing strictness:

- `DISABLED`: Establish an unencrypted connection.
- `PREFERRED`: Establish an encrypted connection if the server supports encrypted connections, falling back to an unencrypted connection if an encrypted connection cannot be established. This is the default if `--ssl-mode` is not specified.

  Connections over Unix socket files are not encrypted with a mode of `PREFERRED`. To enforce encryption for Unix socket-file connections, use a mode of `REQUIRED` or stricter. (However, socket-file transport is secure by default, so encrypting a socket-file connection makes it no more secure and increases CPU load.)
- `REQUIRED`: Establish an encrypted connection if the server supports encrypted connections. The connection attempt fails if an encrypted connection cannot be established.
- `VERIFY_CA`: Like `REQUIRED`, but additionally verify the server Certificate Authority (CA) certificate against the configured CA certificates. The connection attempt fails if no valid matching CA certificates are found.
- `VERIFY_IDENTITY`: Like `VERIFY_CA`, but additionally perform host name identity verification by checking the host name the client uses for connecting to the server against the identity in the certificate that the server sends to the client:
  - As of MySQL 8.0.12, if the client uses OpenSSL 1.0.2 or higher, the client checks whether the host name that it uses for connecting matches either the Subject Alternative Name value or the Common Name value in the server certificate. Host name identity verification also works with certificates that specify the Common Name using wildcards.
  - Otherwise, the client checks whether the host name that it uses for connecting matches the Common Name value in the server certificate.

  The connection fails if there is a mismatch. For encrypted connections, this option helps prevent man-in-the-middle attacks.

Note

Host name identity verification with `VERIFY_IDENTITY` does not work with self-signed certificates that are created automatically by the server or manually using `mysql_ssl_rsa_setup` (see Section 6.3.3.1, “Creating SSL and RSA Certificates and Keys using MySQL”). Such self-signed certificates do not contain the server name as the Common Name value.

Important

The default setting, `--ssl-mode=PREFERRED`, produces an encrypted connection if the other default settings are unchanged. However, to help prevent sophisticated man-in-the-middle attacks, it is important for the client to verify the server’s identity. The settings `--ssl-mode=VERIFY_CA` and `--ssl-mode=VERIFY_IDENTITY` are a better choice than the default setting to help prevent this type of attack. To implement one of these settings, you must first ensure that the CA certificate for the server is reliably available to all the

clients that use it in your environment, otherwise availability issues will result. For this reason, they are not the default setting.

The `--ssl-mode` option interacts with CA certificate options as follows:

- If `--ssl-mode` is not explicitly set otherwise, use of `--ssl-ca` or `--ssl-capath` implies `--ssl-mode=VERIFY_CA`.
- For `--ssl-mode` values of `VERIFY_CA` or `VERIFY_IDENTITY`, `--ssl-ca` or `--ssl-capath` is also required, to supply a CA certificate that matches the one used by the server.
- An explicit `--ssl-mode` option with a value other than `VERIFY_CA` or `VERIFY_IDENTITY`, together with an explicit `--ssl-ca` or `--ssl-capath` option, produces a warning that no verification of the server certificate is performed, despite a CA certificate option being specified.

To require use of encrypted connections by a MySQL account, use `CREATE USER` to create the account with a `REQUIRE SSL` clause, or use `ALTER USER` for an existing account to add a `REQUIRE SSL` clause. This causes connection attempts by clients that use the account to be rejected unless MySQL supports encrypted connections and an encrypted connection can be established.

The `REQUIRE` clause permits other encryption-related options, which can be used to enforce security requirements stricter than `REQUIRE SSL`. For additional details about which command options may or must be specified by clients that connect using accounts configured using the various `REQUIRE` options, see CREATE USER SSL/TLS Options.

- `--ssl-session-data=file_name`

  The path name of the client SSL session data file in PEM format for session reuse.

  When you invoke a MySQL client program with the `--ssl-session-data` option, the client attempts to deserialize session data from the file, if provided, and then use it to establish a new connection. If you supply a file, but the session is not reused, then the connection fails unless you also specified the `--ssl-session-data-continue-on-failed-reuse` option on the command line when you invoked the client program.

  The `mysql` command, `ssl_session_data_print`, generates the session data file (see Section 4.5.1.2, “mysql Client Commands”).

- `ssl-session-data-continue-on-failed-reuse`

  Controls whether a new connection is started to replace an attempted connection that tried but failed to reuse session data specified with the `--ssl-session-data` command-line option. By default, the `--ssl-session-data-continue-on-failed-reuse` command-line option is off, which causes a client program to return a connect failure when session data are supplied and not reused.

  To ensure that a new, unrelated connection opens after session reuse fails silently, invoke MySQL client programs with both the `--ssl-session-data` and `--ssl-session-data-continue-on-failed-reuse` command-line options.

- `--tls-ciphersuites=ciphersuite_list`

  This option specifies which ciphersuites the client permits for encrypted connections that use TLSv1.3. The value is a list of zero or more colon-separated ciphersuite names. For example:

  ```
  mysql --tls-ciphersuites="suite1:suite2:suite3"
  ```

  The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. If this option is not set, the client permits the default set of ciphersuites. If the option is set to

the empty string, no ciphersuites are enabled and encrypted connections cannot be established. For more information, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

This option was added in MySQL 8.0.16.

To specify which ciphersuites the server permits, set the `tls_ciphersuites` system variable.

- `--tls-version=protocol_list`

  This option specifies the TLS protocols the client permits for encrypted connections. The value is a list of one or more comma-separated protocol versions. For example:

  ```
  mysql --tls-version="TLSv1.2,TLSv1.3"
  ```

  The protocols that can be named for this option depend on the SSL library used to compile MySQL, and on the MySQL Server release.

  

  Important

  - Support for the TLSv1 and TLSv1.1 connection protocols is removed from MySQL Server as of MySQL 8.0.28. The protocols were deprecated from MySQL 8.0.26, though MySQL Server clients do not return warnings to the user if a deprecated TLS protocol version is used. From MySQL 8.0.28 onwards, clients, including MySQL Shell, that support the `--tls-version` option cannot make a TLS/SSL connection with the protocol set to TLSv1 or TLSv1.1. If a client attempts to connect using these protocols, for TCP connections, the connection fails, and an error is returned to the client. For socket connections, if `--ssl-mode` is set to `REQUIRED`, the connection fails, otherwise the connection is made but with TLS/SSL disabled. See Removal of Support for the TLSv1 and TLSv1.1 Protocols for more information.
  - Support for the TLSv1.3 protocol is available in MySQL Server as of MySQL 8.0.16, provided that MySQL Server was compiled using OpenSSL 1.1.1 or higher. The server checks the version of OpenSSL at startup, and if it is lower than 1.1.1, TLSv1.3 is removed from the default value for the server system variables relating to the TLS version (such as the `tls_version` system variable).

  Permitted protocols should be chosen such as not to leave “holes” in the list. For example, these values do not have holes:

  ```
  --tls-version="TLSv1,TLSv1.1,TLSv1.2,TLSv1.3"
  --tls-version="TLSv1.1,TLSv1.2,TLSv1.3"
  --tls-version="TLSv1.2,TLSv1.3"
  --tls-version="TLSv1.3"

  From MySQL 8.0.28, only the last two values are suitable.
  ```

  These values do have holes and should not be used:

  ```
  --tls-version="TLSv1,TLSv1.2"
  --tls-version="TLSv1.1,TLSv1.3"
  ```

  For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  To specify which TLS protocols the server permits, set the `tls_version` system variable.

Command Options for Connection Compression

This section describes options that enable client programs to control use of compression for connections to the server. For additional information and examples showing how to use them, see Section 4.2.8, “Connection Compression Control”.

Table 4.5 Connection-Compression Option Summary

Option Name	Description	Introduced	Deprecated
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=`*`value`*

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  This option was added in MySQL 8.0.18.

- `--zstd-compression-level=`*`level`*

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  This option was added in MySQL 8.0.18.

4.2.4 Connecting to the MySQL Server Using Command Options

This section describes use of command-line options to specify how to establish connections to the MySQL server, for clients such as `mysql` or `mysqldump`. For information on establishing connections using URI-like connection strings or key-value pairs, for clients such as MySQL Shell, see Section 4.2.5, “Connecting to the Server Using URI-Like Strings or Key-Value Pairs”. For additional information if you are unable to connect, see Section 6.2.22, “Troubleshooting Problems Connecting to MySQL”.

For a client program to connect to the MySQL server, it must use the proper connection parameters, such as the name of the host where the server is running and the user name and password of your MySQL account. Each connection parameter has a default value, but you can override default values as necessary using program options specified either on the command line or in an option file.

The examples here use the `mysql` client program, but the principles apply to other clients such as `mysqldump`, `mysqladmin`, or `mysqlshow`.

This command invokes `mysql` without specifying any explicit connection parameters:

```
mysql
```

Because there are no parameter options, the default values apply:

- The default host name is `localhost`. On Unix, this has a special meaning, as described later.
- The default user name is `ODBC` on Windows or your Unix login name on Unix.
- No password is sent because neither `--password` nor `-p` is given.
- For `mysql`, the first nonoption argument is taken as the name of the default database. Because there is no such argument, `mysql` selects no default database.

To specify the host name and user name explicitly, as well as a password, supply appropriate options on the command line. To select a default database, add a database-name argument. Examples:

```
mysql --host=localhost --user=myname --password=password mydb
mysql -h localhost -u myname -ppassword mydb
```

For password options, the password value is optional:

- If you use a `--password` or `-p` option and specify a password value, there must be *no space* between `--password=` or `-p` and the password following it.
- If you use `--password` or `-p` but do not specify a password value, the client program prompts you to enter the password. The password is not displayed as you enter it. This is more secure than giving the password on the command line, which might enable other users on your system to see the password line by executing a command such as `ps`. See Section 6.1.2.1, “End-User Guidelines for Password Security”.
- To explicitly specify that there is no password and that the client program should not prompt for one, use the `--skip-password` option.

As just mentioned, including the password value on the command line is a security risk. To avoid this risk, specify the `--password` or `-p` option without any following password value:

```
mysql --host=localhost --user=myname --password mydb
mysql -h localhost -u myname -p mydb
```

When the `--password` or `-p` option is given with no password value, the client program prints a prompt and waits for you to enter the password. (In these examples, `mydb` is *not* interpreted as a password because it is separated from the preceding password option by a space.)

On some systems, the library routine that MySQL uses to prompt for a password automatically limits the password to eight characters. That limitation is a property of the system library, not MySQL. Internally, MySQL does not have any limit for the length of the password. To work around the limitation on systems affected by it, specify your password in an option file (see Section 4.2.2.2, “Using Option Files”). Another workaround is to change your MySQL password to a value that has eight or fewer characters, but that has the disadvantage that shorter passwords tend to be less secure.

Client programs determine what type of connection to make as follows:

- If the host is not specified or is `localhost`, a connection to the local host occurs:
  - On Windows, the client connects using shared memory, if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.
  - On Unix, MySQL programs treat the host name `localhost` specially, in a way that is likely different from what you expect compared to other network-based programs: the client connects using a Unix socket file. The `--socket` option or the `MYSQL_UNIX_PORT` environment variable may be used to specify the socket name.
- On Windows, if `host` is `.` (period), or TCP/IP is not enabled and `--socket` is not specified or the host is empty, the client connects using a named pipe, if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. If named-pipe connections are not supported or if the user making the connection is not a member of the Windows group specified by the `named_pipe_full_access_group` system variable, an error occurs.

- Otherwise, the connection uses TCP/IP.

The `--protocol` option enables you to use a particular transport protocol even when other options normally result in use of a different protocol. That is, `--protocol` specifies the transport protocol explicitly and overrides the preceding rules, even for `localhost`.

Only connection options that are relevant to the selected transport protocol are used or checked. Other connection options are ignored. For example, with `--host=localhost` on Unix, the client attempts to connect to the local server using a Unix socket file, even if a `--port` or `-P` option is given to specify a TCP/IP port number.

To ensure that the client makes a TCP/IP connection to the local server, use `--host` or `-h` to specify a host name value of `127.0.0.1` (instead of `localhost`), or the IP address or name of the local server. You can also specify the transport protocol explicitly, even for `localhost`, by using the `--protocol=TCP` option. Examples:

```
mysql --host=127.0.0.1
mysql --protocol=TCP
```

If the server is configured to accept IPv6 connections, clients can connect to the local server over IPv6 using `--host=::1`. See Section 5.1.13, “IPv6 Support”.

On Windows, to force a MySQL client to use a named-pipe connection, specify the `--pipe` or `--protocol=PIPE` option, or specify `.` (period) as the host name. If the server was not started with the `named_pipe` system variable enabled to support named-pipe connections or if the user making the connection is not a member of the Windows group specified by the `named_pipe_full_access_group` system variable, an error occurs. Use the `--socket` option to specify the name of the pipe if you do not want to use the default pipe name.

Connections to remote servers use TCP/IP. This command connects to the server running on `remote.example.com` using the default port number (3306):

```
mysql --host=remote.example.com
```

To specify a port number explicitly, use the `--port` or `-P` option:

```
mysql --host=remote.example.com --port=13306
```

You can specify a port number for connections to a local server, too. However, as indicated previously, connections to `localhost` on Unix use a socket file by default, so unless you force a TCP/IP connection as previously described, any option that specifies a port number is ignored.

For this command, the program uses a socket file on Unix and the `--port` option is ignored:

```
mysql --port=13306 --host=localhost
```

To cause the port number to be used, force a TCP/IP connection. For example, invoke the program in either of these ways:

```
mysql --port=13306 --host=127.0.0.1
mysql --port=13306 --protocol=TCP
```

For additional information about options that control how client programs establish connections to the server, see Section 4.2.3, “Command Options for Connecting to the Server”.

It is possible to specify connection parameters without entering them on the command line each time you invoke a client program:

- Specify the connection parameters in the `[client]` section of an option file. The relevant section of the file might look like this:

```
[client]
host=host_name
user=user_name
password=password
```

For more information, see Section 4.2.2.2, “Using Option Files”.

- Some connection parameters can be specified using environment variables. Examples:
  - To specify the host for `mysql`, use `MYSQL_HOST`.
  - On Windows, to specify the MySQL user name, use `USER`.

  For a list of supported environment variables, see Section 4.9, “Environment Variables”.

4.2.5 Connecting to the Server Using URI-Like Strings or Key-Value Pairs

This section describes use of URI-like connection strings or key-value pairs to specify how to establish connections to the MySQL server, for clients such as MySQL Shell. For information on establishing connections using command-line options, for clients such as `mysql` or `mysqldump`, see Section 4.2.4, “Connecting to the MySQL Server Using Command Options”. For additional information if you are unable to connect, see Section 6.2.22, “Troubleshooting Problems Connecting to MySQL”.

Note

The term “URI-like” signifies connection-string syntax that is similar to but not identical to the URI (uniform resource identifier) syntax defined by RFC 3986.

The following MySQL clients support connecting to a MySQL server using a URI-like connection string or key-value pairs:

- MySQL Shell
- MySQL Connectors which implement X DevAPI

This section documents all valid URI-like string and key-value pair connection parameters, many of which are similar to those specified with command-line options:

- Parameters specified with a URI-like string use a syntax such as `myuser@example.com:3306/main-schema`. For the full syntax, see Connecting Using URI-Like Connection Strings.
- Parameters specified with key-value pairs use a syntax such as `{user:'myuser', host:'example.com', port:3306, schema:'main-schema'}`. For the full syntax, see Connecting Using Key-Value Pairs.

Connection parameters are not case-sensitive. Each parameter, if specified, can be given only once. If a parameter is specified more than once, an error occurs.

This section covers the following topics:

- Base Connection Parameters
- Additional Connection parameters
- Connecting Using URI-Like Connection Strings
- Connecting Using Key-Value Pairs

Base Connection Parameters

The following discussion describes the parameters available when specifying a connection to MySQL. These parameters can be provided using either a string that conforms to the base URI-like syntax (see Connecting Using URI-Like Connection Strings), or as key-value pairs (see Connecting Using Key-Value Pairs).

- *`scheme`*: The transport protocol to use. Use `mysqlx` for X Protocol connections and `mysql` for classic MySQL protocol connections. If no protocol is specified, the server attempts to guess the

protocol. Connectors that support DNS SRV can use the `mysqlx+srv` scheme (see Connections Using DNS SRV Records).

- *`user`*: The MySQL user account to provide for the authentication process.
- *`password`*: The password to use for the authentication process.

Warning

Specifying an explicit password in the connection specification is insecure and not recommended. Later discussion shows how to cause an interactive prompt for the password to occur.

- *`host`*: The host on which the server instance is running. The value can be a host name, IPv4 address, or IPv6 address. If no host is specified, the default is `localhost`.
- *`port`*: The TCP/IP network port on which the target MySQL server is listening for connections. If no port is specified, the default is 33060 for X Protocol connections and 3306 for classic MySQL protocol connections.
- *`socket`*: The path to a Unix socket file or the name of a Windows named pipe. Values are local file paths. In URI-like strings, they must be encoded, using either percent encoding or by surrounding the path with parentheses. Parentheses eliminate the need to percent encode characters such as the `/` directory separator character. For example, to connect as `root@localhost` using the Unix socket `/tmp/mysql.sock`, specify the path using percent encoding as `root@localhost?socket=%2Ftmp%2Fmysql.sock`, or using parentheses as `root@localhost?socket=(/tmp/mysql.sock)`.
- *`schema`*: The default database for the connection. If no database is specified, the connection has no default database.

The handling of `localhost` on Unix depends on the type of transport protocol. Connections using classic MySQL protocol handle `localhost` the same way as other MySQL clients, which means that `localhost` is assumed to be for socket-based connections. For connections using X Protocol, the behavior of `localhost` differs in that it is assumed to represent the loopback address, for example, IPv4 address 127.0.0.1.

Additional Connection parameters

You can specify options for the connection, either as attributes in a URI-like string by appending *`?attribute=value`*, or as key-value pairs. The following options are available:

- `ssl-mode`: The desired security state for the connection. The following modes are permissible:
  - `DISABLED`
  - `PREFERRED`
  - `REQUIRED`
  - `VERIFY_CA`
  - `VERIFY_IDENTITY`

Important

`VERIFY_CA` and `VERIFY_IDENTITY` are better choices than the default `PREFERRED`, because they help prevent man-in-the-middle attacks.

For information about these modes, see the `--ssl-mode` option description in Command Options for Encrypted Connections.

- `ssl-ca`: The path to the X.509 certificate authority file in PEM format.

- `ssl-capath`: The path to the directory that contains the X.509 certificates authority files in PEM format.
- `ssl-cert`: The path to the X.509 certificate file in PEM format.
- `ssl-cipher`: The encryption cipher to use for connections that use TLS protocols up through TLSv1.2.
- `ssl-crl`: The path to the file that contains certificate revocation lists in PEM format.
- `ssl-crlpath`: The path to the directory that contains certificate revocation-list files in PEM format.
- `ssl-key`: The path to the X.509 key file in PEM format.
- `tls-version`: The TLS protocols permitted for classic MySQL protocol encrypted connections. This option is supported by MySQL Shell only. The value of `tls-version` (singular) is a comma separated list, for example `TLSv1.2,TLSv1.3`. For details, see Section 6.3.2, "Encrypted Connection TLS Protocols and Ciphers". This option depends on the `ssl-mode` option not being set to `DISABLED`.
- `tls-versions`: The permissible TLS protocols for encrypted X Protocol connections. The value of `tls-versions` (plural) is an array such as `[TLSv1.2,TLSv1.3]`. For details, see Section 6.3.2, "Encrypted Connection TLS Protocols and Ciphers". This option depends on the `ssl-mode` option not being set to `DISABLED`.
- `tls-ciphersuites`: The permitted TLS cipher suites. The value of `tls-ciphersuites` is a list of IANA cipher suite names as listed at TLS Ciphersuites. For details, see Section 6.3.2, "Encrypted Connection TLS Protocols and Ciphers". This option depends on the `ssl-mode` option not being set to `DISABLED`.
- `auth-method`: The authentication method to use for the connection. The default is `AUTO`, meaning that the server attempts to guess. The following methods are permissible:
  - `AUTO`
  - `MYSQL41`
  - `SHA256_MEMORY`
  - `FROM_CAPABILITIES`
  - `FALLBACK`
  - `PLAIN`

  For X Protocol connections, any configured `auth-method` is overridden to this sequence of authentication methods: `MYSQL41`, `SHA256_MEMORY`, `PLAIN`.
- `get-server-public-key`: Request from the server the public key required for RSA key pair-based password exchange. Use when connecting to MySQL 8.0 servers over classic MySQL protocol with SSL mode `DISABLED`. You must specify the protocol in this case. For example:

  ```
  mysql://user@localhost:3306?get-server-public-key=true
  ```

  This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `server-public-key-path=`*`file_name`* is given and specifies a valid public key file, it takes precedence over `get-server-public-key`.

For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `server-public-key-path`: The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. Use when connecting to MySQL 8.0 servers over classic MySQL protocol with SSL mode `DISABLED`.

  This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `server-public-key-path=`*`file_name`* is given and specifies a valid public key file, it takes precedence over `get-server-public-key`.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `ssh`: The URI for connection to an SSH server to access a MySQL server instance using SSH tunneling. The URI format is `[user@]host[:port]`. Use the `uri` option to specify the URI of the target MySQL server instance. For information on SSH tunnel connections from MySQL Shell, see Using an SSH Tunnel.

- `uri`: The URI for a MySQL server instance that is to be accessed through an SSH tunnel from the server specified by the `ssh` option. The URI format is `[scheme://][user@]host[:port]`. Do not use the base connection parameters (`scheme`, `user`, `host`, `port`) to specify the MySQL server connection for SSH tunneling, just use the `uri` option.

- `ssh-password`: The password for the connection to the SSH server.

  

  Warning

  Specifying an explicit password in the connection specification is insecure and not recommended. MySQL Shell prompts for a password interactively when one is required.

- `ssh-config-file`: The SSH configuration file for the connection to the SSH server. You can use the MySQL Shell configuration option `ssh.configFile` to set a custom file as the default if this option is not specified. If `ssh.configFile` has not been set, the default is the standard SSH configuration file `~/.ssh/config`.

- `ssh-identity-file`: The identity file to use for the connection to the SSH server. The default if this option is not specified is any identity file configured in an SSH agent (if used), or in the SSH configuration file, or the standard private key file in the SSH configuration folder (`~/.ssh/id_rsa`).

- `ssh-identity-pass`: The passphrase for the identity file specified by the `ssh-identity-file` option.

  

  Warning

  Specifying an explicit password in the connection specification is insecure and not recommended. MySQL Shell prompts for a password interactively when one is required.

- `connect-timeout`: An integer value used to configure the number of seconds that clients, such as MySQL Shell, wait until they stop trying to connect to an unresponsive MySQL server.

- `compression`: This option requests or disables compression for the connection. Up to MySQL 8.0.19 it operates for classic MySQL protocol connections only, and from MySQL 8.0.20 it also operates for X Protocol connections.

  - Up to MySQL 8.0.19, the values for this option are `true` (or 1) which enables compression, and the default `false` (or 0) which disables compression.
  - From MySQL 8.0.20, the values for this option are `required`, which requests compression and fails if the server does not support it; `preferred`, which requests compression and falls back to an uncompressed connection; and `disabled`, which requests an uncompressed connection and fails if the server does not permit those. `preferred` is the default for X Protocol connections, and `disabled` is the default for classic MySQL protocol connections. For information on X Plugin connection compression control, see Section 20.5.5, “Connection Compression with X Plugin”.

  Note that different MySQL clients implement their support for connection compression differently. Consult your client's documentation for details.

- `compression-algorithms` and `compression-level`: These options are available in MySQL Shell 8.0.20 and later for more control over connection compression. You can specify them to select the compression algorithm used for the connection, and the numeric compression level used with that algorithm. You can also use `compression-algorithms` in place of `compression` to request compression for the connection. For information on MySQL Shell's connection compression control, see Using Compressed Connections.

- `connection-attributes`: Controls the key-value pairs that application programs pass to the server at connect time. For general information about connection attributes, see Section 27.12.9, “Performance Schema Connection Attribute Tables”. Clients usually define a default set of attributes, which can be disabled or enabled. For example:

  ```
  mysqlx://user@host?connection-attributes
  mysqlx://user@host?connection-attributes=true
  mysqlx://user@host?connection-attributes=false
  ```

  The default behavior is to send the default attribute set. Applications can specify attributes to be passed in addition to the default attributes. You specify additional connection attributes as a `connection-attributes` parameter in a connection string. The `connection-attributes` parameter value must be empty (the same as specifying `true`), a `Boolean` value (`true` or `false` to enable or disable the default attribute set), or a list or zero or more `key=value` specifiers separated by commas (to be sent in addition to the default attribute set). Within a list, a missing key value evaluates as an empty string. Further examples:

  ```
  mysqlx://user@host?connection-attributes=[attr1=val1,attr2,attr3=]
  mysqlx://user@host?connection-attributes=[]
  ```

  Application-defined attribute names cannot begin with `_` because such names are reserved for internal attributes.

Connecting Using URI-Like Connection Strings

You can specify a connection to MySQL Server using a URI-like string. Such strings can be used with the MySQL Shell with the `--uri` command option, the MySQL Shell `\connect` command, and MySQL Connectors which implement X DevAPI.

Note

The term “URI-like” signifies connection-string syntax that is similar to but not identical to the URI (uniform resource identifier) syntax defined by RFC 3986.

A URI-like connection string has the following syntax:

```
[scheme://][user[:[password]]@]host[:port][/schema][?attribute1=value1&attribute2=value2...
```

Important

Percent encoding must be used for reserved characters in the elements of the URI-like string. For example, if you specify a string that includes the `@` character,

> the character must be replaced by `%40`. If you include a zone ID in an IPv6 address, the `%` character used as the separator must be replaced with `%25`.

The parameters you can use in a URI-like connection string are described at Base Connection Parameters.

MySQL Shell's `shell.parseUri()` and `shell.unparseUri()` methods can be used to deconstruct and assemble a URI-like connection string. Given a URI-like connection string, `shell.parseUri()` returns a dictionary containing each element found in the string. `shell.unparseUri()` converts a dictionary of URI components and connection options into a valid URI-like connection string for connecting to MySQL, which can be used in MySQL Shell or by MySQL Connectors which implement X DevAPI.

If no password is specified in the URI-like string, which is recommended, interactive clients prompt for the password. The following examples show how to specify URI-like strings with the user name *`user_name`*. In each case, the password is prompted for.

- An X Protocol connection to a local server instance listening at port 33065.

```
mysqlx://user_name@localhost:33065
```

- A classic MySQL protocol connection to a local server instance listening at port 3333.

```
mysql://user_name@localhost:3333
```

- An X Protocol connection to a remote server instance, using a host name, an IPv4 address, and an IPv6 address.

```
mysqlx://user_name@server.example.com/
mysqlx://user_name@198.51.100.14:123
mysqlx://user_name@[2001:db8:85a3:8d3:1319:8a2e:370:7348]
```

- An X Protocol connection using a socket, with the path provided using either percent encoding or parentheses.

```
mysqlx://user_name@/path%2Fto%2Fsocket.sock
mysqlx://user_name@(/path/to/socket.sock)
```

- An optional path can be specified, which represents a database.

```
# use 'world' as the default database
mysqlx://user_name@198.51.100.1/world

# use 'world_x' as the default database, encoding _ as %5F
mysqlx://user_name@198.51.100.2:33060/world%5Fx
```

- An optional query can be specified, consisting of values each given as a *`key=value`* pair or as a single *`key`*. To specify multiple values, separate them by `,` characters. A mix of *`key=value`* and *`key`* values is permissible. Values can be of type list, with list values ordered by appearance. Strings must be either percent encoded or surrounded by parentheses. The following are equivalent.

```
ssluser@127.0.0.1?ssl-ca=%2Froot%2Fclientcert%2Fca-cert.pem\
&ssl-cert=%2Froot%2Fclientcert%2Fclient-cert.pem\
&ssl-key=%2Froot%2Fclientcert%2Fclient-key

ssluser@127.0.0.1?ssl-ca=(/root/clientcert/ca-cert.pem)\
&ssl-cert=(/root/clientcert/client-cert.pem)\
&ssl-key=(/root/clientcert/client-key)
```

- To specify a TLS version and ciphersuite to use for encrypted connections:

```
mysql://user_name@198.51.100.2:3306/world%5Fx?\
tls-versions=[TLSv1.2,TLSv1.3]&tls-ciphersuites=[TLS_DHE_PSK_WITH_AES_128_\
GCM_SHA256, TLS_CHACHA20_POLY1305_SHA256]
```

The previous examples assume that connections require a password. With interactive clients, the specified user's password is requested at the login prompt. If the user account has no password (which

is insecure and not recommended), or if socket peer-credential authentication is in use (for example, with Unix socket connections), you must explicitly specify in the connection string that no password is being provided and the password prompt is not required. To do this, place a `:` after the *`user_name`* in the string but do not specify a password after it. For example:

```
mysqlx://user_name:@localhost
```

Connecting Using Key-Value Pairs

In MySQL Shell and some MySQL Connectors which implement X DevAPI, you can specify a connection to MySQL Server using key-value pairs, supplied in language-natural constructs for the implementation. For example, you can supply connection parameters using key-value pairs as a JSON object in JavaScript, or as a dictionary in Python. Regardless of the way the key-value pairs are supplied, the concept remains the same: the keys as described in this section can be assigned values that are used to specify a connection. You can specify connections using key-value pairs in MySQL Shell's `shell.connect()` method or InnoDB Cluster's `dba.createCluster()` method, and with some of the MySQL Connectors which implement X DevAPI.

Generally, key-value pairs are surrounded by `{` and `}` characters and the `,` character is used as a separator between key-value pairs. The `:` character is used between keys and values, and strings must be delimited (for example, using the `'` character). It is not necessary to percent encode strings, unlike URI-like connection strings.

A connection specified as key-value pairs has the following format:

```
{ key: value, key: value, ...}
```

The parameters you can use as keys for a connection are described at Base Connection Parameters.

If no password is specified in the key-value pairs, which is recommended, interactive clients prompt for the password. The following examples show how to specify connections using key-value pairs with the user name `'user_name'`. In each case, the password is prompted for.

- An X Protocol connection to a local server instance listening at port 33065.

  ```
  {user:'user_name', host:'localhost', port:33065}
  ```

- A classic MySQL protocol connection to a local server instance listening at port 3333.

  ```
  {user:'user_name', host:'localhost', port:3333}
  ```

- An X Protocol connection to a remote server instance, using a host name, an IPv4 address, and an IPv6 address.

  ```
  {user:'user_name', host:'server.example.com'}
  {user:'user_name', host:198.51.100.14:123}
  {user:'user_name', host:[2001:db8:85a3:8d3:1319:8a2e:370:7348]}
  ```

- An X Protocol connection using a socket.

  ```
  {user:'user_name', socket:'/path/to/socket/file'}
  ```

- An optional schema can be specified, which represents a database.

  ```
  {user:'user_name', host:'localhost', schema:'world'}
  ```

The previous examples assume that connections require a password. With interactive clients, the specified user's password is requested at the login prompt. If the user account has no password (which is insecure and not recommended), or if socket peer-credential authentication is in use (for example, with Unix socket connections), you must explicitly specify that no password is being provided and the password prompt is not required. To do this, provide an empty string using `''` after the `password` key. For example:

```
{user:'user_name', password:'', host:'localhost'}
```

4.2.6 Connecting to the Server Using DNS SRV Records

In the Domain Name System (DNS), a SRV record (service location record) is a type of resource record that enables a client to specify a name that indicates a service, protocol, and domain. A DNS lookup on the name returns a reply containing the names of multiple available servers in the domain that provide the required service. For information about DNS SRV, including how a record defines the preference order of the listed servers, see RFC 2782.

MySQL supports the use of DNS SRV records for connecting to servers. A client that receives a DNS SRV lookup result attempts to connect to the MySQL server on each of the listed hosts in order of preference, based on the priority and weighting assigned to each host by the DNS administrator. A failure to connect occurs only if the client cannot connect to any of the servers.

When multiple MySQL instances, such as a cluster of servers, provide the same service for your applications, DNS SRV records can be used to assist with failover, load balancing, and replication services. It is cumbersome for applications to directly manage the set of candidate servers for connection attempts, and DNS SRV records provide an alternative:

- DNS SRV records enable a DNS administrator to map a single DNS domain to multiple servers. DNS SRV records also can be updated centrally by administrators when servers are added or removed from the configuration or when their host names are changed.
- Central management of DNS SRV records eliminates the need for individual clients to identify each possible host in connection requests, or for connections to be handled by an additional software component. An application can use the DNS SRV record to obtain information about candidate MySQL servers, instead of managing the server information itself.
- DNS SRV records can be used in combination with connection pooling, in which case connections to hosts that are no longer in the current list of DNS SRV records are removed from the pool when they become idle.

MySQL supports use of DNS SRV records to connect to servers in these contexts:

- Several MySQL Connectors implement DNS SRV support; connector-specific options enable requesting DNS SRV record lookup both for X Protocol connections and for classic MySQL protocol connections. For general information, see Connections Using DNS SRV Records. For details, see the documentation for individual MySQL Connectors.
- The C API provides a `mysql_real_connect_dns_srv()` function that is similar to `mysql_real_connect()`, except that the argument list does not specify the particular host of the MySQL server to connect to. Instead, it names a DNS SRV record that specifies a group of servers. See mysql_real_connect_dns_srv().
- The `mysql` client has a `--dns-srv-name` option to indicate a DNS SRV record that specifies a group of servers. See Section 4.5.1, “mysql — The MySQL Command-Line Client”.

A DNS SRV name consists of a service, protocol, and domain, with the service and protocol each prefixed by an underscore:

```
_service._protocol.domain
```

The following DNS SRV record identifies multiple candidate servers, such as might be used by clients for establishing X Protocol connections:

```
Name                     TTL   Class  Priority Weight Port  Target
_mysqlx._tcp.example.com. 86400 IN SRV 0        5      33060 server1.example.com.
_mysqlx._tcp.example.com. 86400 IN SRV 0        10     33060 server2.example.com.
_mysqlx._tcp.example.com. 86400 IN SRV 10       5      33060 server3.example.com.
_mysqlx._tcp.example.com. 86400 IN SRV 20       5      33060 server4.example.com.
```

Here, `mysqlx` indicates the X Protocol service and `tcp` indicates the TCP protocol. A client can request this DNS SRV record using the name `_mysqlx._tcp.example.com`. The particular syntax

for specifying the name in the connection request depends on the type of client. For example, a client might support specifying the name within a URI-like connection string or as a key-value pair.

A DNS SRV record for classic protocol connections might look like this:

```
Name                     TTL   Class  Priority Weight  Port Target
_mysql._tcp.example.com. 86400 IN SRV 0        5       3306 server1.example.com.
_mysql._tcp.example.com. 86400 IN SRV 0        10      3306 server2.example.com.
_mysql._tcp.example.com. 86400 IN SRV 10       5       3306 server3.example.com.
_mysql._tcp.example.com. 86400 IN SRV 20       5       3306 server4.example.com.
```

Here, the name `mysql` designates the classic MySQL protocol service, and the port is 3306 (the default classic MySQL protocol port) rather than 33060 (the default X Protocol port).

When DNS SRV record lookup is used, clients generally must apply these rules for connection requests (there may be client- or connector-specific exceptions):

- The request must specify the full DNS SRV record name, with the service and protocol names prefixed by underscores.
- The request must not specify multiple host names.
- The request must not specify a port number.
- Only TCP connections are supported. Unix socket files, Windows named pipes, and shared memory cannot be used.

For more information on using DNS SRV based connections in X DevAPI, see Connections Using DNS SRV Records.

4.2.7 Connection Transport Protocols

For programs that use the MySQL client library (for example, `mysql` and `mysqldump`), MySQL supports connections to the server based on several transport protocols: TCP/IP, Unix socket file, named pipe, and shared memory. This section describes how to select these protocols, and how they are similar and different.

- Transport Protocol Selection
- Transport Support for Local and Remote Connections
- Interpretation of localhost
- Encryption and Security Characteristics
- Connection Compression

Transport Protocol Selection

For a given connection, if the transport protocol is not specified explicitly, it is determined implicitly. For example, connections to `localhost` result in a socket file connection on Unix and Unix-like systems, and a TCP/IP connection to `127.0.0.1` otherwise. For additional information, see Section 4.2.4, “Connecting to the MySQL Server Using Command Options”.

To specify the protocol explicitly, use the `--protocol` command option. The following table shows the permissible values for `--protocol` and indicates the applicable platforms for each value. The values are not case-sensitive.

`--protocol` Value	Transport Protocol Used	Applicable Platforms
`TCP`	TCP/IP	All
`SOCKET`	Unix socket file	Unix and Unix-like systems
`PIPE`	Named pipe	Windows

`--protocol` Value	Transport Protocol Used	Applicable Platforms
`MEMORY`	Shared memory	Windows

Transport Support for Local and Remote Connections

TCP/IP transport supports connections to local or remote MySQL servers.

Socket-file, named-pipe, and shared-memory transports support connections only to local MySQL servers. (Named-pipe transport does allow for remote connections, but this capability is not implemented in MySQL.)

Interpretation of localhost

If the transport protocol is not specified explicitly, `localhost` is interpreted as follows:

- On Unix and Unix-like systems, a connection to `localhost` results in a socket-file connection.
- Otherwise, a connection to `localhost` results in a TCP/IP connection to `127.0.0.1`.

If the transport protocol is specified explicitly, `localhost` is interpreted with respect to that protocol. For example, with `--protocol=TCP`, a connection to `localhost` results in a TCP/IP connection to `127.0.0.1` on all platforms.

Encryption and Security Characteristics

TCP/IP and socket-file transports are subject to TLS/SSL encryption, using the options described in Command Options for Encrypted Connections. Named-pipe and shared-memory transports are not subject to TLS/SSL encryption.

A connection is secure by default if made over a transport protocol that is secure by default. Otherwise, for protocols that are subject to TLS/SSL encryption, a connection may be made secure using encryption:

- TCP/IP connections are not secure by default, but can be encrypted to make them secure.
- Socket-file connections are secure by default. They can also be encrypted, but encrypting a socket-file connection makes it no more secure and increases CPU load.
- Named-pipe connections are not secure by default, and are not subject to encryption to make them secure. However, the `named_pipe_full_access_group` system variable is available to control which MySQL users are permitted to use named-pipe connections.
- Shared-memory connections are secure by default.

If the `require_secure_transport` system variable is enabled, the server permits only connections that use some form of secure transport. Per the preceding remarks, connections that use TCP/IP encrypted using TLS/SSL, a socket file, or shared memory are secure connections. TCP/IP connections not encrypted using TLS/SSL and named-pipe connections are not secure.

See also Configuring Encrypted Connections as Mandatory.

Connection Compression

All transport protocols are subject to use of compression on the traffic between the client and server. If both compression and encryption are used for a given connection, compression occurs before encryption. For more information, see Section 4.2.8, “Connection Compression Control”.

4.2.8 Connection Compression Control

Connections to the server can use compression on the traffic between client and server to reduce the number of bytes sent over the connection. By default, connections are uncompressed, but can be compressed if the server and the client agree on a mutually permitted compression algorithm.

Compressed connections originate on the client side but affect CPU load on both the client and server sides because both sides perform compression and decompression operations. Because enabling compression decreases performance, its benefits occur primarily when there is low network bandwidth, network transfer time dominates the cost of compression and decompression operations, and result sets are large.

This section describes the available compression-control configuration parameters and the information sources available for monitoring use of compression. It applies to classic MySQL protocol connections.

Compression control applies to connections to the server by client programs and by servers participating in source/replica replication or Group Replication. Compression control does not apply to connections for `FEDERATED` tables. In the following discussion, “client connection” is shorthand for a connection to the server originating from any source for which compression is supported, unless context indicates a specific connection type.

Note

X Protocol connections to a MySQL Server instance support compression from MySQL 8.0.19, but compression for X Protocol connections operates independently from the compression for classic MySQL protocol connections described here, and is controlled separately. See Section 20.5.5, “Connection Compression with X Plugin” for information on X Protocol connection compression.

- Configuring Connection Compression
- Configuring Legacy Connection Compression
- Monitoring Connection Compression

Configuring Connection Compression

As of MySQL 8.0.18, these configuration parameters are available for controlling connection compression:

- The `protocol_compression_algorithms` system variable configures which compression algorithms the server permits for incoming connections.
- The `--compression-algorithms` and `--zstd-compression-level` command-line options configure permitted compression algorithms and `zstd` compression level for these client programs: `mysql`, `mysqladmin`, `mysqlbinlog`, `mysqlcheck`, `mysqldump`, `mysqlimport`, `mysqlpump`, `mysqlshow`, `mysqlslap`, and `mysqltest`, and `mysql_upgrade`. MySQL Shell also offers these command-line options from its 8.0.20 release.
- The `MYSQL_OPT_COMPRESSION_ALGORITHMS` and `MYSQL_OPT_ZSTD_COMPRESSION_LEVEL` options for the `mysql_options()` function configure permitted compression algorithms and `zstd` compression level for client programs that use the MySQL C API.
- The `MASTER_COMPRESSION_ALGORITHMS` and `MASTER_ZSTD_COMPRESSION_LEVEL` options for the `CHANGE MASTER TO` statement configure permitted compression algorithms and `zstd` compression level for replica servers participating in source/replica replication. From MySQL 8.0.23, use the statement `CHANGE REPLICATION SOURCE TO` and the options `SOURCE_COMPRESSION_ALGORITHMS` and `SOURCE_ZSTD_COMPRESSION_LEVEL` instead.
- The `group_replication_recovery_compression_algorithms` and `group_replication_recovery_zstd_compression_level` system variables configure permitted compression algorithms and `zstd` compression level for Group Replication recovery connections when a new member joins a group and connects to a donor.

Configuration parameters that enable specifying compression algorithms are string-valued and take a list of one or more comma-separated compression algorithm names, in any order, chosen from the following items (not case-sensitive):

- `zlib`: Permit connections that use the `zlib` compression algorithm.
- `zstd`: Permit connections that use the `zstd` compression algorithm (zstd 1.3).
- `uncompressed`: Permit uncompressed connections.

Note

Because `uncompressed` is an algorithm name that may or may not be configured, it is possible to configure MySQL *not* to permit uncompressed connections.

Examples:

- To configure which compression algorithms the server permits for incoming connections, set the `protocol_compression_algorithms` system variable. By default, the server permits all available algorithms. To configure that setting explicitly at startup, use these lines in the server `my.cnf` file:

  ```
  [mysqld]
  protocol_compression_algorithms=zlib,zstd,uncompressed
  ```

  To set and persist the `protocol_compression_algorithms` system variable to that value at runtime, use this statement:

  ```
  SET PERSIST protocol_compression_algorithms='zlib,zstd,uncompressed';
  ```

  `SET PERSIST` sets a value for the running MySQL instance. It also saves the value, causing it to carry over to subsequent server restarts. To change the value for the running MySQL instance without having it carry over to subsequent restarts, use the `GLOBAL` keyword rather than `PERSIST`. See Section 13.7.6.1, “SET Syntax for Variable Assignment”.

- To permit only incoming connections that use `zstd` compression, configure the server at startup like this:

  ```
  [mysqld]
  protocol_compression_algorithms=zstd
  ```

  Or, to make the change at runtime:

  ```
  SET PERSIST protocol_compression_algorithms='zstd';
  ```

- To permit the `mysql` client to initiate `zlib` or `uncompressed` connections, invoke it like this:

  ```
  mysql --compression-algorithms=zlib,uncompressed
  ```

- To configure replicas to connect to the source using `zlib` or `zstd` connections, with a compression level of 7 for `zstd` connections, use a `CHANGE REPLICATION SOURCE TO` statement (from MySQL 8.0.23) or `CHANGE MASTER TO` statement (before MySQL 8.0.23):

  ```
  CHANGE REPLICATION SOURCE TO
    SOURCE_COMPRESSION_ALGORITHMS = 'zlib,zstd',
    SOURCE_ZSTD_COMPRESSION_LEVEL = 7;
  ```

  This assumes that the `replica_compressed_protocol` or `slave_compressed_protocol` system variable is disabled, for reasons described in Configuring Legacy Connection Compression.

For successful connection setup, both sides of the connection must agree on a mutually permitted compression algorithm. The algorithm-negotiation process attempts to use `zlib`, then `zstd`, then `uncompressed`. If the two sides can find no common algorithm, the connection attempt fails.

Because both sides must agree on the compression algorithm, and because `uncompressed` is an algorithm value that is not necessarily permitted, fallback to an uncompressed connection does not necessarily occur. For example, if the server is configured to permit `zstd` and a client is configured to

permit `zlib,uncompressed`, the client cannot connect at all. In this case, no algorithm is common to both sides, so connection attempts fail.

Configuration parameters that enable specifying the `zstd` compression level take an integer value from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

A configurable `zstd` compression level enables choosing between less network traffic and higher CPU load versus more network traffic and lower CPU load. Higher compression levels reduce network congestion but the additional CPU load may reduce server performance.

Configuring Legacy Connection Compression

Prior to MySQL 8.0.18, these configuration parameters are available for controlling connection compression:

- Client programs support a `--compress` command-line option to specify use of compression for the connection to the server.
- For programs that use the MySQL C API, enabling the `MYSQL_OPT_COMPRESS` option for the `mysql_options()` function specifies use of compression for the connection to the server.
- For source/replica replication, enabling the system variable `replica_compressed_protocol` (from MySQL 8.0.26) or `slave_compressed_protocol` (before MySQL 8.0.26) specifies use of compression for replica connections to the source.

In each case, when use of compression is specified, the connection uses the `zlib` compression algorithm if both sides permit it, with fallback to an uncompressed connection otherwise.

As of MySQL 8.0.18, the compression parameters just described become legacy parameters, due to the additional compression parameters introduced for more control over connection compression that are described in Configuring Connection Compression. An exception is MySQL Shell, where the `--compress` command-line option remains current, and can be used to request compression without selecting compression algorithms. For information on MySQL Shell's connection compression control, see Using Compressed Connections.

The legacy compression parameters interact with the newer parameters and their semantics change as follows:

- The meaning of the legacy `--compress` option depends on whether `--compression-algorithms` is specified:
  - When `--compression-algorithms` is not specified, `--compress` is equivalent to specifying a client-side algorithm set of `zlib,uncompressed`.
  - When `--compression-algorithms` is specified, `--compress` is equivalent to specifying an algorithm set of `zlib` and the full client-side algorithm set is the union of `zlib` plus the algorithms specified by `--compression-algorithms`. For example, with both `--compress` and `--compression-algorithms=zlib,zstd`, the permitted-algorithm set is `zlib` plus `zlib,zstd`; that is, `zlib,zstd`. With both `--compress` and `--compression-algorithms=zstd,uncompressed`, the permitted-algorithm set is `zlib` plus `zstd,uncompressed`; that is, `zlib,zstd,uncompressed`.
- The same type of interaction occurs between the legacy `MYSQL_OPT_COMPRESS` option and the `MYSQL_OPT_COMPRESSION_ALGORITHMS` option for the `mysql_options()` C API function.
- If the `replica_compressed_protocol` or `slave_compressed_protocol` system variable is enabled, it takes precedence over `MASTER_COMPRESSION_ALGORITHMS` and connections to the source use `zlib` compression if both source and replica permit that algorithm. If

`replica_compressed_protocol` or `slave_compressed_protocol` is disabled, the value of `MASTER_COMPRESSION_ALGORITHMS` applies.

Note

The legacy compression-control parameters are deprecated as of MySQL 8.0.18; expect it to be removed in a future version of MySQL.

Monitoring Connection Compression

The `Compression` status variable is `ON` or `OFF` to indicate whether the current connection uses compression.

The `mysql` client `\status` command displays a line that says `Protocol: Compressed` if compression is enabled for the current connection. If that line is not present, the connection is uncompressed.

As of 8.0.14, the MySQL Shell `\status` command displays a `Compression:` line that says `Disabled` or `Enabled` to indicate whether the connection is compressed.

As of MySQL 8.0.18, these additional sources of information are available for monitoring connection compression:

- To monitor compression in use for client connections, use the `Compression_algorithm` and `Compression_level` status variables. For the current connection, their values indicate the compression algorithm and compression level, respectively.
- To determine which compression algorithms the server is configured to permit for incoming connections, check the `protocol_compression_algorithms` system variable.
- For source/replica replication connections, the configured compression algorithms and compression level are available from multiple sources:
  - The Performance Schema `replication_connection_configuration` table has `COMPRESSION_ALGORITHMS` and `ZSTD_COMPRESSION_LEVEL` columns.
  - The `mysql.slave_master_info` system table has `Master_compression_algorithms` and `Master_zstd_compression_level` columns. If the `master.info` file exists, it contains lines for those values as well.

4.2.9 Setting Environment Variables

Environment variables can be set at the command prompt to affect the current invocation of your command processor, or set permanently to affect future invocations. To set a variable permanently, you can set it in a startup file or by using the interface provided by your system for this purpose. Consult the documentation for your command interpreter for specific details. Section 4.9, “Environment Variables”, lists all environment variables that affect MySQL program operation.

To specify a value for an environment variable, use the syntax appropriate for your command processor. For example, on Windows, you can set the `USER` variable to specify your MySQL account name. To do so, use this syntax:

```
SET USER=your_name
```

The syntax on Unix depends on your shell. Suppose that you want to specify the TCP/IP port number using the `MYSQL_TCP_PORT` variable. Typical syntax (such as for `sh`, `ksh`, `bash`, `zsh`, and so on) is as follows:

```
MYSQL_TCP_PORT=3306
export MYSQL_TCP_PORT
```

The first command sets the variable, and the `export` command exports the variable to the shell environment so that its value becomes accessible to MySQL and other processes.

For csh and tcsh, use setenv to make the shell variable available to the environment:

```
setenv MYSQL_TCP_PORT 3306
```

The commands to set environment variables can be executed at your command prompt to take effect immediately, but the settings persist only until you log out. To have the settings take effect each time you log in, use the interface provided by your system or place the appropriate command or commands in a startup file that your command interpreter reads each time it starts.

On Windows, you can set environment variables using the System Control Panel (under Advanced).

On Unix, typical shell startup files are .bashrc or .bash_profile for bash, or .tcshrc for tcsh.

Suppose that your MySQL programs are installed in /usr/local/mysql/bin and that you want to make it easy to invoke these programs. To do this, set the value of the PATH environment variable to include that directory. For example, if your shell is bash, add the following line to your .bashrc file:

```
PATH=${PATH}:/usr/local/mysql/bin
```

bash uses different startup files for login and nonlogin shells, so you might want to add the setting to .bashrc for login shells and to .bash_profile for nonlogin shells to make sure that PATH is set regardless.

If your shell is tcsh, add the following line to your .tcshrc file:

```
setenv PATH ${PATH}:/usr/local/mysql/bin
```

If the appropriate startup file does not exist in your home directory, create it with a text editor.

After modifying your PATH setting, open a new console window on Windows or log in again on Unix so that the setting goes into effect.

4.3 Server and Server-Startup Programs

This section describes mysqld, the MySQL server, and several programs that are used to start the server.

4.3.1 mysqld — The MySQL Server

mysqld, also known as MySQL Server, is a single multithreaded program that does most of the work in a MySQL installation. It does not spawn additional processes. MySQL Server manages access to the MySQL data directory that contains databases and tables. The data directory is also the default location for other information such as log files and status files.

Note

Some installation packages contain a debugging version of the server named mysqld-debug. Invoke this version instead of mysqld for debugging support, memory allocation checking, and trace file support (see Section 5.9.1.2, “Creating Trace Files”).

When MySQL server starts, it listens for network connections from client programs and manages access to databases on behalf of those clients.

The mysqld program has many options that can be specified at startup. For a complete list of options, run this command:

```
mysqld --verbose --help
```

MySQL Server also has a set of system variables that affect its operation as it runs. System variables can be set at server startup, and many of them can be changed at runtime to effect dynamic server

reconfiguration. MySQL Server also has a set of status variables that provide information about its operation. You can monitor these status variables to access runtime performance characteristics.

For a full description of MySQL Server command options, system variables, and status variables, see Section 5.1, “The MySQL Server”. For information about installing MySQL and setting up the initial configuration, see Chapter 2, *Installing and Upgrading MySQL*.

4.3.2 mysqld_safe — MySQL Server Startup Script

`mysqld_safe` is the recommended way to start a `mysqld` server on Unix. `mysqld_safe` adds some safety features such as restarting the server when an error occurs and logging runtime information to an error log. A description of error logging is given later in this section.

Note

For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, `mysqld_safe` is not installed because it is unnecessary. For more information, see Section 2.5.9, “Managing MySQL Server with systemd”.

One implication of the non-use of `mysqld_safe` on platforms that use systemd for server management is that use of `[mysqld_safe]` or `[safe_mysqld]` sections in option files is not supported and might lead to unexpected behavior.

`mysqld_safe` tries to start an executable named `mysqld`. To override the default behavior and specify explicitly the name of the server you want to run, specify a `--mysqld` or `--mysqld-version` option to `mysqld_safe`. You can also use `--ledir` to indicate the directory where `mysqld_safe` should look for the server.

Many of the options to `mysqld_safe` are the same as the options to `mysqld`. See Section 5.1.7, “Server Command Options”.

Options unknown to `mysqld_safe` are passed to `mysqld` if they are specified on the command line, but ignored if they are specified in the `[mysqld_safe]` group of an option file. See Section 4.2.2.2, “Using Option Files”.

`mysqld_safe` reads all options from the `[mysqld]`, `[server]`, and `[mysqld_safe]` sections in option files. For example, if you specify a `[mysqld]` section like this, `mysqld_safe` finds and uses the `--log-error` option:

```
[mysqld]
log-error=error.log
```

For backward compatibility, `mysqld_safe` also reads `[safe_mysqld]` sections, but to be current you should rename such sections to `[mysqld_safe]`.

`mysqld_safe` accepts options on the command line and in option files, as described in the following table. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.6 mysqld_safe Options

Option Name	Description
--basedir	Path to MySQL installation directory
--core-file-size	Size of core file that mysqld should be able to create
--datadir	Path to data directory
--defaults-extra-file	Read named option file in addition to usual option files

Option Name	Description
--defaults-file	Read only named option file
--help	Display help message and exit
--ledir	Path to directory where server is located
--log-error	Write error log to named file
--malloc-lib	Alternative malloc library to use for mysqld
--mysqld	Name of server program to start (in ledir directory)
--mysqld-safe-log-timestamps	Timestamp format for logging
--mysqld-version	Suffix for server program name
--nice	Use nice program to set server scheduling priority
--no-defaults	Read no option files
--open-files-limit	Number of files that mysqld should be able to open
--pid-file	Path name of server process ID file
--plugin-dir	Directory where plugins are installed
--port	Port number on which to listen for TCP/IP connections
--skip-kill-mysqld	Do not try to kill stray mysqld processes
--skip-syslog	Do not write error messages to syslog; use error log file
--socket	Socket file on which to listen for Unix socket connections
--syslog	Write error messages to syslog
--syslog-tag	Tag suffix for messages written to syslog
--timezone	Set TZ time zone environment variable to named value
--user	Run mysqld as user having name user_name or numeric user ID user_id

- `--help`

  Display a help message and exit.

- `--basedir=dir_name`

  The path to the MySQL installation directory.

- `--core-file-size=size`

  The size of the core file that `mysqld` should be able to create. The option value is passed to `ulimit -c`.

  

  Note

  The `innodb_buffer_pool_in_core_file` variable can be used to reduce the size of core files on operating systems that support it. For more information, see Section 15.8.3.7, “Excluding Buffer Pool Pages from Core Files”.

- `--datadir=dir_name`

  The path to the data directory.

- `--defaults-extra-file=file_name`

  Read this option file in addition to the usual option files. If the file does not exist or is otherwise inaccessible, the server exits with an error. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory. This must be the first option on the command line if it is used.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, the server exits with an error. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory. This must be the first option on the command line if it is used.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--ledir=dir_name`

  If `mysqld_safe` cannot find the server, use this option to indicate the path name to the directory where the server is located.

  This option is accepted only on the command line, not in option files. On platforms that use systemd, the value can be specified in the value of `MYSQLD_OPTS`. See Section 2.5.9, “Managing MySQL Server with systemd”.

- `--log-error=file_name`

  Write the error log to the given file. See Section 5.4.2, “The Error Log”.

- `--mysqld-safe-log-timestamps`

  This option controls the format for timestamps in log output produced by `mysqld_safe`. The following list describes the permitted values. For any other value, `mysqld_safe` logs a warning and uses `UTC` format.

  - `UTC`, `utc`

    ISO 8601 UTC format (same as `--log_timestamps=UTC` for the server). This is the default.

  - `SYSTEM`, `system`

    ISO 8601 local time format (same as `--log_timestamps=SYSTEM` for the server).

  - `HYPHEN`, `hyphen`

    `YY-MM-DD h:mm:ss` format, as in `mysqld_safe` for MySQL 5.6.

  - `LEGACY`, `legacy`

    `YYMMDD hh:mm:ss` format, as in `mysqld_safe` prior to MySQL 5.6.

- `--malloc-lib=[lib_name]`

  The name of the library to use for memory allocation instead of the system `malloc()` library. The option value must be one of the directories `/usr/lib`, `/usr/lib64`, `/usr/lib/i386-linux-gnu`, or `/usr/lib/x86_64-linux-gnu`.

  The `--malloc-lib` option works by modifying the `LD_PRELOAD` environment value to affect dynamic linking to enable the loader to find the memory-allocation library when `mysqld` runs:

- If the option is not given, or is given without a value (`--malloc-lib=`), `LD_PRELOAD` is not modified and no attempt is made to use `tcmalloc`.
- Prior to MySQL 8.0.21, if the option is given as `--malloc-lib=tcmalloc`, `mysqld_safe` looks for a `tcmalloc` library in `/usr/lib`. If `tmalloc` is found, its path name is added to the beginning of the `LD_PRELOAD` value for `mysqld`. If `tcmalloc` is not found, `mysqld_safe` aborts with an error.

  As of MySQL 8.0.21, `tcmalloc` is not a permitted value for the `--malloc-lib` option.
- If the option is given as `--malloc-lib=/path/to/some/library`, that full path is added to the beginning of the `LD_PRELOAD` value. If the full path points to a nonexistent or unreadable file, `mysqld_safe` aborts with an error.
- For cases where `mysqld_safe` adds a path name to `LD_PRELOAD`, it adds the path to the beginning of any existing value the variable already has.

Note

On systems that manage the server using systemd, `mysqld_safe` is not available. Instead, specify the allocation library by setting `LD_PRELOAD` in `/etc/sysconfig/mysql`.

Linux users can use the `libtcmalloc_minimal.so` library on any platform for which a `tcmalloc` package is installed in `/usr/lib` by adding these lines to the `my.cnf` file:

```
[mysqld_safe]
malloc-lib=tcmalloc
```

To use a specific `tcmalloc` library, specify its full path name. Example:

```
[mysqld_safe]
malloc-lib=/opt/lib/libtcmalloc_minimal.so
```

- `--mysqld=prog_name`

  The name of the server program (in the `ledir` directory) that you want to start. This option is needed if you use the MySQL binary distribution but have the data directory outside of the binary distribution. If `mysqld_safe` cannot find the server, use the `--ledir` option to indicate the path name to the directory where the server is located.

  This option is accepted only on the command line, not in option files. On platforms that use systemd, the value can be specified in the value of `MYSQLD_OPTS`. See Section 2.5.9, “Managing MySQL Server with systemd”.

- `--mysqld-version=suffix`

  This option is similar to the `--mysqld` option, but you specify only the suffix for the server program name. The base name is assumed to be `mysqld`. For example, if you use `--mysqld-version=debug`, `mysqld_safe` starts the `mysqld-debug` program in the `ledir` directory. If the argument to `--mysqld-version` is empty, `mysqld_safe` uses `mysqld` in the `ledir` directory.

  This option is accepted only on the command line, not in option files. On platforms that use systemd, the value can be specified in the value of `MYSQLD_OPTS`. See Section 2.5.9, “Managing MySQL Server with systemd”.

- `--nice=priority`

  Use the `nice` program to set the server's scheduling priority to the given value.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read. This must be the first option on the command line if it is used.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--open-files-limit=`*`count`*

  The number of files that `mysqld` should be able to open. The option value is passed to `ulimit -n`.

  

  Note

  You must start `mysqld_safe` as `root` for this to function properly.

- `--pid-file=`*`file_name`*

  The path name that `mysqld` should use for its process ID file.

- `--plugin-dir=`*`dir_name`*

  The path name of the plugin directory.

- `--port=`*`port_num`*

  The port number that the server should use when listening for TCP/IP connections. The port number must be 1024 or higher unless the server is started by the `root` operating system user.

- `--skip-kill-mysqld`

  Do not try to kill stray `mysqld` processes at startup. This option works only on Linux.

- `--socket=`*`path`*

  The Unix socket file that the server should use when listening for local connections.

- `--syslog`, `--skip-syslog`

  `--syslog` causes error messages to be sent to `syslog` on systems that support the `logger` program. `--skip-syslog` suppresses the use of `syslog`; messages are written to an error log file.

  When `syslog` is used for error logging, the `daemon.err` facility/severity is used for all log messages.

  Using these options to control `mysqld` logging is deprecated. To write error log output to the system log, use the instructions at Section 5.4.2.8, “Error Logging to the System Log”. To control the facility, use the server `log_syslog_facility` system variable.

- `--syslog-tag=`*`tag`*

  For logging to `syslog`, messages from `mysqld_safe` and `mysqld` are written with identifiers of `mysqld_safe` and `mysqld`, respectively. To specify a suffix for the identifiers, use `--syslog-tag=`*`tag`*, which modifies the identifiers to be `mysqld_safe-`*`tag`* and `mysqld-`*`tag`*.

  Using this option to control `mysqld` logging is deprecated. Use the server `log_syslog_tag` system variable instead. See Section 5.4.2.8, “Error Logging to the System Log”.

- `--timezone=`*`timezone`*

  Set the `TZ` time zone environment variable to the given option value. Consult your operating system documentation for legal time zone specification formats.

- `--user={user_name|user_id}`

  Run the `mysqld` server as the user having the name *`user_name`* or the numeric user ID *`user_id`*. (“User” in this context refers to a system login account, not a MySQL user listed in the grant tables.)

If you execute `mysqld_safe` with the `--defaults-file` or `--defaults-extra-file` option to name an option file, the option must be the first one given on the command line or the option file is not used. For example, this command does not use the named option file:

```
mysql> mysqld_safe --port=port_num --defaults-file=file_name
```

Instead, use the following command:

```
mysql> mysqld_safe --defaults-file=file_name --port=port_num
```

The `mysqld_safe` script is written so that it normally can start a server that was installed from either a source or a binary distribution of MySQL, even though these types of distributions typically install the server in slightly different locations. (See Section 2.1.5, “Installation Layouts”.) `mysqld_safe` expects one of the following conditions to be true:

- The server and databases can be found relative to the working directory (the directory from which `mysqld_safe` is invoked). For binary distributions, `mysqld_safe` looks under its working directory for `bin` and `data` directories. For source distributions, it looks for `libexec` and `var` directories. This condition should be met if you execute `mysqld_safe` from your MySQL installation directory (for example, `/usr/local/mysql` for a binary distribution).
- If the server and databases cannot be found relative to the working directory, `mysqld_safe` attempts to locate them by absolute path names. Typical locations are `/usr/local/libexec` and `/usr/local/var`. The actual locations are determined from the values configured into the distribution at the time it was built. They should be correct if MySQL is installed in the location specified at configuration time.

Because `mysqld_safe` tries to find the server and databases relative to its own working directory, you can install a binary distribution of MySQL anywhere, as long as you run `mysqld_safe` from the MySQL installation directory:

```
cd mysql_installation_directory
bin/mysqld_safe &
```

If `mysqld_safe` fails, even when invoked from the MySQL installation directory, specify the `--ledir` and `--datadir` options to indicate the directories in which the server and databases are located on your system.

`mysqld_safe` tries to use the `sleep` and `date` system utilities to determine how many times per second it has attempted to start. If these utilities are present and the attempted starts per second is greater than 5, `mysqld_safe` waits 1 full second before starting again. This is intended to prevent excessive CPU usage in the event of repeated failures. (Bug #11761530, Bug #54035)

When you use `mysqld_safe` to start `mysqld`, `mysqld_safe` arranges for error (and notice) messages from itself and from `mysqld` to go to the same destination.

There are several `mysqld_safe` options for controlling the destination of these messages:

- `--log-error=file_name`: Write error messages to the named error file.
- `--syslog`: Write error messages to `syslog` on systems that support the `logger` program.
- `--skip-syslog`: Do not write error messages to `syslog`. Messages are written to the default error log file (*`host_name`*`.err` in the data directory), or to a named file if the `--log-error` option is given.

If none of these options is given, the default is `--skip-syslog`.

When `mysqld_safe` writes a message, notices go to the logging destination (`syslog` or the error log file) and `stdout`. Errors go to the logging destination and `stderr`.

Note

Controlling `mysqld` logging from `mysqld_safe` is deprecated. Use the server's native `syslog` support instead. For more information, see Section 5.4.2.8, “Error Logging to the System Log”.

4.3.3 mysql.server — MySQL Server Startup Script

MySQL distributions on Unix and Unix-like system include a script named `mysql.server`, which starts the MySQL server using `mysqld_safe`. It can be used on systems such as Linux and Solaris that use System V-style run directories to start and stop system services. It is also used by the macOS Startup Item for MySQL.

`mysql.server` is the script name as used within the MySQL source tree. The installed name might be different (for example, `mysqld` or `mysql`). In the following discussion, adjust the name `mysql.server` as appropriate for your system.

Note

For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, `mysql.server` and `mysqld_safe` are not installed because they are unnecessary. For more information, see Section 2.5.9, “Managing MySQL Server with systemd”.

To start or stop the server manually using the `mysql.server` script, invoke it from the command line with `start` or `stop` arguments:

```
mysql.server start
mysql.server stop
```

`mysql.server` changes location to the MySQL installation directory, then invokes `mysqld_safe`. To run the server as some specific user, add an appropriate `user` option to the `[mysqld]` group of the global `/etc/my.cnf` option file, as shown later in this section. (It is possible that you must edit `mysql.server` if you've installed a binary distribution of MySQL in a nonstandard location. Modify it to change location into the proper directory before it runs `mysqld_safe`. If you do this, your modified version of `mysql.server` may be overwritten if you upgrade MySQL in the future; make a copy of your edited version that you can reinstall.)

`mysql.server stop` stops the server by sending a signal to it. You can also stop the server manually by executing `mysqladmin shutdown`.

To start and stop MySQL automatically on your server, you must add start and stop commands to the appropriate places in your `/etc/rc*` files:

- If you use the Linux server RPM package (`MySQL-server-VERSION.rpm`), or a native Linux package installation, the `mysql.server` script may be installed in the `/etc/init.d` directory with the name `mysqld` or `mysql`. See Section 2.5.4, “Installing MySQL on Linux Using RPM Packages from Oracle”, for more information on the Linux RPM packages.
- If you install MySQL from a source distribution or using a binary distribution format that does not install `mysql.server` automatically, you can install the script manually. It can be found in the `support-files` directory under the MySQL installation directory or in a MySQL source tree. Copy the script to the `/etc/init.d` directory with the name `mysql` and make it executable:

  ```
  cp mysql.server /etc/init.d/mysql
  chmod +x /etc/init.d/mysql
  ```

  After installing the script, the commands needed to activate it to run at system startup depend on your operating system. On Linux, you can use `chkconfig`:

  ```
  chkconfig --add mysql
  ```

On some Linux systems, the following command also seems to be necessary to fully enable the `mysql` script:

```
chkconfig --level 345 mysql on
```

- On FreeBSD, startup scripts generally should go in `/usr/local/etc/rc.d/`. Install the `mysql.server` script as `/usr/local/etc/rc.d/mysql.server.sh` to enable automatic startup. The `rc(8)` manual page states that scripts in this directory are executed only if their base name matches the `*.sh` shell file name pattern. Any other files or directories present within the directory are silently ignored.
- As an alternative to the preceding setup, some operating systems also use `/etc/rc.local` or `/etc/init.d/boot.local` to start additional services on startup. To start up MySQL using this method, append a command like the one following to the appropriate startup file:

```
/bin/sh -c 'cd /usr/local/mysql; ./bin/mysqld_safe --user=mysql &'
```

- For other systems, consult your operating system documentation to see how to install startup scripts.

`mysql.server` reads options from the `[mysql.server]` and `[mysqld]` sections of option files. For backward compatibility, it also reads `[mysql_server]` sections, but to be current you should rename such sections to `[mysql.server]`.

You can add options for `mysql.server` in a global `/etc/my.cnf` file. A typical `my.cnf` file might look like this:

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

[mysql.server]
basedir=/usr/local/mysql
```

The `mysql.server` script supports the options shown in the following table. If specified, they *must* be placed in an option file, not on the command line. `mysql.server` supports only `start` and `stop` as command-line arguments.

Table 4.7 mysql.server Option-File Options

Option Name	Description	Type
`basedir`	Path to MySQL installation directory	Directory name
`datadir`	Path to MySQL data directory	Directory name
`pid-file`	File in which server should write its process ID	File name
`service-startup-timeout`	How long to wait for server startup	Integer

- `basedir=`*`dir_name`*

  The path to the MySQL installation directory.

- `datadir=`*`dir_name`*

  The path to the MySQL data directory.

- `pid-file=`*`file_name`*

  The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.

If this option is not given, `mysql.server` uses a default value of *`host_name`*`.pid`. The PID file value passed to `mysqld_safe` overrides any value specified in the `[mysqld_safe]` option file group. Because `mysql.server` reads the `[mysqld]` option file group but not the `[mysqld_safe]` group, you can ensure that `mysqld_safe` gets the same value when invoked from `mysql.server` as when invoked manually by putting the same `pid-file` setting in both the `[mysqld_safe]` and `[mysqld]` groups.

- `service-startup-timeout=`*`seconds`*

  How long in seconds to wait for confirmation of server startup. If the server does not start within this time, `mysql.server` exits with an error. The default value is 900. A value of 0 means not to wait at all for startup. Negative values mean to wait forever (no timeout).

4.3.4 mysqld_multi — Manage Multiple MySQL Servers

`mysqld_multi` is designed to manage several `mysqld` processes that listen for connections on different Unix socket files and TCP/IP ports. It can start or stop servers, or report their current status.

Note

For some Linux platforms, MySQL installation from RPM or Debian packages includes systemd support for managing MySQL server startup and shutdown. On these platforms, `mysqld_multi` is not installed because it is unnecessary. For information about using systemd to handle multiple MySQL instances, see Section 2.5.9, “Managing MySQL Server with systemd”.

`mysqld_multi` searches for groups named `[mysqld`*`N`*`]` in `my.cnf` (or in the file named by the `--defaults-file` option). *`N`* can be any positive integer. This number is referred to in the following discussion as the option group number, or *`GNR`*. Group numbers distinguish option groups from one another and are used as arguments to `mysqld_multi` to specify which servers you want to start, stop, or obtain a status report for. Options listed in these groups are the same that you would use in the `[mysqld]` group used for starting `mysqld`. (See, for example, Section 2.9.5, “Starting and Stopping MySQL Automatically”.) However, when using multiple servers, it is necessary that each one use its own value for options such as the Unix socket file and TCP/IP port number. For more information on which options must be unique per server in a multiple-server environment, see Section 5.8, “Running Multiple MySQL Instances on One Machine”.

To invoke `mysqld_multi`, use the following syntax:

```
mysqld_multi [options] {start|stop|reload|report} [GNR[,GNR] ...]
```

`start`, `stop`, `reload` (stop and restart), and `report` indicate which operation to perform. You can perform the designated operation for a single server or multiple servers, depending on the *`GNR`* list that follows the option name. If there is no list, `mysqld_multi` performs the operation for all servers in the option file.

Each *`GNR`* value represents an option group number or range of group numbers. The value should be the number at the end of the group name in the option file. For example, the *`GNR`* for a group named `[mysqld17]` is `17`. To specify a range of numbers, separate the first and last numbers by a dash. The *`GNR`* value `10-13` represents groups `[mysqld10]` through `[mysqld13]`. Multiple groups or group ranges can be specified on the command line, separated by commas. There must be no whitespace characters (spaces or tabs) in the *`GNR`* list; anything after a whitespace character is ignored.

This command starts a single server using option group `[mysqld17]`:

```
mysqld_multi start 17
```

This command stops several servers, using option groups `[mysqld8]` and `[mysqld10]` through `[mysqld13]`:

```
mysqld_multi stop 8,10-13
```

For an example of how you might set up an option file, use this command:

```
mysqld_multi --example
```

`mysqld_multi` searches for option files as follows:

- With `--no-defaults`, no option files are read.
- With `--defaults-file=file_name`, only the named file is read.
- Otherwise, option files in the standard list of locations are read, including any file named by the `--defaults-extra-file=file_name` option, if one is given. (If the option is given multiple times, the last value is used.)

For additional information about these and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

Option files read are searched for `[mysqld_multi]` and `[mysqldN]` option groups. The `[mysqld_multi]` group can be used for options to `mysqld_multi` itself. `[mysqldN]` groups can be used for options passed to specific `mysqld` instances.

The `[mysqld]` or `[mysqld_safe]` groups can be used for common options read by all instances of `mysqld` or `mysqld_safe`. You can specify a `--defaults-file=file_name` option to use a different configuration file for that instance, in which case the `[mysqld]` or `[mysqld_safe]` groups from that file are used for that instance.

`mysqld_multi` supports the following options.

- `--help`

  Display a help message and exit.

- `--example`

  Display a sample option file.

- `--log=file_name`

  Specify the name of the log file. If the file exists, log output is appended to it.

- `--mysqladmin=prog_name`

  The `mysqladmin` binary to be used to stop servers.

- `--mysqld=prog_name`

  The `mysqld` binary to be used. Note that you can specify `mysqld_safe` as the value for this option also. If you use `mysqld_safe` to start the server, you can include the `mysqld` or `ledir` options in the corresponding `[mysqldN]` option group. These options indicate the name of the server that `mysqld_safe` should start and the path name of the directory where the server is located. (See the descriptions for these options in Section 4.3.2, “mysqld_safe — MySQL Server Startup Script”.) Example:

  ```
  [mysqld38]
  mysqld = mysqld-debug
  ledir  = /opt/local/mysql/libexec
  ```

- `--no-log`

  Print log information to `stdout` rather than to the log file. By default, output goes to the log file.

- `--password=password`

The password of the MySQL account to use when invoking `mysqladmin`. Note that the password value is not optional for this option, unlike for other MySQL programs.

- `--silent`

  Silent mode; disable warnings.

- `--tcp-ip`

  Connect to each MySQL server through the TCP/IP port instead of the Unix socket file. (If a socket file is missing, the server might still be running, but accessible only through the TCP/IP port.) By default, connections are made using the Unix socket file. This option affects `stop` and `report` operations.

- `--user=`*`user_name`*

  The user name of the MySQL account to use when invoking `mysqladmin`.

- `--verbose`

  Be more verbose.

- `--version`

  Display version information and exit.

Some notes about `mysqld_multi`:

- **Most important**: Before using `mysqld_multi` be sure that you understand the meanings of the options that are passed to the `mysqld` servers and *why* you would want to have separate `mysqld` processes. Beware of the dangers of using multiple `mysqld` servers with the same data directory. Use separate data directories, unless you *know* what you are doing. Starting multiple servers with the same data directory does *not* give you extra performance in a threaded system. See Section 5.8, “Running Multiple MySQL Instances on One Machine”.

  

  Important

  Make sure that the data directory for each server is fully accessible to the Unix account that the specific `mysqld` process is started as. *Do not* use the Unix *`root`* account for this, unless you *know* what you are doing. See Section 6.1.5, “How to Run MySQL as a Normal User”.

- Make sure that the MySQL account used for stopping the `mysqld` servers (with the `mysqladmin` program) has the same user name and password for each server. Also, make sure that the account has the `SHUTDOWN` privilege. If the servers that you want to manage have different user names or passwords for the administrative accounts, you might want to create an account on each server that has the same user name and password. For example, you might set up a common `multi_admin` account by executing the following commands for each server:

  ```
  $> mysql -u root -S /tmp/mysql.sock -p
  Enter password:
  mysql> CREATE USER 'multi_admin'@'localhost' IDENTIFIED BY 'multipass';
  mysql> GRANT SHUTDOWN ON *.* TO 'multi_admin'@'localhost';
  ```

  See Section 6.2, “Access Control and Account Management”. You have to do this for each `mysqld` server. Change the connection parameters appropriately when connecting to each one. Note that the host name part of the account name must permit you to connect as `multi_admin` from the host where you want to run `mysqld_multi`.

- The Unix socket file and the TCP/IP port number must be different for every `mysqld`. (Alternatively, if the host has multiple network addresses, you can set the `bind_address` system variable to cause different servers to listen to different interfaces.)

- The `--pid-file` option is very important if you are using `mysqld_safe` to start `mysqld` (for example, `--mysqld=mysqld_safe`) Every `mysqld` should have its own process ID file. The advantage of using `mysqld_safe` instead of `mysqld` is that `mysqld_safe` monitors its `mysqld` process and restarts it if the process terminates due to a signal sent using `kill -9` or for other reasons, such as a segmentation fault.
- You might want to use the `--user` option for `mysqld`, but to do this you need to run the `mysqld_multi` script as the Unix superuser (`root`). Having the option in the option file doesn't matter; you just get a warning if you are not the superuser and the `mysqld` processes are started under your own Unix account.

The following example shows how you might set up an option file for use with `mysqld_multi`. The order in which the `mysqld` programs are started or stopped depends on the order in which they appear in the option file. Group numbers need not form an unbroken sequence. The first and fifth `[mysqldN]` groups were intentionally omitted from the example to illustrate that you can have “gaps” in the option file. This gives you more flexibility.

```
# This is an example of a my.cnf file for mysqld_multi.
# Usually this file is located in home dir ~/.my.cnf or /etc/my.cnf

[mysqld_multi]
mysqld     = /usr/local/mysql/bin/mysqld_safe
mysqladmin = /usr/local/mysql/bin/mysqladmin
user       = multi_admin
password   = my_password

[mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid-file   = /usr/local/mysql/data2/hostname.pid2
datadir    = /usr/local/mysql/data2
language   = /usr/local/mysql/share/mysql/english
user       = unix_user1

[mysqld3]
mysqld     = /path/to/mysqld_safe
ledir      = /path/to/mysqld-binary/
mysqladmin = /path/to/mysqladmin
socket     = /tmp/mysql.sock3
port       = 3308
pid-file   = /usr/local/mysql/data3/hostname.pid3
datadir    = /usr/local/mysql/data3
language   = /usr/local/mysql/share/mysql/swedish
user       = unix_user2

[mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid-file   = /usr/local/mysql/data4/hostname.pid4
datadir    = /usr/local/mysql/data4
language   = /usr/local/mysql/share/mysql/estonia
user       = unix_user3

[mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid-file   = /usr/local/mysql/data6/hostname.pid6
datadir    = /usr/local/mysql/data6
language   = /usr/local/mysql/share/mysql/japanese
user       = unix_user4
```

See Section 4.2.2.2, “Using Option Files”.

4.4 Installation-Related Programs

The programs in this section are used when installing or upgrading MySQL.

4.4.1 comp_err — Compile MySQL Error Message File

`comp_err` creates the `errmsg.sys` file that is used by `mysqld` to determine the error messages to display for different error codes. `comp_err` normally is run automatically when MySQL is built. It compiles the `errmsg.sys` file from text-format error information in MySQL source distributions:

- As of MySQL 8.0.19, the error information comes from the `messages_to_error_log.txt` and `messages_to_clients.txt` files in the `share` directory.

  For more information about defining error messages, see the comments within those files, along with the `errmsg_readme.txt` file.

- Prior to MySQL 8.0.19, the error information comes from the `errmsg-utf8.txt` file in the `sql/share` directory.

`comp_err` also generates the `mysqld_error.h`, `mysqld_ername.h`, and `mysqld_errmsg.h` header files.

Invoke `comp_err` like this:

```
comp_err [options]
```

`comp_err` supports the following options.

- `--help`, `-?`

  Display a help message and exit.

- `--charset=dir_name`, `-C dir_name`

  The character set directory. The default is `../sql/share/charsets`.

- `--debug=debug_options`, `-# debug_options`

  Write a debugging log. A typical `debug_options` string is `d:t:O,file_name`. The default is `d:t:O,/tmp/comp_err.trace`.

- `--debug-info`, `-T`

  Print some debugging information when the program exits.

- `--errmsg-file=file_name`, `-H file_name`

  The name of the error message file. The default is `mysqld_errmsg.h`. This option was added in MySQL 8.0.18.

- `--header-file=file_name`, `-H file_name`

  The name of the error header file. The default is `mysqld_error.h`.

- `--in-file=file_name`, `-F file_name`

  The name of the input file. The default is `../share/errmsg-utf8.txt`.

  This option was removed in MySQL 8.0.19 and replaced by the `--in-file-errlog` and `--in-file-toclient` options.

- `--in-file-errlog=file_name`, `-e file_name`

  The name of the input file that defines error messages intended to be written to the error log. The default is `../share/messages_to_error_log.txt`.

  This option was added in MySQL 8.0.19.

- `--in-file-toclient=file_name`, `-c file_name`

The name of the input file that defines error messages intended to be written to clients. The default is `../share/messages_to_clients.txt`.

This option was added in MySQL 8.0.19.

- `--name-file=file_name`, `-N file_name`

  The name of the error name file. The default is `mysqld_ername.h`.

- `--out-dir=dir_name`, `-D dir_name`

  The name of the output base directory. The default is `../sql/share/`.

- `--out-file=file_name`, `-O file_name`

  The name of the output file. The default is `errmsg.sys`.

- `--version`, `-V`

  Display version information and exit.

4.4.2 mysql_secure_installation — Improve MySQL Installation Security

This program enables you to improve the security of your MySQL installation in the following ways:

- You can set a password for `root` accounts.
- You can remove `root` accounts that are accessible from outside the local host.
- You can remove anonymous-user accounts.
- You can remove the `test` database (which by default can be accessed by all users, even anonymous users), and privileges that permit anyone to access databases with names that start with `test_`.

`mysql_secure_installation` helps you implement security recommendations similar to those described at Section 2.9.4, “Securing the Initial MySQL Account”.

Normal usage is to connect to the local MySQL server; invoke `mysql_secure_installation` without arguments:

```
mysql_secure_installation
```

When executed, `mysql_secure_installation` prompts you to determine which actions to perform.

The `validate_password` component can be used for password strength checking. If the plugin is not installed, `mysql_secure_installation` prompts the user whether to install it. Any passwords entered later are checked using the plugin if it is enabled.

Most of the usual MySQL client options such as `--host` and `--port` can be used on the command line and in option files. For example, to connect to the local server over IPv6 using port 3307, use this command:

```
mysql_secure_installation --host=::1 --port=3307
```

`mysql_secure_installation` supports the following options, which can be specified on the command line or in the `[mysql_secure_installation]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.8 mysql_secure_installation Options

Option Name	Description	Introduced
--defaults-extra-file	Read named option file in addition to usual option files	

Option Name	Description	Introduced
--defaults-file	Read only named option file	
--defaults-group-suffix	Option group suffix value	
--help	Display help message and exit	
--host	Host on which MySQL server is located	
--no-defaults	Read no option files	
--password	Accepted but always ignored. Whenever mysql_secure_installation is invoked, the user is prompted for a password, regardless	
--port	TCP/IP port number for connection	
--print-defaults	Print default options	
--protocol	Transport protocol to use	
--socket	Unix socket file or Windows named pipe to use	
--ssl-ca	File that contains list of trusted SSL Certificate Authorities	
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files	
--ssl-cert	File that contains X.509 certificate	
--ssl-cipher	Permissible ciphers for connection encryption	
--ssl-crl	File that contains certificate revocation lists	
--ssl-crlpath	Directory that contains certificate revocation-list files	
--ssl-fips-mode	Whether to enable FIPS mode on client side	
--ssl-key	File that contains X.509 key	
--ssl-mode	Desired security state of connection to server	
--ssl-session-data	File that contains SSL session data	8.0.29
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16
--tls-version	Permissible TLS protocols for encrypted connections	
--use-default	Execute with no user interactivity	
--user	MySQL user name to use when connecting to server	

- `--help`, `-?`

  Display a help message and exit.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysql_secure_installation` normally reads the `[client]` and `[mysql_secure_installation]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysql_secure_installation` also reads the `[client_other]` and `[mysql_secure_installation_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--host=host_name`, `-h host_name`

  Connect to the MySQL server on the given host.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--password=password`, `-p password`

  This option is accepted but ignored. Whether or not this option is used, `mysql_secure_installation` always prompts the user for a password.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL

library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--use-default`

  Execute noninteractively. This option can be used for unattended installation operations.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

4.4.3 mysql_ssl_rsa_setup — Create SSL/RSA Files

This program creates the SSL certificate and key files and RSA key-pair files required to support secure connections using SSL and secure password exchange using RSA over unencrypted connections, if those files are missing. `mysql_ssl_rsa_setup` can also be used to create new SSL files if the existing ones have expired.

Note

`mysql_ssl_rsa_setup` uses the `openssl` command, so its use is contingent on having OpenSSL installed on your machine.

Another way to generate SSL and RSA files, for MySQL distributions compiled using OpenSSL, is to have the server generate them automatically. See Section 6.3.3.1, “Creating SSL and RSA Certificates and Keys using MySQL”.

Important

`mysql_ssl_rsa_setup` helps lower the barrier to using SSL by making it easier to generate the required files. However, certificates generated by `mysql_ssl_rsa_setup` are self-signed, which is not very secure. After you gain experience using the files created by `mysql_ssl_rsa_setup`, consider obtaining a CA certificate from a registered certificate authority.

Invoke `mysql_ssl_rsa_setup` like this:

```
mysql_ssl_rsa_setup [options]
```

Typical options are `--datadir` to specify where to create the files, and `--verbose` to see the `openssl` commands that `mysql_ssl_rsa_setup` executes.

`mysql_ssl_rsa_setup` attempts to create SSL and RSA files using a default set of file names. It works as follows:

1. `mysql_ssl_rsa_setup` checks for the `openssl` binary at the locations specified by the `PATH` environment variable. If `openssl` is not found, `mysql_ssl_rsa_setup` does nothing. If `openssl` is present, `mysql_ssl_rsa_setup` looks for default SSL and RSA files in the MySQL data directory specified by the `--datadir` option, or the compiled-in data directory if the `--datadir` option is not given.

2. `mysql_ssl_rsa_setup` checks the data directory for SSL files with the following names:

   ```
   ca.pem
   server-cert.pem
   server-key.pem
   ```

3. If any of those files are present, `mysql_ssl_rsa_setup` creates no SSL files. Otherwise, it invokes `openssl` to create them, plus some additional files:

   ```
   ca.pem              Self-signed CA certificate
   ca-key.pem          CA private key
   server-cert.pem     Server certificate
   ```

```
server-key.pem       Server private key
client-cert.pem      Client certificate
client-key.pem       Client private key
```

These files enable secure client connections using SSL; see Section 6.3.1, “Configuring MySQL to Use Encrypted Connections”.

4. `mysql_ssl_rsa_setup` checks the data directory for RSA files with the following names:

```
private_key.pem      Private member of private/public key pair
public_key.pem       Public member of private/public key pair
```

5. If any of these files are present, `mysql_ssl_rsa_setup` creates no RSA files. Otherwise, it invokes `openssl` to create them. These files enable secure password exchange using RSA over unencrypted connections for accounts authenticated by the `sha256_password` or `caching_sha2_password` plugin; see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

For information about the characteristics of files created by `mysql_ssl_rsa_setup`, see Section 6.3.3.1, “Creating SSL and RSA Certificates and Keys using MySQL”.

At startup, the MySQL server automatically uses the SSL files created by `mysql_ssl_rsa_setup` to enable SSL if no explicit SSL options are given other than `--ssl` (possibly along with `ssl_cipher`). If you prefer to designate the files explicitly, invoke clients with the `--ssl-ca`, `--ssl-cert`, and `--ssl-key` options at startup to name the `ca.pem`, `server-cert.pem`, and `server-key.pem` files, respectively.

The server also automatically uses the RSA files created by `mysql_ssl_rsa_setup` to enable RSA if no explicit RSA options are given.

If the server is SSL-enabled, clients use SSL by default for the connection. To specify certificate and key files explicitly, use the `--ssl-ca`, `--ssl-cert`, and `--ssl-key` options to name the `ca.pem`, `client-cert.pem`, and `client-key.pem` files, respectively. However, some additional client setup may be required first because `mysql_ssl_rsa_setup` by default creates those files in the data directory. The permissions for the data directory normally enable access only to the system account that runs the MySQL server, so client programs cannot use files located there. To make the files available, copy them to a directory that is readable (but *not* writable) by clients:

- For local clients, the MySQL installation directory can be used. For example, if the data directory is a subdirectory of the installation directory and your current location is the data directory, you can copy the files like this:

```
cp ca.pem client-cert.pem client-key.pem ..
```

- For remote clients, distribute the files using a secure channel to ensure they are not tampered with during transit.

If the SSL files used for a MySQL installation have expired, you can use `mysql_ssl_rsa_setup` to create new ones:

1. Stop the server.
2. Rename or remove the existing SSL files. You may wish to make a backup of them first. (The RSA files do not expire, so you need not remove them. `mysql_ssl_rsa_setup` can see that they exist and does not overwrite them.)
3. Run `mysql_ssl_rsa_setup` with the `--datadir` option to specify where to create the new files.
4. Restart the server.

`mysql_ssl_rsa_setup` supports the following command-line options, which can be specified on the command line or in the `[mysql_ssl_rsa_setup]` and `[mysqld]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.9 mysql_ssl_rsa_setup Options

Option Name	Description
--datadir	Path to data directory
--help	Display help message and exit
--suffix	Suffix for X.509 certificate Common Name attribute
--uid	Name of effective user to use for file permissions
--verbose	Verbose mode
--version	Display version information and exit

- `--help`, `?`

  Display a help message and exit.

- `--datadir=`*`dir_name`*

  The path to the directory that `mysql_ssl_rsa_setup` should check for default SSL and RSA files and in which it should create files if they are missing. The default is the compiled-in data directory.

- `--suffix=`*`str`*

  The suffix for the Common Name attribute in X.509 certificates. The suffix value is limited to 17 characters. The default is based on the MySQL version number.

- `--uid=name`, `-v`

  The name of the user who should be the owner of any created files. The value is a user name, not a numeric user ID. In the absence of this option, files created by `mysql_ssl_rsa_setup` are owned by the user who executes it. This option is valid only if you execute the program as `root` on a system that supports the `chown()` system call.

- `--verbose`, `-v`

  Verbose mode. Produce more output about what the program does. For example, the program shows the `openssl` commands it runs, and produces output to indicate whether it skips SSL or RSA file creation because some default file already exists.

- `--version`, `-V`

  Display version information and exit.

4.4.4 mysql_tzinfo_to_sql — Load the Time Zone Tables

The `mysql_tzinfo_to_sql` program loads the time zone tables in the `mysql` database. It is used on systems that have a *zoneinfo* database (the set of files describing time zones). Examples of such systems are Linux, FreeBSD, Solaris, and macOS. One likely location for these files is the `/usr/share/zoneinfo` directory (`/usr/share/lib/zoneinfo` on Solaris). If your system does not have a zoneinfo database, you can use the downloadable package described in Section 5.1.15, “MySQL Server Time Zone Support”.

`mysql_tzinfo_to_sql` can be invoked several ways:

```
mysql_tzinfo_to_sql tz_dir
mysql_tzinfo_to_sql tz_file tz_name
mysql_tzinfo_to_sql --leap tz_file
```

For the first invocation syntax, pass the zoneinfo directory path name to `mysql_tzinfo_to_sql` and send the output into the `mysql` program. For example:

```
mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root mysql
```

`mysql_tzinfo_to_sql` reads your system's time zone files and generates SQL statements from them. `mysql` processes those statements to load the time zone tables.

The second syntax causes `mysql_tzinfo_to_sql` to load a single time zone file *`tz_file`* that corresponds to a time zone name *`tz_name`*:

```
mysql_tzinfo_to_sql tz_file tz_name | mysql -u root mysql
```

If your time zone needs to account for leap seconds, invoke `mysql_tzinfo_to_sql` using the third syntax, which initializes the leap second information. *`tz_file`* is the name of your time zone file:

```
mysql_tzinfo_to_sql --leap tz_file | mysql -u root mysql
```

After running `mysql_tzinfo_to_sql`, it is best to restart the server so that it does not continue to use any previously cached time zone data.

4.4.5 mysql_upgrade — Check and Upgrade MySQL Tables

Note

As of MySQL 8.0.16, the MySQL server performs the upgrade tasks previously handled by `mysql_upgrade` (for details, see Section 2.10.3, “What the MySQL Upgrade Process Upgrades”). Consequently, `mysql_upgrade` is unneeded and is deprecated as of that version; expect it to be removed in a future version of MySQL. Because `mysql_upgrade` no longer performs upgrade tasks, it exits with status 0 unconditionally.

Each time you upgrade MySQL, you should execute `mysql_upgrade`, which looks for incompatibilities with the upgraded MySQL server:

- It upgrades the system tables in the `mysql` schema so that you can take advantage of new privileges or capabilities that might have been added.
- It upgrades the Performance Schema, `INFORMATION_SCHEMA`, and `sys` schema.
- It examines user schemas.

If `mysql_upgrade` finds that a table has a possible incompatibility, it performs a table check and, if problems are found, attempts a table repair. If the table cannot be repaired, see Section 2.10.13, “Rebuilding or Repairing Tables or Indexes” for manual table repair strategies.

`mysql_upgrade` communicates directly with the MySQL server, sending it the SQL statements required to perform an upgrade.

Caution

You should always back up your current MySQL installation *before* performing an upgrade. See Section 7.2, “Database Backup Methods”.

Some upgrade incompatibilities may require special handling *before* upgrading your MySQL installation and running `mysql_upgrade`. See Section 2.10, “Upgrading MySQL”, for instructions on determining whether any such incompatibilities apply to your installation and how to handle them.

Use `mysql_upgrade` like this:

1. Ensure that the server is running.
2. Invoke `mysql_upgrade` to upgrade the system tables in the `mysql` schema and check and repair tables in other schemas:

   ```
   mysql_upgrade [options]
   ```

3. Stop the server and restart it so that any system table changes take effect.

If you have multiple MySQL server instances to upgrade, invoke `mysql_upgrade` with connection parameters appropriate for connecting to each of the desired servers. For example, with servers running on the local host on parts 3306 through 3308, upgrade each of them by connecting to the appropriate port:

```
mysql_upgrade --protocol=tcp -P 3306 [other_options]
mysql_upgrade --protocol=tcp -P 3307 [other_options]
mysql_upgrade --protocol=tcp -P 3308 [other_options]
```

For local host connections on Unix, the `--protocol=tcp` option forces a connection using TCP/IP rather than the Unix socket file.

By default, `mysql_upgrade` runs as the MySQL `root` user. If the `root` password is expired when you run `mysql_upgrade`, it displays a message that your password is expired and that `mysql_upgrade` failed as a result. To correct this, reset the `root` password to unexpire it and run `mysql_upgrade` again. First, connect to the server as `root`:

```
$> mysql -u root -p
Enter password: ****  <- enter root password here
```

Reset the password using `ALTER USER`:

```
mysql> ALTER USER USER() IDENTIFIED BY 'root-password';
```

Then exit `mysql` and run `mysql_upgrade` again:

```
$> mysql_upgrade [options]
```

Note

If you run the server with the `disabled_storage_engines` system variable set to disable certain storage engines (for example, `MyISAM`), `mysql_upgrade` might fail with an error like this:

```
mysql_upgrade: [ERROR] 3161: Storage engine MyISAM is disabled
(Table creation is disallowed).
```

To handle this, restart the server with `disabled_storage_engines` disabled. Then you should be able to run `mysql_upgrade` successfully. After that, restart the server with `disabled_storage_engines` set to its original value.

Unless invoked with the `--upgrade-system-tables` option, `mysql_upgrade` processes all tables in all user schemas as necessary. Table checking might take a long time to complete. Each table is locked and therefore unavailable to other sessions while it is being processed. Check and repair operations can be time-consuming, particularly for large tables. Table checking uses the `FOR UPGRADE` option of the `CHECK TABLE` statement. For details about what this option entails, see Section 13.7.3.2, “CHECK TABLE Statement”.

`mysql_upgrade` marks all checked and repaired tables with the current MySQL version number. This ensures that the next time you run `mysql_upgrade` with the same version of the server, it can be determined whether there is any need to check or repair a given table again.

`mysql_upgrade` saves the MySQL version number in a file named `mysql_upgrade_info` in the data directory. This is used to quickly check whether all tables have been checked for this release so that table-checking can be skipped. To ignore this file and perform the check regardless, use the `--force` option.

Note

The `mysql_upgrade_info` file is deprecated; expect it to be removed in a future version of MySQL.

`mysql_upgrade` checks `mysql.user` system table rows and, for any row with an empty `plugin` column, sets that column to `'mysql_native_password'` if the credentials use a hash format compatible with that plugin. Rows with a pre-4.1 password hash must be upgraded manually.

`mysql_upgrade` does not upgrade the contents of the time zone tables or help tables. For upgrade instructions, see Section 5.1.15, “MySQL Server Time Zone Support”, and Section 5.1.17, “Server-Side Help Support”.

Unless invoked with the `--skip-sys-schema` option, `mysql_upgrade` installs the `sys` schema if it is not installed, and upgrades it to the current version otherwise. An error occurs if a `sys` schema exists but has no `version` view, on the assumption that its absence indicates a user-created schema:

```
A sys schema exists with no sys.version view. If
you have a user created sys schema, this must be renamed for the
upgrade to succeed.
```

To upgrade in this case, remove or rename the existing `sys` schema first.

`mysql_upgrade` supports the following options, which can be specified on the command line or in the `[mysql_upgrade]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.10 mysql_upgrade Options

Option Name	Description	Introduced	Deprecated
--bind-address	Use specified network interface to connect to MySQL Server		
--character-sets-dir	Directory where character sets are installed		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--force	Force execution even if mysql_upgrade has already been executed		

Option Name	Description	Introduced	Deprecated
	for current MySQL version		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--host	Host on which MySQL server is located		
--login-path	Read login path options from .mylogin.cnf		
--max-allowed-packet	Maximum packet length to send to or receive from server		
--net-buffer-length	Buffer size for TCP/IP and socket communication		
--no-defaults	Read no option files		
--password	Password to use when connecting to server		
--pipe	Connect to server using named pipe (Windows only)		
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--server-public-key-path	Path name to file containing RSA public key		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--skip-sys-schema	Do not install or upgrade sys schema		
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		

Option Name	Description	Introduced	Deprecated
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--upgrade-system-tables	Update only system tables, not user schemas		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		
--version-check	Check for proper server version		
--write-binlog	Write all statements to binary log		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`

  Display a short help message and exit.

- `--bind-address=`*`ip_address`*

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=dir_name`

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:O,/tmp/mysql_upgrade.trace`.

- `--debug-check`

  Print some debugging information when the program exits.

- `--debug-info`, `-T`

  Print debugging information and memory and CPU usage statistics when the program exits.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysql_upgrade` normally reads the `[client]` and `[mysql_upgrade]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysql_upgrade` also reads the `[client_other]` and `[mysql_upgrade_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--force`

  Ignore the `mysql_upgrade_info` file and force execution even if `mysql_upgrade` has already been executed for the current version of MySQL.

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--host=host_name`, `-h host_name`

  Connect to the MySQL server on the given host.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--max-allowed-packet=value`

  The maximum size of the buffer for client/server communication. The default value is 24MB. The minimum and maximum values are 4KB and 2GB.

- `--net-buffer-length=value`

  The initial size of the buffer for client/server communication. The default value is 1MB − 1KB. The minimum and maximum values are 4KB and 16MB.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults`

is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysql_upgrade` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysql_upgrade` should not prompt for one, use the `--skip-password` option.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysql_upgrade` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-

based password exchange is not used, as is the case when the client connects to the server using a secure connection.

If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--shared-memory-base-name=name`

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--skip-sys-schema`

  By default, `mysql_upgrade` installs the `sys` schema if it is not installed, and upgrades it to the current version otherwise. The `--skip-sys-schema` option suppresses this behavior.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=ciphersuite_list`

The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--upgrade-system-tables`, `-s`

  Upgrade only the system tables in the `mysql` schema, do not upgrade user schemas.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server. The default user name is `root`.

- `--verbose`

  Verbose mode. Print more information about what the program does.

- `--version-check`, `-k`

  Check the version of the server to which `mysql_upgrade` is connecting to verify that it is the same as the version for which `mysql_upgrade` was built. If not, `mysql_upgrade` exits. This option is enabled by default; to disable the check, use `--skip-version-check`.

- `--write-binlog`

  By default, binary logging by `mysql_upgrade` is disabled. Invoke the program with `--write-binlog` if you want its actions to be written to the binary log.

  When the server is running with global transaction identifiers (GTIDs) enabled (`gtid_mode=ON`), do not enable binary logging by `mysql_upgrade`.

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

4.5 Client Programs

This section describes client programs that connect to the MySQL server.

4.5.1 mysql — The MySQL Command-Line Client

`mysql` is a simple SQL shell with input line editing capabilities. It supports interactive and noninteractive use. When used interactively, query results are presented in an ASCII-table format.

When used noninteractively (for example, as a filter), the result is presented in tab-separated format. The output format can be changed using command options.

If you have problems due to insufficient memory for large result sets, use the `--quick` option. This forces `mysql` to retrieve results from the server a row at a time rather than retrieving the entire result set and buffering it in memory before displaying it. This is done by returning the result set using the `mysql_use_result()` C API function in the client/server library rather than `mysql_store_result()`.

Note

Alternatively, MySQL Shell offers access to the X DevAPI. For details, see MySQL Shell 8.0.

Using `mysql` is very easy. Invoke it from the prompt of your command interpreter as follows:

```
mysql db_name
```

Or:

```
mysql --user=user_name --password db_name
```

In this case, you'll need to enter your password in response to the prompt that `mysql` displays:

```
Enter password: your_password
```

Then type an SQL statement, end it with `;`, `\g`, or `\G` and press Enter.

Typing **Control+C** interrupts the current statement if there is one, or cancels any partial input line otherwise.

You can execute SQL statements in a script file (batch file) like this:

```
mysql db_name < script.sql > output.tab
```

On Unix, the `mysql` client logs statements executed interactively to a history file. See Section 4.5.1.3, “mysql Client Logging”.

4.5.1.1 mysql Client Options

`mysql` supports the following options, which can be specified on the command line or in the `[mysql]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.11 mysql Client Options

Option Name	Description	Introduced	Deprecated
--auto-rehash	Enable automatic rehashing		
--auto-vertical-output	Enable automatic vertical result set display		
--batch	Do not use history file		
--binary-as-hex	Display binary values in hexadecimal notation		
--binary-mode	Disable \r\n - to - \n translation and treatment of \0 as end-of-query		
--bind-address	Use specified network interface to connect to MySQL Server		

Option Name	Description	Introduced	Deprecated
--character-sets-dir	Directory where character sets are installed		
--column-names	Write column names in results		
--column-type-info	Display result set metadata		
--comments	Whether to retain or strip comments in statements sent to the server		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--connect-expired-password	Indicate to server that client can handle expired-password sandbox mode		
--connect-timeout	Number of seconds before connection timeout		
--database	The database to use		
--debug	Write debugging log; supported only if MySQL was built with debugging support		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--delimiter	Set the statement delimiter		

Option Name	Description	Introduced	Deprecated
--dns-srv-name	Use DNS SRV lookup for host information	8.0.22	
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--execute	Execute the statement and quit		
--fido-register-factor	Multifactor authentication factors for which registration must be done	8.0.27	
--force	Continue even if an SQL error occurs		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--histignore	Patterns specifying which statements to ignore for logging		
--host	Host on which MySQL server is located		
--html	Produce HTML output		
--ignore-spaces	Ignore spaces after function names		
--init-command	SQL statement to execute after connecting		
--line-numbers	Write line numbers for errors		
--load-data-local-dir	Directory for files named in LOAD DATA LOCAL statements	8.0.21	
--local-infile	Enable or disable for LOCAL capability for LOAD DATA		
--login-path	Read login path options from .mylogin.cnf		
--max-allowed-packet	Maximum packet length to send to or receive from server		
--max-join-size	The automatic limit for rows in a join when using --safe-updates		
--named-commands	Enable named mysql commands		
--net-buffer-length	Buffer size for TCP/IP and socket communication		

Option Name	Description	Introduced	Deprecated
--network-namespace	Specify network namespace	8.0.22	
--no-auto-rehash	Disable automatic rehashing		
--no-beep	Do not beep when errors occur		
--no-defaults	Read no option files		
--one-database	Ignore statements except those for the default database named on the command line		
--pager	Use the given command for paging query output		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--pipe	Connect to server using named pipe (Windows only)		
--plugin-authentication-kerberos-client-mode	Permit GSSAPI pluggable authentication through the MIT Kerberos library on Windows	8.0.32	
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--prompt	Set the prompt to the specified format		
--protocol	Transport protocol to use		
--quick	Do not cache each query result		
--raw	Write column values without escape conversion		

Option Name	Description	Introduced	Deprecated
--reconnect	If the connection to the server is lost, automatically try to reconnect		
--safe-updates, --i-am-a-dummy	Allow only UPDATE and DELETE statements that specify key values		
--select-limit	The automatic limit for SELECT statements when using --safe-updates		
--server-public-key-path	Path name to file containing RSA public key		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--show-warnings	Show warnings after each statement if there are any		
--sigint-ignore	Ignore SIGINT signals (typically the result of typing Control+C)		
--silent	Silent mode		
--skip-auto-rehash	Disable automatic rehashing		
--skip-column-names	Do not write column names in results		
--skip-line-numbers	Skip line numbers for errors		
--skip-named-commands	Disable named mysql commands		
--skip-pager	Disable paging		
--skip-reconnect	Disable reconnecting		
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		

Option Name	Description	Introduced	Deprecated
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--syslog	Log interactive statements to syslog		
--table	Display output in tabular format		
--tee	Append a copy of output to named file		
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--unbuffered	Flush the buffer after each query		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		
--version	Display version information and exit		
--vertical	Print query output rows vertically (one line per column value)		
--wait	If the connection cannot be established, wait and retry instead of aborting		
--xml	Produce XML output		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`, `-?`

  Display a help message and exit.

- `--auto-rehash`

  Enable automatic rehashing. This option is on by default, which enables database, table, and column name completion. Use `--disable-auto-rehash` to disable rehashing. That causes `mysql` to start faster, but you must issue the `rehash` command or its `\#` shortcut if you want to use name completion.

  To complete a name, enter the first part and press Tab. If the name is unambiguous, `mysql` completes it. Otherwise, you can press Tab again to see the possible names that begin with what you have typed so far. Completion does not occur if there is no default database.

  

  Note

  This feature requires a MySQL client that is compiled with the **readline** library. Typically, the **readline** library is not available on Windows.

- `--auto-vertical-output`

  Cause result sets to be displayed vertically if they are too wide for the current window, and using normal tabular format otherwise. (This applies to statements terminated by `;` or `\G`.)

- `--batch`, `-B`

  Print results using tab as the column separator, with each row on a new line. With this option, `mysql` does not use the history file.

  Batch mode results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the `--raw` option.

- `--binary-as-hex`

  When this option is given, `mysql` displays binary data using hexadecimal notation (`0x`*`value`*). This occurs whether the overall output display format is tabular, vertical, HTML, or XML.

  `--binary-as-hex` when enabled affects display of all binary strings, including those returned by functions such as `CHAR()` and `UNHEX()`. The following example demonstrates this using the ASCII code for `A` (65 decimal, 41 hexadecimal):

  - `--binary-as-hex` disabled:

    ```
    mysql> SELECT CHAR(0x41), UNHEX('41');
    +------------+-------------+
    | CHAR(0x41) | UNHEX('41') |
    +------------+-------------+
    | A          | A           |
    +------------+-------------+
    ```

  - `--binary-as-hex` enabled:

    ```
    mysql> SELECT CHAR(0x41), UNHEX('41');
    +-----------------------+-------------------------+
    | CHAR(0x41)            | UNHEX('41')             |
    +-----------------------+-------------------------+
    | 0x41                  | 0x41                    |
    +-----------------------+-------------------------+
    ```

  To write a binary string expression so that it displays as a character string regardless of whether `--binary-as-hex` is enabled, use these techniques:

  - The `CHAR()` function has a `USING` *`charset`* clause:

```
mysql> SELECT CHAR(0x41 USING utf8mb4);
+--------------------------+
| CHAR(0x41 USING utf8mb4) |
+--------------------------+
| A                        |
+--------------------------+
```

- More generally, use `CONVERT()` to convert an expression to a given character set:

```
mysql> SELECT CONVERT(UNHEX('41') USING utf8mb4);
+-----------------------------------+
| CONVERT(UNHEX('41') USING utf8mb4) |
+-----------------------------------+
| A                                 |
+-----------------------------------+
```

As of MySQL 8.0.19, when `mysql` operates in interactive mode, this option is enabled by default. In addition, output from the `status` (or `\s`) command includes this line when the option is enabled implicitly or explicitly:

```
Binary data as: Hexadecimal
```

To disable hexadecimal notation, use `--skip-binary-as-hex`

- `--binary-mode`

  This option helps when processing `mysqlbinlog` output that may contain `BLOB` values. By default, `mysql` translates `\r\n` in statement strings to `\n` and interprets `\0` as the statement terminator. `--binary-mode` disables both features. It also disables all `mysql` commands except `charset` and `delimiter` in noninteractive mode (for input piped to `mysql` or loaded using the `source` command).

- `--bind-address=`*`ip_address`*

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=`*`dir_name`*

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--column-names`

  Write column names in results.

- `--column-type-info`

  Display result set metadata. This information corresponds to the contents of C API `MYSQL_FIELD` data structures. See C API Basic Data Structures.

- `--comments`, `-c`

  Whether to strip or preserve comments in statements sent to the server. The default is `--skip-comments` (strip comments), enable with `--comments` (preserve comments).

  

  Note

  The `mysql` client always passes optimizer hints to the server, regardless of whether this option is given.

  Comment stripping is deprecated. Expect this feature and the options to control it to be removed in a future MySQL release.

- `--compress`, `-C`

Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

As of MySQL 8.0.18, this option is deprecated. Expect it be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--connect-expired-password`

  Indicate to the server that the client can handle sandbox mode if the account used to connect has an expired password. This can be useful for noninteractive invocations of `mysql` because normally the server disconnects noninteractive clients that attempt to connect using an account with an expired password. (See Section 6.2.16, “Server Handling of Expired Passwords”.)

- `--connect-timeout=value`

  The number of seconds before connection timeout. (Default value is `0`.)

- `--database=db_name`, `-D db_name`

  The database to use. This is useful primarily in an option file.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:o,/tmp/mysql.trace`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`, `-T`

  Print debugging information and memory and CPU usage statistics when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set for the client and connection.

  This option can be useful if the operating system uses one character set and the `mysql` client by default uses another. In this case, output may be formatted incorrectly. You can usually fix such issues by using this option to force the client to use the system character set instead.

  For more information, see Section 10.4, “Connection Character Sets and Collations”, and Section 10.15, “Character Set Configuration”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysql` normally reads the `[client]` and `[mysql]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysql` also reads the `[client_other]` and `[mysql_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--delimiter=str`

  Set the statement delimiter. The default is the semicolon character (`;`).

- `--disable-named-commands`

  Disable named commands. Use the `\*` form only, or use named commands only at the beginning of a line ending with a semicolon (`;`). `mysql` starts with this option *enabled* by default. However, even with this option, long-format commands still work from the first line. See Section 4.5.1.2, “mysql Client Commands”.

- `--dns-srv-name=name`

  Specifies the name of a DNS SRV record that determines the candidate hosts to use for establishing a connection to a MySQL server. For information about DNS SRV support in MySQL, see Section 4.2.6, “Connecting to the Server Using DNS SRV Records”.

  Suppose that DNS is configured with this SRV information for the `example.com` domain:

```
Name                    TTL   Class   Priority Weight Port Target
_mysql._tcp.example.com. 86400 IN SRV  0        5      3306 host1.example.com
_mysql._tcp.example.com. 86400 IN SRV  0        10     3306 host2.example.com
_mysql._tcp.example.com. 86400 IN SRV  10       5      3306 host3.example.com
```

```
_mysql._tcp.example.com. 86400 IN SRV  20       5      3306 host4.example.com
```

To use that DNS SRV record, invoke `mysql` like this:

```
mysql --dns-srv-name=_mysql._tcp.example.com
```

`mysql` then attempts a connection to each server in the group until a successful connection is established. A failure to connect occurs only if a connection cannot be established to any of the servers. The priority and weight values in the DNS SRV record determine the order in which servers should be tried.

When invoked with `--dns-srv-name`, `mysql` attempts to establish TCP connections only.

The `--dns-srv-name` option takes precedence over the `--host` option if both are given. `--dns-srv-name` causes connection establishment to use the `mysql_real_connect_dns_srv()` C API function rather than `mysql_real_connect()`. However, if the `connect` command is subsequently used at runtime and specifies a host name argument, that host name takes precedence over any `--dns-srv-name` option given at `mysql` startup to specify a DNS SRV record.

This option was added in MySQL 8.0.22.

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--execute=`*`statement`*, `-e` *`statement`*

  Execute the statement and quit. The default output format is like that produced with `--batch`. See Section 4.2.2.1, “Using Options on the Command Line”, for some examples. With this option, `mysql` does not use the history file.

- `--fido-register-factor=`*`value`*

  The factor or factors for which FIDO device registration must be performed. This option value must be a single value, or two values separated by commas. Each value must be 2 or 3, so the permitted option values are `'2'`, `'3'`, `'2,3'` and `'3,2'`.

  For example, an account that requires registration for a 3rd authentication factor invokes the `mysql` client as follows:

  ```
  mysql --user=user_name --fido-register-factor=3
  ```

  An account that requires registration for a 2nd and 3rd authentication factor invokes the `mysql` client as follows:

  ```
  mysql --user=user_name --fido-register-factor=2,3
  ```

  If registration is successful, a connection is established. If there is an authentication factor with a pending registration, a connection is placed into pending registration mode when attempting to

connect to the server. In this case, disconnect and reconnect with the correct `--fido-register-factor` value to complete the registration.

Registration is a two step process comprising *initiate registration* and *finish registration* steps. The initiate registration step executes this statement:

```
ALTER USER user factor INITIATE REGISTRATION
```

The statement returns a result set containing a 32 byte challenge, the user name, and the relying party ID (see `authentication_fido_rp_id`).

The finish registration step executes this statement:

```
ALTER USER user factor FINISH REGISTRATION SET CHALLENGE_RESPONSE AS 'auth_string'
```

The statement completes the registration and sends the following information to the server as part of the *`auth_string`*: authenticator data, an optional attestation certificate in X.509 format, and a signature.

The initiate and registration steps must be performed in a single connection, as the challenge received by the client during the initiate step is saved to the client connection handler. Registration would fail if the registration step was performed by a different connection. The `--fido-register-factor` option executes both the initiate and registration steps, which avoids the failure scenario described above and prevents having to execute the `ALTER USER` initiate and registration statements manually.

The `--fido-register-factor` option is only available for the `mysql` client and MySQL Shell. Other MySQL client programs do not support it.

For related information, see Using FIDO Authentication.

- `--force`, `-f`

  Continue even if an SQL error occurs.

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=`*`file_name`* is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--histignore`

  A list of one or more colon-separated patterns specifying statements to ignore for logging purposes. These patterns are added to the default pattern list (`"*IDENTIFIED*:*PASSWORD*"`). The value specified for this option affects logging of statements written to the history file, and to `syslog` if the `--syslog` option is given. For more information, see Section 4.5.1.3, “mysql Client Logging”.

- `--host=host_name`, `-h host_name`

  Connect to the MySQL server on the given host.

  The `--dns-srv-name` option takes precedence over the `--host` option if both are given. `--dns-srv-name` causes connection establishment to use the `mysql_real_connect_dns_srv()` C API function rather than `mysql_real_connect()`. However, if the `connect` command is subsequently used at runtime and specifies a host name argument, that host name takes precedence over any `--dns-srv-name` option given at `mysql` startup to specify a DNS SRV record.

- `--html`, `-H`

  Produce HTML output.

- `--ignore-spaces`, `-i`

  Ignore spaces after function names. The effect of this is described in the discussion for the `IGNORE_SPACE` SQL mode (see Section 5.1.11, “Server SQL Modes”).

- `--init-command=str`

  SQL statement to execute after connecting to the server. If auto-reconnect is enabled, the statement is executed again after reconnection occurs.

- `--line-numbers`

  Write line numbers for errors. Disable this with `--skip-line-numbers`.

- `--load-data-local-dir=dir_name`

  This option affects the client-side `LOCAL` capability for `LOAD DATA` operations. It specifies the directory in which files named in `LOAD DATA LOCAL` statements must be located. The effect of `--load-data-local-dir` depends on whether `LOCAL` data loading is enabled or disabled:

  - If `LOCAL` data loading is enabled, either by default in the MySQL client library or by specifying `--local-infile[=1]`, the `--load-data-local-dir` option is ignored.
  - If `LOCAL` data loading is disabled, either by default in the MySQL client library or by specifying `--local-infile=0`, the `--load-data-local-dir` option applies.

  When `--load-data-local-dir` applies, the option value designates the directory in which local data files must be located. Comparison of the directory path name and the path name of files to be loaded is case-sensitive regardless of the case sensitivity of the underlying file system. If the option value is the empty string, it names no directory, with the result that no files are permitted for local data loading.

  For example, to explicitly disable local data loading except for files located in the `/my/local/data` directory, invoke `mysql` like this:

  ```
  mysql --local-infile=0 --load-data-local-dir=/my/local/data
  ```

  When both `--local-infile` and `--load-data-local-dir` are given, the order in which they are given does not matter.

  Successful use of `LOCAL` load operations within `mysql` also requires that the server permits local loading; see Section 6.1.6, “Security Considerations for LOAD DATA LOCAL”

  The `--load-data-local-dir` option was added in MySQL 8.0.21.

- `--local-infile[={0|1}]`

  By default, `LOCAL` capability for `LOAD DATA` is determined by the default compiled into the MySQL client library. To enable or disable `LOCAL` data loading explicitly, use the `--local-infile` option.

When given with no value, the option enables `LOCAL` data loading. When given as `--local-infile=0` or `--local-infile=1`, the option disables or enables `LOCAL` data loading.

If `LOCAL` capability is disabled, the `--load-data-local-dir` option can be used to permit restricted local loading of files located in a designated directory.

Successful use of `LOCAL` load operations within `mysql` also requires that the server permits local loading; see Section 6.1.6, “Security Considerations for LOAD DATA LOCAL”

- `--login-path=`*`name`*

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--max-allowed-packet=`*`value`*

  The maximum size of the buffer for client/server communication. The default is 16MB, the maximum is 1GB.

- `--max-join-size=`*`value`*

  The automatic limit for rows in a join when using `--safe-updates`. (Default value is 1,000,000.)

- `--named-commands`, `-G`

  Enable named `mysql` commands. Long-format commands are permitted, not just short-format commands. For example, `quit` and `\q` both are recognized. Use `--skip-named-commands` to disable named commands. See Section 4.5.1.2, “mysql Client Commands”.

- `--net-buffer-length=`*`value`*

  The buffer size for TCP/IP and socket communication. (Default value is 16KB.)

- `--network-namespace=`*`name`*

  The network namespace to use for TCP/IP connections. If omitted, the connection uses the default (global) namespace. For information about network namespaces, see Section 5.1.14, “Network Namespace Support”.

  This option was added in MySQL 8.0.22. It is available only on platforms that implement network namespace support.

- `--no-auto-rehash`, `-A`

  This has the same effect as `--skip-auto-rehash`. See the description for `--auto-rehash`.

- `--no-beep`, `-b`

  Do not beep when errors occur.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults`

is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--one-database`, `-o`

  Ignore statements except those that occur while the default database is the one named on the command line. This option is rudimentary and should be used with care. Statement filtering is based only on `USE` statements.

  Initially, `mysql` executes statements in the input because specifying a database *`db_name`* on the command line is equivalent to inserting `USE db_name` at the beginning of the input. Then, for each `USE` statement encountered, `mysql` accepts or rejects following statements depending on whether the database named is the one on the command line. The content of the statements is immaterial.

  Suppose that `mysql` is invoked to process this set of statements:

  ```
  DELETE FROM db2.t2;
  USE db2;
  DROP TABLE db1.t1;
  CREATE TABLE db1.t1 (i INT);
  USE db1;
  INSERT INTO t1 (i) VALUES(1);
  CREATE TABLE db2.t1 (j INT);
  ```

  If the command line is `mysql --force --one-database db1`, `mysql` handles the input as follows:

  - The `DELETE` statement is executed because the default database is `db1`, even though the statement names a table in a different database.
  - The `DROP TABLE` and `CREATE TABLE` statements are not executed because the default database is not `db1`, even though the statements name a table in `db1`.
  - The `INSERT` and `CREATE TABLE` statements are executed because the default database is `db1`, even though the `CREATE TABLE` statement names a table in a different database.

- `--pager[=`*`command`*`]`

  Use the given command for paging query output. If the command is omitted, the default pager is the value of your `PAGER` environment variable. Valid pagers are `less`, `more`, `cat [> filename]`, and so forth. This option works only on Unix and only in interactive mode. To disable paging, use `--skip-pager`. Section 4.5.1.2, “mysql Client Commands”, discusses output paging further.

- `--password[=`*`password`*`]`, `-p[`*`password`*`]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysql` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysql` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysql` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysql` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-authentication-kerberos-client-mode=value`

  On Windows, the `authentication_kerberos_client` authentication plugin supports this plugin option. It provides two possible values that the client user can set at runtime: `SSPI` and `GSSAPI`.

  The default value for the client-side plugin option uses Security Support Provider Interface (SSPI), which is capable of acquiring credentials from the Windows in-memory cache. Alternatively, the client user can select a mode that supports Generic Security Service Application Program Interface (GSSAPI) through the MIT Kerberos library on Windows. GSSAPI is capable of acquiring cached credentials previously generated by using the `kinit` command.

  For more information, see Commands for Windows Clients in GSSAPI Mode.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysql` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--prompt=format_str`

  Set the prompt to the specified format. The default is `mysql>`. The special sequences that the prompt can contain are described in Section 4.5.1.2, “mysql Client Commands”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--quick`, `-q`

  Do not cache each query result, print each row as it is received. This may slow down the server if the output is suspended. With this option, `mysql` does not use the history file.

- `--raw`, `-r`

  For tabular output, the “boxing” around columns enables one column value to be distinguished from another. For nontabular output (such as is produced in batch mode or when the `--batch` or `--silent` option is given), special characters are escaped in the output so they can be identified easily. Newline, tab, `NUL`, and backslash are written as `\n`, `\t`, `\0`, and `\\`. The `--raw` option disables this character escaping.

  The following example demonstrates tabular versus nontabular output and the use of raw mode to disable escaping:

```
% mysql
mysql> SELECT CHAR(92);
+----------+
| CHAR(92) |
+----------+
| \        |
+----------+

% mysql -s
mysql> SELECT CHAR(92);
CHAR(92)
\\

% mysql -s -r
mysql> SELECT CHAR(92);
CHAR(92)
\
```

- `--reconnect`

  If the connection to the server is lost, automatically try to reconnect. A single reconnect attempt is made each time the connection is lost. To suppress reconnection behavior, use `--skip-reconnect`.

- `--safe-updates`, `--i-am-a-dummy`, `-U`

  If this option is enabled, `UPDATE` and `DELETE` statements that do not use a key in the `WHERE` clause or a `LIMIT` clause produce an error. In addition, restrictions are placed on `SELECT` statements that produce (or are estimated to produce) very large result sets. If you have set this option in an option file, you can use `--skip-safe-updates` on the command line to override it. For more information about this option, see Using Safe-Updates Mode (--safe-updates).

- `--select-limit=value`

  The automatic limit for `SELECT` statements when using `--safe-updates`. (Default value is 1,000.)

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--shared-memory-base-name=name`

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--show-warnings`

  Cause warnings to be shown after each statement if there are any. This option applies to interactive and batch mode.

- `--sigint-ignore`

  Ignore `SIGINT` signals (typically the result of typing **Control+C**).

  Without this option, typing **Control+C** interrupts the current statement if there is one, or cancels any partial input line otherwise.

- `--silent`, `-s`

  Silent mode. Produce less output. This option can be given multiple times to produce less and less output.

  This option results in nontabular output format and escaping of special characters. Escaping may be disabled by using raw mode; see the description for the `--raw` option.

- `--skip-column-names`, `-N`

  Do not write column names in results.

- `--skip-line-numbers`, `-L`

  Do not write line numbers for errors. Useful when you want to compare result files that include error messages.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--syslog`, `-j`

  This option causes `mysql` to send interactive statements to the system logging facility. On Unix, this is `syslog`; on Windows, it is the Windows Event Log. The destination where logged messages appear is system dependent. On Linux, the destination is often the `/var/log/messages` file.

  Here is a sample of output generated on Linux by using `--syslog`. This output is formatted for readability; each logged message actually takes a single line.

```
Mar  7 12:39:25 myhost MysqlClient[20824]:
  SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,
  DB_SERVER:'127.0.0.1', DB:'--', QUERY:'USE test;'
Mar  7 12:39:28 myhost MysqlClient[20824]:
  SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,
  DB_SERVER:'127.0.0.1', DB:'test', QUERY:'SHOW TABLES;'
```

  For more information, see Section 4.5.1.3, “mysql Client Logging”.

- `--table`, `-t`

  Display output in table format. This is the default for interactive use, but can be used to produce table output in batch mode.

- `--tee=file_name`

  Append a copy of output to the given file. This option works only in interactive mode. Section 4.5.1.2, “mysql Client Commands”, discusses tee files further.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option

depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--unbuffered`, `-n`

  Flush the buffer after each query.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

- `--verbose`, `-v`

  Verbose mode. Produce more output about what the program does. This option can be given multiple times to produce more and more output. (For example, `-v -v -v` produces table output format even in batch mode.)

- `--version`, `-V`

  Display version information and exit.

- `--vertical`, `-E`

  Print query output rows vertically (one line per column value). Without this option, you can specify vertical output for individual statements by terminating them with `\G`.

- `--wait`, `-w`

  If the connection cannot be established, wait and retry instead of aborting.

- `--xml`, `-X`

  Produce XML output.

```
<field name="column_name">NULL</field>
```

  The output when `--xml` is used with `mysql` matches that of `mysqldump --xml`. See Section 4.5.4, “mysqldump — A Database Backup Program”, for details.

  The XML output also uses an XML namespace, as shown here:

```
$> mysql --xml -uroot -e "SHOW VARIABLES LIKE 'version%'"
<?xml version="1.0"?>

<resultset statement="SHOW VARIABLES LIKE 'version%'" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instanc
<row>
<field name="Variable_name">version</field>
<field name="Value">5.0.40-debug</field>
</row>

<row>
<field name="Variable_name">version_comment</field>
<field name="Value">Source distribution</field>
</row>

<row>
```

```
<field name="Variable_name">version_compile_machine</field>
<field name="Value">i686</field>
</row>

<row>
<field name="Variable_name">version_compile_os</field>
<field name="Value">suse-linux-gnu</field>
</row>
</resultset>
```

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

4.5.1.2 mysql Client Commands

`mysql` sends each SQL statement that you issue to the server to be executed. There is also a set of commands that `mysql` itself interprets. For a list of these commands, type `help` or `\h` at the `mysql>` prompt:

```
mysql> help

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given
               outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing
               binlog with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.
query_attributes Sets string parameters (name1 value1 name2 value2 ...)
for the next query to pick up.
ssl_session_data_print Serializes the current SSL session data to stdout
or file.

For server side help, type 'help contents'
```

If `mysql` is invoked with the `--binary-mode` option, all `mysql` commands are disabled except `charset` and `delimiter` in noninteractive mode (for input piped to `mysql` or loaded using the `source` command).

Each command has both a long and short form. The long form is not case-sensitive; the short form is. The long form can be followed by an optional semicolon terminator, but the short form should not.

The use of short-form commands within multiple-line `/* ... */` comments is not supported. Short-form commands do work within single-line `/*! ... */` version comments, as do `/*+ ... */` optimizer-hint comments, which are stored in object definitions. If there is a concern that optimizer-hint comments may be stored in object definitions so that dump files when reloaded with `mysql` would result in execution of such commands, either invoke `mysql` with the `--binary-mode` option or use a reload client other than `mysql`.

- `help [arg]`, `\h [arg]`, `\? [arg]`, `? [arg]`

  Display a help message listing the available `mysql` commands.

  If you provide an argument to the `help` command, `mysql` uses it as a search string to access server-side help from the contents of the MySQL Reference Manual. For more information, see Section 4.5.1.4, “mysql Client Server-Side Help”.

- `charset charset_name`, `\C charset_name`

  Change the default character set and issue a `SET NAMES` statement. This enables the character set to remain synchronized on the client and server if `mysql` is run with auto-reconnect enabled (which is not recommended), because the specified character set is used for reconnects.

- `clear`, `\c`

  Clear the current input. Use this if you change your mind about executing the statement that you are entering.

- `connect [db_name [host_name]]`, `\r [db_name [host_name]]`

  Reconnect to the server. The optional database name and host name arguments may be given to specify the default database or the host where the server is running. If omitted, the current values are used.

  If the `connect` command specifies a host name argument, that host takes precedence over any `--dns-srv-name` option given at `mysql` startup to specify a DNS SRV record.

- `delimiter str`, `\d str`

  Change the string that `mysql` interprets as the separator between SQL statements. The default is the semicolon character (`;`).

  The delimiter string can be specified as an unquoted or quoted argument on the `delimiter` command line. Quoting can be done with either single quote (`'`), double quote (`"`), or backtick (`` ` ``) characters. To include a quote within a quoted string, either quote the string with a different quote character or escape the quote with a backslash (`\`) character. Backslash should be avoided outside of quoted strings because it is the escape character for MySQL. For an unquoted argument, the delimiter is read up to the first space or end of line. For a quoted argument, the delimiter is read up to the matching quote on the line.

  `mysql` interprets instances of the delimiter string as a statement delimiter anywhere it occurs, except within quoted strings. Be careful about defining a delimiter that might occur within other words. For example, if you define the delimiter as `X`, it is not possible to use the word `INDEX` in statements. `mysql` interprets this as `INDE` followed by the delimiter `X`.

  When the delimiter recognized by `mysql` is set to something other than the default of `;`, instances of that character are sent to the server without interpretation. However, the server itself still interprets `;` as a statement delimiter and processes statements accordingly. This behavior on the server side comes into play for multiple-statement execution (see Multiple Statement Execution Support), and

for parsing the body of stored procedures and functions, triggers, and events (see Section 25.1, “Defining Stored Programs”).

- `edit`, `\e`

  Edit the current input statement. `mysql` checks the values of the `EDITOR` and `VISUAL` environment variables to determine which editor to use. The default editor is `vi` if neither variable is set.

  The `edit` command works only in Unix.

- `ego`, `\G`

  Send the current statement to the server to be executed and display the result using vertical format.

- `exit`, `\q`

  Exit `mysql`.

- `go`, `\g`

  Send the current statement to the server to be executed.

- `nopager`, `\n`

  Disable output paging. See the description for `pager`.

  The `nopager` command works only in Unix.

- `notee`, `\t`

  Disable output copying to the tee file. See the description for `tee`.

- `nowarning`, `\w`

  Disable display of warnings after each statement.

- `pager [`*`command`*`]`, `\P [`*`command`*`]`

  Enable output paging. By using the `--pager` option when you invoke `mysql`, it is possible to browse or search query results in interactive mode with Unix programs such as `less`, `more`, or any other similar program. If you specify no value for the option, `mysql` checks the value of the `PAGER` environment variable and sets the pager to that. Pager functionality works only in interactive mode.

  Output paging can be enabled interactively with the `pager` command and disabled with `nopager`. The command takes an optional argument; if given, the paging program is set to that. With no argument, the pager is set to the pager that was set on the command line, or `stdout` if no pager was specified.

  Output paging works only in Unix because it uses the `popen()` function, which does not exist on Windows. For Windows, the `tee` option can be used instead to save query output, although it is not as convenient as `pager` for browsing output in some situations.

- `print`, `\p`

  Print the current input statement without executing it.

- `prompt [`*`str`*`]`, `\R [`*`str`*`]`

  Reconfigure the `mysql` prompt to the given string. The special character sequences that can be used in the prompt are described later in this section.

  If you specify the `prompt` command with no argument, `mysql` resets the prompt to the default of `mysql>`.

- `query_attributes name value [name value ...]`

  Define query attributes that apply to the next query sent to the server. For discussion of the purpose and use of query attributes, see Section 9.6, “Query Attributes”.

  The `query_attributes` command follows these rules:

  - The format and quoting rules for attribute names and values are the same as for the `delimiter` command.
  - The command permits up to 32 attribute name/value pairs. Names and values may be up to 1024 characters long. If a name is given without a value, an error occurs.
  - If multiple `query_attributes` commands are issued prior to query execution, only the last command applies. After sending the query, `mysql` clears the attribute set.
  - If multiple attributes are defined with the same name, attempts to retrieve the attribute value have an undefined result.
  - An attribute defined with an empty name cannot be retrieved by name.
  - If a reconnect occurs while `mysql` executes the query, `mysql` restores the attributes after reconnecting so the query can be executed again with the same attributes.

- `quit`, `\q`

  Exit `mysql`.

- `rehash`, `\#`

  Rebuild the completion hash that enables database, table, and column name completion while you are entering statements. (See the description for the `--auto-rehash` option.)

- `resetconnection`, `\x`

  Reset the connection to clear the session state. This includes clearing any current query attributes defined using the `query_attributes` command.

  Resetting a connection has effects similar to `mysql_change_user()` or an auto-reconnect except that the connection is not closed and reopened, and re-authentication is not done. See mysql_change_user(), and Automatic Reconnection Control.

  This example shows how `resetconnection` clears a value maintained in the session state:

```
mysql> SELECT LAST_INSERT_ID(3);
+------------------+
| LAST_INSERT_ID(3) |
+------------------+
|                3 |
+------------------+

mysql> SELECT LAST_INSERT_ID();
+-----------------+
| LAST_INSERT_ID() |
+-----------------+
|               3 |
+-----------------+

mysql> resetconnection;

mysql> SELECT LAST_INSERT_ID();
+-----------------+
| LAST_INSERT_ID() |
+-----------------+
|               0 |
```

```
+-----------------+
```

- `source` *`file_name`*, `\.` *`file_name`*

  Read the named file and executes the statements contained therein. On Windows, specify path name separators as `/` or `\\`.

  Quote characters are taken as part of the file name itself. For best results, the name should not include space characters.

- `ssl_session_data_print [`*`file_name`*`]`

  Fetches, serializes, and optionally stores the session data of a successful connection. The optional file name and arguments may be given to specify the file to store serialized session data. If omitted, the session data is printed to `stdout`.

  If the MySQL session is configured for reuse, session data from the file is deserialized and supplied to the `connect` command to reconnect. When the session is reused successfully, the `status` command contains a row showing `SSL session reused: true` while the client remains reconnected to the server.

- `status`, `\s`

  Provide status information about the connection and the server you are using. If you are running with `--safe-updates` enabled, `status` also prints the values for the `mysql` variables that affect your queries.

- `system` *`command`*, `\!` *`command`*

  Execute the given command using your default command interpreter.

  Prior to MySQL 8.0.19, the `system` command works only in Unix. As of 8.0.19, it also works on Windows.

- `tee [`*`file_name`*`]`, `\T [`*`file_name`*`]`

  By using the `--tee` option when you invoke `mysql`, you can log statements and their output. All the data displayed on the screen is appended into a given file. This can be very useful for debugging purposes also. `mysql` flushes results to the file after each statement, just before it prints its next prompt. Tee functionality works only in interactive mode.

  You can enable this feature interactively with the `tee` command. Without a parameter, the previous file is used. The `tee` file can be disabled with the `notee` command. Executing `tee` again re-enables logging.

- `use` *`db_name`*, `\u` *`db_name`*

  Use *`db_name`* as the default database.

- `warnings`, `\W`

  Enable display of warnings after each statement (if there are any).

Here are a few tips about the `pager` command:

- You can use it to write to a file and the results go only to the file:

```
mysql> pager cat > /tmp/log.txt
```

  You can also pass any options for the program that you want to use as your pager:

```
mysql> pager less -n -i -S
```

- In the preceding example, note the `-S` option. You may find it very useful for browsing wide query results. Sometimes a very wide result set is difficult to read on the screen. The `-S` option to `less` can make the result set much more readable because you can scroll it horizontally using the left-arrow and right-arrow keys. You can also use `-S` interactively within `less` to switch the horizontal-browse mode on and off. For more information, read the `less` manual page:

  ```
  man less
  ```

- The `-F` and `-X` options may be used with `less` to cause it to exit if output fits on one screen, which is convenient when no scrolling is necessary:

  ```
  mysql> pager less -n -i -S -F -X
  ```

- You can specify very complex pager commands for handling query output:

  ```
  mysql> pager cat | tee /dr1/tmp/res.txt \
            | tee /dr2/tmp/res2.txt | less -n -i -S
  ```

  In this example, the command would send query results to two files in two different directories on two different file systems mounted on `/dr1` and `/dr2`, yet still display the results onscreen using `less`.

You can also combine the `tee` and `pager` functions. Have a `tee` file enabled and `pager` set to `less`, and you are able to browse the results using the `less` program and still have everything appended into a file the same time. The difference between the Unix `tee` used with the `pager` command and the `mysql` built-in `tee` command is that the built-in `tee` works even if you do not have the Unix `tee` available. The built-in `tee` also logs everything that is printed on the screen, whereas the Unix `tee` used with `pager` does not log quite that much. Additionally, `tee` file logging can be turned on and off interactively from within `mysql`. This is useful when you want to log some queries to a file, but not others.

The `prompt` command reconfigures the default `mysql>` prompt. The string for defining the prompt can contain the following special sequences.

Option	Description
`\C`	The current connection identifier
`\c`	A counter that increments for each statement you issue
`\D`	The full current date
`\d`	The default database
`\h`	The server host
`\l`	The current delimiter
`\m`	Minutes of the current time
`\n`	A newline character
`\O`	The current month in three-letter format (Jan, Feb, …)
`\o`	The current month in numeric format
`\P`	am/pm
`\p`	The current TCP/IP port or socket file
`\R`	The current time, in 24-hour military time (0–23)
`\r`	The current time, standard 12-hour time (1–12)
`\S`	Semicolon
`\s`	Seconds of the current time
`\T`	Print an asterisk (`*`) if the current session is inside a transaction block (from MySQL 8.0.28)

Option	Description
`\t`	A tab character
`\U`	Your full *`user_name@host_name`* account name
`\u`	Your user name
`\v`	The server version
`\w`	The current day of the week in three-letter format (Mon, Tue, …)
`\Y`	The current year, four digits
`\y`	The current year, two digits
`\_`	A space
`\ `	A space (a space follows the backslash)
`\'`	Single quote
`\"`	Double quote
`\\`	A literal `\` backslash character
`\x`	*`x`*, for any "*`x`*" not listed above

You can set the prompt in several ways:

- *Use an environment variable.* You can set the `MYSQL_PS1` environment variable to a prompt string. For example:

```
export MYSQL_PS1="(\u@\h) [\d]> "
```

- *Use a command-line option.* You can set the `--prompt` option on the command line to `mysql`. For example:

```
$> mysql --prompt="(\u@\h) [\d]> "
(user@host) [database]>
```

- *Use an option file.* You can set the `prompt` option in the `[mysql]` group of any MySQL option file, such as `/etc/my.cnf` or the `.my.cnf` file in your home directory. For example:

```
[mysql]
prompt=(\\u@\\h) [\\d]>\\_
```

  In this example, note that the backslashes are doubled. If you set the prompt using the `prompt` option in an option file, it is advisable to double the backslashes when using the special prompt options. There is some overlap in the set of permissible prompt options and the set of special escape sequences that are recognized in option files. (The rules for escape sequences in option files are listed in Section 4.2.2.2, "Using Option Files".) The overlap may cause you problems if you use single backslashes. For example, `\s` is interpreted as a space rather than as the current seconds value. The following example shows how to define a prompt within an option file to include the current time in *`hh:mm:ss>`* format:

```
[mysql]
prompt="\\r:\\m:\\s> "
```

- *Set the prompt interactively.* You can change your prompt interactively by using the `prompt` (or `\R`) command. For example:

```
mysql> prompt (\u@\h) [\d]>\_
PROMPT set to '(\u@\h) [\d]>\_'
(user@host) [database]>
(user@host) [database]> prompt
Returning to default PROMPT of mysql>
mysql>
```

4.5.1.3 mysql Client Logging

The `mysql` client can do these types of logging for statements executed interactively:

- On Unix, `mysql` writes the statements to a history file. By default, this file is named `.mysql_history` in your home directory. To specify a different file, set the value of the `MYSQL_HISTFILE` environment variable.
- On all platforms, if the `--syslog` option is given, `mysql` writes the statements to the system logging facility. On Unix, this is `syslog`; on Windows, it is the Windows Event Log. The destination where logged messages appear is system dependent. On Linux, the destination is often the `/var/log/messages` file.

The following discussion describes characteristics that apply to all logging types and provides information specific to each logging type.

- How Logging Occurs
- Controlling the History File
- syslog Logging Characteristics

How Logging Occurs

For each enabled logging destination, statement logging occurs as follows:

- Statements are logged only when executed interactively. Statements are noninteractive, for example, when read from a file or a pipe. It is also possible to suppress statement logging by using the `--batch` or `--execute` option.
- Statements are ignored and not logged if they match any pattern in the “ignore” list. This list is described later.
- `mysql` logs each nonignored, nonempty statement line individually.
- If a nonignored statement spans multiple lines (not including the terminating delimiter), `mysql` concatenates the lines to form the complete statement, maps newlines to spaces, and logs the result, plus a delimiter.

Consequently, an input statement that spans multiple lines can be logged twice. Consider this input:

```
mysql> SELECT
    -> 'Today is'
    -> ,
    -> CURDATE()
    -> ;
```

In this case, `mysql` logs the “SELECT”, “'Today is'”, “,”, “CURDATE()”, and “;” lines as it reads them. It also logs the complete statement, after mapping `SELECT\n'Today is'\n,\nCURDATE()` to `SELECT 'Today is' , CURDATE()`, plus a delimiter. Thus, these lines appear in logged output:

```
SELECT
'Today is'
,
CURDATE()
;
SELECT 'Today is' , CURDATE();
```

`mysql` ignores for logging purposes statements that match any pattern in the “ignore” list. By default, the pattern list is `"*IDENTIFIED*:*PASSWORD*"`, to ignore statements that refer to passwords. Pattern matching is not case-sensitive. Within patterns, two characters are special:

- `?` matches any single character.
- `*` matches any sequence of zero or more characters.

To specify additional patterns, use the `--histignore` option or set the `MYSQL_HISTIGNORE` environment variable. (If both are specified, the option value takes precedence.) The value should be a list of one or more colon-separated patterns, which are appended to the default pattern list.

Patterns specified on the command line might need to be quoted or escaped to prevent your command interpreter from treating them specially. For example, to suppress logging for `UPDATE` and `DELETE` statements in addition to statements that refer to passwords, invoke `mysql` like this:

```
mysql --histignore="*UPDATE*:*DELETE*"
```

Controlling the History File

The `.mysql_history` file should be protected with a restrictive access mode because sensitive information might be written to it, such as the text of SQL statements that contain passwords. See Section 6.1.2.1, “End-User Guidelines for Password Security”. Statements in the file are accessible from the `mysql` client when the **up-arrow** key is used to recall the history. See Disabling Interactive History.

If you do not want to maintain a history file, first remove `.mysql_history` if it exists. Then use either of the following techniques to prevent it from being created again:

- Set the `MYSQL_HISTFILE` environment variable to `/dev/null`. To cause this setting to take effect each time you log in, put it in one of your shell's startup files.
- Create `.mysql_history` as a symbolic link to `/dev/null`; this need be done only once:

  ```
  ln -s /dev/null $HOME/.mysql_history
  ```

syslog Logging Characteristics

If the `--syslog` option is given, `mysql` writes interactive statements to the system logging facility. Message logging has the following characteristics.

Logging occurs at the “information” level. This corresponds to the `LOG_INFO` priority for `syslog` on Unix/Linux `syslog` capability and to `EVENTLOG_INFORMATION_TYPE` for the Windows Event Log. Consult your system documentation for configuration of your logging capability.

Message size is limited to 1024 bytes.

Messages consist of the identifier `MysqlClient` followed by these values:

- `SYSTEM_USER`

  The operating system user name (login name) or `--` if the user is unknown.

- `MYSQL_USER`

  The MySQL user name (specified with the `--user` option) or `--` if the user is unknown.

- `CONNECTION_ID`:

  The client connection identifier. This is the same as the `CONNECTION_ID()` function value within the session.

- `DB_SERVER`

  The server host or `--` if the host is unknown.

- `DB`

  The default database or `--` if no database has been selected.

- `QUERY`

  The text of the logged statement.

Here is a sample of output generated on Linux by using `--syslog`. This output is formatted for readability; each logged message actually takes a single line.

```
Mar  7 12:39:25 myhost MysqlClient[20824]:
  SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,
  DB_SERVER:'127.0.0.1', DB:'--', QUERY:'USE test;'
Mar  7 12:39:28 myhost MysqlClient[20824]:
  SYSTEM_USER:'oscar', MYSQL_USER:'my_oscar', CONNECTION_ID:23,
  DB_SERVER:'127.0.0.1', DB:'test', QUERY:'SHOW TABLES;'
```

4.5.1.4 mysql Client Server-Side Help

```
mysql> help search_string
```

If you provide an argument to the `help` command, `mysql` uses it as a search string to access server-side help from the contents of the MySQL Reference Manual. The proper operation of this command requires that the help tables in the `mysql` database be initialized with help topic information (see Section 5.1.17, “Server-Side Help Support”).

If there is no match for the search string, the search fails:

```
mysql> help me

Nothing found
Please try to run 'help contents' for a list of all accessible topics
```

Use `help contents` to see a list of the help categories:

```
mysql> help contents
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the
following categories:
   Account Management
   Administration
   Data Definition
   Data Manipulation
   Data Types
   Functions
   Functions and Modifiers for Use with GROUP BY
   Geographic Features
   Language Structure
   Plugins
   Storage Engines
   Stored Routines
   Table Maintenance
   Transactions
   Triggers
```

If the search string matches multiple items, `mysql` shows a list of matching topics:

```
mysql> help logs
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following topics:
   SHOW
   SHOW BINARY LOGS
   SHOW ENGINE
   SHOW LOGS
```

Use a topic as the search string to see the help entry for that topic:

```
mysql> help show binary logs
Name: 'SHOW BINARY LOGS'
Description:
Syntax:
SHOW BINARY LOGS
SHOW MASTER LOGS

Lists the binary log files on the server. This statement is used as
part of the procedure described in [purge-binary-logs], that shows how
to determine which logs can be purged.
```

```
mysql> SHOW BINARY LOGS;
+---------------+-----------+-----------+
| Log_name      | File_size | Encrypted |
+---------------+-----------+-----------+
| binlog.000015 |    724935 | Yes       |
| binlog.000016 |    733481 | Yes       |
+---------------+-----------+-----------+
```

The search string can contain the wildcard characters `%` and `_`. These have the same meaning as for pattern-matching operations performed with the `LIKE` operator. For example, `HELP rep%` returns a list of topics that begin with `rep`:

```
mysql> HELP rep%
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
topics:
   REPAIR TABLE
   REPEAT FUNCTION
   REPEAT LOOP
   REPLACE
   REPLACE FUNCTION
```

4.5.1.5 Executing SQL Statements from a Text File

The `mysql` client typically is used interactively, like this:

```
mysql db_name
```

However, it is also possible to put your SQL statements in a file and then tell `mysql` to read its input from that file. To do so, create a text file *`text_file`* that contains the statements you wish to execute. Then invoke `mysql` as shown here:

```
mysql db_name < text_file
```

If you place a `USE db_name` statement as the first statement in the file, it is unnecessary to specify the database name on the command line:

```
mysql < text_file
```

If you are already running `mysql`, you can execute an SQL script file using the `source` command or `\.` command:

```
mysql> source file_name
mysql> \. file_name
```

Sometimes you may want your script to display progress information to the user. For this you can insert statements like this:

```
SELECT '<info_to_display>' AS ' ';
```

The statement shown outputs `<info_to_display>`.

You can also invoke `mysql` with the `--verbose` option, which causes each statement to be displayed before the result that it produces.

`mysql` ignores Unicode byte order mark (BOM) characters at the beginning of input files. Previously, it read them and sent them to the server, resulting in a syntax error. Presence of a BOM does not cause `mysql` to change its default character set. To do that, invoke `mysql` with an option such as `--default-character-set=utf8mb4`.

For more information about batch mode, see Section 3.5, “Using mysql in Batch Mode”.

4.5.1.6 mysql Client Tips

This section provides information about techniques for more effective use of `mysql` and about `mysql` operational behavior.

Input-Line Editing

`mysql` supports input-line editing, which enables you to modify the current input line in place or recall previous input lines. For example, the **left-arrow** and **right-arrow** keys move horizontally within the current input line, and the **up-arrow** and **down-arrow** keys move up and down through the set of previously entered lines. **Backspace** deletes the character before the cursor and typing new characters enters them at the cursor position. To enter the line, press **Enter**.

On Windows, the editing key sequences are the same as supported for command editing in console windows. On Unix, the key sequences depend on the input library used to build `mysql` (for example, the `libedit` or `readline` library).

Documentation for the `libedit` and `readline` libraries is available online. To change the set of key sequences permitted by a given input library, define key bindings in the library startup file. This is a file in your home directory: `.editrc` for `libedit` and `.inputrc` for `readline`.

For example, in `libedit`, **Control+W** deletes everything before the current cursor position and **Control+U** deletes the entire line. In `readline`, **Control+W** deletes the word before the cursor and **Control+U** deletes everything before the current cursor position. If `mysql` was built using `libedit`, a user who prefers the `readline` behavior for these two keys can put the following lines in the `.editrc` file (creating the file if necessary):

```
bind "^W" ed-delete-prev-word
bind "^U" vi-kill-line-prev
```

To see the current set of key bindings, temporarily put a line that says only `bind` at the end of `.editrc`. `mysql` shows the bindings when it starts.

Disabling Interactive History

The **up-arrow** key enables you to recall input lines from current and previous sessions. In cases where a console is shared, this behavior may be unsuitable. `mysql` supports disabling the interactive history partially or fully, depending on the host platform.

On Windows, the history is stored in memory. **Alt+F7** deletes all input lines stored in memory for the current history buffer. It also deletes the list of sequential numbers in front of the input lines displayed with **F7** and recalled (by number) with **F9**. New input lines entered after you press **Alt+F7** repopulate the current history buffer. Clearing the buffer does not prevent logging to the Windows Event Viewer, if the `--syslog` option was used to start `mysql`. Closing the console window also clears the current history buffer.

To disable interactive history on Unix, first delete the `.mysql_history` file, if it exists (previous entries are recalled otherwise). Then start `mysql` with the `--histignore="*"` option to ignore all new input lines. To re-enable the recall (and logging) behavior, restart `mysql` without the option.

If you prevent the `.mysql_history` file from being created (see Controlling the History File) and use `--histignore="*"` to start the `mysql` client, the interactive history recall facility is disabled fully. Alternatively, if you omit the `--histignore` option, you can recall the input lines entered during the current session.

Unicode Support on Windows

Windows provides APIs based on UTF-16LE for reading from and writing to the console; the `mysql` client for Windows is able to use these APIs. The Windows installer creates an item in the MySQL menu named `MySQL command line client - Unicode`. This item invokes the `mysql` client with properties set to communicate through the console to the MySQL server using Unicode.

To take advantage of this support manually, run `mysql` within a console that uses a compatible Unicode font and set the default character set to a Unicode character set that is supported for communication with the server:

1. Open a console window.
2. Go to the console window properties, select the font tab, and choose Lucida Console or some other compatible Unicode font. This is necessary because console windows start by default using a DOS raster font that is inadequate for Unicode.
3. Execute `mysql.exe` with the `--default-character-set=utf8mb4` (or `utf8mb3`) option. This option is necessary because `utf16le` is one of the character sets that cannot be used as the client character set. See Impermissible Client Character Sets.

With those changes, `mysql` uses the Windows APIs to communicate with the console using UTF-16LE, and communicate with the server using UTF-8. (The menu item mentioned previously sets the font and character set as just described.)

To avoid those steps each time you run `mysql`, you can create a shortcut that invokes `mysql.exe`. The shortcut should set the console font to Lucida Console or some other compatible Unicode font, and pass the `--default-character-set=utf8mb4` (or `utf8mb3`) option to `mysql.exe`.

Alternatively, create a shortcut that only sets the console font, and set the character set in the `[mysql]` group of your `my.ini` file:

```
[mysql]
default-character-set=utf8mb4   # or utf8mb3
```

Displaying Query Results Vertically

Some query results are much more readable when displayed vertically, instead of in the usual horizontal table format. Queries can be displayed vertically by terminating the query with \G instead of a semicolon. For example, longer text values that include newlines often are much easier to read with vertical output:

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 LIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Jones
    reply: jones@example.com
  mail_to: "John Smith" <smith@example.com>
      sbj: UTF-8
      txt: >>>>> "John" == John Smith writes:

John> Hi.  I think this is a good idea.  Is anyone familiar
John> with UTF-8 or Unicode? Otherwise, I'll put this on my
John> TODO list and see what happens.

Yes, please do that.

Regards,
Jones
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

Using Safe-Updates Mode (--safe-updates)

For beginners, a useful startup option is `--safe-updates` (or `--i-am-a-dummy`, which has the same effect). Safe-updates mode is helpful for cases when you might have issued an `UPDATE` or `DELETE` statement but forgotten the `WHERE` clause indicating which rows to modify. Normally, such statements update or delete all rows in the table. With `--safe-updates`, you can modify rows only by specifying the key values that identify them, or a `LIMIT` clause, or both. This helps prevent accidents. Safe-updates mode also restricts `SELECT` statements that produce (or are estimated to produce) very large result sets.

The `--safe-updates` option causes `mysql` to execute the following statement when it connects to the MySQL server, to set the session values of the `sql_safe_updates`, `sql_select_limit`, and `max_join_size` system variables:

```
SET sql_safe_updates=1, sql_select_limit=1000, max_join_size=1000000;
```

The `SET` statement affects statement processing as follows:

- Enabling `sql_safe_updates` causes `UPDATE` and `DELETE` statements to produce an error if they do not specify a key constraint in the `WHERE` clause, or provide a `LIMIT` clause, or both. For example:

  ```
  UPDATE tbl_name SET not_key_column=val WHERE key_column=val;

  UPDATE tbl_name SET not_key_column=val LIMIT 1;
  ```

- Setting `sql_select_limit` to 1,000 causes the server to limit all `SELECT` result sets to 1,000 rows unless the statement includes a `LIMIT` clause.

- Setting `max_join_size` to 1,000,000 causes multiple-table `SELECT` statements to produce an error if the server estimates it must examine more than 1,000,000 row combinations.

To specify result set limits different from 1,000 and 1,000,000, you can override the defaults by using the `--select-limit` and `--max-join-size` options when you invoke `mysql`:

```
mysql --safe-updates --select-limit=500 --max-join-size=10000
```

It is possible for `UPDATE` and `DELETE` statements to produce an error in safe-updates mode even with a key specified in the `WHERE` clause, if the optimizer decides not to use the index on the key column:

- Range access on the index cannot be used if memory usage exceeds that permitted by the `range_optimizer_max_mem_size` system variable. The optimizer then falls back to a table scan. See Limiting Memory Use for Range Optimization.

- If key comparisons require type conversion, the index may not be used (see Section 8.3.1, “How MySQL Uses Indexes”). Suppose that an indexed string column `c1` is compared to a numeric value using `WHERE c1 = 2222`. For such comparisons, the string value is converted to a number and the operands are compared numerically (see Section 12.3, “Type Conversion in Expression Evaluation”), preventing use of the index. If safe-updates mode is enabled, an error occurs.

As of MySQL 8.0.13, safe-updates mode also includes these behaviors:

- `EXPLAIN` with `UPDATE` and `DELETE` statements does not produce safe-updates errors. This enables use of `EXPLAIN` plus `SHOW WARNINGS` to see why an index is not used, which can be helpful in cases such as when a `range_optimizer_max_mem_size` violation or type conversion occurs and the optimizer does not use an index even though a key column was specified in the `WHERE` clause.

- When a safe-updates error occurs, the error message includes the first diagnostic that was produced, to provide information about the reason for failure. For example, the message may indicate that the `range_optimizer_max_mem_size` value was exceeded or type conversion occurred, either of which can preclude use of an index.

- For multiple-table deletes and updates, an error is produced with safe updates enabled only if any target table uses a table scan.

Disabling mysql Auto-Reconnect

If the `mysql` client loses its connection to the server while sending a statement, it immediately and automatically tries to reconnect once to the server and send the statement again. However, even if `mysql` succeeds in reconnecting, your first connection has ended and all your previous session objects and settings are lost: temporary tables, the autocommit mode, and user-defined and session variables. Also, any current transaction rolls back. This behavior may be dangerous for you, as in the following example where the server was shut down and restarted between the first and second statements without you knowing it:

```
mysql> SET @a=1;
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO t VALUES(@a);
ERROR 2006: MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    1
Current database: test

Query OK, 1 row affected (1.30 sec)

mysql> SELECT * FROM t;
+------+
| a    |
+------+
| NULL |
+------+
1 row in set (0.05 sec)
```

The `@a` user variable has been lost with the connection, and after the reconnection it is undefined. If it is important to have `mysql` terminate with an error if the connection has been lost, you can start the `mysql` client with the `--skip-reconnect` option.

For more information about auto-reconnect and its effect on state information when a reconnection occurs, see Automatic Reconnection Control.

mysql Client Parser Versus Server Parser

The `mysql` client uses a parser on the client side that is not a duplicate of the complete parser used by the `mysqld` server on the server side. This can lead to differences in treatment of certain constructs. Examples:

- The server parser treats strings delimited by `"` characters as identifiers rather than as plain strings if the `ANSI_QUOTES` SQL mode is enabled.

  The `mysql` client parser does not take the `ANSI_QUOTES` SQL mode into account. It treats strings delimited by `"`, `'`, and `` ` `` characters the same, regardless of whether `ANSI_QUOTES` is enabled.

- Within `/*! ... */` and `/*+ ... */` comments, the `mysql` client parser interprets short-form `mysql` commands. The server parser does not interpret them because these commands have no meaning on the server side.

  If it is desirable for `mysql` not to interpret short-form commands within comments, a partial workaround is to use the `--binary-mode` option, which causes all `mysql` commands to be disabled except `\C` and `\d` in noninteractive mode (for input piped to `mysql` or loaded using the `source` command).

4.5.2 mysqladmin — A MySQL Server Administration Program

`mysqladmin` is a client for performing administrative operations. You can use it to check the server's configuration and current status, to create and drop databases, and more.

Invoke `mysqladmin` like this:

```
mysqladmin [options] command [command-arg] [command [command-arg]] ...
```

`mysqladmin` supports the following commands. Some of the commands take an argument following the command name.

- `create db_name`

  Create a new database named *`db_name`*.

- `debug`

  Prior to MySQL 8.0.20, tell the server to write debug information to the error log. The connected user must have the `SUPER` privilege. Format and content of this information is subject to change.

  This includes information about the Event Scheduler. See Section 25.4.5, “Event Scheduler Status”.

- `drop db_name`

  Delete the database named *`db_name`* and all its tables.

- `extended-status`

  Display the server status variables and their values.

- `flush-hosts`

  Flush all information in the host cache. See Section 5.1.12.3, “DNS Lookups and the Host Cache”.

- `flush-logs [log_type ...]`

  Flush all logs.

  The `mysqladmin flush-logs` command permits optional log types to be given, to specify which logs to flush. Following the `flush-logs` command, you can provide a space-separated list of one or more of the following log types: `binary`, `engine`, `error`, `general`, `relay`, `slow`. These correspond to the log types that can be specified for the `FLUSH LOGS` SQL statement.

- `flush-privileges`

  Reload the grant tables (same as `reload`).

- `flush-status`

  Clear status variables.

- `flush-tables`

  Flush all tables.

- `flush-threads`

  Flush the thread cache.

- `kill id,id,...`

  Kill server threads. If multiple thread ID values are given, there must be no spaces in the list.

  To kill threads belonging to other users, the connected user must have the `CONNECTION_ADMIN` privilege (or the deprecated `SUPER` privilege).

- `password new_password`

  Set a new password. This changes the password to *`new_password`* for the account that you use with `mysqladmin` for connecting to the server. Thus, the next time you invoke `mysqladmin` (or any other client program) using the same account, you must specify the new password.

Warning

Setting a password using `mysqladmin` should be considered *insecure*. On some systems, your password becomes visible to system status programs such as `ps` that may be invoked by other users to display command lines. MySQL clients typically overwrite the command-line password argument with zeros during their initialization sequence. However, there is still a brief interval during which the value is visible. Also, on some systems this overwriting strategy is ineffective and the password remains visible to `ps`. (SystemV Unix systems and perhaps others are subject to this problem.)

If the *`new_password`* value contains spaces or other characters that are special to your command interpreter, you need to enclose it within quotation marks. On Windows, be sure to use double quotation marks rather than single quotation marks; single quotation marks are not stripped from the password, but rather are interpreted as part of the password. For example:

```
mysqladmin password "my new password"
```

The new password can be omitted following the `password` command. In this case, `mysqladmin` prompts for the password value, which enables you to avoid specifying the password on the command line. Omitting the password value should be done only if `password` is the final command on the `mysqladmin` command line. Otherwise, the next argument is taken as the password.

Caution

Do not use this command used if the server was started with the `--skip-grant-tables` option. No password change is applied. This is true even if you precede the `password` command with `flush-privileges` on the same command line to re-enable the grant tables because the flush operation occurs after you connect. However, you can use `mysqladmin flush-privileges` to re-enable the grant table and then use a separate `mysqladmin password` command to change the password.

- `ping`

  Check whether the server is available. The return status from `mysqladmin` is 0 if the server is running, 1 if it is not. This is 0 even in case of an error such as `Access denied`, because this means that the server is running but refused the connection, which is different from the server not running.

- `processlist`

  Show a list of active server threads. This is like the output of the `SHOW PROCESSLIST` statement. If the `--verbose` option is given, the output is like that of `SHOW FULL PROCESSLIST`. (See Section 13.7.7.29, “SHOW PROCESSLIST Statement”.)

- `reload`

  Reload the grant tables.

- `refresh`

  Flush all tables and close and open log files.

- `shutdown`

  Stop the server.

- `start-replica`

  Start replication on a replica server. Use this command from MySQL 8.0.26.

- `start-slave`

  Start replication on a replica server. Use this command before MySQL 8.0.26.

- `status`

  Display a short server status message.

- `stop-replica`

  Stop replication on a replica server. Use this command from MySQL 8.0.26.

- `stop-slave`

  Stop replication on a replica server. Use this command before MySQL 8.0.26.

- `variables`

  Display the server system variables and their values.

- `version`

  Display version information from the server.

All commands can be shortened to any unique prefix. For example:

```
$> mysqladmin proc stat
+----+-------+-----------+----+---------+------+-------+------------------+
| Id | User  | Host      | db | Command | Time | State | Info             |
+----+-------+-----------+----+---------+------+-------+------------------+
| 51 | jones | localhost |    | Query   | 0    |       | show processlist |
+----+-------+-----------+----+---------+------+-------+------------------+
Uptime: 1473624  Threads: 1  Questions: 39487
Slow queries: 0  Opens: 541  Flush tables: 1
Open tables: 19  Queries per second avg: 0.0268
```

The `mysqladmin status` command result displays the following values:

- `Uptime`

  The number of seconds the MySQL server has been running.

- `Threads`

  The number of active threads (clients).

- `Questions`

  The number of questions (queries) from clients since the server was started.

- `Slow queries`

  The number of queries that have taken more than `long_query_time` seconds. See Section 5.4.5, “The Slow Query Log”.

- `Opens`

  The number of tables the server has opened.

- `Flush tables`

  The number of `flush-*`, `refresh`, and `reload` commands the server has executed.

- `Open tables`

  The number of tables that currently are open.

If you execute `mysqladmin shutdown` when connecting to a local server using a Unix socket file, `mysqladmin` waits until the server's process ID file has been removed, to ensure that the server has stopped properly.

`mysqladmin` supports the following options, which can be specified on the command line or in the `[mysqladmin]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.12 mysqladmin Options

Option Name	Description	Introduced	Deprecated
--bind-address	Use specified network interface to connect to MySQL Server		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--connect-timeout	Number of seconds before connection timeout		
--count	Number of iterations to make for repeated command execution		
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--force	Continue even if an SQL error occurs		
--get-server-public-key	Request RSA public key from server		

Option Name	Description	Introduced	Deprecated
--help	Display help message and exit		
--host	Host on which MySQL server is located		
--login-path	Read login path options from .mylogin.cnf		
--no-beep	Do not beep when errors occur		
--no-defaults	Read no option files		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--pipe	Connect to server using named pipe (Windows only)		
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--relative	Show the difference between the current and previous values when used with the --sleep option		
--server-public-key-path	Path name to file containing RSA public key		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--show-warnings	Show warnings after statement execution		

Option Name	Description	Introduced	Deprecated
--shutdown-timeout	The maximum number of seconds to wait for server shutdown		
--silent	Silent mode		
--sleep	Execute commands repeatedly, sleeping for delay seconds in between		
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		

Option Name	Description	Introduced	Deprecated
--version	Display version information and exit		
--vertical	Print query output rows vertically (one line per column value)		
--wait	If the connection cannot be established, wait and retry instead of aborting		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`, `-?`

  Display a help message and exit.

- `--bind-address=ip_address`

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=dir_name`

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--connect-timeout=value`

  The maximum number of seconds before connection timeout. The default value is 43200 (12 hours).

- `--count=N`, `-c N`

  The number of iterations to make for repeated command execution if the `--sleep` option is given.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:o,/tmp/mysqladmin.trace`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`

  Print debugging information and memory and CPU usage statistics when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysqladmin` normally reads the `[client]` and `[mysqladmin]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqladmin` also reads the `[client_other]` and `[mysqladmin_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--force`, `-f`

  Do not ask for confirmation for the `drop db_name` command. With multiple commands, continue even if an error occurs.

- `--get-server-public-key`

Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, "Caching SHA-2 Pluggable Authentication".

- `--host=host_name`, `-h host_name`

  Connect to the MySQL server on the given host.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A "login path" is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, "mysql_config_editor — MySQL Configuration Utility".

  For additional information about this and other option-file options, see Section 4.2.2.3, "Command-Line Options that Affect Option-File Handling".

- `--no-beep`, `-b`

  Suppress the warning beep that is emitted by default for errors such as a failure to connect to the server.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, "mysql_config_editor — MySQL Configuration Utility".

  For additional information about this and other option-file options, see Section 4.2.2.3, "Command-Line Options that Affect Option-File Handling".

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqladmin` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, "End-User Guidelines for Password Security".

  To explicitly specify that there is no password and that `mysqladmin` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysql` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqladmin` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqladmin` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--relative`, `-r`

  Show the difference between the current and previous values when used with the `--sleep` option. This option works only with the `extended-status` command.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--shared-memory-base-name=name`

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--show-warnings`

  Show warnings resulting from execution of statements sent to the server.

- `--shutdown-timeout=value`

  The maximum number of seconds to wait for server shutdown. The default value is 3600 (1 hour).

- `--silent`, `-s`

  Exit silently if a connection to the server cannot be established.

- `--sleep=delay`, `-i delay`

  Execute commands repeatedly, sleeping for *`delay`* seconds in between. The `--count` option determines the number of iterations. If `--count` is not given, `mysqladmin` executes commands indefinitely until interrupted.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

  If you are using the `Rewriter` plugin with MySQL 8.0.31 or later, you should grant this user the `SKIP_QUERY_REWRITE` privilege.

- `--verbose`, `-v`

  Verbose mode. Print more information about what the program does.

- `--version`, `-V`

  Display version information and exit.

- `--vertical`, `-E`

  Print output vertically. This is similar to `--relative`, but prints output vertically.

- `--wait[=count]`, `-w[count]`

  If the connection cannot be established, wait and retry instead of aborting. If a `count` value is given, it indicates the number of times to retry. The default is one time.

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

4.5.3 mysqlcheck — A Table Maintenance Program

The `mysqlcheck` client performs table maintenance: It checks, repairs, optimizes, or analyzes tables.

Each table is locked and therefore unavailable to other sessions while it is being processed, although for check operations, the table is locked with a `READ` lock only (see Section 13.3.6, “LOCK TABLES and UNLOCK TABLES Statements”, for more information about `READ` and `WRITE` locks). Table maintenance operations can be time-consuming, particularly for large tables. If you use the `--databases` or `--all-databases` option to process all tables in one or more databases, an invocation of `mysqlcheck` might take a long time. (This is also true for the MySQL upgrade procedure if it determines that table checking is needed because it processes tables the same way.)

`mysqlcheck` must be used when the `mysqld` server is running, which means that you do not have to stop the server to perform table maintenance.

`mysqlcheck` uses the SQL statements `CHECK TABLE`, `REPAIR TABLE`, `ANALYZE TABLE`, and `OPTIMIZE TABLE` in a convenient way for the user. It determines which statements to use for the operation you want to perform, and then sends the statements to the server to be executed. For details about which storage engines each statement works with, see the descriptions for those statements in Section 13.7.3, “Table Maintenance Statements”.

All storage engines do not necessarily support all four maintenance operations. In such cases, an error message is displayed. For example, if `test.t` is an `MEMORY` table, an attempt to check it produces this result:

```
$> mysqlcheck test t
test.t
note     : The storage engine for the table doesn't support check
```

If `mysqlcheck` is unable to repair a table, see Section 2.10.13, “Rebuilding or Repairing Tables or Indexes” for manual table repair strategies. This is the case, for example, for `InnoDB` tables, which can be checked with `CHECK TABLE`, but not repaired with `REPAIR TABLE`.

Caution

It is best to make a backup of a table before performing a table repair operation; under some circumstances the operation might cause data loss. Possible causes include but are not limited to file system errors.

There are three general ways to invoke `mysqlcheck`:

```
mysqlcheck [options] db_name [tbl_name ...]
mysqlcheck [options] --databases db_name ...
mysqlcheck [options] --all-databases
```

If you do not name any tables following *`db_name`* or if you use the `--databases` or `--all-databases` option, entire databases are checked.

`mysqlcheck` has a special feature compared to other client programs. The default behavior of checking tables (`--check`) can be changed by renaming the binary. If you want to have a tool that repairs tables by default, you should just make a copy of `mysqlcheck` named `mysqlrepair`, or make a symbolic link to `mysqlcheck` named `mysqlrepair`. If you invoke `mysqlrepair`, it repairs tables.

The names shown in the following table can be used to change `mysqlcheck` default behavior.

Command	Meaning
`mysqlrepair`	The default option is `--repair`
`mysqlanalyze`	The default option is `--analyze`
`mysqloptimize`	The default option is `--optimize`

`mysqlcheck` supports the following options, which can be specified on the command line or in the `[mysqlcheck]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.13 mysqlcheck Options

Option Name	Description	Introduced	Deprecated
--all-databases	Check all tables in all databases		
--all-in-1	Execute a single statement for each database that names all the tables from that database		
--analyze	Analyze the tables		
--auto-repair	If a checked table is corrupted, automatically fix it		
--bind-address	Use specified network interface to connect to MySQL Server		
--character-sets-dir	Directory where character sets are installed		
--check	Check the tables for errors		
--check-only-changed	Check only tables that have changed since the last check		
--check-upgrade	Invoke CHECK TABLE with the FOR UPGRADE option		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--databases	Interpret all arguments as database names		
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		

Option Name	Description	Introduced	Deprecated
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--extended	Check and repair tables		
--fast	Check only tables that have not been closed properly		
--force	Continue even if an SQL error occurs		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--host	Host on which MySQL server is located		
--login-path	Read login path options from .mylogin.cnf		
--medium-check	Do a check that is faster than an --extended operation		
--no-defaults	Read no option files		
--optimize	Optimize the tables		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password	8.0.27	

Option Name	Description	Introduced	Deprecated
	to use when connecting to server		
--pipe	Connect to server using named pipe (Windows only)		
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--quick	The fastest method of checking		
--repair	Perform a repair that can fix almost anything except unique keys that are not unique		
--server-public-key-path	Path name to file containing RSA public key		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--silent	Silent mode		
--skip-database	Omit this database from performed operations		
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		

Option Name	Description	Introduced	Deprecated
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tables	Overrides the --databases or -B option		
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--use-frm	For repair operations on MyISAM tables		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		
--version	Display version information and exit		
--write-binlog	Log ANALYZE, OPTIMIZE, REPAIR statements to binary log. --skip-write-binlog adds NO_WRITE_TO_BINLOG to these statements		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`, `-?`

  Display a help message and exit.

- `--all-databases`, `-A`

  Check all tables in all databases. This is the same as using the `--databases` option and naming all the databases on the command line, except that the `INFORMATION_SCHEMA` and `performance_schema` databases are not checked. They can be checked by explicitly naming them with the `--databases` option.

- `--all-in-1`, `-1`

  Instead of issuing a statement for each table, execute a single statement for each database that names all the tables from that database to be processed.

- `--analyze`, `-a`

  Analyze the tables.

- `--auto-repair`

  If a checked table is corrupted, automatically fix it. Any necessary repairs are done after all tables have been checked.

- `--bind-address=ip_address`

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=dir_name`

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--check`, `-c`

  Check the tables for errors. This is the default operation.

- `--check-only-changed`, `-C`

  Check only tables that have changed since the last check or that have not been closed properly.

- `--check-upgrade`, `-g`

  Invoke `CHECK TABLE` with the `FOR UPGRADE` option to check tables for incompatibilities with the current version of the server.

- `--compress`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--databases`, `-B`

  Process all tables in the named databases. Normally, `mysqlcheck` treats the first name argument on the command line as a database name and any following names as table names. With this option, it treats all name arguments as database names.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:o`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

Print some debugging information when the program exits.

This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`

  Print debugging information and memory and CPU usage statistics when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysqlcheck` normally reads the `[client]` and `[mysqlcheck]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqlcheck` also reads the `[client_other]` and `[mysqlcheck_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--extended`, `-e`

  If you are using this option to check tables, it ensures that they are 100% consistent but takes a long time.

  If you are using this option to repair tables, it runs an extended repair that may not only take a long time to execute, but may produce a lot of garbage rows also!

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--fast`, `-F`

  Check only tables that have not been closed properly.

- `--force`, `-f`

  Continue even if an SQL error occurs.

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--host=host_name`, `-h host_name`

  Connect to the MySQL server on the given host.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--medium-check`, `-m`

  Do a check that is faster than an `--extended` operation. This finds only 99.99% of all errors, which should be good enough in most cases.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--optimize`, `-o`

  Optimize the tables.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlcheck` prompts for one. If given, there must be *no space* between `--`

`password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

To explicitly specify that there is no password and that `mysqlcheck` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlcheck` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlcheck` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqlcheck` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--quick`, `-q`

  If you are using this option to check tables, it prevents the check from scanning the rows to check for incorrect links. This is the fastest check method.

  If you are using this option to repair tables, it tries to repair only the index tree. This is the fastest repair method.

- `--repair`, `-r`

  Perform a repair that can fix almost anything except unique keys that are not unique.

- `--server-public-key-path=`*`file_name`*

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=`*`file_name`* is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--shared-memory-base-name=`*`name`*

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--silent`, `-s`

  Silent mode. Print only error messages.

- `--skip-database=`*`db_name`*

  Do not include the named database (case-sensitive) in the operations performed by `mysqlcheck`.

- `--socket=`*`path`*, `-S` *`path`*

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tables`

  Override the `--databases` or `-B` option. All name arguments following the option are regarded as table names.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--use-frm`

  For repair operations on `MyISAM` tables, get the table structure from the data dictionary so that the table can be repaired even if the `.MYI` header is corrupted.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

- `--verbose`, `-v`

  Verbose mode. Print information about the various stages of program operation.

- `--version`, `-V`

  Display version information and exit.

- `--write-binlog`

  This option is enabled by default, so that `ANALYZE TABLE`, `OPTIMIZE TABLE`, and `REPAIR TABLE` statements generated by `mysqlcheck` are written to the binary log. Use `--skip-write-binlog` to cause `NO_WRITE_TO_BINLOG` to be added to the statements so that they are not logged. Use the `--skip-write-binlog` when these statements should not be sent to replicas or run when using the binary logs for recovery from backup.

- `--zstd-compression-level=`*`level`*

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

4.5.4 mysqldump — A Database Backup Program

The `mysqldump` client utility performs logical backups, producing a set of SQL statements that can be executed to reproduce the original database object definitions and table data. It dumps one or more MySQL databases for backup or transfer to another SQL server. The `mysqldump` command can also generate output in CSV, other delimited text, or XML format.

Tip

Consider using the MySQL Shell dump utilities, which provide parallel dumping with multiple threads, file compression, and progress information display, as well as cloud features such as Oracle Cloud Infrastructure Object Storage streaming, and MySQL Database Service compatibility checks and modifications. Dumps can be easily imported into a MySQL Server instance or a MySQL Database Service DB System using the MySQL Shell load dump utilities. Installation instructions for MySQL Shell can be found here.

- Performance and Scalability Considerations
- Invocation Syntax
- Option Syntax - Alphabetical Summary
- Connection Options
- Option-File Options
- DDL Options
- Debug Options
- Help Options
- Internationalization Options
- Replication Options
- Format Options
- Filtering Options
- Performance Options
- Transactional Options

- Option Groups
- Examples
- Restrictions

`mysqldump` requires at least the `SELECT` privilege for dumped tables, `SHOW VIEW` for dumped views, `TRIGGER` for dumped triggers, `LOCK TABLES` if the `--single-transaction` option is not used, and (as of MySQL 8.0.21) `PROCESS` if the `--no-tablespaces` option is not used. Certain options might require other privileges as noted in the option descriptions.

To reload a dump file, you must have the privileges required to execute the statements that it contains, such as the appropriate `CREATE` privileges for objects created by those statements.

`mysqldump` output can include `ALTER DATABASE` statements that change the database collation. These may be used when dumping stored programs to preserve their character encodings. To reload a dump file containing such statements, the `ALTER` privilege for the affected database is required.

Note

A dump made using PowerShell on Windows with output redirection creates a file that has UTF-16 encoding:

```
mysqldump [options] > dump.sql
```

However, UTF-16 is not permitted as a connection character set (see Impermissible Client Character Sets), so the dump file cannot be loaded correctly. To work around this issue, use the `--result-file` option, which creates the output in ASCII format:

```
mysqldump [options] --result-file=dump.sql
```

It is not recommended to load a dump file when GTIDs are enabled on the server (`gtid_mode=ON`), if your dump file includes system tables. `mysqldump` issues DML instructions for the system tables which use the non-transactional MyISAM storage engine, and this combination is not permitted when GTIDs are enabled.

Performance and Scalability Considerations

`mysqldump` advantages include the convenience and flexibility of viewing or even editing the output before restoring. You can clone databases for development and DBA work, or produce slight variations of an existing database for testing. It is not intended as a fast or scalable solution for backing up substantial amounts of data. With large data sizes, even if the backup step takes a reasonable time, restoring the data can be very slow because replaying the SQL statements involves disk I/O for insertion, index creation, and so on.

For large-scale backup and restore, a physical backup is more appropriate, to copy the data files in their original format so that they can be restored quickly.

If your tables are primarily `InnoDB` tables, or if you have a mix of `InnoDB` and `MyISAM` tables, consider using `mysqlbackup`, which is available as part of MySQL Enterprise. This tool provides high performance for `InnoDB` backups with minimal disruption; it can also back up tables from `MyISAM` and other storage engines; it also provides a number of convenient options to accommodate different backup scenarios. See Section 30.2, “MySQL Enterprise Backup Overview”.

`mysqldump` can retrieve and dump table contents row by row, or it can retrieve the entire content from a table and buffer it in memory before dumping it. Buffering in memory can be a problem if you are dumping large tables. To dump tables row by row, use the `--quick` option (or `--opt`, which enables `--quick`). The `--opt` option (and hence `--quick`) is enabled by default, so to enable memory buffering, use `--skip-quick`.

If you are using a recent version of `mysqldump` to generate a dump to be reloaded into a very old MySQL server, use the `--skip-opt` option instead of the `--opt` or `--extended-insert` option.

For additional information about `mysqldump`, see Section 7.4, “Using mysqldump for Backups”.

Invocation Syntax

There are in general three ways to use `mysqldump`—in order to dump a set of one or more tables, a set of one or more complete databases, or an entire MySQL server—as shown here:

```
mysqldump [options] db_name [tbl_name ...]
mysqldump [options] --databases db_name ...
mysqldump [options] --all-databases
```

To dump entire databases, do not name any tables following *`db_name`*, or use the `--databases` or `--all-databases` option.

To see a list of the options your version of `mysqldump` supports, issue the command `mysqldump --help`.

Option Syntax - Alphabetical Summary

`mysqldump` supports the following options, which can be specified on the command line or in the `[mysqldump]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.14 mysqldump Options

Option Name	Description	Introduced	Deprecated
--add-drop-database	Add DROP DATABASE statement before each CREATE DATABASE statement		
--add-drop-table	Add DROP TABLE statement before each CREATE TABLE statement		
--add-drop-trigger	Add DROP TRIGGER statement before each CREATE TRIGGER statement		
--add-locks	Surround each table dump with LOCK TABLES and UNLOCK TABLES statements		
--all-databases	Dump all tables in all databases		
--allow-keywords	Allow creation of column names that are keywords		
--apply-replica-statements	Include STOP REPLICA prior to CHANGE REPLICATION SOURCE TO statement and START REPLICA at end of output	8.0.26	
--apply-slave-statements	Include STOP SLAVE prior to CHANGE MASTER statement and START SLAVE at end of output		8.0.26

Option Name	Description	Introduced	Deprecated
--bind-address	Use specified network interface to connect to MySQL Server		
--character-sets-dir	Directory where character sets are installed		
--column-statistics	Write ANALYZE TABLE statements to generate statistics histograms		
--comments	Add comments to dump file		
--compact	Produce more compact output		
--compatible	Produce output that is more compatible with other database systems or with older MySQL servers		
--complete-insert	Use complete INSERT statements that include column names		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--create-options	Include all MySQL-specific table options in CREATE TABLE statements		
--databases	Interpret all name arguments as database names		
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		

Option Name	Description	Introduced	Deprecated
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--delete-master-logs	On a replication source server, delete the binary logs after performing the dump operation		8.0.26
--delete-source-logs	On a replication source server, delete the binary logs after performing the dump operation	8.0.26	
--disable-keys	For each table, surround INSERT statements with statements to disable and enable keys		
--dump-date	Include dump date as "Dump completed on" comment if --comments is given		
--dump-replica	Include CHANGE REPLICATION SOURCE TO statement that lists binary log coordinates of replica's source	8.0.26	
--dump-slave	Include CHANGE MASTER statement that lists binary log coordinates of replica's source		8.0.26
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--events	Dump events from dumped databases		
--extended-insert	Use multiple-row INSERT syntax		
--fields-enclosed-by	This option is used with the --tab option and has the same meaning as the corresponding clause for LOAD DATA		
--fields-escaped-by	This option is used with the --tab option and has the same meaning as the corresponding clause for LOAD DATA		
--fields-optionally-enclosed-by	This option is used with the --tab option and has the same meaning		

Option Name	Description	Introduced	Deprecated
	as the corresponding clause for LOAD DATA		
--fields-terminated-by	This option is used with the --tab option and has the same meaning as the corresponding clause for LOAD DATA		
--flush-logs	Flush MySQL server log files before starting dump		
--flush-privileges	Emit a FLUSH PRIVILEGES statement after dumping mysql database		
--force	Continue even if an SQL error occurs during a table dump		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--hex-blob	Dump binary columns using hexadecimal notation		
--host	Host on which MySQL server is located		
--ignore-error	Ignore specified errors		
--ignore-table	Do not dump given table		
--include-master-host-port	Include MASTER_HOST/ MASTER_PORT options in CHANGE MASTER statement produced with --dump-slave		8.0.26
--include-source-host-port	Include SOURCE_HOST and SOURCE_PORT options in CHANGE REPLICATION SOURCE TO statement produced with --dump-replica	8.0.26	
--insert-ignore	Write INSERT IGNORE rather than INSERT statements		
--lines-terminated-by	This option is used with the --tab option and has the same meaning as the corresponding clause for LOAD DATA		

Option Name	Description	Introduced	Deprecated
--lock-all-tables	Lock all tables across all databases		
--lock-tables	Lock all tables before dumping them		
--log-error	Append warnings and errors to named file		
--login-path	Read login path options from .mylogin.cnf		
--master-data	Write the binary log file name and position to the output		8.0.26
--max-allowed-packet	Maximum packet length to send to or receive from server		
--mysqld-long-query-time	Session value for slow query threshold	8.0.30	
--net-buffer-length	Buffer size for TCP/IP and socket communication		
--network-timeout	Increase network timeouts to permit larger table dumps		
--no-autocommit	Enclose the INSERT statements for each dumped table within SET autocommit = 0 and COMMIT statements		
--no-create-db	Do not write CREATE DATABASE statements		
--no-create-info	Do not write CREATE TABLE statements that re-create each dumped table		
--no-data	Do not dump table contents		
--no-defaults	Read no option files		
--no-set-names	Same as --skip-set-charset		
--no-tablespaces	Do not write any CREATE LOGFILE GROUP or CREATE TABLESPACE statements in output		
--opt	Shorthand for --add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick --set-charset		

Option Name	Description	Introduced	Deprecated
--order-by-primary	Dump each table's rows sorted by its primary key, or by its first unique index		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--pipe	Connect to server using named pipe (Windows only)		
--plugin-authentication-kerberos-client-mode	Permit GSSAPI pluggable authentication through the MIT Kerberos library on Windows	8.0.32	
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--quick	Retrieve rows for a table from the server a row at a time		
--quote-names	Quote identifiers within backtick characters		
--replace	Write REPLACE statements rather than INSERT statements		
--result-file	Direct output to a given file		
--routines	Dump stored routines (procedures and functions) from dumped databases		
--server-public-key-path	Path name to file containing RSA public key		

Option Name	Description	Introduced	Deprecated
--set-charset	Add SET NAMES default_character_set to output		
--set-gtid-purged	Whether to add SET @@GLOBAL.GTID_PURGED to output		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--show-create-skip-secondary-engine	Exclude SECONDARY ENGINE clause from CREATE TABLE statements	8.0.18	
--single-transaction	Issue a BEGIN SQL statement before dumping data from server		
--skip-add-drop-table	Do not add a DROP TABLE statement before each CREATE TABLE statement		
--skip-add-locks	Do not add locks		
--skip-comments	Do not add comments to dump file		
--skip-compact	Do not produce more compact output		
--skip-disable-keys	Do not disable keys		
--skip-extended-insert	Turn off extended-insert		
--skip-generated-invisible-primary-key	Do not include generated invisible primary keys in dump file	8.0.30	
--skip-opt	Turn off options set by --opt		
--skip-quick	Do not retrieve rows for a table from the server a row at a time		
--skip-quote-names	Do not quote identifiers		
--skip-set-charset	Do not write SET NAMES statement		
--skip-triggers	Do not dump triggers		
--skip-tz-utc	Turn off tz-utc		
--socket	Unix socket file or Windows named pipe to use		
--source-data	Write the binary log file name and position to the output	8.0.26	

Option Name	Description	Introduced	Deprecated
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tab	Produce tab-separated data files		
--tables	Override --databases or -B option		
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--triggers	Dump triggers for each dumped table		
--tz-utc	Add SET TIME_ZONE='+00:00' to dump file		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		
--version	Display version information and exit		

Option Name	Description	Introduced	Deprecated
--where	Dump only rows selected by given WHERE condition		
--xml	Produce XML output		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

Connection Options

The `mysqldump` command logs into a MySQL server to extract information. The following options specify how to connect to the MySQL server, either on the same machine or a remote system.

- `--bind-address=ip_address`

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--host=host_name`, `-h host_name`

  Dump data from the MySQL server on the given host. The default host is `localhost`.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqldump` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqldump` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqldump` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqldump` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-authentication-kerberos-client-mode=value`

  On Windows, the `authentication_kerberos_client` authentication plugin supports this plugin option. It provides two possible values that the client user can set at runtime: `SSPI` and `GSSAPI`.

  The default value for the client-side plugin option uses Security Support Provider Interface (SSPI), which is capable of acquiring credentials from the Windows in-memory cache. Alternatively, the client user can select a mode that supports Generic Security Service Application Program Interface (GSSAPI) through the MIT Kerberos library on Windows. GSSAPI is capable of acquiring cached credentials previously generated by using the `kinit` command.

  For more information, see Commands for Windows Clients in GSSAPI Mode.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqldump` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

  If you are using the `Rewriter` plugin with MySQL 8.0.31 or later, you should grant this user the `SKIP_QUERY_REWRITE` privilege.

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

Option-File Options

These options are used to control which option files to read.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If `file_name` is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If `file_name` is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of `str`. For example, `mysqldump` normally reads the `[client]` and `[mysqldump]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqldump` also reads the `[client_other]` and `[mysqldump_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

DDL Options

Usage scenarios for `mysqldump` include setting up an entire new MySQL instance (including database tables), and replacing data inside an existing instance with existing databases and tables. The following options let you specify which things to tear down and set up when restoring a dump, by encoding various DDL statements within the dump file.

- `--add-drop-database`

Write a `DROP DATABASE` statement before each `CREATE DATABASE` statement. This option is typically used in conjunction with the `--all-databases` or `--databases` option because no `CREATE DATABASE` statements are written unless one of those options is specified.

Note

In MySQL 8.0, the `mysql` schema is considered a system schema that cannot be dropped by end users. If `--add-drop-database` is used with `--all-databases` or with `--databases` where the list of schemas to be dumped includes `mysql`, the dump file contains a `DROP DATABASE` `` `mysql` `` statement that causes an error when the dump file is reloaded.

Instead, to use `--add-drop-database`, use `--databases` with a list of schemas to be dumped, where the list does not include `mysql`.

- `--add-drop-table`

  Write a `DROP TABLE` statement before each `CREATE TABLE` statement.

- `--add-drop-trigger`

  Write a `DROP TRIGGER` statement before each `CREATE TRIGGER` statement.

- `--all-tablespaces`, `-Y`

  Adds to a table dump all SQL statements needed to create any tablespaces used by an `NDB` table. This information is not otherwise included in the output from `mysqldump`. This option is currently relevant only to NDB Cluster tables.

- `--no-create-db`, `-n`

  Suppress the `CREATE DATABASE` statements that are otherwise included in the output if the `--databases` or `--all-databases` option is given.

- `--no-create-info`, `-t`

  Do not write `CREATE TABLE` statements that create each dumped table.

  

  Note

  This option does *not* exclude statements creating log file groups or tablespaces from `mysqldump` output; however, you can use the `--no-tablespaces` option for this purpose.

- `--no-tablespaces`, `-y`

  This option suppresses all `CREATE LOGFILE GROUP` and `CREATE TABLESPACE` statements in the output of `mysqldump`.

- `--replace`

  Write `REPLACE` statements rather than `INSERT` statements.

Debug Options

The following options print debugging information, encode debugging information in the dump file, or let the dump operation proceed regardless of potential problems.

- `--allow-keywords`

  Permit creation of column names that are keywords. This works by prefixing each column name with the table name.

- `--comments`, `-i`

  Write additional information in the dump file such as program version, server version, and host. This option is enabled by default. To suppress this additional information, use `--skip-comments`.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default value is `d:t:o,/tmp/mysqldump.trace`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`

  Print debugging information and memory and CPU usage statistics when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--dump-date`

  If the `--comments` option is given, `mysqldump` produces a comment at the end of the dump of the following form:

  ```
  -- Dump completed on DATE
  ```

  However, the date causes dump files taken at different times to appear to be different, even if the data are otherwise identical. `--dump-date` and `--skip-dump-date` control whether the date is added to the comment. The default is `--dump-date` (include the date in the comment). `--skip-dump-date` suppresses date printing.

- `--force`, `-f`

  Ignore all errors; continue even if an SQL error occurs during a table dump.

  One use for this option is to cause `mysqldump` to continue executing even when it encounters a view that has become invalid because the definition refers to a table that has been dropped. Without `--force`, `mysqldump` exits with an error message. With `--force`, `mysqldump` prints the error message, but it also writes an SQL comment containing the view definition to the dump output and continues executing.

  If the `--ignore-error` option is also given to ignore specific errors, `--force` takes precedence.

- `--log-error=file_name`

  Log warnings and errors by appending them to the named file. The default is to do no logging.

- `--skip-comments`

  See the description for the `--comments` option.

- `--verbose`, `-v`

  Verbose mode. Print more information about what the program does.

Help Options

The following options display information about the `mysqldump` command itself.

- `--help`, `-?`

  Display a help message and exit.

- `--version`, `-V`

  Display version information and exit.

Internationalization Options

The following options change how the `mysqldump` command represents character data with national language settings.

- `--character-sets-dir=`*`dir_name`*

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--default-character-set=`*`charset_name`*

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”. If no character set is specified, `mysqldump` uses `utf8mb4`.

- `--no-set-names`, `-N`

  Turns off the `--set-charset` setting, the same as specifying `--skip-set-charset`.

- `--set-charset`

  Write `SET NAMES` *`default_character_set`* to the output. This option is enabled by default. To suppress the `SET NAMES` statement, use `--skip-set-charset`.

Replication Options

The `mysqldump` command is frequently used to create an empty instance, or an instance including data, on a replica server in a replication configuration. The following options apply to dumping and restoring data on replication source servers and replicas.

- `--apply-replica-statements`

  From MySQL 8.0.26, use `--apply-replica-statements`, and before MySQL 8.0.26, use `--apply-slave-statements`. Both options have the same effect. For a replica dump produced with the `--dump-replica` or `--dump-slave` option, the options add a `STOP REPLICA` (or before MySQL 8.0.22, `STOP SLAVE`) statement before the statement with the binary log coordinates, and a `START REPLICA` statement at the end of the output.

- `--apply-slave-statements`

  Use this option before MySQL 8.0.26 rather than `--apply-replica-statements`. Both options have the same effect.

- `--delete-source-logs`

  From MySQL 8.0.26, use `--delete-source-logs`, and before MySQL 8.0.26, use `--delete-master-logs`. Both options have the same effect. On a replication source server, the options delete the binary logs by sending a `PURGE BINARY LOGS` statement to the server after performing the dump operation. The options require the `RELOAD` privilege as well as privileges sufficient to execute that statement. The options automatically enable `--source-data` or `--master-data`.

- `--delete-master-logs`

Use this option before MySQL 8.0.26 rather than `--delete-source-logs`. Both options have the same effect.

- `--dump-replica[=value]`

  From MySQL 8.0.26, use `--dump-replica`, and before MySQL 8.0.26, use `--dump-slave`. Both options have the same effect. The options are similar to `--source-data`, except that they are used to dump a replica server to produce a dump file that can be used to set up another server as a replica that has the same source as the dumped server. The options cause the dump output to include a `CHANGE REPLICATION SOURCE TO` statement (from MySQL 8.0.23) or `CHANGE MASTER TO` statement (before MySQL 8.0.23) that indicates the binary log coordinates (file name and position) of the dumped replica's source. The `CHANGE REPLICATION SOURCE TO` statement reads the values of `Relay_Master_Log_File` and `Exec_Master_Log_Pos` from the `SHOW REPLICA STATUS` output and uses them for `SOURCE_LOG_FILE` and `SOURCE_LOG_POS` respectively. These are the replication source server coordinates from which the replica starts replicating.

  

  Note

  Inconsistencies in the sequence of transactions from the relay log which have been executed can cause the wrong position to be used. See Section 17.5.1.34, “Replication and Transaction Inconsistencies” for more information.

  `--dump-replica` or `--dump-slave` cause the coordinates from the source to be used rather than those of the dumped server, as is done by the `--source-data` or `--master-data` option. In addition, specifying this option causes the `--source-data` or `--master-data` option to be overridden, if used, and effectively ignored.

  

  Warning

  `--dump-replica` and `--dump-slave` should not be used if the server where the dump is going to be applied uses `gtid_mode=ON` and `SOURCE_AUTO_POSITION=1` or `MASTER_AUTO_POSITION=1`.

  The option value is handled the same way as for `--source-data`. Setting no value or 1 causes a `CHANGE REPLICATION SOURCE TO` statement (from MySQL 8.0.23) or `CHANGE MASTER TO` statement (before MySQL 8.0.23) to be written to the dump. Setting 2 causes the statement to be written but encased in SQL comments. It has the same effect as `--source-data` in terms of enabling or disabling other options and in how locking is handled.

  `--dump-replica` and `--dump-slave` cause `mysqldump` to stop the replication SQL thread before the dump and restart it again after.

  `--dump-replica` and `--dump-slave` send a `SHOW REPLICA STATUS` statement to the server to obtain information, so they require privileges sufficient to execute that statement.

  `--apply-replica-statements` and `--include-source-host-port` options can be used in conjunction with `--dump-replica` and `--dump-slave`.

- `--dump-slave[=value]`

  Use this option before MySQL 8.0.26 rather than `--dump-replica`. Both options have the same effect.

- `--include-source-host-port`

  From MySQL 8.0.26, use `--include-source-host-port`, and before MySQL 8.0.26, use `--include-master-host-port`. Both options have the same effect. The options add the `SOURCE_HOST` | `MASTER_HOST` and `SOURCE_PORT` | `MASTER_PORT` options for the host name and TCP/IP port number of the replica's source, to the `CHANGE REPLICATION SOURCE TO` statement

(from MySQL 8.0.23) or `CHANGE MASTER TO` statement (before MySQL 8.0.23) in a replica dump produced with the `--dump-replica` or `--dump-slave` option.

- `--include-master-host-port`

  Use this option before MySQL 8.0.26 rather than `--include-source-host-port`. Both options have the same effect.

- `--source-data[=value]`

  From MySQL 8.0.26, use `--source-data`, and before MySQL 8.0.26, use `--master-data`. Both options have the same effect. The options are used to dump a replication source server to produce a dump file that can be used to set up another server as a replica of the source. The options cause the dump output to include a `CHANGE REPLICATION SOURCE TO` statement (from MySQL 8.0.23) or `CHANGE MASTER TO` statement (before MySQL 8.0.23) that indicates the binary log coordinates (file name and position) of the dumped server. These are the replication source server coordinates from which the replica should start replicating after you load the dump file into the replica.

  If the option value is 2, the `CHANGE REPLICATION SOURCE TO` | `CHANGE MASTER TO` statement is written as an SQL comment, and thus is informative only; it has no effect when the dump file is reloaded. If the option value is 1, the statement is not written as a comment and takes effect when the dump file is reloaded. If no option value is specified, the default value is 1.

  `--source-data` and `--master-data` send a `SHOW MASTER STATUS` statement to the server to obtain information, so they require privileges sufficient to execute that statement. This option also requires the `RELOAD` privilege and the binary log must be enabled.

  `--source-data` and `--master-data` automatically turn off `--lock-tables`. They also turn on `--lock-all-tables`, unless `--single-transaction` also is specified, in which case, a global read lock is acquired only for a short time at the beginning of the dump (see the description for `--single-transaction`). In all cases, any action on logs happens at the exact moment of the dump.

  It is also possible to set up a replica by dumping an existing replica of the source, using the `--dump-replica` or `--dump-slave` option, which overrides `--source-data` and `--master-data` and causes them to be ignored.

- `--master-data[=value]`

  Use this option before MySQL 8.0.26 rather than `--source-data`. Both options have the same effect.

- `--set-gtid-purged=value`

  This option is for servers that use GTID-based replication (`gtid_mode=ON`). It controls the inclusion of a `SET @@GLOBAL.gtid_purged` statement in the dump output, which updates the value of `gtid_purged` on a server where the dump file is reloaded, to add the GTID set from the source server's `gtid_executed` system variable. `gtid_purged` holds the GTIDs of all transactions that have been applied on the server, but do not exist on any binary log file on the server. `mysqldump` therefore adds the GTIDs for the transactions that were executed on the source server, so that the target server records these transactions as applied, although it does not have them in its binary logs. `--set-gtid-purged` also controls the inclusion of a `SET @@SESSION.sql_log_bin=0` statement, which disables binary logging while the dump file is being reloaded. This statement prevents new GTIDs from being generated and assigned to the transactions in the dump file as they are executed, so that the original GTIDs for the transactions are used.

  If you do not set the `--set-gtid-purged` option, the default is that a `SET @@GLOBAL.gtid_purged` statement is included in the dump output if GTIDs are enabled on the server you are backing up, and the set of GTIDs in the global value of the `gtid_executed` system

variable is not empty. A `SET @@SESSION.sql_log_bin=0` statement is also included if GTIDs are enabled on the server.

You can either replace the value of `gtid_purged` with a specified GTID set, or add a plus sign (+) to the statement to append a specified GTID set to the GTID set that is already held by `gtid_purged`. The `SET @@GLOBAL.gtid_purged` statement recorded by `mysqldump` includes a plus sign (`+`) in a version-specific comment, such that MySQL 8.0 (and later) adds the GTID set from the dump file to the existing `gtid_purged` value.

It is important to note that the value that is included by `mysqldump` for the `SET @@GLOBAL.gtid_purged` statement includes the GTIDs of all transactions in the `gtid_executed` set on the server, even those that changed suppressed parts of the database, or other databases on the server that were not included in a partial dump. This can mean that after the `gtid_purged` value has been updated on the server where the dump file is replayed, GTIDs are present that do not relate to any data on the target server. If you do not replay any further dump files on the target server, the extraneous GTIDs do not cause any problems with the future operation of the server, but they make it harder to compare or reconcile GTID sets on different servers in the replication topology. If you do replay a further dump file on the target server that contains the same GTIDs (for example, another partial dump from the same origin server), any `SET @@GLOBAL.gtid_purged` statement in the second dump file fails. In this case, either remove the statement manually before replaying the dump file, or output the dump file without the statement.

Using this option with the `--single-transaction` option can lead to inconsistencies in the output. If `--set-gtid-purged=ON` is required, it can be used with `--lock-all-tables`, but this can prevent parallel queries while `mysqldump` is being run.

If the `SET @@GLOBAL.gtid_purged` statement would not have the desired result on your target server, you can exclude the statement from the output, or (from MySQL 8.0.17) include it but comment it out so that it is not actioned automatically. You can also include the statement but manually edit it in the dump file to achieve the desired result.

The possible values for the `--set-gtid-purged` option are as follows:

`AUTO`
: The default value. If GTIDs are enabled on the server you are backing up and `gtid_executed` is not empty, `SET @@GLOBAL.gtid_purged` is added to the output, containing the GTID set from `gtid_executed`. If GTIDs are enabled, `SET @@SESSION.sql_log_bin=0` is added to the output. If GTIDs are not enabled on the server, the statements are not added to the output.

`OFF`
: `SET @@GLOBAL.gtid_purged` is not added to the output, and `SET @@SESSION.sql_log_bin=0` is not added to the output. For a server where GTIDs are not in use, use this option or `AUTO`. Only use this option for a server where GTIDs are in use if you are sure that the required GTID set is already present in `gtid_purged` on the target server and should not be changed, or if you plan to identify and add any missing GTIDs manually.

`ON`
: If GTIDs are enabled on the server you are backing up, `SET @@GLOBAL.gtid_purged` is added to the output (unless `gtid_executed` is empty), and `SET @@SESSION.sql_log_bin=0` is added to the output. An error occurs if you set this option but GTIDs are not enabled on the server. For a server where GTIDs are in use, use this option or `AUTO`, unless you are sure that the GTIDs in `gtid_executed` are not needed on the target server.

`COMMENTED`	Available from MySQL 8.0.17. If GTIDs are enabled on the server you are backing up, `SET @@GLOBAL.gtid_purged` is added to the output (unless `gtid_executed` is empty), but it is commented out. This means that the value of `gtid_executed` is available in the output, but no action is taken automatically when the dump file is reloaded. `SET @@SESSION.sql_log_bin=0` is added to the output, and it is not commented out. With `COMMENTED`, you can control the use of the `gtid_executed` set manually or through automation. For example, you might prefer to do this if you are migrating data to another server that already has different active databases.

Format Options

The following options specify how to represent the entire dump file or certain kinds of data in the dump file. They also control whether certain optional information is written to the dump file.

- `--compact`

  Produce more compact output. This option enables the `--skip-add-drop-table`, `--skip-add-locks`, `--skip-comments`, `--skip-disable-keys`, and `--skip-set-charset` options.

- `--compatible=`*name*

  Produce output that is more compatible with other database systems or with older MySQL servers. The only permitted value for this option is `ansi`, which has the same meaning as the corresponding option for setting the server SQL mode. See Section 5.1.11, “Server SQL Modes”.

- `--complete-insert`, `-c`

  Use complete `INSERT` statements that include column names.

- `--create-options`

  Include all MySQL-specific table options in the `CREATE TABLE` statements.

- `--fields-terminated-by=...`, `--fields-enclosed-by=...`, `--fields-optionally-enclosed-by=...`, `--fields-escaped-by=...`

  These options are used with the `--tab` option and have the same meaning as the corresponding `FIELDS` clauses for `LOAD DATA`. See Section 13.2.9, “LOAD DATA Statement”.

- `--hex-blob`

  Dump binary columns using hexadecimal notation (for example, `'abc'` becomes `0x616263`). The affected data types are `BINARY`, `VARBINARY`, `BLOB` types, `BIT`, all spatial data types, and other non-binary data types when used with the `binary` character set.

  The `--hex-blob` option is ignored when the `--tab` is used.

- `--lines-terminated-by=...`

  This option is used with the `--tab` option and has the same meaning as the corresponding `LINES` clause for `LOAD DATA`. See Section 13.2.9, “LOAD DATA Statement”.

- `--quote-names`, `-Q`

  Quote identifiers (such as database, table, and column names) within ` characters. If the `ANSI_QUOTES` SQL mode is enabled, identifiers are quoted within " characters. This option is enabled by default. It can be disabled with `--skip-quote-names`, but this option should be given after any option such as `--compatible` that may enable `--quote-names`.

- `--result-file=file_name`, `-r file_name`

  Direct output to the named file. The result file is created and its previous contents overwritten, even if an error occurs while generating the dump.

  This option should be used on Windows to prevent newline `\n` characters from being converted to `\r\n` carriage return/newline sequences.

- `--show-create-skip-secondary-engine=value`

  Excludes the `SECONDARY ENGINE` clause from `CREATE TABLE` statements. It does so by enabling the `show_create_table_skip_secondary_engine` system variable for the duration of the dump operation. Alternatively, you can enable the `show_create_table_skip_secondary_engine` system variable prior to using `mysqldump`.

  This option was added in MySQL 8.0.18. Attempting a `mysqldump` operation with the `--show-create-skip-secondary-engine` option on a release prior to MySQL 8.0.18 that does not support the `show_create_table_skip_secondary_engine` variable causes an error.

- `--tab=dir_name`, `-T dir_name`

  Produce tab-separated text-format data files. For each dumped table, `mysqldump` creates a `tbl_name.sql` file that contains the `CREATE TABLE` statement that creates the table, and the server writes a `tbl_name.txt` file that contains its data. The option value is the directory in which to write the files.

  

  Note

  This option should be used only when `mysqldump` is run on the same machine as the `mysqld` server. Because the server creates `*.txt` files in the directory that you specify, the directory must be writable by the server and the MySQL account that you use must have the `FILE` privilege. Because `mysqldump` creates `*.sql` in the same directory, it must be writable by your system login account.

  By default, the `.txt` data files are formatted using tab characters between column values and a newline at the end of each line. The format can be specified explicitly using the `--fields-xxx` and `--lines-terminated-by` options.

  Column values are converted to the character set specified by the `--default-character-set` option.

- `--tz-utc`

  This option enables `TIMESTAMP` columns to be dumped and reloaded between servers in different time zones. `mysqldump` sets its connection time zone to UTC and adds `SET TIME_ZONE='+00:00'` to the dump file. Without this option, `TIMESTAMP` columns are dumped and reloaded in the time zones local to the source and destination servers, which can cause the values to change if the servers are in different time zones. `--tz-utc` also protects against changes due to daylight saving time. `--tz-utc` is enabled by default. To disable it, use `--skip-tz-utc`.

- `--xml`, `-X`

  Write dump output as well-formed XML.

  `NULL`, `'NULL'`, and Empty Values: For a column named *`column_name`*, the `NULL` value, an empty string, and the string value `'NULL'` are distinguished from one another in the output generated by this option as follows.

Value:	XML Representation:
`NULL` (*unknown value*)	`<field name="column_name" xsi:nil="true" />`
`''` (*empty string*)	`<field name="column_name"></field>`
`'NULL'` (*string value*)	`<field name="column_name">NULL</field>`

  The output from the `mysql` client when run using the `--xml` option also follows the preceding rules. (See Section 4.5.1.1, “mysql Client Options”.)

  XML output from `mysqldump` includes the XML namespace, as shown here:

```
$> mysqldump --xml -u root world City
<?xml version="1.0"?>
<mysqldump xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<database name="world">
<table_structure name="City">
<field Field="ID" Type="int(11)" Null="NO" Key="PRI" Extra="auto_increment" />
<field Field="Name" Type="char(35)" Null="NO" Key="" Default="" Extra="" />
<field Field="CountryCode" Type="char(3)" Null="NO" Key="" Default="" Extra="" />
<field Field="District" Type="char(20)" Null="NO" Key="" Default="" Extra="" />
<field Field="Population" Type="int(11)" Null="NO" Key="" Default="0" Extra="" />
<key Table="City" Non_unique="0" Key_name="PRIMARY" Seq_in_index="1" Column_name="ID"
Collation="A" Cardinality="4079" Null="" Index_type="BTREE" Comment="" />
<options Name="City" Engine="MyISAM" Version="10" Row_format="Fixed" Rows="4079"
Avg_row_length="67" Data_length="273293" Max_data_length="18858823439613951"
Index_length="43008" Data_free="0" Auto_increment="4080"
Create_time="2007-03-31 01:47:01" Update_time="2007-03-31 01:47:02"
Collation="latin1_swedish_ci" Create_options="" Comment="" />
</table_structure>
<table_data name="City">
<row>
<field name="ID">1</field>
<field name="Name">Kabul</field>
<field name="CountryCode">AFG</field>
<field name="District">Kabol</field>
<field name="Population">1780000</field>
</row>

...

<row>
<field name="ID">4079</field>
<field name="Name">Rafah</field>
<field name="CountryCode">PSE</field>
<field name="District">Rafah</field>
<field name="Population">92020</field>
</row>
</table_data>
</database>
</mysqldump>
```

Filtering Options

The following options control which kinds of schema objects are written to the dump file: by category, such as triggers or events; by name, for example, choosing which databases and tables to dump; or even filtering rows from the table data using a `WHERE` clause.

- `--all-databases`, `-A`

  Dump all tables in all databases. This is the same as using the `--databases` option and naming all the databases on the command line.

  

  Note

  See the `--add-drop-database` description for information about an incompatibility of that option with `--all-databases`.

  Prior to MySQL 8.0, the `--routines` and `--events` options for `mysqldump` and `mysqlpump` were not required to include stored routines and events when using the `--all-databases` option: The dump included the `mysql` system database, and therefore also the `mysql.proc` and `mysql.event` tables containing stored routine and event definitions. As of MySQL 8.0, the `mysql.event` and `mysql.proc` tables are not used. Definitions for the corresponding objects are stored in data dictionary tables, but those tables are not dumped. To include stored routines and events in a dump made using `--all-databases`, use the `--routines` and `--events` options explicitly.

- `--databases`, `-B`

  Dump several databases. Normally, `mysqldump` treats the first name argument on the command line as a database name and following names as table names. With this option, it treats all name arguments as database names. `CREATE DATABASE` and `USE` statements are included in the output before each new database.

  This option may be used to dump the `performance_schema` database, which normally is not dumped even with the `--all-databases` option. (Also use the `--skip-lock-tables` option.)

  

  Note

  See the `--add-drop-database` description for information about an incompatibility of that option with `--databases`.

- `--events`, `-E`

  Include Event Scheduler events for the dumped databases in the output. This option requires the `EVENT` privileges for those databases.

  The output generated by using `--events` contains `CREATE EVENT` statements to create the events.

- `--ignore-error=`*`error[,error]...`*

  Ignore the specified errors. The option value is a list of comma-separated error numbers specifying the errors to ignore during `mysqldump` execution. If the `--force` option is also given to ignore all errors, `--force` takes precedence.

- `--ignore-table=`*`db_name.tbl_name`*

  Do not dump the given table, which must be specified using both the database and table names. To ignore multiple tables, use this option multiple times. This option also can be used to ignore views.

- `--no-data`, `-d`

  Do not write any table row information (that is, do not dump table contents). This is useful if you want to dump only the `CREATE TABLE` statement for the table (for example, to create an empty copy of the table by loading the dump file).

- `--routines`, `-R`

  Include stored routines (procedures and functions) for the dumped databases in the output. This option requires the global `SELECT` privilege.

  The output generated by using `--routines` contains `CREATE PROCEDURE` and `CREATE FUNCTION` statements to create the routines.

- `--skip-generated-invisible-primary-key`

  This option is available beginning with MySQL 8.0.30, and causes generated invisible primary keys to be excluded from the output. For more information, see Section 13.1.20.11, “Generated Invisible Primary Keys”.

- `--tables`

  Override the `--databases` or `-B` option. `mysqldump` regards all name arguments following the option as table names.

- `--triggers`

  Include triggers for each dumped table in the output. This option is enabled by default; disable it with `--skip-triggers`.

  To be able to dump a table's triggers, you must have the `TRIGGER` privilege for the table.

  Multiple triggers are permitted. `mysqldump` dumps triggers in activation order so that when the dump file is reloaded, triggers are created in the same activation order. However, if a `mysqldump` dump file contains multiple triggers for a table that have the same trigger event and action time, an error occurs for attempts to load the dump file into an older server that does not support multiple triggers. (For a workaround, see Downgrade Notes; you can convert triggers to be compatible with older servers.)

- `--where='`*`where_condition`*`'`, `-w '`*`where_condition`*`'`

  Dump only rows selected by the given `WHERE` condition. Quotes around the condition are mandatory if it contains spaces or other characters that are special to your command interpreter.

  Examples:

```
--where="user='jimf'"
-w"userid>1"
-w"userid<1"
```

Performance Options

The following options are the most relevant for the performance particularly of the restore operations. For large data sets, restore operation (processing the `INSERT` statements in the dump file) is the most time-consuming part. When it is urgent to restore data quickly, plan and test the performance of this stage in advance. For restore times measured in hours, you might prefer an alternative backup and restore solution, such as MySQL Enterprise Backup for `InnoDB`-only and mixed-use databases.

Performance is also affected by the transactional options, primarily for the dump operation.

- `--column-statistics`

  Add `ANALYZE TABLE` statements to the output to generate histogram statistics for dumped tables when the dump file is reloaded. This option is disabled by default because histogram generation for large tables can take a long time.

- `--disable-keys`, `-K`

For each table, surround the `INSERT` statements with `/*!40000 ALTER TABLE tbl_name DISABLE KEYS */;` and `/*!40000 ALTER TABLE tbl_name ENABLE KEYS */;` statements. This makes loading the dump file faster because the indexes are created after all rows are inserted. This option is effective only for nonunique indexes of `MyISAM` tables.

- `--extended-insert`, `-e`

  Write `INSERT` statements using multiple-row syntax that includes several `VALUES` lists. This results in a smaller dump file and speeds up inserts when the file is reloaded.

- `--insert-ignore`

  Write `INSERT IGNORE` statements rather than `INSERT` statements.

- `--max-allowed-packet=value`

  The maximum size of the buffer for client/server communication. The default is 24MB, the maximum is 1GB.

  

  Note

  The value of this option is specific to `mysqldump` and should not be confused with the MySQL server's `max_allowed_packet` system variable; the server value cannot be exceeded by a single packet from `mysqldump`, regardless of any setting for the `mysqldump` option, even if the latter is larger.

- `--mysqld-long-query-time=value`

  Set the session value of the `long_query_time` system variable. Use this option, which is available from MySQL 8.0.30, if you want to increase the time allowed for `mysqldump`'s queries before they are logged to the slow query log file. `mysqldump` performs a full table scan, which means its queries can often exceed a global `long_query_time` setting that is useful for regular queries. The default global setting is 10 seconds.

  You can use `--mysqld-long-query-time` to specify a session value from 0 (meaning that every query from `mysqldump` is logged to the slow query log) to 31536000, which is 365 days in seconds. For `mysqldump`'s option, you can only specify whole seconds. When you do not specify this option, the server's global setting applies to `mysqldump`'s queries.

- `--net-buffer-length=value`

  The initial size of the buffer for client/server communication. When creating multiple-row `INSERT` statements (as with the `--extended-insert` or `--opt` option), `mysqldump` creates rows up to `--net-buffer-length` bytes long. If you increase this variable, ensure that the MySQL server `net_buffer_length` system variable has a value at least this large.

- `--network-timeout`, `-M`

  Enable large tables to be dumped by setting `--max-allowed-packet` to its maximum value and network read and write timeouts to a large value. This option is enabled by default. To disable it, use `--skip-network-timeout`.

- `--opt`

  This option, enabled by default, is shorthand for the combination of `--add-drop-table --add-locks --create-options --disable-keys --extended-insert --lock-tables --quick`

`--set-charset`. It gives a fast dump operation and produces a dump file that can be reloaded into a MySQL server quickly.

Because the `--opt` option is enabled by default, you only specify its converse, the `--skip-opt` to turn off several default settings. See the discussion of `mysqldump` option groups for information about selectively enabling or disabling a subset of the options affected by `--opt`.

- `--quick`, `-q`

  This option is useful for dumping large tables. It forces `mysqldump` to retrieve rows for a table from the server a row at a time rather than retrieving the entire row set and buffering it in memory before writing it out.

- `--skip-opt`

  See the description for the `--opt` option.

Transactional Options

The following options trade off the performance of the dump operation, against the reliability and consistency of the exported data.

- `--add-locks`

  Surround each table dump with `LOCK TABLES` and `UNLOCK TABLES` statements. This results in faster inserts when the dump file is reloaded. See Section 8.2.5.1, “Optimizing INSERT Statements”.

- `--flush-logs`, `-F`

  Flush the MySQL server log files before starting the dump. This option requires the `RELOAD` privilege. If you use this option in combination with the `--all-databases` option, the logs are flushed *for each database dumped*. The exception is when using `--lock-all-tables`, `--source-data` or `--master-data`, or `--single-transaction`. In these cases, the logs are flushed only once, corresponding to the moment that all tables are locked by `FLUSH TABLES WITH READ LOCK`. If you want your dump and the log flush to happen at exactly the same moment, you should use `--flush-logs` together with `--lock-all-tables`, `--source-data` or `--master-data`, or `--single-transaction`.

- `--flush-privileges`

  Add a `FLUSH PRIVILEGES` statement to the dump output after dumping the `mysql` database. This option should be used any time the dump contains the `mysql` database and any other database that depends on the data in the `mysql` database for proper restoration.

  Because the dump file contains a `FLUSH PRIVILEGES` statement, reloading the file requires privileges sufficient to execute that statement.

  

  Note

  For upgrades to MySQL 5.7 or higher from older versions, do not use `--flush-privileges`. For upgrade instructions in this case, see Section 2.10.4, “Changes in MySQL 8.0”.

- `--lock-all-tables`, `-x`

  Lock all tables across all databases. This is achieved by acquiring a global read lock for the duration of the whole dump. This option automatically turns off `--single-transaction` and `--lock-tables`.

- `--lock-tables`, `-l`

For each dumped database, lock all tables to be dumped before dumping them. The tables are locked with `READ LOCAL` to permit concurrent inserts in the case of `MyISAM` tables. For transactional tables such as `InnoDB`, `--single-transaction` is a much better option than `--lock-tables` because it does not need to lock the tables at all.

Because `--lock-tables` locks tables for each database separately, this option does not guarantee that the tables in the dump file are logically consistent between databases. Tables in different databases may be dumped in completely different states.

Some options, such as `--opt`, automatically enable `--lock-tables`. If you want to override this, use `--skip-lock-tables` at the end of the option list.

- `--no-autocommit`

  Enclose the `INSERT` statements for each dumped table within `SET autocommit = 0` and `COMMIT` statements.

- `--order-by-primary`

  Dump each table's rows sorted by its primary key, or by its first unique index, if such an index exists. This is useful when dumping a `MyISAM` table to be loaded into an `InnoDB` table, but makes the dump operation take considerably longer.

- `--shared-memory-base-name=`*name*

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--single-transaction`

  This option sets the transaction isolation mode to `REPEATABLE READ` and sends a `START TRANSACTION` SQL statement to the server before dumping data. It is useful only with transactional tables such as `InnoDB`, because then it dumps the consistent state of the database at the time when `START TRANSACTION` was issued without blocking any applications.

  The RELOAD or FLUSH_TABLES privilege is required with `--single-transaction` if both gtid_mode=ON and --set-gtid=purged=ON|AUTO. This requirement was added in MySQL 8.0.32.

  When using this option, you should keep in mind that only `InnoDB` tables are dumped in a consistent state. For example, any `MyISAM` or `MEMORY` tables dumped while using this option may still change state.

  While a `--single-transaction` dump is in process, to ensure a valid dump file (correct table contents and binary log coordinates), no other connection should use the following statements: `ALTER TABLE`, `CREATE TABLE`, `DROP TABLE`, `RENAME TABLE`, `TRUNCATE TABLE`. A consistent read is not isolated from those statements, so use of them on a table to be dumped can cause the `SELECT` that is performed by `mysqldump` to retrieve the table contents to obtain incorrect contents or fail.

  The `--single-transaction` option and the `--lock-tables` option are mutually exclusive because `LOCK TABLES` causes any pending transactions to be committed implicitly.

  Using `--single-transaction` together with the `--set-gtid-purged` option is not recommended; doing so can lead to inconsistencies in the output of `mysqldump`.

  To dump large tables, combine the `--single-transaction` option with the `--quick` option.

Option Groups

- The `--opt` option turns on several settings that work together to perform a fast dump operation. All of these settings are on by default, because `--opt` is on by default. Thus you rarely if ever specify `--opt`. Instead, you can turn these settings off as a group by specifying `--skip-opt`, then optionally re-enable certain settings by specifying the associated options later on the command line.
- The `--compact` option turns off several settings that control whether optional statements and comments appear in the output. Again, you can follow this option with other options that re-enable certain settings, or turn all the settings on by using the `--skip-compact` form.

When you selectively enable or disable the effect of a group option, order is important because options are processed first to last. For example, `--disable-keys --lock-tables --skip-opt` would not have the intended effect; it is the same as `--skip-opt` by itself.

Examples

To make a backup of an entire database:

```
mysqldump db_name > backup-file.sql
```

To load the dump file back into the server:

```
mysql db_name < backup-file.sql
```

Another way to reload the dump file:

```
mysql -e "source /path-to-backup/backup-file.sql" db_name
```

`mysqldump` is also very useful for populating databases by copying data from one MySQL server to another:

```
mysqldump --opt db_name | mysql --host=remote_host -C db_name
```

You can dump several databases with one command:

```
mysqldump --databases db_name1 [db_name2 ...] > my_databases.sql
```

To dump all databases, use the `--all-databases` option:

```
mysqldump --all-databases > all_databases.sql
```

For `InnoDB` tables, `mysqldump` provides a way of making an online backup:

```
mysqldump --all-databases --master-data --single-transaction > all_databases.sql

or from MySQL 8.0.26:
mysqldump --all-databases --source-data --single-transaction > all_databases.sql
```

This backup acquires a global read lock on all tables (using `FLUSH TABLES WITH READ LOCK`) at the beginning of the dump. As soon as this lock has been acquired, the binary log coordinates are read and the lock is released. If long updating statements are running when the `FLUSH` statement is issued, the MySQL server may get stalled until those statements finish. After that, the dump becomes lock free and does not disturb reads and writes on the tables. If the update statements that the MySQL server receives are short (in terms of execution time), the initial lock period should not be noticeable, even with many updates.

For point-in-time recovery (also known as “roll-forward,” when you need to restore an old backup and replay the changes that happened since that backup), it is often useful to rotate the binary log (see Section 5.4.4, “The Binary Log”) or at least know the binary log coordinates to which the dump corresponds:

```
mysqldump --all-databases --master-data=2 > all_databases.sql
```

```
or from MySQL 8.0.26:
mysqldump --all-databases --source-data=2 > all_databases.sql
```

Or:

```
mysqldump --all-databases --flush-logs --master-data=2 > all_databases.sql

or from MySQL 8.0.26:
mysqldump --all-databases --flush-logs --source-data=2 > all_databases.sql
```

The `--source-data` or `--master-data` option can be used simultaneously with the `--single-transaction` option, which provides a convenient way to make an online backup suitable for use prior to point-in-time recovery if tables are stored using the `InnoDB` storage engine.

For more information on making backups, see Section 7.2, “Database Backup Methods”, and Section 7.3, “Example Backup and Recovery Strategy”.

- To select the effect of `--opt` except for some features, use the `--skip` option for each feature. To disable extended inserts and memory buffering, use `--opt --skip-extended-insert --skip-quick`. (Actually, `--skip-extended-insert --skip-quick` is sufficient because `--opt` is on by default.)
- To reverse `--opt` for all features except disabling of indexes and table locking, use `--skip-opt --disable-keys --lock-tables`.

Restrictions

`mysqldump` does not dump the `performance_schema` or `sys` schema by default. To dump any of these, name them explicitly on the command line. You can also name them with the `--databases` option. For `performance_schema`, also use the `--skip-lock-tables` option.

`mysqldump` does not dump the `INFORMATION_SCHEMA` schema.

`mysqldump` does not dump `InnoDB CREATE TABLESPACE` statements.

`mysqldump` does not dump the NDB Cluster `ndbinfo` information database.

`mysqldump` includes statements to recreate the `general_log` and `slow_query_log` tables for dumps of the `mysql` database. Log table contents are not dumped.

If you encounter problems backing up views due to insufficient privileges, see Section 25.9, “Restrictions on Views” for a workaround.

4.5.5 mysqlimport — A Data Import Program

The `mysqlimport` client provides a command-line interface to the `LOAD DATA` SQL statement. Most options to `mysqlimport` correspond directly to clauses of `LOAD DATA` syntax. See Section 13.2.9, “LOAD DATA Statement”.

Invoke `mysqlimport` like this:

```
mysqlimport [options] db_name textfile1 [textfile2 ...]
```

For each text file named on the command line, `mysqlimport` strips any extension from the file name and uses the result to determine the name of the table into which to import the file's contents. For example, files named `patient.txt`, `patient.text`, and `patient` all would be imported into a table named `patient`.

`mysqlimport` supports the following options, which can be specified on the command line or in the `[mysqlimport]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.15 mysqlimport Options

Option Name	Description	Introduced	Deprecated
--bind-address	Use specified network interface to connect to MySQL Server		
--columns	This option takes a comma-separated list of column names as its value		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--delete	Empty the table before importing the text file		
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--fields-enclosed-by	This option has the same meaning as the corresponding clause for LOAD DATA		
--fields-escaped-by	This option has the same meaning as the corresponding clause for LOAD DATA		
--fields-optionally-enclosed-by	This option has the same meaning as the corresponding clause for LOAD DATA		

Option Name	Description	Introduced	Deprecated
--fields-terminated-by	This option has the same meaning as the corresponding clause for LOAD DATA		
--force	Continue even if an SQL error occurs		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--host	Host on which MySQL server is located		
--ignore	See the description for the --replace option		
--ignore-lines	Ignore the first N lines of the data file		
--lines-terminated-by	This option has the same meaning as the corresponding clause for LOAD DATA		
--local	Read input files locally from the client host		
--lock-tables	Lock all tables for writing before processing any text files		
--login-path	Read login path options from .mylogin.cnf		
--low-priority	Use LOW_PRIORITY when loading the table		
--no-defaults	Read no option files		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--pipe	Connect to server using named pipe (Windows only)		
--plugin-dir	Directory where plugins are installed		

Option Name	Description	Introduced	Deprecated
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--replace	The --replace and --ignore options control handling of input rows that duplicate existing rows on unique key values		
--server-public-key-path	Path name to file containing RSA public key		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--silent	Produce output only when errors occur		
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	

Option Name	Description	Introduced	Deprecated
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--use-threads	Number of threads for parallel file-loading		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		
--version	Display version information and exit		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`, `-?`

  Display a help message and exit.

- `--bind-address=`*`ip_address`*

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=`*`dir_name`*

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--columns=`*`column_list`*, `-c` *`column_list`*

  This option takes a list of comma-separated column names as its value. The order of the column names indicates how to match data file columns with table columns.

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=`*`value`*

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:o`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`

  Print debugging information and memory and CPU usage statistics when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysqlimport` normally reads the `[client]` and `[mysqlimport]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqlimport` also reads the `[client_other]` and `[mysqlimport_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--delete`, `-D`

  Empty the table before importing the text file.

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--fields-terminated-by=...`, `--fields-enclosed-by=...`, `--fields-optionally-enclosed-by=...`, `--fields-escaped-by=...`

  These options have the same meaning as the corresponding clauses for `LOAD DATA`. See Section 13.2.9, “LOAD DATA Statement”.

- `--force`, `-f`

  Ignore errors. For example, if a table for a text file does not exist, continue processing any remaining files. Without `--force`, `mysqlimport` exits if a table does not exist.

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=`*`file_name`* is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--host=`*`host_name`*, `-h` *`host_name`*

  Import data to the MySQL server on the given host. The default host is `localhost`.

- `--ignore`, `-i`

  See the description for the `--replace` option.

- `--ignore-lines=`*`N`*

  Ignore the first *`N`* lines of the data file.

- `--lines-terminated-by=...`

  This option has the same meaning as the corresponding clause for `LOAD DATA`. For example, to import Windows files that have lines terminated with carriage return/linefeed pairs, use `--lines-terminated-by="\r\n"`. (You might have to double the backslashes, depending on the escaping conventions of your command interpreter.) See Section 13.2.9, “LOAD DATA Statement”.

- `--local`, `-L`

  By default, files are read by the server on the server host. With this option, `mysqlimport` reads input files locally on the client host.

  Successful use of `LOCAL` load operations within `mysqlimport` also requires that the server permits local loading; see Section 6.1.6, “Security Considerations for LOAD DATA LOCAL”

- `--lock-tables`, `-l`

  Lock *all* tables for writing before processing any text files. This ensures that all tables are synchronized on the server.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--low-priority`

  Use `LOW_PRIORITY` when loading the table. This affects only storage engines that use only table-level locking (such as `MyISAM`, `MEMORY`, and `MERGE`).

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlimport` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlimport` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlimport` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlimport` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqlimport` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--replace`, `-r`

  The `--replace` and `--ignore` options control handling of input rows that duplicate existing rows on unique key values. If you specify `--replace`, new rows replace existing rows that have the same unique key value. If you specify `--ignore`, input rows that duplicate an existing row on a unique key value are skipped. If you do not specify either option, an error occurs when a duplicate key value is found, and the rest of the text file is ignored.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-

based password exchange is not used, as is the case when the client connects to the server using a secure connection.

If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, "SHA-256 Pluggable Authentication", and Section 6.4.1.2, "Caching SHA-2 Pluggable Authentication".

- `--shared-memory-base-name=name`

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--silent`, `-s`

  Silent mode. Produce output only when errors occur.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, "FIPS Support".

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable "strict" FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

- `--use-threads=N`

  Load files in parallel using `N` threads.

- `--verbose`, `-v`

  Verbose mode. Print more information about what the program does.

- `--version`, `-V`

  Display version information and exit.

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

Here is a sample session that demonstrates use of `mysqlimport`:

```
$> mysql -e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
$> ed
a
100     Max Sydow
101     Count Dracula
.
w imptest.txt
32
q
$> od -c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   C   o   u   n   t       D   r   a   c   u   l   a  \n
0000040
$> mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$> mysql -e 'SELECT * FROM imptest' test
+------+--------------+
| id   | n            |
+------+--------------+
|  100 | Max Sydow    |
|  101 | Count Dracula |
```

```
+------+---------------+
```

4.5.6 mysqlpump — A Database Backup Program

The `mysqlpump` client utility performs logical backups, producing a set of SQL statements that can be executed to reproduce the original database object definitions and table data. It dumps one or more MySQL databases for backup or transfer to another SQL server.

Tip

Consider using the MySQL Shell dump utilities, which provide parallel dumping with multiple threads, file compression, and progress information display, as well as cloud features such as Oracle Cloud Infrastructure Object Storage streaming, and MySQL Database Service compatibility checks and modifications. Dumps can be easily imported into a MySQL Server instance or a MySQL Database Service DB System using the MySQL Shell load dump utilities. Installation instructions for MySQL Shell can be found here.

`mysqlpump` features include:

- Parallel processing of databases, and of objects within databases, to speed up the dump process
- Better control over which databases and database objects (tables, stored programs, user accounts) to dump
- Dumping of user accounts as account-management statements (`CREATE USER`, `GRANT`) rather than as inserts into the `mysql` system database
- Capability of creating compressed output
- Progress indicator (the values are estimates)
- For dump file reloading, faster secondary index creation for `InnoDB` tables by adding indexes after rows are inserted

Note

`mysqlpump` uses MySQL features introduced in MySQL 5.7, and thus assumes use with MySQL 5.7 or higher.

`mysqlpump` requires at least the `SELECT` privilege for dumped tables, `SHOW VIEW` for dumped views, `TRIGGER` for dumped triggers, and `LOCK TABLES` if the `--single-transaction` option is not used. The `SELECT` privilege on the `mysql` system database is required to dump user definitions. Certain options might require other privileges as noted in the option descriptions.

To reload a dump file, you must have the privileges required to execute the statements that it contains, such as the appropriate `CREATE` privileges for objects created by those statements.

Note

A dump made using PowerShell on Windows with output redirection creates a file that has UTF-16 encoding:

```
mysqlpump [options] > dump.sql
```

However, UTF-16 is not permitted as a connection character set (see Section 10.4, “Connection Character Sets and Collations”), so the dump file cannot be loaded correctly. To work around this issue, use the `--result-file` option, which creates the output in ASCII format:

```
mysqlpump [options] --result-file=dump.sql
```

mysqlpump Invocation Syntax

By default, `mysqlpump` dumps all databases (with certain exceptions noted in mysqlpump Restrictions). To specify this behavior explicitly, use the `--all-databases` option:

```
mysqlpump --all-databases
```

To dump a single database, or certain tables within that database, name the database on the command line, optionally followed by table names:

```
mysqlpump db_name
mysqlpump db_name tbl_name1 tbl_name2 ...
```

To treat all name arguments as database names, use the `--databases` option:

```
mysqlpump --databases db_name1 db_name2 ...
```

By default, `mysqlpump` does not dump user account definitions, even if you dump the `mysql` system database that contains the grant tables. To dump grant table contents as logical definitions in the form of `CREATE USER` and `GRANT` statements, use the `--users` option and suppress all database dumping:

```
mysqlpump --exclude-databases=% --users
```

In the preceding command, `%` is a wildcard that matches all database names for the `--exclude-databases` option.

`mysqlpump` supports several options for including or excluding databases, tables, stored programs, and user definitions. See mysqlpump Object Selection.

To reload a dump file, execute the statements that it contains. For example, use the `mysql` client:

```
mysqlpump [options] > dump.sql
mysql < dump.sql
```

The following discussion provides additional `mysqlpump` usage examples.

To see a list of the options `mysqlpump` supports, issue the command `mysqlpump --help`.

mysqlpump Option Summary

`mysqlpump` supports the following options, which can be specified on the command line or in the `[mysqlpump]` and `[client]` groups of an option file. (Prior to MySQL 8.0.20, `mysqlpump` read the `[mysql_dump]` group rather than `[mysqlpump]`. As of 8.0.20, `[mysql_dump]` is still accepted but is deprecated.) For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.16 mysqlpump Options

Option Name	Description	Introduced	Deprecated
--add-drop-database	Add DROP DATABASE statement before each CREATE DATABASE statement		

Option Name	Description	Introduced	Deprecated
--add-drop-table	Add DROP TABLE statement before each CREATE TABLE statement		
--add-drop-user	Add DROP USER statement before each CREATE USER statement		
--add-locks	Surround each table dump with LOCK TABLES and UNLOCK TABLES statements		
--all-databases	Dump all databases		
--bind-address	Use specified network interface to connect to MySQL Server		
--character-sets-dir	Directory where character sets are installed		
--column-statistics	Write ANALYZE TABLE statements to generate statistics histograms		
--complete-insert	Use complete INSERT statements that include column names		
--compress	Compress all information sent between client and server		8.0.18
--compress-output	Output compression algorithm		
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--databases	Interpret all name arguments as database names		
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--default-character-set	Specify default character set		

Option Name	Description	Introduced	Deprecated
--default-parallelism	Default number of threads for parallel processing		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--defer-table-indexes	For reloading, defer index creation until after loading table rows		
--events	Dump events from dumped databases		
--exclude-databases	Databases to exclude from dump		
--exclude-events	Events to exclude from dump		
--exclude-routines	Routines to exclude from dump		
--exclude-tables	Tables to exclude from dump		
--exclude-triggers	Triggers to exclude from dump		
--exclude-users	Users to exclude from dump		
--extended-insert	Use multiple-row INSERT syntax		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--hex-blob	Dump binary columns using hexadecimal notation		
--host	Host on which MySQL server is located		
--include-databases	Databases to include in dump		
--include-events	Events to include in dump		
--include-routines	Routines to include in dump		
--include-tables	Tables to include in dump		
--include-triggers	Triggers to include in dump		

Option Name	Description	Introduced	Deprecated
--include-users	Users to include in dump		
--insert-ignore	Write INSERT IGNORE rather than INSERT statements		
--log-error-file	Append warnings and errors to named file		
--login-path	Read login path options from .mylogin.cnf		
--max-allowed-packet	Maximum packet length to send to or receive from server		
--net-buffer-length	Buffer size for TCP/IP and socket communication		
--no-create-db	Do not write CREATE DATABASE statements		
--no-create-info	Do not write CREATE TABLE statements that re-create each dumped table		
--no-defaults	Read no option files		
--parallel-schemas	Specify schema-processing parallelism		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--replace	Write REPLACE statements rather than INSERT statements		

Option Name	Description	Introduced	Deprecated
--result-file	Direct output to a given file		
--routines	Dump stored routines (procedures and functions) from dumped databases		
--server-public-key-path	Path name to file containing RSA public key		
--set-charset	Add SET NAMES default_character_set to output		
--set-gtid-purged	Whether to add SET @@GLOBAL.GTID_PURGED to output		
--single-transaction	Dump tables within single transaction		
--skip-definer	Omit DEFINER and SQL SECURITY clauses from view and stored program CREATE statements		
--skip-dump-rows	Do not dump table rows		
--skip-generated-invisible-primary-key	Do not dump information about generated invisible primary keys	8.0.30	
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		

Option Name	Description	Introduced	Deprecated
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--triggers	Dump triggers for each dumped table		
--tz-utc	Add SET TIME_ZONE='+00:00' to dump file		
--user	MySQL user name to use when connecting to server		
--users	Dump user accounts		
--version	Display version information and exit		
--watch-progress	Display progress indicator		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

mysqlpump Option Descriptions

- `--help`, `-?`

  Display a help message and exit.

- `--add-drop-database`

  Write a `DROP DATABASE` statement before each `CREATE DATABASE` statement.

  

  Note

  In MySQL 8.0, the `mysql` schema is considered a system schema that cannot be dropped by end users. If `--add-drop-database` is used with `--all-databases` or with `--databases` where the list of schemas to be dumped includes `mysql`, the dump file contains a `DROP DATABASE` `` `mysql` `` statement that causes an error when the dump file is reloaded.

  Instead, to use `--add-drop-database`, use `--databases` with a list of schemas to be dumped, where the list does not include `mysql`.

- `--add-drop-table`

Write a `DROP TABLE` statement before each `CREATE TABLE` statement.

- `--add-drop-user`

  Write a `DROP USER` statement before each `CREATE USER` statement.

- `--add-locks`

  Surround each table dump with `LOCK TABLES` and `UNLOCK TABLES` statements. This results in faster inserts when the dump file is reloaded. See Section 8.2.5.1, “Optimizing INSERT Statements”.

  This option does not work with parallelism because `INSERT` statements from different tables can be interleaved and `UNLOCK TABLES` following the end of the inserts for one table could release locks on tables for which inserts remain.

  `--add-locks` and `--single-transaction` are mutually exclusive.

- `--all-databases`, `-A`

  Dump all databases (with certain exceptions noted in mysqlpump Restrictions). This is the default behavior if no other is specified explicitly.

  `--all-databases` and `--databases` are mutually exclusive.

  

  Note

  See the `--add-drop-database` description for information about an incompatibility of that option with `--all-databases`.

  Prior to MySQL 8.0, the `--routines` and `--events` options for `mysqldump` and `mysqlpump` were not required to include stored routines and events when using the `--all-databases` option: The dump included the `mysql` system database, and therefore also the `mysql.proc` and `mysql.event` tables containing stored routine and event definitions. As of MySQL 8.0, the `mysql.event` and `mysql.proc` tables are not used. Definitions for the corresponding objects are stored in data dictionary tables, but those tables are not dumped. To include stored routines and events in a dump made using `--all-databases`, use the `--routines` and `--events` options explicitly.

- `--bind-address=`*`ip_address`*

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=`*`path`*

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--column-statistics`

  Add `ANALYZE TABLE` statements to the output to generate histogram statistics for dumped tables when the dump file is reloaded. This option is disabled by default because histogram generation for large tables can take a long time.

- `--complete-insert`

  Write complete `INSERT` statements that include column names.

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compress-output=`*`algorithm`*

  By default, `mysqlpump` does not compress output. This option specifies output compression using the specified algorithm. Permitted algorithms are `LZ4` and `ZLIB`.

  To uncompress compressed output, you must have an appropriate utility. If the system commands `lz4` and `openssl zlib` are not available, MySQL distributions include `lz4_decompress` and `zlib_decompress` utilities that can be used to decompress `mysqlpump` output that was compressed using the `--compress-output=LZ4` and `--compress-output=ZLIB` options. For more information, see Section 4.8.1, “lz4_decompress — Decompress mysqlpump LZ4-Compressed Output”, and Section 4.8.3, “zlib_decompress — Decompress mysqlpump ZLIB-Compressed Output”.

- `--compression-algorithms=`*`value`*

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--databases`, `-B`

  Normally, `mysqlpump` treats the first name argument on the command line as a database name and any following names as table names. With this option, it treats all name arguments as database names. `CREATE DATABASE` statements are included in the output before each new database.

  `--all-databases` and `--databases` are mutually exclusive.

  

  Note

  See the `--add-drop-database` description for information about an incompatibility of that option with `--databases`.

- `--debug[=`*`debug_options`*`]`, `-# [`*`debug_options`*`]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,`*`file_name`*. The default is `d:t:O,/tmp/mysqlpump.trace`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`, `-T`

  Print debugging information and memory and CPU usage statistics when the program exits.

This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”. If no character set is specified, `mysqlpump` uses `utf8mb4`.

- `--default-parallelism=N`

  The default number of threads for each parallel processing queue. The default is 2.

  The `--parallel-schemas` option also affects parallelism and can be used to override the default number of threads. For more information, see mysqlpump Parallel Processing.

  With `--default-parallelism=0` and no `--parallel-schemas` options, `mysqlpump` runs as a single-threaded process and creates no queues.

  With parallelism enabled, it is possible for output from different databases to be interleaved.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysqlpump` normally reads the `[client]` and `[mysqlpump]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqlpump` also reads the `[client_other]` and `[mysqlpump_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defer-table-indexes`

  In the dump output, defer index creation for each table until after its rows have been loaded. This works for all storage engines, but for `InnoDB` applies only for secondary indexes.

  This option is enabled by default; use `--skip-defer-table-indexes` to disable it.

- `--events`

Include Event Scheduler events for the dumped databases in the output. Event dumping requires the `EVENT` privileges for those databases.

The output generated by using `--events` contains `CREATE EVENT` statements to create the events.

This option is enabled by default; use `--skip-events` to disable it.

- `--exclude-databases=db_list`

  Do not dump the databases in *`db_list`*, which is a list of one or more comma-separated database names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--exclude-events=event_list`

  Do not dump the databases in *`event_list`*, which is a list of one or more comma-separated event names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--exclude-routines=routine_list`

  Do not dump the events in *`routine_list`*, which is a list of one or more comma-separated routine (stored procedure or function) names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--exclude-tables=table_list`

  Do not dump the tables in *`table_list`*, which is a list of one or more comma-separated table names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--exclude-triggers=trigger_list`

  Do not dump the triggers in *`trigger_list`*, which is a list of one or more comma-separated trigger names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--exclude-users=user_list`

  Do not dump the user accounts in *`user_list`*, which is a list of one or more comma-separated account names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--extended-insert=N`

  Write `INSERT` statements using multiple-row syntax that includes several `VALUES` lists. This results in a smaller dump file and speeds up inserts when the file is reloaded.

  The option value indicates the number of rows to include in each `INSERT` statement. The default is 250. A value of 1 produces one `INSERT` statement per table row.

- `--get-server-public-key`

  Request from the server the public key required for RSA key pair-based password exchange. This option applies to clients that authenticate with the `caching_sha2_password` authentication plugin. For that plugin, the server does not send the public key unless requested. This option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based

password exchange is not used, as is the case when the client connects to the server using a secure connection.

If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--hex-blob`

  Dump binary columns using hexadecimal notation (for example, `'abc'` becomes `0x616263`). The affected data types are `BINARY`, `VARBINARY`, `BLOB` types, `BIT`, all spatial data types, and other non-binary data types when used with the `binary` character set.

- `--host=host_name`, `-h host_name`

  Dump data from the MySQL server on the given host.

- `--include-databases=db_list`

  Dump the databases in *`db_list`*, which is a list of one or more comma-separated database names. The dump includes all objects in the named databases. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--include-events=event_list`

  Dump the events in *`event_list`*, which is a list of one or more comma-separated event names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--include-routines=routine_list`

  Dump the routines in *`routine_list`*, which is a list of one or more comma-separated routine (stored procedure or function) names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--include-tables=table_list`

  Dump the tables in *`table_list`*, which is a list of one or more comma-separated table names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--include-triggers=trigger_list`

  Dump the triggers in *`trigger_list`*, which is a list of one or more comma-separated trigger names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--include-users=user_list`

  Dump the user accounts in *`user_list`*, which is a list of one or more comma-separated user names. Multiple instances of this option are additive. For more information, see mysqlpump Object Selection.

- `--insert-ignore`

  Write `INSERT IGNORE` statements rather than `INSERT` statements.

- `--log-error-file=file_name`

  Log warnings and errors by appending them to the named file. If this option is not given, `mysqlpump` writes warnings and errors to the standard error output.

- `--login-path=name`

Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--max-allowed-packet=N`

  The maximum size of the buffer for client/server communication. The default is 24MB, the maximum is 1GB.

- `--net-buffer-length=N`

  The initial size of the buffer for client/server communication. When creating multiple-row `INSERT` statements (as with the `--extended-insert` option), `mysqlpump` creates rows up to *`N`* bytes long. If you use this option to increase the value, ensure that the MySQL server `net_buffer_length` system variable has a value at least this large.

- `--no-create-db`

  Suppress any `CREATE DATABASE` statements that might otherwise be included in the output.

- `--no-create-info`, `-t`

  Do not write `CREATE TABLE` statements that create each dumped table.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--parallel-schemas=[N:]db_list`

  Create a queue for processing the databases in *`db_list`*, which is a list of one or more comma-separated database names. If *`N`* is given, the queue uses *`N`* threads. If *`N`* is not given, the `--default-parallelism` option determines the number of queue threads.

  Multiple instances of this option create multiple queues. `mysqlpump` also creates a default queue to use for databases not named in any `--parallel-schemas` option, and for dumping user definitions if command options select them. For more information, see mysqlpump Parallel Processing.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlpump` prompts for one. If given, there must be *no space* between `--`

`password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

To explicitly specify that there is no password and that `mysqlpump` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlpump` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlpump` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqlpump` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--replace`

  Write `REPLACE` statements rather than `INSERT` statements.

- `--result-file=file_name`

  Direct output to the named file. The result file is created and its previous contents overwritten, even if an error occurs while generating the dump.

  This option should be used on Windows to prevent newline `\n` characters from being converted to `\r\n` carriage return/newline sequences.

- `--routines`

  Include stored routines (procedures and functions) for the dumped databases in the output. This option requires the global `SELECT` privilege.

  The output generated by using `--routines` contains `CREATE PROCEDURE` and `CREATE FUNCTION` statements to create the routines.

  This option is enabled by default; use `--skip-routines` to disable it.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--set-charset`

  Write `SET NAMES default_character_set` to the output.

  This option is enabled by default. To disable it and suppress the `SET NAMES` statement, use `--skip-set-charset`.

- `--set-gtid-purged=value`

  This option enables control over global transaction ID (GTID) information written to the dump file, by indicating whether to add a `SET @@GLOBAL.gtid_purged` statement to the output. This option may also cause a statement to be written to the output that disables binary logging while the dump file is being reloaded.

  The following table shows the permitted option values. The default value is `AUTO`.

Value	Meaning
`OFF`	Add no `SET` statement to the output.
`ON`	Add a `SET` statement to the output. An error occurs if GTIDs are not enabled on the server.

Value	Meaning
AUTO	Add a SET statement to the output if GTIDs are enabled on the server.

The --set-gtid-purged option has the following effect on binary logging when the dump file is reloaded:

- --set-gtid-purged=OFF: SET @@SESSION.SQL_LOG_BIN=0; is not added to the output.
- --set-gtid-purged=ON: SET @@SESSION.SQL_LOG_BIN=0; is added to the output.
- --set-gtid-purged=AUTO: SET @@SESSION.SQL_LOG_BIN=0; is added to the output if GTIDs are enabled on the server you are backing up (that is, if AUTO evaluates to ON).

- --single-transaction

  This option sets the transaction isolation mode to REPEATABLE READ and sends a START TRANSACTION SQL statement to the server before dumping data. It is useful only with transactional tables such as InnoDB, because then it dumps the consistent state of the database at the time when START TRANSACTION was issued without blocking any applications.

  When using this option, you should keep in mind that only InnoDB tables are dumped in a consistent state. For example, any MyISAM or MEMORY tables dumped while using this option may still change state.

  While a --single-transaction dump is in process, to ensure a valid dump file (correct table contents and binary log coordinates), no other connection should use the following statements: ALTER TABLE, CREATE TABLE, DROP TABLE, RENAME TABLE, TRUNCATE TABLE. A consistent read is not isolated from those statements, so use of them on a table to be dumped can cause the SELECT that is performed by mysqlpump to retrieve the table contents to obtain incorrect contents or fail.

  --add-locks and --single-transaction are mutually exclusive.

- --skip-definer

  Omit DEFINER and SQL SECURITY clauses from the CREATE statements for views and stored programs. The dump file, when reloaded, creates objects that use the default DEFINER and SQL SECURITY values. See Section 25.6, “Stored Object Access Control”.

- --skip-dump-rows, -d

  Do not dump table rows.

- --skip-generated-invisible-primary-key

  This option is available beginning with MySQL 8.0.30, and causes generated invisible primary keys (GIPKs) to be excluded from the dump. See Section 13.1.20.11, “Generated Invisible Primary Keys”, for more information about GIPKs and GIPK mode.

- --socket=*path*, -S *path*

  For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the named_pipe system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the named_pipe_full_access_group system variable.

- --ssl*

Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--triggers`

  Include triggers for each dumped table in the output.

  This option is enabled by default; use `--skip-triggers` to disable it.

- `--tz-utc`

  This option enables `TIMESTAMP` columns to be dumped and reloaded between servers in different time zones. `mysqlpump` sets its connection time zone to UTC and adds `SET TIME_ZONE='+00:00'` to the dump file. Without this option, `TIMESTAMP` columns are dumped and reloaded in the time zones local to the source and destination servers, which can cause the values to change if the servers are in different time zones. `--tz-utc` also protects against changes due to daylight saving time.

  This option is enabled by default; use `--skip-tz-utc` to disable it.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

  If you are using the `Rewriter` plugin with MySQL 8.0.31 or later, you should grant this user the `SKIP_QUERY_REWRITE` privilege.

- `--users`

  Dump user accounts as logical definitions in the form of `CREATE USER` and `GRANT` statements.

  User definitions are stored in the grant tables in the `mysql` system database. By default, `mysqlpump` does not include the grant tables in `mysql` database dumps. To dump the contents of the grant tables as logical definitions, use the `--users` option and suppress all database dumping:

  ```
  mysqlpump --exclude-databases=% --users
  ```

- `--version`, `-V`

  Display version information and exit.

- `--watch-progress`

  Periodically display a progress indicator that provides information about the completed and total number of tables, rows, and other objects.

  This option is enabled by default; use `--skip-watch-progress` to disable it.

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

mysqlpump Object Selection

`mysqlpump` has a set of inclusion and exclusion options that enable filtering of several object types and provide flexible control over which objects to dump:

- `--include-databases` and `--exclude-databases` apply to databases and all objects within them.
- `--include-tables` and `--exclude-tables` apply to tables. These options also affect triggers associated with tables unless the trigger-specific options are given.
- `--include-triggers` and `--exclude-triggers` apply to triggers.
- `--include-routines` and `--exclude-routines` apply to stored procedures and functions. If a routine option matches a stored procedure name, it also matches a stored function of the same name.
- `--include-events` and `--exclude-events` apply to Event Scheduler events.
- `--include-users` and `--exclude-users` apply to user accounts.

Any inclusion or exclusion option may be given multiple times. The effect is additive. Order of these options does not matter.

The value of each inclusion and exclusion option is a list of comma-separated names of the appropriate object type. For example:

```
--exclude-databases=test,world
--include-tables=customer,invoice
```

Wildcard characters are permitted in the object names:

- `%` matches any sequence of zero or more characters.
- `_` matches any single character.

For example, `--include-tables=t%,__tmp` matches all table names that begin with `t` and all five-character table names that end with `tmp`.

For users, a name specified without a host part is interpreted with an implied host of `%`. For example, `u1` and `u1@%` are equivalent. This is the same equivalence that applies in MySQL generally (see Section 6.2.4, “Specifying Account Names”).

Inclusion and exclusion options interact as follows:

- By default, with no inclusion or exclusion options, `mysqlpump` dumps all databases (with certain exceptions noted in mysqlpump Restrictions).
- If inclusion options are given in the absence of exclusion options, only the objects named as included are dumped.
- If exclusion options are given in the absence of inclusion options, all objects are dumped except those named as excluded.
- If inclusion and exclusion options are given, all objects named as excluded and not named as included are not dumped. All other objects are dumped.

If multiple databases are being dumped, it is possible to name tables, triggers, and routines in a specific database by qualifying the object names with the database name. The following command dumps databases `db1` and `db2`, but excludes tables `db1.t1` and `db2.t2`:

```
mysqlpump --include-databases=db1,db2 --exclude-tables=db1.t1,db2.t2
```

The following options provide alternative ways to specify which databases to dump:

- The `--all-databases` option dumps all databases (with certain exceptions noted in mysqlpump Restrictions). It is equivalent to specifying no object options at all (the default `mysqlpump` action is to dump everything).

  `--include-databases=%` is similar to `--all-databases`, but selects all databases for dumping, even those that are exceptions for `--all-databases`.

- The `--databases` option causes `mysqlpump` to treat all name arguments as names of databases to dump. It is equivalent to an `--include-databases` option that names the same databases.

mysqlpump Parallel Processing

`mysqlpump` can use parallelism to achieve concurrent processing. You can select concurrency between databases (to dump multiple databases simultaneously) and within databases (to dump multiple objects from a given database simultaneously).

By default, `mysqlpump` sets up one queue with two threads. You can create additional queues and control the number of threads assigned to each one, including the default queue:

- `--default-parallelism=N` specifies the default number of threads used for each queue. In the absence of this option, *`N`* is 2.

  The default queue always uses the default number of threads. Additional queues use the default number of threads unless you specify otherwise.

- `--parallel-schemas=[N:]db_list` sets up a processing queue for dumping the databases named in `db_list` and optionally specifies how many threads the queue uses. `db_list` is a list of comma-separated database names. If the option argument begins with `N:`, the queue uses `N` threads. Otherwise, the `--default-parallelism` option determines the number of queue threads.

  Multiple instances of the `--parallel-schemas` option create multiple queues.

  Names in the database list are permitted to contain the same `%` and `_` wildcard characters supported for filtering options (see mysqlpump Object Selection).

`mysqlpump` uses the default queue for processing any databases not named explicitly with a `--parallel-schemas` option, and for dumping user definitions if command options select them.

In general, with multiple queues, `mysqlpump` uses parallelism between the sets of databases processed by the queues, to dump multiple databases simultaneously. For a queue that uses multiple threads, `mysqlpump` uses parallelism within databases, to dump multiple objects from a given database simultaneously. Exceptions can occur; for example, `mysqlpump` may block queues while it obtains from the server lists of objects in databases.

With parallelism enabled, it is possible for output from different databases to be interleaved. For example, `INSERT` statements from multiple tables dumped in parallel can be interleaved; the statements are not written in any particular order. This does not affect reloading because output statements qualify object names with database names or are preceded by `USE` statements as required.

The granularity for parallelism is a single database object. For example, a single table cannot be dumped in parallel using multiple threads.

Examples:

```
mysqlpump --parallel-schemas=db1,db2 --parallel-schemas=db3
```

`mysqlpump` sets up a queue to process `db1` and `db2`, another queue to process `db3`, and a default queue to process all other databases. All queues use two threads.

```
mysqlpump --parallel-schemas=db1,db2 --parallel-schemas=db3
          --default-parallelism=4
```

This is the same as the previous example except that all queues use four threads.

```
mysqlpump --parallel-schemas=5:db1,db2 --parallel-schemas=3:db3
```

The queue for `db1` and `db2` uses five threads, the queue for `db3` uses three threads, and the default queue uses the default of two threads.

As a special case, with `--default-parallelism=0` and no `--parallel-schemas` options, `mysqlpump` runs as a single-threaded process and creates no queues.

mysqlpump Restrictions

`mysqlpump` does not dump the `performance_schema`, `ndbinfo`, or `sys` schema by default. To dump any of these, name them explicitly on the command line. You can also name them with the `--databases` or `--include-databases` option.

`mysqlpump` does not dump the `INFORMATION_SCHEMA` schema.

`mysqlpump` does not dump `InnoDB` `CREATE TABLESPACE` statements.

`mysqlpump` dumps user accounts in logical form using `CREATE USER` and `GRANT` statements (for example, when you use the `--include-users` or `--users` option). For this reason, dumps of the `mysql` system database do not by default include the grant tables that contain user definitions: `user`, `db`, `tables_priv`, `columns_priv`, `procs_priv`, or `proxies_priv`. To dump any of the grant tables, name the `mysql` database followed by the table names:

```
mysqlpump mysql user db ...
```

4.5.7 mysqlshow — Display Database, Table, and Column Information

The `mysqlshow` client can be used to quickly see which databases exist, their tables, or a table's columns or indexes.

`mysqlshow` provides a command-line interface to several SQL `SHOW` statements. See Section 13.7.7, “SHOW Statements”. The same information can be obtained by using those statements directly. For example, you can issue them from the `mysql` client program.

Invoke `mysqlshow` like this:

```
mysqlshow [options] [db_name [tbl_name [col_name]]]
```

- If no database is given, a list of database names is shown.
- If no table is given, all matching tables in the database are shown.
- If no column is given, all matching columns and column types in the table are shown.

The output displays only the names of those databases, tables, or columns for which you have some privileges.

If the last argument contains shell or SQL wildcard characters (`*`, `?`, `%`, or `_`), only those names that are matched by the wildcard are shown. If a database name contains any underscores, those should be escaped with a backslash (some Unix shells require two) to get a list of the proper tables or columns. `*` and `?` characters are converted into SQL `%` and `_` wildcard characters. This might cause some confusion when you try to display the columns for a table with a `_` in the name, because in this case, `mysqlshow` shows you only the table names that match the pattern. This is easily fixed by adding an extra `%` last on the command line as a separate argument.

`mysqlshow` supports the following options, which can be specified on the command line or in the `[mysqlshow]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.17 mysqlshow Options

Option Name	Description	Introduced	Deprecated
--bind-address	Use specified network interface to connect to MySQL Server		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--count	Show the number of rows per table		
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		

Option Name	Description	Introduced	Deprecated
--default-character-set	Specify default character set		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--host	Host on which MySQL server is located		
--keys	Show table indexes		
--login-path	Read login path options from .mylogin.cnf		
--no-defaults	Read no option files		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--pipe	Connect to server using named pipe (Windows only)		
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--server-public-key-path	Path name to file containing RSA public key		

Option Name	Description	Introduced	Deprecated
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--show-table-type	Show a column indicating the table type		
--socket	Unix socket file or Windows named pipe to use		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--status	Display extra information about each table		
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		

Option Name	Description	Introduced	Deprecated
--version	Display version information and exit		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`, `-?`

  Display a help message and exit.

- `--bind-address=ip_address`

  On a computer having multiple network interfaces, use this option to select which interface to use for connecting to the MySQL server.

- `--character-sets-dir=dir_name`

  The directory where character sets are installed. See Section 10.15, “Character Set Configuration”.

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--count`

  Show the number of rows per table. This can be slow for non-`MyISAM` tables.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:o`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`

  Print debugging information and memory and CPU usage statistics when the program exits.

This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-character-set=charset_name`

  Use *`charset_name`* as the default character set. See Section 10.15, “Character Set Configuration”.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysqlshow` normally reads the `[client]` and `[mysqlshow]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqlshow` also reads the `[client_other]` and `[mysqlshow_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--get-server-public-key`

  Request from the server the RSA public key that it uses for key pair-based password exchange. This option applies to clients that connect to the server using an account that authenticates with the `caching_sha2_password` authentication plugin. For connections by such accounts, the server does not send the public key to the client unless requested. The option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not needed, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--host=host_name`, `-h host_name`

Connect to the MySQL server on the given host.

- `--keys`, `-k`

  Show table indexes.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlshow` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlshow` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlshow` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlshow` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqlshow` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--shared-memory-base-name=name`

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--show-table-type`, `-t`

  Show a column indicating the table type, as in `SHOW FULL TABLES`. The type is `BASE TABLE` or `VIEW`.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

  

  Note

  If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--status`, `-i`

  Display extra information about each table.

- `--tls-ciphersuites=ciphersuite_list`

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=protocol_list`

The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--user=user_name`, `-u user_name`

  The user name of the MySQL account to use for connecting to the server.

- `--verbose`, `-v`

  Verbose mode. Print more information about what the program does. This option can be used multiple times to increase the amount of information.

- `--version`, `-V`

  Display version information and exit.

- `--zstd-compression-level=level`

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

4.5.8 mysqlslap — A Load Emulation Client

`mysqlslap` is a diagnostic program designed to emulate client load for a MySQL server and to report the timing of each stage. It works as if multiple clients are accessing the server.

Invoke `mysqlslap` like this:

```
mysqlslap [options]
```

Some options such as `--create` or `--query` enable you to specify a string containing an SQL statement or a file containing statements. If you specify a file, by default it must contain one statement per line. (That is, the implicit statement delimiter is the newline character.) Use the `--delimiter` option to specify a different delimiter, which enables you to specify statements that span multiple lines or place multiple statements on a single line. You cannot include comments in a file; `mysqlslap` does not understand them.

`mysqlslap` runs in three stages:

1. Create schema, table, and optionally any stored programs or data to use for the test. This stage uses a single client connection.

2. Run the load test. This stage can use many client connections.

3. Clean up (disconnect, drop table if specified). This stage uses a single client connection.

Examples:

Supply your own create and query SQL statements, with 50 clients querying and 200 selects for each (enter the command on a single line):

```
mysqlslap --delimiter=";"
  --create="CREATE TABLE a (b int);INSERT INTO a VALUES (23)"
```

```
  --query="SELECT * FROM a" --concurrency=50 --iterations=200
```

Let `mysqlslap` build the query SQL statement with a table of two `INT` columns and three `VARCHAR` columns. Use five clients querying 20 times each. Do not create the table or insert the data (that is, use the previous test's schema and data):

```
mysqlslap --concurrency=5 --iterations=20
  --number-int-cols=2 --number-char-cols=3
  --auto-generate-sql
```

Tell the program to load the create, insert, and query SQL statements from the specified files, where the `create.sql` file has multiple table creation statements delimited by `';'` and multiple insert statements delimited by `';'`. The `--query` file should contain multiple queries delimited by `';'`. Run all the load statements, then run all the queries in the query file with five clients (five times each):

```
mysqlslap --concurrency=5
  --iterations=5 --query=query.sql --create=create.sql
  --delimiter=";"
```

`mysqlslap` supports the following options, which can be specified on the command line or in the `[mysqlslap]` and `[client]` groups of an option file. For information about option files used by MySQL programs, see Section 4.2.2.2, “Using Option Files”.

Table 4.18 mysqlslap Options

Option Name	Description	Introduced	Deprecated
--auto-generate-sql	Generate SQL statements automatically when they are not supplied in files or using command options		
--auto-generate-sql-add-autoincrement	Add AUTO_INCREMENT column to automatically generated tables		
--auto-generate-sql-execute-number	Specify how many queries to generate automatically		
--auto-generate-sql-guid-primary	Add a GUID-based primary key to automatically generated tables		
--auto-generate-sql-load-type	Specify the test load type		
--auto-generate-sql-secondary-indexes	Specify how many secondary indexes to add to automatically generated tables		
--auto-generate-sql-unique-query-number	How many different queries to generate for automatic tests		
--auto-generate-sql-unique-write-number	How many different queries to generate for --auto-generate-sql-write-number		
--auto-generate-sql-write-number	How many row inserts to perform on each thread		

Option Name	Description	Introduced	Deprecated
--commit	How many statements to execute before committing		
--compress	Compress all information sent between client and server		8.0.18
--compression-algorithms	Permitted compression algorithms for connections to server	8.0.18	
--concurrency	Number of clients to simulate when issuing the SELECT statement		
--create	File or string containing the statement to use for creating the table		
--create-schema	Schema in which to run the tests		
--csv	Generate output in comma-separated values format		
--debug	Write debugging log		
--debug-check	Print debugging information when program exits		
--debug-info	Print debugging information, memory, and CPU statistics when program exits		
--default-auth	Authentication plugin to use		
--defaults-extra-file	Read named option file in addition to usual option files		
--defaults-file	Read only named option file		
--defaults-group-suffix	Option group suffix value		
--delimiter	Delimiter to use in SQL statements		
--detach	Detach (close and reopen) each connection after each N statements		
--enable-cleartext-plugin	Enable cleartext authentication plugin		
--engine	Storage engine to use for creating the table		

Option Name	Description	Introduced	Deprecated
--get-server-public-key	Request RSA public key from server		
--help	Display help message and exit		
--host	Host on which MySQL server is located		
--iterations	Number of times to run the tests		
--login-path	Read login path options from .mylogin.cnf		
--no-defaults	Read no option files		
--no-drop	Do not drop any schema created during the test run		
--number-char-cols	Number of VARCHAR columns to use if --auto-generate-sql is specified		
--number-int-cols	Number of INT columns to use if --auto-generate-sql is specified		
--number-of-queries	Limit each client to approximately this number of queries		
--only-print	Do not connect to databases. mysqlslap only prints what it would have done		
--password	Password to use when connecting to server		
--password1	First multifactor authentication password to use when connecting to server	8.0.27	
--password2	Second multifactor authentication password to use when connecting to server	8.0.27	
--password3	Third multifactor authentication password to use when connecting to server	8.0.27	
--pipe	Connect to server using named pipe (Windows only)		
--plugin-dir	Directory where plugins are installed		
--port	TCP/IP port number for connection		

Option Name	Description	Introduced	Deprecated
--post-query	File or string containing the statement to execute after the tests have completed		
--post-system	String to execute using system() after the tests have completed		
--pre-query	File or string containing the statement to execute before running the tests		
--pre-system	String to execute using system() before running the tests		
--print-defaults	Print default options		
--protocol	Transport protocol to use		
--query	File or string containing the SELECT statement to use for retrieving data		
--server-public-key-path	Path name to file containing RSA public key		
--shared-memory-base-name	Shared-memory name for shared-memory connections (Windows only)		
--silent	Silent mode		
--socket	Unix socket file or Windows named pipe to use		
--sql-mode	Set SQL mode for client session		
--ssl-ca	File that contains list of trusted SSL Certificate Authorities		
--ssl-capath	Directory that contains trusted SSL Certificate Authority certificate files		
--ssl-cert	File that contains X.509 certificate		
--ssl-cipher	Permissible ciphers for connection encryption		
--ssl-crl	File that contains certificate revocation lists		
--ssl-crlpath	Directory that contains certificate revocation-list files		

Option Name	Description	Introduced	Deprecated
--ssl-fips-mode	Whether to enable FIPS mode on client side		
--ssl-key	File that contains X.509 key		
--ssl-mode	Desired security state of connection to server		
--ssl-session-data	File that contains SSL session data	8.0.29	
--ssl-session-data-continue-on-failed-reuse	Whether to establish connections if session reuse fails	8.0.29	
--tls-ciphersuites	Permissible TLSv1.3 ciphersuites for encrypted connections	8.0.16	
--tls-version	Permissible TLS protocols for encrypted connections		
--user	MySQL user name to use when connecting to server		
--verbose	Verbose mode		
--version	Display version information and exit		
--zstd-compression-level	Compression level for connections to server that use zstd compression	8.0.18	

- `--help`, `-?`

  Display a help message and exit.

- `--auto-generate-sql`, `-a`

  Generate SQL statements automatically when they are not supplied in files or using command options.

- `--auto-generate-sql-add-autoincrement`

  Add an `AUTO_INCREMENT` column to automatically generated tables.

- `--auto-generate-sql-execute-number=`*`N`*

  Specify how many queries to generate automatically.

- `--auto-generate-sql-guid-primary`

  Add a GUID-based primary key to automatically generated tables.

- `--auto-generate-sql-load-type=`*`type`*

  Specify the test load type. The permissible values are `read` (scan tables), `write` (insert into tables), `key` (read primary keys), `update` (update primary keys), or `mixed` (half inserts, half scanning selects). The default is `mixed`.

- `--auto-generate-sql-secondary-indexes=N`

  Specify how many secondary indexes to add to automatically generated tables. By default, none are added.

- `--auto-generate-sql-unique-query-number=N`

  How many different queries to generate for automatic tests. For example, if you run a `key` test that performs 1000 selects, you can use this option with a value of 1000 to run 1000 unique queries, or with a value of 50 to perform 50 different selects. The default is 10.

- `--auto-generate-sql-unique-write-number=N`

  How many different queries to generate for `--auto-generate-sql-write-number`. The default is 10.

- `--auto-generate-sql-write-number=N`

  How many row inserts to perform. The default is 100.

- `--commit=N`

  How many statements to execute before committing. The default is 0 (no commits are done).

- `--compress`, `-C`

  Compress all information sent between the client and the server if possible. See Section 4.2.8, “Connection Compression Control”.

  As of MySQL 8.0.18, this option is deprecated. Expect it to be removed in a future version of MySQL. See Configuring Legacy Connection Compression.

- `--compression-algorithms=value`

  The permitted compression algorithms for connections to the server. The available algorithms are the same as for the `protocol_compression_algorithms` system variable. The default value is `uncompressed`.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

- `--concurrency=N`, `-c N`

  The number of parallel clients to simulate.

- `--create=value`

  The file or string containing the statement to use for creating the table.

- `--create-schema=value`

  The schema in which to run the tests.

  

  Note

  If the `--auto-generate-sql` option is also given, `mysqlslap` drops the schema at the end of the test run. To avoid this, use the `--no-drop` option as well.

- `--csv[=file_name]`

  Generate output in comma-separated values format. The output goes to the named file, or to the standard output if no file is given.

- `--debug[=debug_options]`, `-# [debug_options]`

  Write a debugging log. A typical *`debug_options`* string is `d:t:o,file_name`. The default is `d:t:o,/tmp/mysqlslap.trace`.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-check`

  Print some debugging information when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--debug-info`, `-T`

  Print debugging information and memory and CPU usage statistics when the program exits.

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--default-auth=plugin`

  A hint about which client-side authentication plugin to use. See Section 6.2.17, “Pluggable Authentication”.

- `--defaults-extra-file=file_name`

  Read this option file after the global option file but (on Unix) before the user option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-file=file_name`

  Use only the given option file. If the file does not exist or is otherwise inaccessible, an error occurs. If *`file_name`* is not an absolute path name, it is interpreted relative to the current directory.

  Exception: Even with `--defaults-file`, client programs read `.mylogin.cnf`.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--defaults-group-suffix=str`

  Read not only the usual option groups, but also groups with the usual names and a suffix of *`str`*. For example, `mysqlslap` normally reads the `[client]` and `[mysqlslap]` groups. If this option is given as `--defaults-group-suffix=_other`, `mysqlslap` also reads the `[client_other]` and `[mysqlslap_other]` groups.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--delimiter=str`, `-F str`

  The delimiter to use in SQL statements supplied in files or using command options.

- `--detach=N`

  Detach (close and reopen) each connection after each *`N`* statements. The default is 0 (connections are not detached).

- `--enable-cleartext-plugin`

  Enable the `mysql_clear_password` cleartext authentication plugin. (See Section 6.4.1.4, “Client-Side Cleartext Pluggable Authentication”.)

- `--engine=engine_name`, `-e engine_name`

  The storage engine to use for creating tables.

- `--get-server-public-key`

  Request from the server the RSA public key that it uses for key pair-based password exchange. This option applies to clients that connect to the server using an account that authenticates with the `caching_sha2_password` authentication plugin. For connections by such accounts, the server does not send the public key to the client unless requested. The option is ignored for accounts that do not authenticate with that plugin. It is also ignored if RSA-based password exchange is not needed, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For information about the `caching_sha2_password` plugin, see Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--host=host_name`, `-h host_name`

  Connect to the MySQL server on the given host.

- `--iterations=N`, `-i N`

  The number of times to run the tests.

- `--login-path=name`

  Read options from the named login path in the `.mylogin.cnf` login path file. A “login path” is an option group containing options that specify which MySQL server to connect to and which account to authenticate as. To create or modify a login path file, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--no-drop`

  Prevent `mysqlslap` from dropping any schema it creates during the test run.

- `--no-defaults`

  Do not read any option files. If program startup fails due to reading unknown options from an option file, `--no-defaults` can be used to prevent them from being read.

  The exception is that the `.mylogin.cnf` file is read in all cases, if it exists. This permits passwords to be specified in a safer way than on the command line even when `--no-defaults` is used. To create `.mylogin.cnf`, use the `mysql_config_editor` utility. See Section 4.6.7, “mysql_config_editor — MySQL Configuration Utility”.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--number-char-cols=N`, `-x N`

  The number of `VARCHAR` columns to use if `--auto-generate-sql` is specified.

- `--number-int-cols=N`, `-y N`

  The number of `INT` columns to use if `--auto-generate-sql` is specified.

- `--number-of-queries=N`

  Limit each client to approximately this many queries. Query counting takes into account the statement delimiter. For example, if you invoke `mysqlslap` as follows, the `;` delimiter is recognized so that each instance of the query string counts as two queries. As a result, 5 rows (not 10) are inserted.

  ```
  mysqlslap --delimiter=";" --number-of-queries=10
            --query="use test;insert into t values(null)"
  ```

- `--only-print`

  Do not connect to databases. `mysqlslap` only prints what it would have done.

- `--password[=password]`, `-p[password]`

  The password of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlslap` prompts for one. If given, there must be *no space* between `--password=` or `-p` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlslap` should not prompt for one, use the `--skip-password` option.

- `--password1[=pass_val]`

  The password for multifactor authentication factor 1 of the MySQL account used for connecting to the server. The password value is optional. If not given, `mysqlslap` prompts for one. If given, there must be *no space* between `--password1=` and the password following it. If no password option is specified, the default is to send no password.

  Specifying a password on the command line should be considered insecure. To avoid giving the password on the command line, use an option file. See Section 6.1.2.1, “End-User Guidelines for Password Security”.

  To explicitly specify that there is no password and that `mysqlslap` should not prompt for one, use the `--skip-password1` option.

  `--password1` and `--password` are synonymous, as are `--skip-password1` and `--skip-password`.

- `--password2[=pass_val]`

  The password for multifactor authentication factor 2 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--password3[=pass_val]`

  The password for multifactor authentication factor 3 of the MySQL account used for connecting to the server. The semantics of this option are similar to the semantics for `--password1`; see the description of that option for details.

- `--pipe`, `-W`

  On Windows, connect to the server using a named pipe. This option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--plugin-dir=dir_name`

  The directory in which to look for plugins. Specify this option if the `--default-auth` option is used to specify an authentication plugin but `mysqlslap` does not find it. See Section 6.2.17, “Pluggable Authentication”.

- `--port=port_num`, `-P port_num`

  For TCP/IP connections, the port number to use.

- `--post-query=value`

  The file or string containing the statement to execute after the tests have completed. This execution is not counted for timing purposes.

- `--post-system=str`

  The string to execute using `system()` after the tests have completed. This execution is not counted for timing purposes.

- `--pre-query=value`

  The file or string containing the statement to execute before running the tests. This execution is not counted for timing purposes.

- `--pre-system=str`

  The string to execute using `system()` before running the tests. This execution is not counted for timing purposes.

- `--print-defaults`

  Print the program name and all options that it gets from option files.

  For additional information about this and other option-file options, see Section 4.2.2.3, “Command-Line Options that Affect Option-File Handling”.

- `--protocol={TCP|SOCKET|PIPE|MEMORY}`

  The transport protocol to use for connecting to the server. It is useful when the other connection parameters normally result in use of a protocol other than the one you want. For details on the permissible values, see Section 4.2.7, “Connection Transport Protocols”.

- `--query=value`, `-q value`

  The file or string containing the `SELECT` statement to use for retrieving data.

- `--server-public-key-path=file_name`

  The path name to a file in PEM format containing a client-side copy of the public key required by the server for RSA key pair-based password exchange. This option applies to clients that authenticate with the `sha256_password` or `caching_sha2_password` authentication plugin. This option is ignored for accounts that do not authenticate with one of those plugins. It is also ignored if RSA-based password exchange is not used, as is the case when the client connects to the server using a secure connection.

  If `--server-public-key-path=file_name` is given and specifies a valid public key file, it takes precedence over `--get-server-public-key`.

  For `sha256_password`, this option applies only if MySQL was built using OpenSSL.

  For information about the `sha256_password` and `caching_sha2_password` plugins, see Section 6.4.1.3, “SHA-256 Pluggable Authentication”, and Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”.

- `--shared-memory-base-name=name`

  On Windows, the shared-memory name to use for connections made using shared memory to a local server. The default value is `MYSQL`. The shared-memory name is case-sensitive.

  This option applies only if the server was started with the `shared_memory` system variable enabled to support shared-memory connections.

- `--silent`, `-s`

  Silent mode. No output.

- `--socket=path`, `-S path`

  For connections to `localhost`, the Unix socket file to use, or, on Windows, the name of the named pipe to use.

  On Windows, this option applies only if the server was started with the `named_pipe` system variable enabled to support named-pipe connections. In addition, the user making the connection must be a member of the Windows group specified by the `named_pipe_full_access_group` system variable.

- `--sql-mode=mode`

  Set the SQL mode for the client session.

- `--ssl*`

  Options that begin with `--ssl` specify whether to connect to the server using encryption and indicate where to find SSL keys and certificates. See Command Options for Encrypted Connections.

- `--ssl-fips-mode={OFF|ON|STRICT}`

  Controls whether to enable FIPS mode on the client side. The `--ssl-fips-mode` option differs from other `--ssl-xxx` options in that it is not used to establish encrypted connections, but rather to affect which cryptographic operations to permit. See Section 6.8, “FIPS Support”.

  These `--ssl-fips-mode` values are permitted:

  - `OFF`: Disable FIPS mode.
  - `ON`: Enable FIPS mode.
  - `STRICT`: Enable “strict” FIPS mode.

Note

If the OpenSSL FIPS Object Module is not available, the only permitted value for `--ssl-fips-mode` is `OFF`. In this case, setting `--ssl-fips-mode` to `ON` or `STRICT` causes the client to produce a warning at startup and to operate in non-FIPS mode.

- `--tls-ciphersuites=`*`ciphersuite_list`*

  The permissible ciphersuites for encrypted connections that use TLSv1.3. The value is a list of one or more colon-separated ciphersuite names. The ciphersuites that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

  This option was added in MySQL 8.0.16.

- `--tls-version=`*`protocol_list`*

  The permissible TLS protocols for encrypted connections. The value is a list of one or more comma-separated protocol names. The protocols that can be named for this option depend on the SSL library used to compile MySQL. For details, see Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”.

- `--user=`*`user_name`*, `-u` *`user_name`*

  The user name of the MySQL account to use for connecting to the server.

- `--verbose`, `-v`

  Verbose mode. Print more information about what the program does. This option can be used multiple times to increase the amount of information.

- `--version`, `-V`

  Display version information and exit.

- `--zstd-compression-level=`*`level`*

  The compression level to use for connections to the server that use the `zstd` compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default `zstd` compression level is 3. The compression level setting has no effect on connections that do not use `zstd` compression.

  For more information, see Section 4.2.8, “Connection Compression Control”.

  This option was added in MySQL 8.0.18.

4.6 Administrative and Utility Programs

This section describes administrative programs and programs that perform miscellaneous utility operations.

4.6.1 ibd2sdi — InnoDB Tablespace SDI Extraction Utility

`ibd2sdi` is a utility for extracting serialized dictionary information (SDI) from `InnoDB` tablespace files. SDI data is present in all persistent `InnoDB` tablespace files.

`ibd2sdi` can be run on file-per-table tablespace files (`*.ibd` files), general tablespace files (`*.ibd` files), system tablespace files (`ibdata*` files), and the data dictionary tablespace (`mysql.ibd`). It is not supported for use with temporary tablespaces or undo tablespaces.

`ibd2sdi` can be used at runtime or while the server is offline. During DDL operations, `ROLLBACK` operations, and undo log purge operations related to SDI, there may be a short interval of time when `ibd2sdi` fails to read SDI data stored in the tablespace.

`ibd2sdi` performs an uncommitted read of SDI from the specified tablespace. Redo logs and undo logs are not accessed.

Invoke the `ibd2sdi` utility like this:

```
ibd2sdi [options] file_name1 [file_name2 file_name3 ...]
```

`ibd2sdi` supports multi-file tablespaces like the `InnoDB` system tablespace, but it cannot be run on more than one tablespace at a time. For multi-file tablespaces, specify each file:

```
ibd2sdi ibdata1 ibdata2
```

The files of a multi-file tablespace must be specified in order of the ascending page number. If two successive files have the same space ID, the later file must start with the last page number of the previous file + 1.

`ibd2sdi` outputs SDI (containing id, type, and data fields) in `JSON` format.

ibd2sdi Options

`ibd2sdi` supports the following options:

- `--help`, `-h`

  Display a help message and exit. For example:

```
Usage: ./ibd2sdi [-v] [-c <strict-check>] [-d <dump file name>] [-n] filename1 [filenames]
See http://dev.mysql.com/doc/refman/8.0/en/ibd2sdi.html for usage hints.
  -h, --help          Display this help and exit.
  -v, --version       Display version information and exit.
  -#, --debug[=name]  Output debug log. See
                      http://dev.mysql.com/doc/refman/8.0/en/dbug-package.html
  -d, --dump-file=name
                      Dump the tablespace SDI into the file passed by user.
                      Without the filename, it will default to stdout
  -s, --skip-data     Skip retrieving data from SDI records. Retrieve only id
                      and type.
  -i, --id=#          Retrieve the SDI record matching the id passed by user.
  -t, --type=#        Retrieve the SDI records matching the type passed by
                      user.
  -c, --strict-check=name
                      Specify the strict checksum algorithm by the user.
                      Allowed values are innodb, crc32, none.
  -n, --no-check      Ignore the checksum verification.
  -p, --pretty        Pretty format the SDI output.If false, SDI would be not
                      human readable but it will be of less size
                      (Defaults to on; use --skip-pretty to disable.)

Variables (--variable-name=value)
and boolean options {FALSE|TRUE}  Value (after reading options)
--------------------------------- ----------------------------------------
debug                             (No default value)
dump-file                         (No default value)
skip-data                         FALSE
id                                0
type                              0
strict-check                      crc32
no-check                          FALSE
pretty                            TRUE
```

- `--version`, `-v`

  Display version information and exit. For example:

```
ibd2sdi  Ver 8.0.3-dmr for Linux on x86_64 (Source distribution)
```

- `--debug[=debug_options]`, `-# [debug_options]`

  Prints a debug log. For debug options, refer to Section 5.9.4, “The DBUG Package”.

  ```
  ibd2sdi --debug=d:t /tmp/ibd2sdi.trace
  ```

  This option is available only if MySQL was built using `WITH_DEBUG`. MySQL release binaries provided by Oracle are *not* built using this option.

- `--dump-file=`, `-d`

  Dumps serialized dictionary information (SDI) into the specified dump file. If a dump file is not specified, the tablespace SDI is dumped to `stdout`.

  ```
  ibd2sdi --dump-file=file_name ../data/test/t1.ibd
  ```

- `--skip-data`, `-s`

  Skips retrieval of `data` field values from the serialized dictionary information (SDI) and only retrieves the `id` and `type` field values, which are primary keys for SDI records.

  ```
  $> ibd2sdi --skip-data ../data/test/t1.ibd
  ["ibd2sdi"
  ,
  {
   "type": 1,
   "id": 330
  }
  ,
  {
   "type": 2,
   "id": 7
  }
  ]
  ```

- `--id=#`, `-i #`

  Retrieves serialized dictionary information (SDI) matching the specified table or tablespace object id. An object id is unique to the object type. Table and tablespace object IDs are also found in the `id` column of the `mysql.tables` and `mysql.tablespace` data dictionary tables. For information about data dictionary tables, see Section 14.1, “Data Dictionary Schema”.

  ```
  $> ibd2sdi --id=7 ../data/test/t1.ibd
  ["ibd2sdi"
  ,
  {
   "type": 2,
   "id": 7,
   "object":
    {
      "mysqld_version_id": 80003,
      "dd_version": 80003,
      "sdi_version": 1,
      "dd_object_type": "Tablespace",
      "dd_object": {
          "name": "test/t1",
          "comment": "",
          "options": "",
          "se_private_data": "flags=16417;id=2;server_version=80003;space_version=1;",
          "engine": "InnoDB",
          "files": [
              {
                  "ordinal_position": 1,
                  "filename": "./test/t1.ibd",
                  "se_private_data": "id=2;"
              }
          ]
      }
  }
  ```

```
}
]
```

- `--type=#`, `-t #`

  Retrieves serialized dictionary information (SDI) matching the specified object type. SDI is provided for table (type=1) and tablespace (type=2) objects.

```
$> ibd2sdi --type=2 ../data/test/t1.ibd
["ibd2sdi"
,
{
 "type": 2,
 "id": 7,
 "object":
  {
    "mysqld_version_id": 80003,
    "dd_version": 80003,
    "sdi_version": 1,
    "dd_object_type": "Tablespace",
    "dd_object": {
        "name": "test/t1",
        "comment": "",
        "options": "",
        "se_private_data": "flags=16417;id=2;server_version=80003;space_version=1;",
        "engine": "InnoDB",
        "files": [
            {
                "ordinal_position": 1,
                "filename": "./test/t1.ibd",
                "se_private_data": "id=2;"
            }
        ]
    }
}
}
]
```

- `--strict-check`, `-c`

  Specifies a strict checksum algorithm for validating the checksum of pages that are read. Options include `innodb`, `crc32`, and `none`.

  In this example, the strict version of the `innodb` checksum algorithm is specified:

```
ibd2sdi --strict-check=innodb ../data/test/t1.ibd
```

  In this example, the strict version of `crc32` checksum algorithm is specified:

```
ibd2sdi -c crc32 ../data/test/t1.ibd
```

  If you do not specify the `--strict-check` option, validation is performed against non-strict `innodb`, `crc32` and `none` checksums.

- `--no-check`, `-n`

  Skips checksum validation for pages that are read.

```
ibd2sdi --no-check ../data/test/t1.ibd
```

- `--pretty`, `-p`

  Outputs SDI data in JSON pretty print format. Enabled by default. If disabled, SDI is not human readable but is smaller in size. Use `--skip-pretty` to disable.

```
ibd2sdi --skip-pretty ../data/test/t1.ibd
```

4.6.2 innochecksum — Offline InnoDB File Checksum Utility

`innochecksum` prints checksums for `InnoDB` files. This tool reads an `InnoDB` tablespace file, calculates the checksum for each page, compares the calculated checksum to the stored checksum, and reports mismatches, which indicate damaged pages. It was originally developed to speed up verifying the integrity of tablespace files after power outages but can also be used after file copies. Because checksum mismatches cause `InnoDB` to deliberately shut down a running server, it may be preferable to use this tool rather than waiting for an in-production server to encounter the damaged pages.

`innochecksum` cannot be used on tablespace files that the server already has open. For such files, you should use `CHECK TABLE` to check tables within the tablespace. Attempting to run `innochecksum` on a tablespace that the server already has open results in an `Unable to lock file` error.

If checksum mismatches are found, restore the tablespace from backup or start the server and attempt to use `mysqldump` to make a backup of the tables within the tablespace.

Invoke `innochecksum` like this:

```
innochecksum [options] file_name
```

innochecksum Options

`innochecksum` supports the following options. For options that refer to page numbers, the numbers are zero-based.

- `--help`, `-?`

  Displays command line help. Example usage:

  ```
  innochecksum --help
  ```

- `--info`, `-I`

  Synonym for `--help`. Displays command line help. Example usage:

  ```
  innochecksum --info
  ```

- `--version`, `-V`

  Displays version information. Example usage:

  ```
  innochecksum --version
  ```

- `--verbose`, `-v`

  Verbose mode; prints a progress indicator to the log file every five seconds. In order for the progress indicator to be printed, the log file must be specified using the `--log option`. To turn on `verbose` mode, run:

  ```
  innochecksum --verbose
  ```

  To turn off verbose mode, run:

  ```
  innochecksum --verbose=FALSE
  ```

  The `--verbose` option and `--log` option can be specified at the same time. For example:

  ```
  innochecksum --verbose --log=/var/lib/mysql/test/logtest.txt
  ```

  To locate the progress indicator information in the log file, you can perform the following search:

  ```
  cat ./logtest.txt | grep -i "okay"
  ```

  The progress indicator information in the log file appears similar to the following: