**Реплікація даних**

[Реплікація даних 1](#_Toc57749738)

[Оператори реплікації Replication Statements 1](#_Toc57749739)

[SQL Statements for Controlling Source Servers 2](#_Toc57749740)

[PURGE BINARY LOGS Statement 2](#_Toc57749741)

[RESET MASTER Statement 3](#_Toc57749742)

[SET sql\_log\_bin Statement 4](#_Toc57749743)

[SQL Statements for Controlling Replica Servers 4](#_Toc57749744)

[CHANGE MASTER TO Statement 4](#_Toc57749745)

[CHANGE REPLICATION FILTER Statement 12](#_Toc57749746)

[MASTER\_POS\_WAIT() Statement 14](#_Toc57749747)

[RESET REPLICA | SLAVE Statement 14](#_Toc57749748)

[RESET SLAVE | REPLICA Statement 16](#_Toc57749749)

[START REPLICA | SLAVE Statement 16](#_Toc57749750)

[START SLAVE | REPLICA Statement 19](#_Toc57749751)

[STOP REPLICA | SLAVE Statement 20](#_Toc57749752)

[STOP SLAVE | REPLICA Statement 21](#_Toc57749753)

[Functions which Configure the Source List 21](#_Toc57749754)

[SQL Statements for Controlling Group Replication 23](#_Toc57749755)

[START GROUP\_REPLICATION Statement 23](#_Toc57749756)

[STOP GROUP\_REPLICATION Statement 23](#_Toc57749757)

[Function which Configures Group Replication Primary 24](#_Toc57749758)

[Functions which Configure the Group Replication Mode 24](#_Toc57749759)

[Functions to Inspect and Configure the Maximum Consensus Instances of a Group 25](#_Toc57749760)

[Version 26](#_Toc57749761)

# Реплікація даних

## Оператори реплікації Replication Statements

**MySQL replication** is a process that allows you to easily maintain multiple copies of a **MySQL** data by having them copied automatically from a master to a slave database. ... For the process to work you will need two IP addresses: one of the master server and and one of the slave.

Replication enables you to maintain identical data on multiple servers. This has several benefits, such as enabling client query load to be distributed over servers, availability of data even if a given server is taken offline or fails, and the ability to make backups with no impact on the source by using a replica.

Replication can be controlled through the SQL interface using the statements described in this section. Statements are split into a group which controls source servers, a group which controls replica servers, and a group which can be applied to any replication servers.

### SQL Statements for Controlling Source Servers

This section discusses statements for managing replication source servers. [Section 13.4.2, “SQL Statements for Controlling Replica Servers”](https://dev.mysql.com/doc/refman/8.0/en/replication-statements-replica.html), discusses statements for managing replica servers.

In addition to the statements described here, the following [SHOW](https://dev.mysql.com/doc/refman/8.0/en/show.html) statements are used with source servers in replication..

* [SHOW BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/show-binary-logs.html)
* [SHOW BINLOG EVENTS](https://dev.mysql.com/doc/refman/8.0/en/show-binlog-events.html)
* [SHOW MASTER STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-master-status.html)
* [SHOW REPLICAS | SHOW SLAVE HOSTS](https://dev.mysql.com/doc/refman/8.0/en/show-replicas.html)

#### PURGE BINARY LOGS Statement

PURGE { BINARY | MASTER } LOGS {

 TO 'log\_name'

 | BEFORE datetime\_expr

}

The binary log is a set of files that contain information about data modifications made by the MySQL server. The log consists of a set of binary log files, plus an index file.

The [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) statement deletes all the binary log files listed in the log index file prior to the specified log file name or date. BINARY and MASTER are synonyms. Deleted log files also are removed from the list recorded in the index file, so that the given log file becomes the first in the list.

[PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) requires the [BINLOG\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_binlog-admin) privilege. This statement has no effect if the server was not started with the [--log-bin](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#option_mysqld_log-bin) option to enable binary logging.

Examples:

PURGE BINARY LOGS TO 'mysql-bin.010';

PURGE BINARY LOGS BEFORE '2019-04-02 22:46:26';

The BEFORE variant's ***datetime\_expr*** argument should evaluate to a [DATETIME](https://dev.mysql.com/doc/refman/8.0/en/datetime.html) value (a value in '***YYYY-MM-DD hh:mm:ss***' format).

This statement is safe to run while replicas are replicating. You need not stop them. If you have an active replica that currently is reading one of the log files you are trying to delete, this statement does not delete the log file that is in use or any log files later than that one, but it deletes any earlier log files. A warning message is issued in this situation. However, if a replica is not connected and you happen to purge one of the log files it has yet to read, the replica cannot replicate after it reconnects.

To safely purge binary log files, follow this procedure:

1. On each replica, use [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html) to check which log file it is reading.
2. Obtain a listing of the binary log files on the source with [SHOW BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/show-binary-logs.html).
3. Determine the earliest log file among all the replicas. This is the target file. If all the replicas are up to date, this is the last log file on the list.
4. Make a backup of all the log files you are about to delete. (This step is optional, but always advisable.)
5. Purge all log files up to but not including the target file.

PURGE BINARY LOGS TO and PURGE BINARY LOGS BEFORE both fail with an error when binary log files listed in the .index file had been removed from the system by some other means (such as using **rm** on Linux). (Bug #18199, Bug #18453) To handle such errors, edit the .index file (which is a simple text file) manually to ensure that it lists only the binary log files that are actually present, then run again the [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) statement that failed.

Binary log files are automatically removed after the server's binary log expiration period. Removal of the files can take place at startup and when the binary log is flushed. The default binary log expiration period is 30 days. You can specify an alternative expiration period using the [binlog\_expire\_logs\_seconds](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html%22%20%5Cl%20%22sysvar_binlog_expire_logs_seconds) system variable. If you are using replication, you should specify an expiration period that is no lower than the maximum amount of time your replicas might lag behind the source.

#### RESET MASTER Statement

RESET MASTER [TO binary\_log\_file\_index\_number]

**Warning**

Use this statement with caution to ensure you do not lose any wanted binary log file data and GTID execution history.

[RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) requires the [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload) privilege.

For a server where binary logging is enabled ([log\_bin](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html%22%20%5Cl%20%22sysvar_log_bin) is ON), RESET MASTER deletes all existing binary log files and resets the binary log index file, resetting the server to its state before binary logging was started. A new empty binary log file is created so that binary logging can be restarted.

For a server where GTIDs are in use ([gtid\_mode](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_mode) is ON), issuing RESET MASTER resets the GTID execution history. The value of the [gtid\_purged](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_purged) system variable is set to an empty string (''), the global value (but not the session value) of the [gtid\_executed](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_executed) system variable is set to an empty string, and the mysql.gtid\_executed table is cleared (see [mysql.gtid\_executed Table](https://dev.mysql.com/doc/refman/8.0/en/replication-gtids-concepts.html%22%20%5Cl%20%22replication-gtids-gtid-executed-table%22%20%5Co%20%22mysql.gtid_executed%20Table)). If the GTID-enabled server has binary logging enabled, [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) also resets the binary log as described above. Note that [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) is the method to reset the GTID execution history even if the GTID-enabled server is a replica where binary logging is disabled; [RESET REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) has no effect on the GTID execution history. For more information on resetting the GTID execution history, see [Resetting the GTID Execution History](https://dev.mysql.com/doc/refman/8.0/en/replication-gtids-lifecycle.html#replication-gtids-execution-history).

Issuing RESET MASTER without the optional TO clause deletes all binary log files listed in the index file, resets the binary log index file to be empty, and creates a new binary log file starting at 1. Use the optional TO clause to start the binary log file index from a number other than 1 after the reset.

Using RESET MASTER with the TO clause to specify a binary log file index number to start from simplifies failover by providing a single statement alternative to the [FLUSH BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/flush.html#flush-binary-logs) and [PURGE BINARY LOGS TO](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) statements. Check that you are using a reasonable value for the index number. If you enter an incorrect value, you can correct this by issuing another [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) statement with or without the TO clause. If you do not correct a value that is out of range, the server cannot be restarted.

The following example demonstrates TO clause usage:

RESET MASTER TO 1234;

SHOW BINARY LOGS;

+-------------------+-----------+-----------+

| Log\_name | File\_size | Encrypted |

+-------------------+-----------+-----------+

| source-bin.001234 | 154 | No |

+-------------------+-----------+-----------+

**Important**

The effects of [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) without the TO clause differ from those of [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) in 2 key ways:

1. [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) removes all binary log files that are listed in the index file, leaving only a single, empty binary log file with a numeric suffix of .000001, whereas the numbering is not reset by [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html).
2. [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) is not intended to be used while any replicas are running. The behavior of [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) when used while replicas are running is undefined (and thus unsupported), whereas [PURGE BINARY LOGS](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html) may be safely used while replicas are running.

See also [Section 13.4.1.1, “PURGE BINARY LOGS Statement”](https://dev.mysql.com/doc/refman/8.0/en/purge-binary-logs.html).

[RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) without the TO clause can prove useful when you first set up a source and replica, so that you can verify the setup as follows:

1. Start the source and replica, and start replication (see [Section 17.1.2, “Setting Up Binary Log File Position Based Replication”](https://dev.mysql.com/doc/refman/8.0/en/replication-howto.html)).
2. Execute a few test queries on the source.
3. Check that the queries were replicated to the replica.
4. When replication is running correctly, issue [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) followed by [RESET REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) on the replica, then verify that no unwanted data from the test queries exists on the replica.
5. Issue [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html) on the source to clean up the test queries.

After verifying the setup, resetting the source and replica and ensuring that no unwanted data or binary log files generated by testing remain on the source or replica, you can start the replica and begin replicating.

#### SET sql\_log\_bin Statement

SET sql\_log\_bin = {OFF|ON}

The [sql\_log\_bin](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html%22%20%5Cl%20%22sysvar_sql_log_bin) variable controls whether logging to the binary log is enabled for the current session (assuming that the binary log itself is enabled). The default value is ON. To disable or enable binary logging for the current session, set the session [sql\_log\_bin](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html%22%20%5Cl%20%22sysvar_sql_log_bin) variable to OFF or ON.

Set this variable to OFF for a session to temporarily disable binary logging while making changes to the source that you do not want replicated to the replica.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](https://dev.mysql.com/doc/refman/8.0/en/system-variable-privileges.html).

It is not possible to set the session value of [sql\_log\_bin](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html%22%20%5Cl%20%22sysvar_sql_log_bin) within a transaction or subquery.

Setting this variable to *OFF* prevents new GTIDs from being assigned to transactions in the binary log. If you are using GTIDs for replication, this means that even when binary logging is later enabled again, the GTIDs written into the log from this point do not account for any transactions that occurred in the meantime, so in effect those transactions are lost.

[**mysqldump**](https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html) adds a SET @@SESSION.sql\_log\_bin=0 statement to a dump file from a server where GTIDs are in use, which disables binary logging while the dump file is being reloaded. The 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.

### SQL Statements for Controlling Replica Servers

#### CHANGE MASTER TO Statement

CHANGE MASTER TO option [, option] ... [ channel\_option ]

option: {

 MASTER\_BIND = 'interface\_name'

 | MASTER\_HOST = 'host\_name'

 | MASTER\_USER = 'user\_name'

 | MASTER\_PASSWORD = 'password'

 | MASTER\_PORT = port\_num

 | PRIVILEGE\_CHECKS\_USER = {'account' | NULL}

 | REQUIRE\_ROW\_FORMAT = {0|1}

 | REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK = {STREAM | ON | OFF}

 | MASTER\_LOG\_FILE = 'source\_log\_name'

 | MASTER\_LOG\_POS = source\_log\_pos

 | MASTER\_AUTO\_POSITION = {0|1}

 | RELAY\_LOG\_FILE = 'relay\_log\_name'

 | RELAY\_LOG\_POS = relay\_log\_pos

 | MASTER\_HEARTBEAT\_PERIOD = interval

 | MASTER\_CONNECT\_RETRY = interval

 | MASTER\_RETRY\_COUNT = count

 | SOURCE\_CONNECTION\_AUTO\_FAILOVER = {0|1}

 | MASTER\_DELAY = interval

 | MASTER\_COMPRESSION\_ALGORITHMS = 'value'

 | MASTER\_ZSTD\_COMPRESSION\_LEVEL = level

 | MASTER\_SSL = {0|1}

 | MASTER\_SSL\_CA = 'ca\_file\_name'

 | MASTER\_SSL\_CAPATH = 'ca\_directory\_name'

 | MASTER\_SSL\_CERT = 'cert\_file\_name'

 | MASTER\_SSL\_CRL = 'crl\_file\_name'

 | MASTER\_SSL\_CRLPATH = 'crl\_directory\_name'

 | MASTER\_SSL\_KEY = 'key\_file\_name'

 | MASTER\_SSL\_CIPHER = 'cipher\_list'

 | MASTER\_SSL\_VERIFY\_SERVER\_CERT = {0|1}

 | MASTER\_TLS\_VERSION = 'protocol\_list'

 | MASTER\_TLS\_CIPHERSUITES = 'ciphersuite\_list'

 | MASTER\_PUBLIC\_KEY\_PATH = 'key\_file\_name'

 | GET\_MASTER\_PUBLIC\_KEY = {0|1}

 | NETWORK\_NAMESPACE = 'namespace'

 | IGNORE\_SERVER\_IDS = (server\_id\_list)

}

channel\_option:

 FOR CHANNEL channel

server\_id\_list:

 [server\_id [, server\_id] ... ]

[CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) changes the parameters that the replica server uses for connecting to the source, for reading the source's binary log, and reading the replica's relay log. It also updates the contents of the replication metadata repositories (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](https://dev.mysql.com/doc/refman/8.0/en/replica-logs.html)). [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) requires the [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege).

You can issue CHANGE MASTER TO statements on a running replica without first stopping it, depending on the states of the replication SQL thread and replication I/O thread. The rules governing such use are provided later in this section.

When using a multithreaded replica (in other words [slave\_parallel\_workers](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_parallel_workers) is greater than 0), stopping the replica can cause “gaps” in the sequence of transactions that have been executed from the relay log, regardless of whether the replica was stopped intentionally or otherwise. When such gaps exist, issuing [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) fails. The solution in this situation is to issue [START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) which ensures that the gaps are closed.

The optional FOR CHANNEL ***channel*** clause enables you to name which replication channel the statement applies to. Providing a FOR CHANNEL ***channel*** clause applies the CHANGE MASTER TO statement to a specific replication channel, and is used to add a new channel or modify an existing channel. For example, to add a new channel called channel2:

CHANGE MASTER TO MASTER\_HOST=host1, MASTER\_PORT=3002 FOR CHANNEL 'channel2'

If no clause is named and no extra channels exist, the statement applies to the default channel.

When using multiple replication channels, if a CHANGE MASTER TO statement does not name a channel using a FOR CHANNEL ***channel*** clause, an error occurs. See [Section 17.2.2, “Replication Channels”](https://dev.mysql.com/doc/refman/8.0/en/replication-channels.html) for more information.

Options not specified retain their value, except as indicated in the following discussion. Thus, in most cases, there is no need to specify options that do not change.

MASTER\_HOST, MASTER\_USER, MASTER\_PASSWORD, MASTER\_PORT, and NETWORK\_NAMESPACE provide information to the replica about how to connect to its source:

* MASTER\_HOST and MASTER\_PORT are the host name (or IP address) of the source server and its TCP/IP port.

**Note**

Replication cannot use Unix socket files. You must be able to connect to the source MySQL server using TCP/IP.

If you specify the MASTER\_HOST or MASTER\_PORT option, the replica assumes that the source server is different from before (even if the option value is the same as its current value.) In this case, the old values for the source's binary log file name and position are considered no longer applicable, so if you do not specify MASTER\_LOG\_FILE and MASTER\_LOG\_POS in the statement, MASTER\_LOG\_FILE='' and MASTER\_LOG\_POS=4 are silently appended to it.

Setting MASTER\_HOST='' (that is, setting its value explicitly to an empty string) is not the same as not setting MASTER\_HOST at all. Trying to set MASTER\_HOST to an empty string fails with an error.

Values used for MASTER\_HOST and other CHANGE MASTER TO options are checked for linefeed (\n or 0x0A) characters; the presence of such characters in these values causes the statement to fail with [ER\_MASTER\_INFO](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_master_info). (Bug #11758581, Bug #50801)

* MASTER\_USER and MASTER\_PASSWORD are the user name and password of the replication user account to use for connecting to the source. If you specify MASTER\_PASSWORD, MASTER\_USER is also required. The password used for a replication user account in a CHANGE MASTER TO statement is limited to 32 characters in length; trying to use a password of more than 32 characters causes CHANGE MASTER TO to fail.

It is possible to set an empty user name by specifying MASTER\_USER='', but the replication channel cannot be started with an empty user name. In releases before MySQL 8.0.21, only set an empty MASTER\_USER user name if you need to clear previously used credentials from the replication metadata repositories for security purposes. Do not use the channel afterwards, due to a bug in these releases that can substitute a default user name if an empty user name is read from the repositories (for example, during an automatic restart of a Group Replication channel). From MySQL 8.0.21, it is valid to set an empty MASTER\_USER user name and use the channel afterwards if you always provide user credentials using the [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) statement or [START GROUP\_REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/start-group-replication.html) statement that starts the replication channel. This approach means that the replication channel always needs operator intervention to restart, but the user credentials are not recorded in the replication metadata repositories.

The text of a running [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement, including values for MASTER\_USER and MASTER\_PASSWORD, can be seen in the output of a concurrent [SHOW PROCESSLIST](https://dev.mysql.com/doc/refman/8.0/en/show-processlist.html) statement. (The complete text of a [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) statement is also visible to [SHOW PROCESSLIST](https://dev.mysql.com/doc/refman/8.0/en/show-processlist.html).)

* NETWORK\_NAMESPACE specifies the network namespace to use for TCP/IP connections to the source. If this option is omitted, connections from the replica use the default (global) namespace.

For information about network namespaces, see [Section 5.1.14, “Network Namespace Support”](https://dev.mysql.com/doc/refman/8.0/en/network-namespace-support.html).

This option was added in MySQL 8.0.22. On platforms that do not implement network namespace support, failure occurs when the replica attempts to connect to the source.

REQUIRE\_ROW\_FORMAT (available as of MySQL 8.0.19) permits only row-based replication events to be processed by the replication channel. This option prevents the replication applier from taking actions such as creating temporary tables and executing LOAD DATA INFILE requests, which increases the security of the channel. Group Replication channels are automatically created with REQUIRE\_ROW\_FORMAT set, and you cannot change the option for those channels. For more information, see [Section 17.3.3, “Replication Privilege Checks”](https://dev.mysql.com/doc/refman/8.0/en/replication-privilege-checks.html).

PRIVILEGE\_CHECKS\_USER (available as of MySQL 8.0.18) names a user account that supplies a security context for the specified channel. NULL, which is the default, means no security context is used. The use of row-based binary logging is strongly recommended when PRIVILEGE\_CHECKS\_USER is set, and you can set REQUIRE\_ROW\_FORMAT to enforce this. For example, to start privilege checks on the channel channel\_1 on a running replica, issue the following statements:

mysql> STOP REPLICA | SLAVE FOR CHANNEL 'channel\_1';

mysql> CHANGE MASTER TO

 PRIVILEGE\_CHECKS\_USER = 'priv\_repl'@'%.example.com',

 REQUIRE\_ROW\_FORMAT = 1,

 FOR CHANNEL 'channel\_1';

mysql> START REPLICA | SLAVE FOR CHANNEL 'channel\_1';

The user name and host name for the user account must follow the syntax described in  [“Specifying Account Names”](https://dev.mysql.com/doc/refman/8.0/en/account-names.html), and the user must not be an anonymous user (with a blank user name) or the CURRENT\_USER. The account must have the [REPLICATION\_APPLIER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-applier) privilege, plus the required privileges to execute the transactions replicated on the channel. When you restart the replication channel, the privilege checks are applied from that point on. If you do not specify a channel and no other channels exist, the statement is applied to the default channel.

REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK (available as of MySQL 8.0.20) enables a replica to select its own policy for primary key checks. When the option is set to ON for a replication channel, the replica always uses the value ON for the [sql\_require\_primary\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_sql_require_primary_key) system variable in replication operations, requiring a primary key. When the option is set to OFF, the replica always uses the value OFF for the [sql\_require\_primary\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_sql_require_primary_key) system variable in replication operations, so that a primary key is never required, even if the source required one. When the REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK option is set to STREAM, which is the default, the replica uses whatever value is replicated from the source for each transaction.

* For multisource replication, setting REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK to ON or OFF enables a replica to normalize behavior across the replication channels for different sources, and keep a consistent setting for the [sql\_require\_primary\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_sql_require_primary_key) system variable. Using ON safeguards against the accidental loss of primary keys when multiple sources update the same set of tables. Using OFF allows sources that can manipulate primary keys to work alongside sources that cannot.
* When PRIVILEGE\_CHECKS\_USER is set, setting REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK to ON or OFF means that the user account does not need session administration level privileges to set restricted session variables, which are required to change the value of [sql\_require\_primary\_key](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_sql_require_primary_key) to match the source's setting for each transaction. For more information, see [Section 17.3.3, “Replication Privilege Checks”](https://dev.mysql.com/doc/refman/8.0/en/replication-privilege-checks.html%22%20%5Co%20%2217.3.3%C2%A0Replication%20Privilege%20Checks).

MASTER\_COMPRESSION\_ALGORITHMS and MASTER\_ZSTD\_COMPRESSION\_LEVEL (available as of MySQL 8.0.18) enable control over the use of compression for connections to the source:

* MASTER\_COMPRESSION\_ALGORITHMS specifies the permitted compression algorithms. The available algorithms are the same as for the [protocol\_compression\_algorithms](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_protocol_compression_algorithms) system variable. The default value is uncompressed.

The value of MASTER\_COMPRESSION\_ALGORITHMS applies only if the [slave\_compressed\_protocol](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_compressed_protocol) system variable is disabled. If [slave\_compressed\_protocol](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_compressed_protocol) is enabled, it takes precedence over MASTER\_COMPRESSION\_ALGORITHMS and connections to the source use zlib compression if both source and replica support that algorithm.

* MASTER\_ZSTD\_COMPRESSION\_LEVEL is the compression level to use for connections 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.

Binary log transaction compression (available as of MySQL 8.0.20), which is activated by the [binlog\_transaction\_compression](https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html%22%20%5Cl%20%22sysvar_binlog_transaction_compression) system variable, can also be used to save bandwidth. If you do this in combination with connection compression, connection compression has less opportunity to act on the data, but can still compress headers and those events and transaction payloads that are uncompressed.

The MASTER\_SSL\_***xxx*** options and the MASTER\_TLS\_***xxx*** options specify how the replica uses encryption and ciphers to secure the replication connection. These options can be changed even on replicas that are compiled without SSL support. They are saved to the source metadata repository, but are ignored if the replica does not have SSL support enabled. The MASTER\_SSL\_***xxx*** and MASTER\_TLS\_***xxx*** options perform the same functions as the --ssl-***xxx*** and --tls-***xxx*** client options described in [Command Options for Encrypted Connections](https://dev.mysql.com/doc/refman/8.0/en/connection-options.html#encrypted-connection-options).

**Important**

To connect to the source using a replication user account that authenticates with the caching\_sha2\_password plugin, you must either set up a secure connection as described in  [“Setting Up Replication to Use Encrypted Connections”](https://dev.mysql.com/doc/refman/8.0/en/replication-solutions-encrypted-connections.html), or enable the unencrypted connection to support password exchange using an RSA key pair. The caching\_sha2\_password authentication plugin is the default for new users created from MySQL 8.0 If the user account that you create or use for replication (as specified by the MASTER\_USER option) uses this authentication plugin, and you are not using a secure connection, you must enable RSA key pair-based password exchange for a successful connection.

To enable RSA key pair-based password exchange, specify either the MASTER\_PUBLIC\_KEY\_PATH or the GET\_MASTER\_PUBLIC\_KEY=1 option. Either of these options provides the RSA public key to the replica:

* MASTER\_PUBLIC\_KEY\_PATH indicates the path name to a file containing a replica-side copy of the public key required by the source for RSA key pair-based password exchange. The file must be in PEM format. This option applies to replicas that authenticate with the sha256\_password or caching\_sha2\_password authentication plugin. (For sha256\_password, MASTER\_PUBLIC\_KEY\_PATH can be used only if MySQL was built using OpenSSL.)
* GET\_MASTER\_PUBLIC\_KEY indicates whether to request from the source the public key required for RSA key pair-based password exchange. This option applies to replicas that authenticate with the caching\_sha2\_password authentication plugin. For connections by accounts that authenticate using this plugin, the source does not send the public key unless requested, so it must be requested or specified in the client. If MASTER\_PUBLIC\_KEY\_PATH is given and specifies a valid public key file, it takes precedence over GET\_MASTER\_PUBLIC\_KEY.

The MASTER\_HEARTBEAT\_PERIOD, MASTER\_CONNECT\_RETRY, MASTER\_RETRY\_COUNT , and (from MySQL 8.0.22) SOURCE\_CONNECTION\_AUTO\_FAILOVER options control how the replica recognizes that the connection to the source has been lost and makes attempts to reconnect.

* The [slave\_net\_timeout](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_net_timeout) system variable specifies the number of seconds that the replica waits for either more data or a heartbeat signal from the source, before the replica considers the connection broken, aborts the read, and tries to reconnect. The default value is 60 seconds (one minute).
* The heartbeat interval, which stops the connection timeout occurring in the absence of data if the connection is still good, is controlled by the MASTER\_HEARTBEAT\_PERIOD option. A heartbeat signal is sent to the replica after that number of seconds, and the waiting period is reset whenever the source's binary log is updated with an event. Heartbeats are therefore sent by the source only if there are no unsent events in the binary log file for a period longer than this. The heartbeat interval ***interval*** is a decimal value having the range 0 to 4294967 seconds and a resolution in milliseconds; the smallest nonzero value is 0.001. Setting ***interval*** to 0 disables heartbeats altogether. The heartbeat interval defaults to half the value of the [slave\_net\_timeout](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_net_timeout) system variable. It is recorded in the source metadata repository and shown in the [replication\_connection\_configuration](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-replication-connection-configuration-table.html%22%20%5Co%20%2227.12.11.1%C2%A0The%20replication_connection_configuration%20Table) Performance Schema table. Issuing [RESET REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) resets the heartbeat interval to the default value.

Note that a change to the value or default setting of [slave\_net\_timeout](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_net_timeout) does not automatically change the heartbeat interval, whether that has been set explicitly or is using a previously calculated default. A warning is issued if you set @@GLOBAL.slave\_net\_timeout to a value less than that of the current heartbeat interval. If [slave\_net\_timeout](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_net_timeout) is changed, you must also issue [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) to adjust the heartbeat interval to an appropriate value so that the heartbeat signal occurs before the connection timeout. If you do not do this, the heartbeat signal has no effect, and if no data is received from the source, the replica can make repeated reconnection attempts, creating zombie dump threads.

* If the replica does need to reconnect, the first retry occurs immediately after the timeout. MASTER\_CONNECT\_RETRY specifies the interval between reconnection attempts, and MASTER\_RETRY\_COUNT limits the number of reconnection attempts. If both the default settings are used, the replica waits 60 seconds between reconnection attempts (MASTER\_CONNECT\_RETRY=60), and keeps attempting to reconnect at this rate for 60 days (MASTER\_RETRY\_COUNT=86400). These values are recorded in the source metadata repository and shown in the [replication\_connection\_configuration](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-replication-connection-configuration-table.html) Performance Schema table. MASTER\_RETRY\_COUNT supersedes the [--master-retry-count](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_master-retry-count) server startup option.
* From MySQL 8.0.22, you can set the SOURCE\_CONNECTION\_AUTO\_FAILOVER option to activate the asynchronous connection failover mechanism for a replication channel if one or more alternative replication source servers are available (so when there are multiple MySQL servers or groups of servers that share the replicated data). The asynchronous connection failover mechanism takes over after the reconnection attempts controlled by MASTER\_CONNECT\_RETRY and MASTER\_RETRY\_COUNT are exhausted. It reconnects the replica to an alternative source chosen from a specified source list, which you manage using the [asynchronous\_connection\_failover\_add\_source](https://dev.mysql.com/doc/refman/8.0/en/replication-functions-source-list.html#udf_asynchronous-connection-failover-add-source) and [asynchronous\_connection\_failover\_delete\_source](https://dev.mysql.com/doc/refman/8.0/en/replication-functions-source-list.html%22%20%5Cl%20%22udf_asynchronous-connection-failover-delete-source) UDFs..

**Important**

* 1. You can only set SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1 when GTID auto-positioning is in use (MASTER\_AUTO\_POSITION = 1).
	2. When you set SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1, set MASTER\_RETRY\_COUNT to a minimal number that just allows a few retry attempts with the same source, in case the connection failure is caused by a transient network outage. Otherwise the asynchronous connection failover mechanism cannot be activated promptly.
	3. When you set SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1, the replication metadata repositories must contain the credentials for a replication user account that can be used to connect to all the servers on the source list for the replication channel. These credentials can be set using the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement with the MASTER\_USER and MASTER\_PASSWORD options..

MASTER\_DELAY specifies how many seconds behind the source the replica must lag. An event received from the source is not executed until at least ***interval*** seconds later than its execution on the source. The default is 0. An error occurs if ***interval*** is not a nonnegative integer in the range from 0 to 2**31**−1.. A CHANGE MASTER TO statement employing the MASTER\_DELAY option can be executed on a running replica when the replication SQL thread is stopped.

MASTER\_BIND is for use on replicas that have multiple network interfaces, and determines which of the replica's network interfaces is chosen for connecting to the source. The address configured with this option, if any, can be seen in the Master\_Bind column of the output from [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html). In the source metadata repository table mysql.slave\_master\_info, the value can be seen as the Master\_bind column. The ability to bind a replica to a specific network interface is also supported by NDB Cluster.

MASTER\_LOG\_FILE and MASTER\_LOG\_POS are the coordinates at which the replication I/O thread should begin reading from the source the next time the thread starts. RELAY\_LOG\_FILE and RELAY\_LOG\_POS are the coordinates at which the replication SQL thread should begin reading from the relay log the next time the thread starts. If you specify either of MASTER\_LOG\_FILE or MASTER\_LOG\_POS, you cannot specify RELAY\_LOG\_FILE or RELAY\_LOG\_POS. If you specify either of MASTER\_LOG\_FILE or MASTER\_LOG\_POS, you also cannot specify MASTER\_AUTO\_POSITION = 1 (described later in this section). If neither of MASTER\_LOG\_FILE or MASTER\_LOG\_POS is specified, the replica uses the last coordinates of the replication SQL thread before [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) was issued. This ensures that there is no discontinuity in replication, even if the replication SQL thread was late compared to the replication I/O thread, when you merely want to change, say, the password to use.

RELAY\_LOG\_FILE can use either an absolute or relative path, and uses the same base name as MASTER\_LOG\_FILE. A CHANGE MASTER TO statement employing RELAY\_LOG\_FILE, RELAY\_LOG\_POS, or both options can be executed on a running replica when the replication SQL thread is stopped. Relay logs are preserved if at least one of the replication SQL thread and the replication I/O thread is running. If both threads are stopped, all relay log files are deleted unless at least one of RELAY\_LOG\_FILE or RELAY\_LOG\_POS is specified. Note that the Group Replication applier channel (group\_replication\_applier) has no I/O thread, only an SQL thread. For this channel, the relay logs are not preserved when the SQL thread is stopped.

When MASTER\_AUTO\_POSITION = 1 is used with CHANGE MASTER TO, the replica attempts to connect to the source using the GTID-based replication protocol. This option can be used with CHANGE MASTER TO only if both the replication SQL thread and replication I/O thread are stopped. Both the replica and the source must have GTIDs enabled ([GTID\_MODE=ON](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html#sysvar_gtid_mode), ON\_PERMISSIVE, or OFF\_PERMISSIVE on the replica, and [GTID\_MODE=ON](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html#sysvar_gtid_mode) on the source). Auto-positioning is used for the connection, so the coordinates represented by MASTER\_LOG\_FILE and MASTER\_LOG\_POS are not used, and the use of either or both of these options together with MASTER\_AUTO\_POSITION = 1 causes an error. If multi-source replication is enabled on the replica, you need to set the MASTER\_AUTO\_POSITION = 1 option for each applicable replication channel.

With MASTER\_AUTO\_POSITION = 1 set, in the initial connection handshake, the replica sends a GTID set containing the transactions that it has already received, committed, or both. The source responds by sending all transactions recorded in its binary log whose GTID is not included in the GTID set sent by the replica. This exchange ensures that the source only sends the transactions with a GTID that the replica has not already recorded or committed. If the replica receives transactions from more than one source, as in the case of a diamond topology, the auto-skip function ensures that the transactions are not applied twice..

If any of the transactions that should be sent by the source have been purged from the source's binary log, or added to the set of GTIDs in the [gtid\_purged](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_purged) system variable by another method, the source sends the error **ER\_MASTER\_HAS\_PURGED\_REQUIRED\_GTIDS** to the replica, and replication does not start. The GTIDs of the missing purged transactions are identified and listed in the source's error log in the warning message **ER\_FOUND\_MISSING\_GTIDS**. Also, if during the exchange of transactions it is found that the replica has recorded or committed transactions with the source's UUID in the GTID, but the source itself has not committed them, the source sends the error **ER\_SLAVE\_HAS\_MORE\_GTIDS\_THAN\_MASTER** to the replica and replication does not start.

You can see whether replication is running with auto-positioning enabled by checking the Performance Schema [replication\_connection\_status](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-replication-connection-status-table.html%22%20%5Co%20%2227.12.11.2%C2%A0The%20replication_connection_status%20Table) table or the output of [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html). Disabling the MASTER\_AUTO\_POSITION option again makes the replica revert to file-based replication, in which case you must also specify one or both of the MASTER\_LOG\_FILE or MASTER\_LOG\_POS options.

IGNORE\_SERVER\_IDS takes a comma-separated list of 0 or more server IDs. Events originating from the corresponding servers are ignored, with the exception of log rotation and deletion events, which are still recorded in the relay log.

In circular replication, the originating server normally acts as the terminator of its own events, so that they are not applied more than once. Thus, this option is useful in circular replication when one of the servers in the circle is removed. Suppose that you have a circular replication setup with 4 servers, having server IDs 1, 2, 3, and 4, and server 3 fails. When bridging the gap by starting replication from server 2 to server 4, you can include IGNORE\_SERVER\_IDS = (3) in the [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement that you issue on server 4 to tell it to use server 2 as its source instead of server 3. Doing so causes it to ignore and not to propagate any statements that originated with the server that is no longer in use.

If IGNORE\_SERVER\_IDS contains the server's own ID and the server was started with the [--replicate-same-server-id](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-same-server-id) option enabled, an error results.

**Note**

When global transaction identifiers (GTIDs) are used for replication, transactions that have already been applied are automatically ignored, so the IGNORE\_SERVER\_IDS function is not required and is deprecated. If [gtid\_mode=ON](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_mode) is set for the server, a deprecation warning is issued if you include the IGNORE\_SERVER\_IDS option in a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement.

The source metadata repository and the output of [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html) provide the list of servers that are currently ignored.

If a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement is issued without any IGNORE\_SERVER\_IDS option, any existing list is preserved. To clear the list of ignored servers, it is necessary to use the option with an empty list:

CHANGE MASTER TO IGNORE\_SERVER\_IDS = ();

[RESET REPLICA | SLAVE ALL](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) clears IGNORE\_SERVER\_IDS.

**Note**

A deprecation warning is issued if SET GTID\_MODE=ON is issued when any channel has existing server IDs set with IGNORE\_SERVER\_IDS. Before starting GTID-based replication, check for and clear all ignored server ID lists on the servers involved. The [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html) statement displays the list of ignored IDs, if there is one. If you do receive the deprecation warning, you can still clear a list after [gtid\_mode=ON](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_mode) is set by issuing a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement containing the IGNORE\_SERVER\_IDS option with an empty list.

Invoking [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) causes the previous values for MASTER\_HOST, MASTER\_PORT, MASTER\_LOG\_FILE, and MASTER\_LOG\_POS to be written to the error log, along with other information about the replica's state prior to execution.

CHANGE MASTER TO causes an implicit commit of an ongoing transaction.

From MySQL 5.7, the strict requirement to execute [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) prior to issuing any [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement (and [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) afterward) is removed. Instead of depending on whether the replica is stopped, the behavior of CHANGE MASTER TO depends on the states of the replication SQL thread and replication I/O thread; which of these threads is stopped or running now determines the options that can or cannot be used with a CHANGE MASTER TO statement at a given point in time. The rules for making this determination are listed here:

* If the SQL thread is stopped, you can execute CHANGE MASTER TO using any combination that is otherwise allowed of RELAY\_LOG\_FILE, RELAY\_LOG\_POS, and MASTER\_DELAY options, even if the replication I/O thread is running. No other options may be used with this statement when the I/O thread is running.
* If the I/O thread is stopped, you can execute CHANGE MASTER TO using any of the options for this statement (in any allowed combination) except RELAY\_LOG\_FILE, RELAY\_LOG\_POS, MASTER\_DELAY, or MASTER\_AUTO\_POSITION = 1 even when the SQL thread is running.
* Both the SQL thread and the I/O thread must be stopped before issuing a CHANGE MASTER TO statement that employs MASTER\_AUTO\_POSITION = 1.

You can check the current state of the replication SQL thread and replication I/O thread using [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html). Note that the Group Replication applier channel (group\_replication\_applier) has no I/O thread, only an SQL thread.

If you are using statement-based replication and temporary tables, it is possible for a CHANGE MASTER TO statement following a [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) statement to leave behind temporary tables on the replica. A warning ([ER\_WARN\_OPEN\_TEMP\_TABLES\_MUST\_BE\_ZERO](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warn_open_temp_tables_must_be_zero)) is now issued whenever this occurs. You can avoid this in such cases by making sure that the value of the [Slave\_open\_temp\_tables](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html%22%20%5Cl%20%22statvar_Slave_open_temp_tables) system status variable is equal to 0 prior to executing such a CHANGE MASTER TO statement.

[CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) is useful for setting up a replica when you have the snapshot of the source and have recorded the source's binary log coordinates corresponding to the time of the snapshot. After loading the snapshot into the replica to synchronize it with the source, you can run CHANGE MASTER TO MASTER\_LOG\_FILE='***log\_name***', MASTER\_LOG\_POS=***log\_pos*** on the replica to specify the coordinates at which the replica should begin reading the source's binary log.

The following example changes the source server the replica uses and establishes the source's binary log coordinates from which the replica begins reading. This is used when you want to set up the replica to replicate the source:

CHANGE MASTER TO

 MASTER\_HOST='source2.example.com',

 MASTER\_USER='replication',

 MASTER\_PASSWORD='password',

 MASTER\_PORT=3306,

 MASTER\_LOG\_FILE='source2-bin.001',

 MASTER\_LOG\_POS=4,

 MASTER\_CONNECT\_RETRY=10;

The next example shows an operation that is less frequently employed. It is used when the replica has relay log files that you want it to execute again for some reason. To do this, the source need not be reachable. You need only use [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) and start the SQL thread ([START REPLICA | SLAVE SQL\_THREAD](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html)):

CHANGE MASTER TO

 RELAY\_LOG\_FILE='replica-relay-bin.006',

 RELAY\_LOG\_POS=4025;

The following table shows the maximum permissible length for the string-valued options.

| **Option** | **Maximum Length** |
| --- | --- |
| MASTER\_HOST | 255 (60 prior to MySQL 8.0.17) |
| MASTER\_USER | 96 |
| MASTER\_PASSWORD | 32 |
| MASTER\_LOG\_FILE | 511 |
| RELAY\_LOG\_FILE | 511 |
| MASTER\_SSL\_CA | 511 |
| MASTER\_SSL\_CAPATH | 511 |
| MASTER\_SSL\_CERT | 511 |
| MASTER\_SSL\_CRL | 511 |
| MASTER\_SSL\_CRLPATH | 511 |
| MASTER\_SSL\_KEY | 511 |
| MASTER\_SSL\_CIPHER | 511 |
| MASTER\_TLS\_VERSION | 511 |
| MASTER\_TLS\_CIPHERSUITES | 4000 |
| MASTER\_PUBLIC\_KEY\_PATH | 511 |
| MASTER\_COMPRESSION\_ALGORITHMS | 99 |
| NETWORK\_NAMESPACE | 64 |

#### CHANGE REPLICATION FILTER Statement

CHANGE REPLICATION FILTER filter[, filter]

 [, ...] [FOR CHANNEL channel]

filter: {

 REPLICATE\_DO\_DB = (db\_list)

 | REPLICATE\_IGNORE\_DB = (db\_list)

 | REPLICATE\_DO\_TABLE = (tbl\_list)

 | REPLICATE\_IGNORE\_TABLE = (tbl\_list)

 | REPLICATE\_WILD\_DO\_TABLE = (wild\_tbl\_list)

 | REPLICATE\_WILD\_IGNORE\_TABLE = (wild\_tbl\_list)

 | REPLICATE\_REWRITE\_DB = (db\_pair\_list)

}

db\_list:

 db\_name[, db\_name][, ...]

tbl\_list:

 db\_name*.table\_name*[, db\_name*.table\_name*][, ...]

wild\_tbl\_list:

 'db\_pattern.table\_pattern'[, 'db\_pattern.table\_pattern'][, ...]

db\_pair\_list:

 (db\_pair)[, (db\_pair)][, ...]

db\_pair:

 from\_db, to\_db

CHANGE REPLICATION FILTER sets one or more replication filtering rules on the replica in the same way as starting the replica **[mysqld](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html%22%20%5Co%20%224.3.1%C2%A0mysqld%20%E2%80%94%20The%20MySQL%20Server)** with replication filtering options such as [--replicate-do-db](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-do-db) or [--replicate-wild-ignore-table](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-wild-ignore-table). Unlike the case with the server options, this statement does not require restarting the server to take effect, only that the replication SQL thread be stopped using [STOP REPLICA | SLAVE SQL\_THREAD](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) first (and restarted with [START REPLICA | SLAVE SQL\_THREAD](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) afterwards). [CHANGE REPLICATION FILTER](https://dev.mysql.com/doc/refman/8.0/en/change-replication-filter.html) requires the [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege). Use the FOR CHANNEL ***channel*** clause to make a replication filter specific to a replication channel, for example on a multi-source replica. Filters applied without a specific FOR CHANNEL clause are considered global filters, meaning that they are applied to all replication channels.

**Note**

Global replication filters cannot be set on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be set on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be set on the group\_replication\_applier or group\_replication\_recovery channels.

The following list shows the CHANGE REPLICATION FILTER options and how they relate to --replicate-\* server options:

* REPLICATE\_DO\_DB: Include updates based on database name. Equivalent to [--replicate-do-db](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-do-db).
* REPLICATE\_IGNORE\_DB: Exclude updates based on database name. Equivalent to [--replicate-ignore-db](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-ignore-db).
* REPLICATE\_DO\_TABLE: Include updates based on table name. Equivalent to [--replicate-do-table](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-do-table).
* REPLICATE\_IGNORE\_TABLE: Exclude updates based on table name. Equivalent to [--replicate-ignore-table](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-ignore-table).
* REPLICATE\_WILD\_DO\_TABLE: Include updates based on wildcard pattern matching table name. Equivalent to [--replicate-wild-do-table](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-wild-do-table).
* REPLICATE\_WILD\_IGNORE\_TABLE: Exclude updates based on wildcard pattern matching table name. Equivalent to [--replicate-wild-ignore-table](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-wild-ignore-table).
* REPLICATE\_REWRITE\_DB: Perform updates on replica after substituting new name on replica for specified database on source. Equivalent to [--replicate-rewrite-db](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-rewrite-db).

The precise effects of REPLICATE\_DO\_DB and REPLICATE\_IGNORE\_DB filters are dependent on whether statement-based or row-based replication is in effect.

Multiple replication filtering rules can be created in a single CHANGE REPLICATION FILTER statement by separating the rules with commas, as shown here:

CHANGE REPLICATION FILTER

 REPLICATE\_DO\_DB = (d1), REPLICATE\_IGNORE\_DB = (d2);

Issuing the statement just shown is equivalent to starting the replica **[mysqld](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html%22%20%5Co%20%224.3.1%C2%A0mysqld%20%E2%80%94%20The%20MySQL%20Server)** with the options [--replicate-do-db=d1](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-do-db) [--replicate-ignore-db=d2](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-ignore-db).

On a multi-source replica, which uses multiple replication channels to process transaction from different sources, use the FOR CHANNEL ***channel*** clause to set a replication filter on a replication channel:

CHANGE REPLICATION FILTER REPLICATE\_DO\_DB = (d1) FOR CHANNEL channel\_1;

This enables you to create a channel specific replication filter to filter out selected data from a source. When a FOR CHANNEL clause is provided, the replication filter statement acts on that replication channel, removing any existing replication filter which has the same filter type as the specified replication filters, and replacing them with the specified filter. Filter types not explicitly listed in the statement are not modified. If issued against a replication channel which is not configured, the statement fails with an ER\_SLAVE\_CONFIGURATION error. If issued against Group Replication channels, the statement fails with an ER\_SLAVE\_CHANNEL\_OPERATION\_NOT\_ALLOWED error.

On a replica with multiple replication channels configured, issuing [CHANGE REPLICATION FILTER](https://dev.mysql.com/doc/refman/8.0/en/change-replication-filter.html) with no FOR CHANNEL clause configures the replication filter for every configured replication channel, and for the global replication filters. For every filter type, if the filter type is listed in the statement, then any existing filter rules of that type are replaced by the filter rules specified in the most recently issued statement, otherwise the old value of the filter type is retained.

If the same filtering rule is specified multiple times, only the last such rule is actually used. For example, the two statements shown here have exactly the same effect, because the first REPLICATE\_DO\_DB rule in the first statement is ignored:

CHANGE REPLICATION FILTER

 REPLICATE\_DO\_DB = (db1, db2), REPLICATE\_DO\_DB = (db3, db4);

CHANGE REPLICATION FILTER

 REPLICATE\_DO\_DB = (db3, db4);

**Caution**

This behavior differs from that of the --replicate-\* filter options where specifying the same option multiple times causes the creation of multiple filter rules.

Names of tables and database not containing any special characters need not be quoted. Values used with REPLICATION\_WILD\_TABLE and REPLICATION\_WILD\_IGNORE\_TABLE are string expressions, possibly containing (special) wildcard characters, and so must be quoted. This is shown in the following example statements:

CHANGE REPLICATION FILTER

 REPLICATE\_WILD\_DO\_TABLE = ('db1.old%');

CHANGE REPLICATION FILTER

 REPLICATE\_WILD\_IGNORE\_TABLE = ('db1.new%', 'db2.new%');

Values used with REPLICATE\_REWRITE\_DB represent pairs of database names; each such value must be enclosed in parentheses. The following statement rewrites statements occurring on database db1 on the source to database db2 on the replica:

CHANGE REPLICATION FILTER REPLICATE\_REWRITE\_DB = ((db1, db2));

The statement just shown contains two sets of parentheses, one enclosing the pair of database names, and the other enclosing the entire list. This is perhaps more easily seen in the following example, which creates two rewrite-db rules, one rewriting database dbA to dbB, and one rewriting database dbC to dbD:

CHANGE REPLICATION FILTER

 REPLICATE\_REWRITE\_DB = ((dbA, dbB), (dbC, dbD));

The [CHANGE REPLICATION FILTER](https://dev.mysql.com/doc/refman/8.0/en/change-replication-filter.html) statement replaces replication filtering rules only for the filter types and replication channels affected by the statement, and leaves other rules and channels unchanged. If you want to unset all filters of a given type, set the filter's value to an explicitly empty list, as shown in this example, which removes all existing REPLICATE\_DO\_DB and REPLICATE\_IGNORE\_DB rules:

CHANGE REPLICATION FILTER

 REPLICATE\_DO\_DB = (), REPLICATE\_IGNORE\_DB = ();

Setting a filter to empty in this way removes all existing rules, does not create any new ones, and does not restore any rules set at mysqld startup using --replicate-\* options on the command line or in the configuration file.

The [RESET REPLICA | SLAVE ALL](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) statement removes channel specific replication filters that were set on channels deleted by the statement. When the deleted channel or channels are recreated, any global replication filters specified for the replica are copied to them, and no channel specific replication filters are applied.

#### MASTER\_POS\_WAIT() Statement

SELECT MASTER\_POS\_WAIT('source\_log\_file', source\_log\_pos [, timeout][, channel])

This is actually a function, not a statement. It is used to ensure that the replica has read and executed events up to a given position in the source's binary log.

#### RESET REPLICA | SLAVE Statement

RESET {REPLICA | SLAVE} [ALL] [channel\_option]

channel\_option:

 FOR CHANNEL channel

RESET REPLICA | SLAVE makes the replica forget its position in the source's binary log. From MySQL 8.0.22, use [RESET REPLICA](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) in place of [RESET SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-slave.html), which is deprecated from that release. In releases before MySQL 8.0.22, use [RESET SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-slave.html).

This statement is meant to be used for a clean start; it clears the replication metadata repositories, deletes all the relay log files, and starts a new relay log file. It also resets to 0 the replication delay specified with the MASTER\_DELAY option of [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html).

**Note**

All relay log files are deleted, even if they have not been completely executed by the replication SQL thread. (This is a condition likely to exist on a replica if you have issued a [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) statement or if the replica is highly loaded.)

For a server where GTIDs are in use ([gtid\_mode](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html%22%20%5Cl%20%22sysvar_gtid_mode) is ON), issuing RESET REPLICA | SLAVE has no effect on the GTID execution history. The statement does not change the values of gtid\_executed or gtid\_purged, or the mysql.gtid\_executed table. If you need to reset the GTID execution history, use [RESET MASTER](https://dev.mysql.com/doc/refman/8.0/en/reset-master.html), even if the GTID-enabled server is a replica where binary logging is disabled.

RESET REPLICA | SLAVE requires the [RELOAD](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_reload) privilege.

To use RESET REPLICA | SLAVE, the replication SQL thread and replication I/O thread must be stopped, so on a running replica use [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) before issuing RESET REPLICA | SLAVE. To use RESET REPLICA | SLAVE on a Group Replication group member, the member status must be OFFLINE, meaning that the plugin is loaded but the member does not currently belong to any group. A group member can be taken offline by using a [STOP GROUP REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/stop-group-replication.html) statement.

The optional FOR CHANNEL ***channel*** clause enables you to name which replication channel the statement applies to. Providing a FOR CHANNEL ***channel*** clause applies the RESET REPLICA | SLAVE statement to a specific replication channel. Combining a FOR CHANNEL ***channel*** clause with the ALL option deletes the specified channel. If no channel is named and no extra channels exist, the statement applies to the default channel. Issuing a RESET REPLICA | SLAVE ALL statement without a FOR CHANNEL ***channel*** clause when multiple replication channels exist deletes all replication channels and recreates only the default channel.

RESET REPLICA | SLAVE does not change any replication connection parameters, which include the source's host name and port, the replication user account and its password, the PRIVILEGE\_CHECKS\_USER account, the REQUIRE\_ROW\_FORMAT option, and the REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK option. If you want to change any of the replication connection parameters, you can do this using a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement after the server start. If you want to remove all of the replication connection parameters, use RESET REPLICA | SLAVE ALL. RESET REPLICA | SLAVE ALL also clears the IGNORE\_SERVER\_IDS list set by [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html). When you have used RESET REPLICA | SLAVE ALL, if you want to use the instance as a replica again, you need to issue a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement after the server start to specify new connection parameters.

In the event of an unexpected server exit or deliberate restart after issuing RESET REPLICA | SLAVE but before issuing [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html), retention of the replication connection parameters depends on the repository used for the replication metadata:

* When [master\_info\_repository=TABLE](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#sysvar_master_info_repository) and [relay\_log\_info\_repository=TABLE](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#sysvar_relay_log_info_repository) are set on the server (which are the default settings from MySQL 8.0), replication connection parameters are preserved in the crash-safe InnoDB tables mysql.slave\_master\_info and mysql.slave\_relay\_log\_info as part of the RESET REPLICA | SLAVE operation. They are also retained in memory. In the event of an unexpected server exit or deliberate restart after issuing RESET REPLICA | SLAVE but before issuing [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html), the replication connection parameters are retrieved from the tables and reapplied to the channel. This situation applies from MySQL 8.0.13 for the source metadata repository, and from MySQL 8.0.19 for the replica metadata repository.
* If [master\_info\_repository=FILE](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#sysvar_master_info_repository) and [relay\_log\_info\_repository=FILE](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#sysvar_relay_log_info_repository) are set on the server, or the MySQL Server release is earlier than those specified above, replication connection parameters are only retained in memory. If the replica **[mysqld](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html%22%20%5Co%20%224.3.1%C2%A0mysqld%20%E2%80%94%20The%20MySQL%20Server)** is restarted immediately after issuing RESET REPLICA | SLAVE due to an unexpected server exit or deliberate restart, the connection parameters are lost. In that case, you must issue a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement after the server start to respecify the connection parameters before issuing [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html). Note that the FILE setting for these options is deprecated; expect it to be removed in a future release.

RESET REPLICA | SLAVE does not change any replication filter settings (such as [--replicate-ignore-table](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_replicate-ignore-table)) for channels affected by the statement. However, RESET REPLICA | SLAVE ALL removes the replication filters that were set on the channels deleted by the statement. When the deleted channel or channels are recreated, any global replication filters specified for the replica are copied to them, and no channel specific replication filters are applied.

RESET REPLICA | SLAVE causes an implicit commit of an ongoing transaction.

If the replication SQL thread was in the middle of replicating temporary tables when it was stopped, and RESET REPLICA | SLAVE is issued, these replicated temporary tables are deleted on the replica.

RESET REPLICA | SLAVE does not reset the heartbeat period or SSL\_VERIFY\_SERVER\_CERT.

**Note**

When used on an NDB Cluster replica SQL node, RESET REPLICA | SLAVE clears the mysql.ndb\_apply\_status table. You should keep in mind when using this statement that ndb\_apply\_status uses the [NDB](https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster.html) storage engine and so is shared by all SQL nodes attached to the cluster.

You can override this behavior by issuing [SET](https://dev.mysql.com/doc/refman/8.0/en/set-variable.html) GLOBAL @@[ndb\_clear\_apply\_status=OFF](https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster-options-variables.html%22%20%5Cl%20%22sysvar_ndb_clear_apply_status) prior to executing RESET REPLICA | SLAVE, which keeps the replica from purging the ndb\_apply\_status table in such cases.

#### RESET SLAVE | REPLICA Statement

RESET {SLAVE | REPLICA} [ALL] [channel\_option]

channel\_option:

 FOR CHANNEL channel

Makes the replica forget its position in the source's binary log. From MySQL 8.0.22, [RESET SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-slave.html) is deprecated and the alias [RESET REPLICA](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) should be used instead. In releases before MySQL 8.0.22, use [RESET SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-slave.html). The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [RESET REPLICA](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) for a description of the statement.

#### START REPLICA | SLAVE Statement

START {REPLICA | SLAVE} [thread\_types] [until\_option] [connection\_options] [channel\_option]

thread\_types:

 [thread\_type [, thread\_type] ... ]

thread\_type:

 IO\_THREAD | SQL\_THREAD

until\_option:

 UNTIL { {SQL\_BEFORE\_GTIDS | SQL\_AFTER\_GTIDS} = gtid\_set

 | MASTER\_LOG\_FILE = 'log\_name', MASTER\_LOG\_POS = log\_pos

 | RELAY\_LOG\_FILE = 'log\_name', RELAY\_LOG\_POS = log\_pos

 | SQL\_AFTER\_MTS\_GAPS }

connection\_options:

 [USER='user\_name'] [PASSWORD='user\_pass'] [DEFAULT\_AUTH='plugin\_name'] [PLUGIN\_DIR='plugin\_dir']

channel\_option:

 FOR CHANNEL channel

gtid\_set:

 uuid\_set [, uuid\_set] ...

 | ''

uuid\_set:

 uuid:interval[:interval]...

uuid:

 hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh

h:

 [0-9,A-F]

interval:

 n[-n]

 (n >= 1)

START REPLICA | SLAVE starts one or both of the replication threads. From MySQL 8.0.22, use [START REPLICA](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) in place of [START SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-slave.html), which is deprecated from that release. In releases before MySQL 8.0.22, use [START SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-slave.html).

START REPLICA | SLAVE with no ***thread\_type*** options starts both of the replication threads. The replication I/O thread reads events from the source server and stores them in the relay log. The replication SQL thread reads events from the relay log and executes them. START REPLICA | SLAVE requires the [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege).

If START REPLICA | SLAVE succeeds in starting the replication threads, it returns without any error. However, even in that case, it might be that the replication threads start and then later stop (for example, because they do not manage to connect to the source or read its binary log, or some other problem). START REPLICA | SLAVE does not warn you about this. You must check the replica's error log for error messages generated by the replication threads, or check that they are running satisfactorily with [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html).

START REPLICA | SLAVE causes an implicit commit of an ongoing transaction.

[gtid\_next](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html#sysvar_gtid_next) must be set to AUTOMATIC before issuing this statement.

The optional FOR CHANNEL ***channel*** clause enables you to name which replication channel the statement applies to. Providing a FOR CHANNEL ***channel*** clause applies the START REPLICA | SLAVE statement to a specific replication channel. If no clause is named and no extra channels exist, the statement applies to the default channel. If a START REPLICA | SLAVE statement does not have a channel defined when using multiple channels, this statement starts the specified threads for all channels. This statement is disallowed for the group\_replication\_recovery channel.

You can add IO\_THREAD and SQL\_THREAD options to the statement to name which of the threads to start. Note that the Group Replication applier channel (group\_replication\_applier) has no I/O thread, only an SQL thread. Specifying the IO\_THREAD or SQL\_THREAD options when you start this channel has no benefit.

START REPLICA | SLAVE supports pluggable user-password authentication with the USER, PASSWORD, DEFAULT\_AUTH and PLUGIN\_DIR options, as described in the following list:

* USER: User name. Cannot be set to an empty or null string, or left unset if PASSWORD is used.
* PASSWORD: Password.
* DEFAULT\_AUTH: Name of plugin; default is MySQL native authentication.
* PLUGIN\_DIR: Location of plugin.

You cannot use the SQL\_THREAD option when specifying any of USER, PASSWORD, DEFAULT\_AUTH, or PLUGIN\_DIR, unless the IO\_THREAD option is also provided.

If an insecure connection is used with any these options, the server issues the warning Sending passwords in plain text without SSL/TLS is extremely insecure.

START REPLICA | SLAVE ... UNTIL supports two additional options for use with global transaction identifiers (GTIDs) . Each of these takes a set of one or more global transaction identifiers ***gtid\_set*** as an argument (see [GTID Sets](https://dev.mysql.com/doc/refman/8.0/en/replication-gtids-concepts.html#replication-gtids-concepts-gtid-sets), for more information).

When no ***thread\_type*** is specified, START REPLICA | SLAVE UNTIL SQL\_BEFORE\_GTIDS causes the replication SQL thread to process transactions until it has reached the first transaction whose GTID is listed in the ***gtid\_set***. START REPLICA | SLAVE UNTIL SQL\_AFTER\_GTIDS causes the replication threads to process all transactions until the ***last*** transaction in the ***gtid\_set*** has been processed by both threads. In other words, START REPLICA | SLAVE UNTIL SQL\_BEFORE\_GTIDS causes the replication SQL thread to process all transactions occurring before the first GTID in the ***gtid\_set*** is reached, and START REPLICA | SLAVE UNTIL SQL\_AFTER\_GTIDS causes the replication threads to handle all transactions, including those whose GTIDs are found in ***gtid\_set***, until each has encountered a transaction whose GTID is not part of the set. SQL\_BEFORE\_GTIDS and SQL\_AFTER\_GTIDS each support the SQL\_THREAD and IO\_THREAD options, although using IO\_THREAD with them currently has no effect.

For example, START REPLICA | SLAVE SQL\_THREAD UNTIL SQL\_BEFORE\_GTIDS = 3E11FA47-71CA-11E1-9E33-C80AA9429562:11-56 causes the replication SQL thread to process all transactions originating from the source whose [server\_uuid](https://dev.mysql.com/doc/refman/8.0/en/replication-options.html%22%20%5Cl%20%22sysvar_server_uuid) is 3E11FA47-71CA-11E1-9E33-C80AA9429562 until it encounters the transaction having sequence number 11; it then stops without processing this transaction. In other words, all transactions up to and including the transaction with sequence number 10 are processed. Executing START REPLICA | SLAVE SQL\_THREAD UNTIL SQL\_AFTER\_GTIDS = 3E11FA47-71CA-11E1-9E33-C80AA9429562:11-56, on the other hand, would cause the replication SQL thread to obtain all transactions just mentioned from the source, including all of the transactions having the sequence numbers 11 through 56, and then to stop without processing any additional transactions; that is, the transaction having sequence number 56 would be the last transaction fetched by the replication SQL thread.

When using a multithreaded replica with slave\_preserve\_commit\_order=0 set, there is a chance of gaps in the sequence of transactions that have been executed from the relay log in the following cases:

* killing the coordinator thread
* after an error occurs in the applier threads
* [**mysqld**](https://dev.mysql.com/doc/refman/8.0/en/mysqld.html) shuts down unexpectedly

Use the START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS statement to cause a multithreaded replica's worker threads to only run until no more gaps are found in the relay log, and then to stop. This statement can take an SQL\_THREAD option, but the effects of the statement remain unchanged. It has no effect on the replication I/O thread (and cannot be used with the IO\_THREAD option).

Issuing START REPLICA | SLAVE on a multithreaded replica with gaps in the sequence of transactions executed from the relay log generates a warning. In such a situation, the solution is to use START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS, then issue [RESET REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/reset-replica.html) to remove any remaining relay logs.

To change a failed multithreaded replica to single-threaded mode, you can issue the following series of statements, in the order shown:

START {REPLICA | SLAVE} UNTIL SQL\_AFTER\_MTS\_GAPS;

SET @@GLOBAL.slave\_parallel\_workers = 0;

START {REPLICA | SLAVE} SQL\_THREAD;

**Note**

It is possible to view the entire text of a running START REPLICA | SLAVE statement, including any USER or PASSWORD values used, in the output of [SHOW PROCESSLIST](https://dev.mysql.com/doc/refman/8.0/en/show-processlist.html). This is also true for the text of a running [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement, including any values it employs for MASTER\_USER or MASTER\_PASSWORD.

START REPLICA | SLAVE sends an acknowledgment to the user after both the replication I/O thread and the replication SQL thread have started. However, the replication I/O thread may not yet have connected. For this reason, a successful START REPLICA | SLAVE causes [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html) to show Replica\_SQL\_Running=Yes, but this does not guarantee that Replica\_IO\_Running=Yes (because Replica\_IO\_Running=Yes only if the I/O thread is running and connected).

An UNTIL clause (***until\_option***, in the preceding grammar) may be added to specify that the replica should start and run until the replication SQL thread reaches a given point in the source's binary log, specified by the MASTER\_LOG\_POS and MASTER\_LOG\_FILE options, or a given point in the replica's relay log, indicated with the RELAY\_LOG\_POS and RELAY\_LOG\_FILE options. For compressed transaction payloads, the position must be based on the compressed Transaction\_payload\_event. When the SQL thread reaches the point specified, it stops. If the SQL\_THREAD option is specified in the statement, it starts only the SQL thread. Otherwise, it starts both replication threads. If the SQL thread is running, the UNTIL clause is ignored and a warning is issued. You cannot use an UNTIL clause with the IO\_THREAD option.

It is also possible with START REPLICA | SLAVE UNTIL to specify a stop point relative to a given GTID or set of GTIDs using one of the options SQL\_BEFORE\_GTIDS or SQL\_AFTER\_GTIDS, as explained previously in this section. When using one of these options, you can specify SQL\_THREAD, IO\_THREAD, both of these, or neither of them. If you specify only SQL\_THREAD, then only the replication SQL thread is affected by the statement; if only IO\_THREAD is used, then only the replication I/O thread is affected. If both SQL\_THREAD and IO\_THREAD are used, or if neither of them is used, then both the SQL and I/O threads are affected by the statement.

For an UNTIL clause, you must specify any one of the following:

* Both a log file name and a position in that file
* Either of SQL\_BEFORE\_GTIDS or SQL\_AFTER\_GTIDS
* SQL\_AFTER\_MTS\_GAPS

Do not mix source and relay log options. Do not mix log file options with GTID options.

The UNTIL clause is not supported for multithreaded replicas except when also using SQL\_AFTER\_MTS\_GAPS. If UNTIL is used on a multithreaded replica without SQL\_AFTER\_MTS\_GAPS, the replica operates in single-threaded (sequential) mode for replication until the point specified by the UNTIL clause is reached.

Any UNTIL condition is reset by a subsequent [STOP REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) statement, a START REPLICA | SLAVE statement that includes no UNTIL clause, or a server restart.

When specifying a log file and position, you can use the IO\_THREAD option with START REPLICA | SLAVE ... UNTIL even though only the SQL thread is affected by this statement. The IO\_THREAD option is ignored in such cases. The preceding restriction does not apply when using one of the GTID options (SQL\_BEFORE\_GTIDS and SQL\_AFTER\_GTIDS); the GTID options support both SQL\_THREAD and IO\_THREAD, as explained previously in this section.

The UNTIL clause can be useful for debugging replication, or to cause replication to proceed until just before the point where you want to avoid having the replica replicate an event. For example, if an unwise [DROP TABLE](https://dev.mysql.com/doc/refman/8.0/en/drop-table.html) statement was executed on the source, you can use UNTIL to tell the replica to execute up to that point but no farther. To find what the event is, use **[mysqlbinlog](https://dev.mysql.com/doc/refman/8.0/en/mysqlbinlog.html%22%20%5Co%20%224.6.8%C2%A0mysqlbinlog%20%E2%80%94%20Utility%20for%20Processing%20Binary%20Log%20Files)** with the source's binary log or the replica's relay log, or by using a [SHOW BINLOG EVENTS](https://dev.mysql.com/doc/refman/8.0/en/show-binlog-events.html) statement.

If you are using UNTIL to have the replica process replicated queries in sections, it is recommended that you start the replica with the [--skip-slave-start](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html#option_mysqld_skip-slave-start) option to prevent the SQL thread from running when the replica server starts. It is probably best to use this option in an option file rather than on the command line, so that an unexpected server restart does not cause it to be forgotten.

The [SHOW REPLICA | SLAVE STATUS](https://dev.mysql.com/doc/refman/8.0/en/show-replica-status.html) statement includes output fields that display the current values of the UNTIL condition.

#### START SLAVE | REPLICA Statement

START {SLAVE | REPLICA} [thread\_types] [until\_option] [connection\_options] [channel\_option]

thread\_types:

 [thread\_type [, thread\_type] ... ]

thread\_type:

 IO\_THREAD | SQL\_THREAD

until\_option:

 UNTIL { {SQL\_BEFORE\_GTIDS | SQL\_AFTER\_GTIDS} = gtid\_set

 | MASTER\_LOG\_FILE = 'log\_name', MASTER\_LOG\_POS = log\_pos

 | RELAY\_LOG\_FILE = 'log\_name', RELAY\_LOG\_POS = log\_pos

 | SQL\_AFTER\_MTS\_GAPS }

connection\_options:

 [USER='user\_name'] [PASSWORD='user\_pass'] [DEFAULT\_AUTH='plugin\_name'] [PLUGIN\_DIR='plugin\_dir']

channel\_option:

 FOR CHANNEL channel

gtid\_set:

 uuid\_set [, uuid\_set] ...

 | ''

uuid\_set:

 uuid:interval[:interval]...

uuid:

 hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh

h:

 [0-9,A-F]

interval:

 n[-n]

 (n >= 1)

Starts the replication threads. From MySQL 8.0.22, [START SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-slave.html) is deprecated and the alias [START REPLICA](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) should be used instead. The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [START REPLICA](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html) for a description of the statement.

#### STOP REPLICA | SLAVE Statement

STOP {REPLICA | SLAVE} [thread\_types] [channel\_option]

thread\_types:

 [thread\_type [, thread\_type] ... ]

thread\_type: IO\_THREAD | SQL\_THREAD

channel\_option:

 FOR CHANNEL channel

Stops the replication threads. From MySQL 8.0.22, use [STOP REPLICA](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) in place of [STOP SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-slave.html), which is now deprecated. In releases before MySQL 8.0.22, use [STOP SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-slave.html).

STOP REPLICA | SLAVE requires the [REPLICATION\_SLAVE\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_replication-slave-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege). Recommended best practice is to execute STOP REPLICA | SLAVE on the replica before stopping the replica server.

Like [START REPLICA | SLAVE](https://dev.mysql.com/doc/refman/8.0/en/start-replica.html), this statement may be used with the IO\_THREAD and SQL\_THREAD options to name the replication thread or threads to be stopped. Note that the Group Replication applier channel (group\_replication\_applier) has no replication I/O thread, only a replication SQL thread. Using the SQL\_THREAD option therefore stops this channel completely.

STOP REPLICA | SLAVE causes an implicit commit of an ongoing transaction.

[gtid\_next](https://dev.mysql.com/doc/refman/8.0/en/replication-options-gtids.html#sysvar_gtid_next) must be set to AUTOMATIC before issuing this statement.

You can control how long STOP REPLICA | SLAVE waits before timing out by setting the [rpl\_stop\_slave\_timeout](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_rpl_stop_slave_timeout) system variable. This can be used to avoid deadlocks between STOP REPLICA | SLAVE and other SQL statements using different client connections to the replica. When the timeout value is reached, the issuing client returns an error message and stops waiting, but the STOP REPLICA | SLAVE instruction remains in effect. Once the replication threads are no longer busy, the STOP REPLICA | SLAVE statement is executed and the replica stops.

Some [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statements are allowed while the replica is running, depending on the states of the replication threads. However, using STOP REPLICA | SLAVE prior to executing [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) in such cases is still supported.

The optional FOR CHANNEL ***channel*** clause enables you to name which replication channel the statement applies to. Providing a FOR CHANNEL ***channel*** clause applies the STOP REPLICA | SLAVE statement to a specific replication channel. If no channel is named and no extra channels exist, the statement applies to the default channel. If a STOP REPLICA | SLAVE statement does not name a channel when using multiple channels, this statement stops the specified threads for all channels. This statement cannot be used with the group\_replication\_recovery channel..

When the replica is multithreaded ([slave\_parallel\_workers](https://dev.mysql.com/doc/refman/8.0/en/replication-options-replica.html%22%20%5Cl%20%22sysvar_slave_parallel_workers) is a nonzero value), any gaps in the sequence of transactions executed from the relay log are closed as part of stopping the worker threads. If the replica is stopped unexpectedly (for example due to an error in a worker thread, or another thread issuing [KILL](https://dev.mysql.com/doc/refman/8.0/en/kill.html)) while a STOP REPLICA | SLAVE statement is executing, the sequence of executed transactions from the relay log may become inconsistent.

When the source is using the row-based binary logging format, you should execute STOP REPLICA | SLAVE or STOP REPLICA | SLAVE SQL\_THREAD on the replica prior to shutting down the replica server if you are replicating any tables that use a nontransactional storage engine. If the current replication event group has modified one or more nontransactional tables, STOP REPLICA | SLAVE waits for up to 60 seconds for the event group to complete, unless you issue a [KILL QUERY](https://dev.mysql.com/doc/refman/8.0/en/kill.html) or [KILL CONNECTION](https://dev.mysql.com/doc/refman/8.0/en/kill.html) statement for the replication SQL thread. If the event group remains incomplete after the timeout, an error message is logged.

When the source is using the statement-based binary logging format, changing the source while it has open temporary tables is potentially unsafe. This is one of the reasons why statement-based replication of temporary tables is not recommended. You can find out whether there are any temporary tables on the replica by checking the value of [Slave\_open\_temp\_tables](https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html%22%20%5Cl%20%22statvar_Slave_open_temp_tables); when using statement-based replication, this value should be 0 before executing [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html). If there are any temporary tables open on the replica, issuing a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement after issuing a STOP REPLICA | SLAVE causes an [ER\_WARN\_OPEN\_TEMP\_TABLES\_MUST\_BE\_ZERO](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warn_open_temp_tables_must_be_zero) warning.

#### STOP SLAVE | REPLICA Statement

STOP {SLAVE | REPLICA} [thread\_types] [channel\_option]

thread\_types:

 [thread\_type [, thread\_type] ... ]

thread\_type: IO\_THREAD | SQL\_THREAD

channel\_option:

 FOR CHANNEL channel

Stops the replication threads. From MySQL 8.0.22, [STOP SLAVE](https://dev.mysql.com/doc/refman/8.0/en/stop-slave.html) is deprecated and the alias [STOP REPLICA](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) should be used instead. The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [STOP REPLICA](https://dev.mysql.com/doc/refman/8.0/en/stop-replica.html) for a description of the statement.

#### Functions which Configure the Source List

The following functions, which are available from MySQL 8.0.22, enable you to add and remove replication source servers from the source list for a replication channel. The asynchronous connection failover mechanism automatically establishes an asynchronous (source to replica) replication connection to a new source from the appropriate list after the existing connection from the replica to its source fails.

The source lists are stored in the mysql.replication\_asynchronous\_connection\_failover table, and can be viewed in the Performance Schema table [replication\_asynchronous\_connection\_failover](https://dev.mysql.com/doc/refman/8.0/en/performance-schema-replication-asynchronous-connection-failover-table.html%22%20%5Co%20%2227.12.11.3%C2%A0The%20replication_asynchronous_connection_failover%20Table).

* [asynchronous\_connection\_failover\_add\_source()](https://dev.mysql.com/doc/refman/8.0/en/replication-functions-source-list.html%22%20%5Cl%20%22udf_asynchronous-connection-failover-add-source)

Add configuration information for a replication source server to the source list for a replication channel.

Syntax:

asynchronous\_connection\_failover\_add\_source(channel, host, port, network\_namespace, weight)

Arguments:

* + ***channel***: The replication channel for which this replication source server is part of the source list.
	+ ***host***: The host name for this replication source server.
	+ ***port***: The port number for this replication source server.
	+ ***network\_namespace***: The network namespace for this replication source server. If you specify an empty string, connections use the default (global) network namespace.
	+ ***weight***: The priority of this replication source server in the replication channel's source list. The priority is from 1 to 100, with 100 being the highest, and 50 being the default. When the asynchronous connection failover mechanism activates, the source with the highest priority setting among the alternative sources listed in the source list for the channel is chosen for the first connection attempt. If this attempt does not work, the replica tries with all the listed sources in descending order of priority, then starts again from the highest priority source. If multiple sources have the same priority, the replica orders them randomly.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT asynchronous\_connection\_failover\_add\_source('channel2', '127.0.0.1', 3310, '', 80);

+-------------------------------------------------------------------------------------------------+

| asynchronous\_connection\_failover\_add\_source('channel2', '127.0.0.1', 3310, '', 80) |

+-------------------------------------------------------------------------------------------------+

| Source configuration details successfully inserted. |

+-------------------------------------------------------------------------------------------------+

For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](https://dev.mysql.com/doc/refman/8.0/en/replication-asynchronous-connection-failover.html).

* [asynchronous\_connection\_failover\_delete\_source()](https://dev.mysql.com/doc/refman/8.0/en/replication-functions-source-list.html%22%20%5Cl%20%22udf_asynchronous-connection-failover-delete-source)

Remove configuration information for a replication source server from the source list for a replication channel.

Syntax:

asynchronous\_connection\_failover\_delete\_source(channel, host, port, network\_namespace)

Arguments:

* + ***channel***: The replication channel for which this replication source server was part of the source list.
	+ ***host***: The host name for this replication source server.
	+ ***port***: The port number for this replication source server.
	+ ***network\_namespace***: The network namespace for this replication source server. An empty string means connections use the default (global) network namespace.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT asynchronous\_connection\_failover\_delete\_source('channel2', '127.0.0.1', 3310, '');

+------------------------------------------------------------------------------------------------+

| asynchronous\_connection\_failover\_delete\_source('channel2', '127.0.0.1', 3310, '') |

+------------------------------------------------------------------------------------------------+

| Source configuration details successfully deleted. |

+------------------------------------------------------------------------------------------------+

For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](https://dev.mysql.com/doc/refman/8.0/en/replication-asynchronous-connection-failover.html).

### SQL Statements for Controlling Group Replication

This section provides information about the statements used for controlling group replication.

#### START GROUP\_REPLICATION Statement

 START GROUP\_REPLICATION

 [USER='user\_name']

 [, PASSWORD='user\_pass']

 [, DEFAULT\_AUTH='plugin\_name']

Starts group replication. This statement requires the [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_group-replication-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege). If [super\_read\_only=ON](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_super_read_only) is set and the member should join as a primary, [super\_read\_only](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_super_read_only) is set to OFF once Group Replication successfully starts.

From MySQL 8.0.21, you can specify user credentials for distributed recovery on the START GROUP\_REPLICATION statement using the USER, PASSWORD, and DEFAULT\_AUTH options, as follows:

* USER: The replication user for distributed recovery. You cannot specify an empty or null string, or omit the USER option if PASSWORD is specified.
* PASSWORD: The password for the replication user account. The password cannot be encrypted, but it is masked in the query log.
* DEFAULT\_AUTH: The name of the authentication plugin used for the replication user account. If you do not specify this option, MySQL native authentication (the mysql\_native\_password plugin) is assumed. This option acts as a hint to the server, and the donor for distributed recovery overrides it if a different plugin is associated with the user account on that server. The authentication plugin used by default when you create user accounts in MySQL 8 is the caching SHA-2 authentication plugin (caching\_sha2\_password).

These credentials are used for distributed recovery on the group\_replication\_recovery channel. When you specify user credentials on START GROUP\_REPLICATION, the credentials are saved in memory only, and are removed by a STOP GROUP\_REPLICATION statement or server shutdown. You must issue a START GROUP\_REPLICATION statement to provide the credentials again. This method is therefore not compatible with starting Group Replication automatically on server start, as specified by the [group\_replication\_start\_on\_boot](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html%22%20%5Cl%20%22sysvar_group_replication_start_on_boot) system variable.

User credentials specified on START GROUP\_REPLICATION take precedence over any user credentials set for the group\_replication\_recovery channel using a [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) statement. Note that user credentials set using [CHANGE MASTER TO](https://dev.mysql.com/doc/refman/8.0/en/change-master-to.html) are stored in the replication metadata repositories, and are used when START GROUP\_REPLICATION is specified without user credentials, including automatic starts if the [group\_replication\_start\_on\_boot](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html%22%20%5Cl%20%22sysvar_group_replication_start_on_boot) system variable is set to ON. To gain the security benefits of specifying user credentials on START GROUP\_REPLICATION, ensure that [group\_replication\_start\_on\_boot](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html%22%20%5Cl%20%22sysvar_group_replication_start_on_boot) is set to OFF (the default is ON), and clear any user credentials previously set for the group\_replication\_recovery channel, following the instructions.

#### STOP GROUP\_REPLICATION Statement

STOP GROUP\_REPLICATION

Stops Group Replication. This statement requires the [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_group-replication-admin) privilege (or the deprecated [SUPER](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_super) privilege). As soon as you issue [STOP GROUP\_REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/stop-group-replication.html) the member is set to [super\_read\_only=ON](https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html%22%20%5Cl%20%22sysvar_super_read_only), which ensures that no writes can be made to the member while Group Replication stops. Any other replication channels running on the member are also stopped. Any user credentials that you specified on the [START GROUP\_REPLICATION](https://dev.mysql.com/doc/refman/8.0/en/start-group-replication.html) statement when starting Group Replication on this member are removed from memory, and must be supplied when you start Group Replication again.

**Warning**

Use this statement with extreme caution because it removes the server instance from the group, meaning it is no longer protected by Group Replication's consistency guarantee mechanisms. To be completely safe, ensure that your applications can no longer connect to the instance before issuing this statement to avoid any chance of stale reads.

####  Function which Configures Group Replication Primary

The following function enables you to configure which member of a single-primary replication group is the primary.

* [group\_replication\_set\_as\_primary()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-new-primary.html%22%20%5Cl%20%22udf_group-replication-set-as-primary)

Appoints a specific member of the group as the new primary, overriding any election process. Pass in ***member\_uuid*** which is the [server\_uuid](https://dev.mysql.com/doc/refman/8.0/en/replication-options.html%22%20%5Cl%20%22sysvar_server_uuid) of the member that you want to become the new primary. Must be issued on a member of a replication group running in single-primary mode.

Syntax:

STRING group\_replication\_set\_as\_primary(member\_uuid)

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT group\_replication\_set\_as\_primary(member\_uuid)

For more information, see [Section 18.4.1.1, “Changing a Group's Primary Member”](https://dev.mysql.com/doc/refman/8.0/en/group-replication-changing-primary-member.html)

####  Functions which Configure the Group Replication Mode

The following functions enable you to control the mode which a replication group is running in, either single-primary or multi-primary mode.

* [group\_replication\_switch\_to\_single\_primary\_mode()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-mode.html%22%20%5Cl%20%22udf_group-replication-switch-to-single-primary-mode)

Changes a group running in multi-primary mode to single-primary mode, without the need to stop Group Replication. Must be issued on a member of a replication group running in multi-primary mode. When you change to single-primary mode, strict consistency checks are also disabled on all group members, as required in single-primary mode ([group\_replication\_enforce\_update\_everywhere\_checks=OFF](https://dev.mysql.com/doc/refman/8.0/en/group-replication-options.html%22%20%5Cl%20%22sysvar_group_replication_enforce_update_everywhere_checks)).

Syntax:

STRING group\_replication\_switch\_to\_single\_primary\_mode([str])

Arguments:

* + ***str***: A string containing the UUID of a member of the group which should become the new single primary. Other members of the group become secondaries.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT group\_replication\_switch\_to\_single\_primary\_mode(member\_uuid);

* [group\_replication\_switch\_to\_multi\_primary\_mode()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-mode.html%22%20%5Cl%20%22udf_group-replication-switch-to-multi-primary-mode)

Changes a group running in single-primary mode to multi-primary mode. Must be issued on a member of a replication group running in single-primary mode.

Syntax:

STRING group\_replication\_switch\_to\_multi\_primary\_mode()

This function has no parameters.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT group\_replication\_switch\_to\_multi\_primary\_mode()

All members which belong to the group become primaries.

#### Functions to Inspect and Configure the Maximum Consensus Instances of a Group

The following functions enable you to inspect and configure the maximum number of consensus instances that a group can execute in parallel.

* [group\_replication\_get\_write\_concurrency()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-maximum-consensus.html%22%20%5Cl%20%22udf_group-replication-get-write-concurrency)

Check the maximum number of consensus instances that a group can execute in parallel.

Syntax:

INT group\_replication\_get\_write\_concurrency()

This function has no parameters.

Return value:

The maximum number of consensus instances currently set for the group.

Example:

SELECT group\_replication\_get\_write\_concurrency()

* [group\_replication\_set\_write\_concurrency()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-maximum-consensus.html%22%20%5Cl%20%22udf_group-replication-set-write-concurrency)

Configures the maximum number of consensus instances that a group can execute in parallel. The [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_group-replication-admin) privilege is required to use this UDF.

Syntax:

STRING group\_replication\_set\_write\_concurrency(instances)

Arguments:

* + ***members***: Sets the maximum number of consensus instances that a group can execute in parallel. Default value is 10, valid values are integers in the range of 10 to 200.

Return value:

Any resulting error as a string.

Example:

SELECT group\_replication\_set\_write\_concurrency(instances);

#### Version

The following functions enable you to inspect and configure the Group Replication communication protocol version that is used by a replication group.

* [group\_replication\_get\_communication\_protocol()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-communication-protocol.html%22%20%5Cl%20%22udf_group-replication-get-communication-protocol)

Inspect the Group Replication communication protocol version that is currently in use for a group.

Syntax:

STRING group\_replication\_get\_communication\_protocol()

This function has no parameters.

Return value:

The oldest MySQL Server version that can join this group and use the group's communication protocol. Versions from MySQL 5.7.14 allow compression of messages, and versions from MySQL 8.0.16 also allow fragmentation of messages. Note that the [group\_replication\_get\_communication\_protocol()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-communication-protocol.html%22%20%5Cl%20%22udf_group-replication-get-communication-protocol) UDF returns the minimum MySQL version that the group supports, which might differ from the version number that was passed to the [group\_replication\_set\_communication\_protocol()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-communication-protocol.html%22%20%5Cl%20%22udf_group-replication-set-communication-protocol) UDF, and from the MySQL Server version that is installed on the member where you use the UDF.

If the protocol cannot be inspected because this server instance does not belong to a replication group, an error is returned as a string.

Example:

SELECT group\_replication\_get\_communication\_protocol();

+------------------------------------------------+

| group\_replication\_get\_communication\_protocol() |

+------------------------------------------------+

| 8.0.16 |

+------------------------------------------------+

* [group\_replication\_set\_communication\_protocol()](https://dev.mysql.com/doc/refman/8.0/en/group-replication-functions-for-communication-protocol.html%22%20%5Cl%20%22udf_group-replication-set-communication-protocol)

Downgrade the Group Replication communication protocol version of a group so that members at earlier releases can join, or upgrade the Group Replication communication protocol version of a group after upgrading MySQL Server on all members. The [GROUP\_REPLICATION\_ADMIN](https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#priv_group-replication-admin) privilege is required to use this UDF, and all existing group members must be online when you issue the statement, with no loss of majority.

**Note**

For MySQL InnoDB cluster, the communication protocol version is managed automatically whenever the cluster topology is changed using AdminAPI operations. You do not have to use these UDFs yourself for an InnoDB cluster.

Syntax:

STRING group\_replication\_set\_communication\_protocol(version)

Arguments:

* + ***version***: For a downgrade, specify the MySQL Server version of the prospective group member that has the oldest installed server version. In this case, the command makes the group fall back to a communication protocol compatible with that server version if possible. The minimum server version that you can specify is MySQL 5.7.14. For an upgrade, specify the new MySQL Server version to which the existing group members have been upgraded.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT group\_replication\_set\_communication\_protocol("5.7.25");