MySQL Replication
Abstract

This is the MySQL Replication extract from the MySQL 5.1 Reference Manual.

For legal information, see the Legal Notices.

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

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Document generated on: 2015-12-22 (revision: 45958)
# Table of Contents

Preface and Legal Notices ........................................................................................................... vii  
1 Replication ............................................................................................................................... 1  
2 Replication Configuration ......................................................................................................... 3  
   2.1 How to Set Up Replication ................................................................................................. 4  
      2.1.1 Setting the Replication Master Configuration ............................................................. 5  
      2.1.2 Setting the Replication Slave Configuration ............................................................... 6  
      2.1.3 Creating a User for Replication ................................................................................... 6  
      2.1.4 Obtaining the Replication Master Binary Log Coordinates ......................................... 7  
      2.1.5 Creating a Data Snapshot Using mysqldump ............................................................... 8  
      2.1.6 Creating a Data Snapshot Using Raw Data Files ....................................................... 8  
      2.1.7 Setting Up Replication with New Master and Slaves ................................................... 10  
      2.1.8 Setting Up Replication with Existing Data ................................................................. 10  
      2.1.9 Introducing Additional Slaves to an Existing Replication Environment ................... 12  
      2.1.10 Setting the Master Configuration on the Slave ......................................................... 13  
   2.2 Replication Formats ............................................................................................................. 13  
      2.2.1 Advantages and Disadvantages of Statement-Based and Row-Based Replication ........ 14  
      2.2.2 Usage of Row-Based Logging and Replication ......................................................... 18  
      2.2.3 Determination of Safe and Unsafe Statements in Binary Logging .............................. 19  
   2.3 Replication and Binary Logging Options and Variables ..................................................... 21  
      2.3.1 Replication and Binary Logging Option and Variable Reference ................................. 22  
      2.3.2 Replication Master Options and Variables .................................................................. 32  
      2.3.3 Replication Slave Options and Variables .................................................................... 35  
      2.3.4 Binary Log Options and Variables ............................................................................. 59  
   2.4 Common Replication Administration Tasks ......................................................................... 72  
      2.4.1 Checking Replication Status ....................................................................................... 72  
      2.4.2 Pausing Replication on the Slave ............................................................................... 74  
   3 Replication Solutions ............................................................................................................. 75  
   3.1 Using Replication for Backups ............................................................................................ 75  
      3.1.1 Backing Up a Slave Using mysqldump ....................................................................... 76  
      3.1.2 Backing Up Raw Data from a Slave .......................................................................... 76  
      3.1.3 Backing Up a Master or Slave by Making It Read Only ............................................. 77  
   3.2 Using Replication with Different Master and Slave Storage Engines ............................... 79  
   3.3 Using Replication for Scale-Out ......................................................................................... 80  
   3.4 Replicating Different Databases to Different Slaves .......................................................... 81  
   3.5 Improving Replication Performance ............................................................................... 83  
   3.6 Switching Masters During Failover .................................................................................. 84  
   3.7 Setting Up Replication Using SSL .................................................................................... 86  
   4 Replication Notes and Tips ................................................................................................... 89  
   4.1 Replication Features and Issues ......................................................................................... 89  
      4.1.1 Replication and AUTO_INCREMENT .......................................................................... 90  
      4.1.2 Replication and BLACKHOLE Tables ....................................................................... 91  
      4.1.3 Replication and Character Sets .................................................................................. 91  
      4.1.4 Replication and CHECKSUM TABLE ........................................................................ 92  
      4.1.5 Replication of CREATE ... IF NOT EXISTS Statements .......................................... 92  
      4.1.6 Replication of CREATE TABLE ... SELECT Statements .......................................... 92  
      4.1.7 Replication of CURRENT_USER() ............................................................................ 93  
      4.1.8 Replication of CREATE SERVER, ALTER SERVER, and DROP SERVER .................. 94  
      4.1.9 Replication of DROP ... IF EXISTS Statements ......................................................... 94  
      4.1.10 Replication with Differing Table Definitions on Master and Slave ............................ 94  
      4.1.11 Replication and DIRECTORY Table Options ............................................................ 100  
      4.1.12 Replication of Invoked Features .............................................................................. 100
# MySQL Replication

4.1.13 Replication and Floating-Point Values .............................................................. 102
4.1.14 Replication and FLUSH ......................................................................................... 102
4.1.15 Replication and System Functions ................................................................. 102
4.1.16 Replication and LIMIT ....................................................................................... 105
4.1.17 Replication and LOAD DATA INFILE ............................................................... 105
4.1.18 Replication and the Slow Query Log ............................................................... 105
4.1.19 Replication and REPAIR TABLE .......................................................................... 105
4.1.20 Replication and Master or Slave Shutdowns .................................................... 105
4.1.21 Replication and max_allowed_packet ............................................................ 106
4.1.22 Replication and MEMORY Tables ...................................................................... 106
4.1.23 Replication and Temporary Tables .................................................................... 107
4.1.24 Replication of the mysql System Database .................................................... 108
4.1.25 Replication and the Query Optimizer .............................................................. 108
4.1.26 Replication and Partitioning ................................................................................ 108
4.1.27 Replication and Reserved Words ......................................................................... 108
4.1.28 SET PASSWORD and Row-Based Replication .................................................. 109
4.1.29 Slave Errors During Replication ....................................................................... 109
4.1.30 Replication of Server-Side Help Tables ............................................................. 109
4.1.31 Replication and Server SQL Mode ...................................................................... 111
4.1.32 Replication Retries and Timeouts ...................................................................... 111
4.1.33 Replication and TIMESTAMP .............................................................................. 112
4.1.34 Replication and Time Zones .............................................................................. 112
4.1.35 Replication and Transactions ............................................................................ 112
4.1.36 Replication and Triggers ..................................................................................... 113
4.1.37 Replication and TRUNCATE TABLE .................................................................... 114
4.1.38 Replication and Variables .................................................................................. 114
4.1.39 Replication and Views ....................................................................................... 116

4.2 Replication Compatibility Between MySQL Versions ............................................ 116
4.3 Upgrading a Replication Setup ................................................................................ 117
4.4 Troubleshooting Replication ................................................................................... 118
4.5 How to Report Replication Bugs or Problems ....................................................... 119

5 Replication Implementation ....................................................................................... 121
5.1 Replication Implementation Details ......................................................................... 121
5.2 Replication Relay and Status Logs ........................................................................... 123
5.2.1 The Slave Relay Log ........................................................................................... 123
5.2.2 Slave Status Logs ............................................................................................... 124
5.3 How Servers Evaluate Replication Filtering Rules ............................................... 126
5.3.1 Evaluation of Database-Level Replication and Binary Logging Options .......... 127
5.3.2 Evaluation of Table-Level Replication Options ............................................... 129
5.3.3 Replication Rule Application .............................................................................. 131

A Licenses for Third-Party Components ..................................................................... 133
A.1 ANTLR 3 License ....................................................................................................... 135
A.2 dtoa.c License .......................................................................................................... 135
A.3 Editline Library (libedit) License ............................................................................ 136
A.4 FindGTest.cmake License ...................................................................................... 138
A.5 Fred Fish's Dbug Library License ........................................................................... 139
A.6 getarg License .......................................................................................................... 140
A.7 GNU General Public License Version 2.0, June 1991 ............................................. 140
A.8 GNU Lesser General Public License Version 2.1, February 1999 ......................... 146
A.9 GNU Libtool License ................................................................................................ 154
A.10 GNU Readline License .......................................................................................... 154
A.11 Google Controlling Master Thread I/O Rate Patch License .................................. 155
A.12 Google PerfTools (TCMalloc utility) License ....................................................... 155
A.13 Google SMP Patch License ................................................................................... 156
Preface and Legal Notices

This is the MySQL Replication extract from the MySQL 5.1 Reference Manual.

Legal Notices

Copyright © 1997, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

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

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

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

Access to Oracle Support

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

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

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.
Chapter 1 Replication

Replication enables data from one MySQL database server (the master) to be replicated to one or more MySQL database servers (the slaves). Replication is asynchronous - slaves need not be connected permanently to receive updates from the master. This means that updates can occur over long-distance connections and even over temporary or intermittent connections such as a dial-up service. Depending on the configuration, you can replicate all databases, selected databases, or even selected tables within a database.

For answers to some questions often asked by those who are new to MySQL Replication, see MySQL 5.1 FAQ: Replication.

The target uses for replication in MySQL include:

- Scale-out solutions - spreading the load among multiple slaves to improve performance. In this environment, all writes and updates must take place on the master server. Reads, however, may take place on one or more slaves. This model can improve the performance of writes (since the master is dedicated to updates), while dramatically increasing read speed across an increasing number of slaves.

- Data security - because data is replicated to the slave, and the slave can pause the replication process, it is possible to run backup services on the slave without corrupting the corresponding master data.

- Analytics - live data can be created on the master, while the analysis of the information can take place on the slave without affecting the performance of the master.

- Long-distance data distribution - if a branch office would like to work with a copy of your main data, you can use replication to create a local copy of the data for their use without requiring permanent access to the master.

Replication in MySQL features support for one-way, asynchronous replication, in which one server acts as the master, while one or more other servers act as slaves. This is in contrast to the synchronous replication which is a characteristic of MySQL Cluster (see MySQL Cluster NDB 6.1 - 7.1).

There are a number of solutions available for setting up replication between two servers, but the best method to use depends on the presence of data and the engine types you are using. For more information on the available options, see Section 2.1, “How to Set Up Replication”.

There are two core types of replication format, Statement Based Replication (SBR), which replicates entire SQL statements, and Row Based Replication (RBR), which replicates only the changed rows. You may also use a third variety, Mixed Based Replication (MBR). For more information on the different replication formats, see Section 2.2, “Replication Formats”. From MySQL 5.1.12 to MySQL 5.1.28, mixed format is the default. Beginning with MySQL 5.1.29, statement-based format is the default.

Replication is controlled through a number of different options and variables. These control the core operation of the replication, timeouts, and the databases and filters that can be applied on databases and tables. For more information on the available options, see Section 2.3, “Replication and Binary Logging Options and Variables”.

You can use replication to solve a number of different problems, including problems with performance, supporting the backup of different databases, and as part of a larger solution to alleviate system failures. For information on how to address these issues, see Chapter 3, Replication Solutions.

For notes and tips on how different data types and statements are treated during replication, including details of replication features, version compatibility, upgrades, and problems and their resolution, including an FAQ, see Chapter 4, Replication Notes and Tips.
For detailed information on the implementation of replication, how replication works, the process and contents of the binary log, background threads and the rules used to decide how statements are recorded and replication, see Chapter 5, *Replication Implementation*. 
Chapter 2 Replication Configuration

Table of Contents

2.1 How to Set Up Replication ................................................................. 4
   2.1.1 Setting the Replication Master Configuration .................................. 5
   2.1.2 Setting the Replication Slave Configuration .................................... 6
   2.1.3 Creating a User for Replication ..................................................... 6
   2.1.4 Obtaining the Replication Master Binary Log Coordinates ..................... 7
   2.1.5 Creating a Data Snapshot Using mysqldump ...................................... 8
   2.1.6 Creating a Data Snapshot Using Raw Data Files ................................. 8
   2.1.7 Setting Up Replication with New Master and Slaves ........................... 10
   2.1.8 Setting Up Replication with Existing Data ....................................... 10
   2.1.9 Introducing Additional Slaves to an Existing Replication Environment ......... 12
   2.1.10 Setting the Master Configuration on the Slave .................................. 13

2.2 Replication Formats ........................................................................... 13
   2.2.1 Advantages and Disadvantages of Statement-Based and Row-Based Replication 14
   2.2.2 Usage of Row-Based Logging and Replication ..................................... 18
   2.2.3 Determination of Safe and Unsafe Statements in Binary Logging ............... 19

2.3 Replication and Binary Logging Options and Variables ......................... 21
   2.3.1 Replication and Binary Logging Option and Variable Reference ............... 22
   2.3.2 Replication Master Options and Variables ....................................... 32
   2.3.3 Replication Slave Options and Variables ......................................... 35
   2.3.4 Binary Log Options and Variables ................................................. 59

2.4 Common Replication Administration Tasks ........................................ 72
   2.4.1 Checking Replication Status ......................................................... 72
   2.4.2 Pausing Replication on the Slave .................................................. 74

Replication between servers in MySQL is based on the binary logging mechanism. The MySQL instance operating as the master (the source of the database changes) writes updates and changes as “events” to the binary log. The information in the binary log is stored in different logging formats according to the database changes being recorded. Slaves are configured to read the binary log from the master and to execute the events in the binary log on the slave's local database.

The master is “dumb” in this scenario. Once binary logging has been enabled, all statements are recorded in the binary log. Each slave receives a copy of the entire contents of the binary log. It is the responsibility of the slave to decide which statements in the binary log should be executed; you cannot configure the master to log only certain events. If you do not specify otherwise, all events in the master binary log are executed on the slave. If required, you can configure the slave to process only events that apply to particular databases or tables.

Each slave keeps a record of the binary log coordinates: The file name and position within the file that it has read and processed from the master. This means that multiple slaves can be connected to the master and executing different parts of the same binary log. Because the slaves control this process, individual slaves can be connected and disconnected from the server without affecting the master's operation. Also, because each slave remembers the position within the binary log, it is possible for slaves to be disconnected, reconnect and then “catch up” by continuing from the recorded position.

Both the master and each slave must be configured with a unique ID (using the server-id [21] option). In addition, each slave must be configured with information about the master host name, log file name, and position within that file. These details can be controlled from within a MySQL session using the CHANGE MASTER TO statement on the slave. The details are stored within the slave's master.info file.
How to Set Up Replication

This section describes the setup and configuration required for a replication environment, including step-by-step instructions for creating a new replication environment. The major components of this section are:

- For a guide to setting up two or more servers for replication, Section 2.1, “How to Set Up Replication”, deals with the configuration of the systems and provides methods for copying data between the master and slaves.

- Events in the binary log are recorded using a number of formats. These are referred to as statement-based replication (SBR) or row-based replication (RBR). A third type, mixed-format replication (MIXED), uses SBR or RBR replication automatically to take advantage of the benefits of both SBR and RBR formats when appropriate. The different formats are discussed in Section 2.2, “Replication Formats”.

- Detailed information on the different configuration options and variables that apply to replication is provided in Section 2.3, “Replication and Binary Logging Options and Variables”.

- Once started, the replication process should require little administration or monitoring. However, for advice on common tasks that you may want to execute, see Section 2.4, “Common Replication Administration Tasks”.

2.1 How to Set Up Replication

This section describes how to set up complete replication of a MySQL server. There are a number of different methods for setting up replication, and the exact method to use depends on how you are setting up replication, and whether you already have data within your master database.

There are some generic tasks that are common to all replication setups:

- On the master, you must enable binary logging and configure a unique server ID. This might require a server restart. See Section 2.1.1, “Setting the Replication Master Configuration”.

- On each slave that you want to connect to the master, you must configure a unique server ID. This might require a server restart. See Section 2.1.2, “Setting the Replication Slave Configuration”.

- You may want to create a separate user that will be used by your slaves to authenticate with the master to read the binary log for replication. The step is optional. See Section 2.1.3, “Creating a User for Replication”.

- Before creating a data snapshot or starting the replication process, you should record the position of the binary log on the master. You will need this information when configuring the slave so that the slave knows where within the binary log to start executing events. See Section 2.1.4, “Obtaining the Replication Master Binary Log Coordinates”.

- If you already have data on your master and you want to use it to synchronize your slave, you will need to create a data snapshot. You can create a snapshot using mysqldump (see Section 2.1.5, “Creating a Data Snapshot Using mysqldump”) or by copying the data files directly (see Section 2.1.6, “Creating a Data Snapshot Using Raw Data Files”).

- You will need to configure the slave with settings for connecting to the master, such as the host name, login credentials, and binary log file name and position. See Section 2.1.10, “Setting the Master Configuration on the Slave”.

Once you have configured the basic options, you will need to follow the instructions for your replication setup. A number of alternatives are provided:

- If you are establishing a new MySQL master and one or more slaves, you need only set up the configuration, as you have no data to exchange. For guidance on setting up replication in this situation, see Section 2.1.7, “Setting Up Replication with New Master and Slaves”.
• If you are already running a MySQL server, and therefore already have data that must be transferred to your slaves before replication starts, have not previously configured the binary log and are able to shut down your MySQL server for a short period during the process, see Section 2.1.8, “Setting Up Replication with Existing Data”.

• If you are adding slaves to an existing replication environment, you can set up the slaves without affecting the master. See Section 2.1.9, “Introducing Additional Slaves to an Existing Replication Environment”.

If you will be administering MySQL replication servers, we suggest that you read this entire chapter through and try all statements mentioned in SQL Statements for Controlling Master Servers, and SQL Statements for Controlling Slave Servers. You should also familiarize yourself with the replication startup options described in Section 2.3, “Replication and Binary Logging Options and Variables”.

Note

Note that certain steps within the setup process require the SUPER privilege. If you do not have this privilege, it might not be possible to enable replication.

2.1.1 Setting the Replication Master Configuration

On a replication master, you must enable binary logging and establish a unique server ID. If this has not already been done, this part of master setup requires a server restart.

Binary logging must be enabled on the master because the binary log is the basis for sending data changes from the master to its slaves. If binary logging is not enabled, replication will not be possible.

Each server within a replication group must be configured with a unique server ID. This ID is used to identify individual servers within the group, and must be a positive integer between 1 and \(2^{32} - 1\). How you organize and select the numbers is entirely up to you.

To configure the binary log and server ID options, you will need to shut down your MySQL server and edit the my.cnf or my.ini file. Add the following options to the configuration file within the [mysqld] section. If these options already exist, but are commented out, uncomment the options and alter them according to your needs. For example, to enable binary logging using a log file name prefix of mysql-bin, and configure a server ID of 1, use these lines:

```plaintext
[mysqld]
log-bin=mysql-bin
server-id=1
```

After making the changes, restart the server.

Note

If you omit server-id [21] (or set it explicitly to its default value of 0), a master refuses connections from all slaves.

Note

For the greatest possible durability and consistency in a replication setup using InnoDB with transactions, you should use `innodb_flush_log_at_trx_commit=1` and `sync_binlog=1` in the master my.cnf file.
2.1.2 Setting the Replication Slave Configuration

On a replication slave, you must establish a unique server ID. If this has not already been done, this part of slave setup requires a server restart.

If the slave server ID is not already set, or the current value conflicts with the value that you have chosen for the master server, you should shut down your slave server and edit the configuration to specify a unique server ID. For example:

```plaintext
[mysqld]
server-id=2
```

After making the changes, restart the server.

If you are setting up multiple slaves, each one must have a unique server-id value that differs from that of the master and from each of the other slaves. Think of server-id values as something similar to IP addresses: These IDs uniquely identify each server instance in the community of replication partners.

Note

If you omit server-id (or set it explicitly to its default value of 0), a slave refuses to connect to a master.

You do not have to enable binary logging on the slave for replication to be enabled. However, if you enable binary logging on the slave, you can use the binary log for data backups and crash recovery on the slave, and also use the slave as part of a more complex replication topology (for example, where the slave acts as a master to other slaves).

2.1.3 Creating a User for Replication

Each slave must connect to the master using a MySQL user name and password, so there must be a user account on the master that the slave can use to connect. Any account can be used for this operation, providing it has been granted the REPLICATION SLAVE privilege. You may wish to create a different account for each slave, or connect to the master using the same account for each slave.

You need not create an account specifically for replication. However, you should be aware that the user name and password will be stored in plain text within the master.info file (see Section 5.2.2, "Slave Status Logs"). Therefore, you may want to create a separate account that has privileges only for the replication process, to minimize the possibility of compromise to other accounts.

To create a new account, use CREATE USER. To grant this account the privileges required for replication, use the GRANT statement. If you create an account solely for the purposes of replication, that account needs only the REPLICATION SLAVE privilege. For example, to set up a new user, repl, that can connect for replication from any host within the mydomain.com domain, issue these statements on the master:

```sql
mysql> CREATE USER 'repl'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
```
Obtaining the Replication Master Binary Log Coordinates

mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%.mydomain.com';

See Account Management Statements, for more information on statements for manipulation of user accounts.

2.1.4 Obtaining the Replication Master Binary Log Coordinates

To configure replication on the slave you must determine the master's current coordinates within its binary log. You will need this information so that when the slave starts the replication process, it is able to start processing events from the binary log at the correct point.

If you have existing data on your master that you want to synchronize on your slaves before starting the replication process, you must stop processing statements on the master, and then obtain its current binary log coordinates and dump its data, before permitting the master to continue executing statements. If you do not stop the execution of statements, the data dump and the master status information that you use will not match and you will end up with inconsistent or corrupted databases on the slaves.

To obtain the master binary log coordinates, follow these steps:

1. Start a session on the master by connecting to it with the command-line client, and flush all tables and block write statements by executing the `FLUSH TABLES WITH READ LOCK` statement:

   mysql> FLUSH TABLES WITH READ LOCK;

   For InnoDB tables, note that `FLUSH TABLES WITH READ LOCK` also blocks COMMIT operations.

   Warning
   Leave the client from which you issued the `FLUSH TABLES` statement running so that the read lock remains in effect. If you exit the client, the lock is released.

2. In a different session on the master, use the `SHOW MASTER STATUS` statement to determine the current binary log file name and position:

   mysql > SHOW MASTER STATUS;

   +------------------+----------+--------------+------------------+
   | File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
   +------------------+----------+--------------+------------------+
   | mysql-bin.000003 | 73       | test         | manual,mysql     |
   +------------------+----------+--------------+------------------+

   The File column shows the name of the log file and Position shows the position within the file. In this example, the binary log file is `mysql-bin.000003` and the position is 73. Record these values. You need them later when you are setting up the slave. They represent the replication coordinates at which the slave should begin processing new updates from the master.

   If the master has been running previously without binary logging enabled, the log file name and position values displayed by `SHOW MASTER STATUS` or `mysqldump --master-data` will be empty. In that case, the values that you need to use later when specifying the slave's log file and position are the empty string (`''`) and 4.

   You now have the information you need to enable the slave to start reading from the binary log in the correct place to start replication.

   If you have existing data that needs be to synchronized with the slave before you start replication, leave the client running so that the lock remains in place and then proceed to Section 2.1.5, “Creating a Data
Creating a Data Snapshot Using mysqldump

Snapshot Using mysqldump”, or Section 2.1.6, “Creating a Data Snapshot Using Raw Data Files”. The idea here is to prevent any further changes so that the data copied to the slaves is in synchrony with the master.

If you are setting up a brand new master and slave replication group, you can exit the first session to release the read lock.

2.1.5 Creating a Data Snapshot Using mysqldump

One way to create a snapshot of the data in an existing master database is to use the mysqldump tool to create a dump of all the databases you want to replicate. Once the data dump has been completed, you then import this data into the slave before starting the replication process.

The example shown here dumps all databases to a file named dbdump.db, and includes the --master-data option which automatically appends the CHANGE MASTER TO statement required on the slave to start the replication process:

\[ \text{shell> } \text{mysqldump --all-databases --master-data > dbdump.db} \]

If you do not use --master-data, then it is necessary to lock all tables in a separate session manually (using FLUSH TABLES WITH READ LOCK) prior to running mysqldump, then exiting or running UNLOCK TABLES from the second session to release the locks. You must also obtain binary log position information matching the snapshot, using SHOW MASTER STATUS, and use this to issue the appropriate CHANGE MASTER TO statement when starting the slave.

When choosing databases to include in the dump, remember that you need to filter out databases on each slave that you do not want to include in the replication process.

To import the data, either copy the dump file to the slave, or access the file from the master when connecting remotely to the slave.

2.1.6 Creating a Data Snapshot Using Raw Data Files

If your database is large, copying the raw data files can be more efficient than using mysqldump and importing the file on each slave. This technique skips the overhead of updating indexes as the INSERT statements are replayed.

Using this method with tables in storage engines with complex caching or logging algorithms requires extra steps to produce a perfect “point in time” snapshot: the initial copy command might leave out cache information and logging updates, even if you have acquired a global read lock. How the storage engine responds to this depends on its crash recovery abilities.

This method also does not work reliably if the master and slave have different values for ft_stopword_file, ft_min_word_len, or ft_max_word_len and you are copying tables having full-text indexes.

If you use InnoDB tables, you can use the mysqlbackup command from the MySQL Enterprise Backup component to produce a consistent snapshot. This command records the log name and offset corresponding to the snapshot to be later used on the slave. MySQL Enterprise Backup is a commercial product that is included as part of a MySQL Enterprise subscription. See MySQL Enterprise Backup Overview for detailed information.

Otherwise, use the cold backup technique to obtain a reliable binary snapshot of InnoDB tables: copy all data files after doing a slow shutdown of the MySQL Server.
To create a raw data snapshot of MyISAM tables, you can use standard copy tools such as cp or copy, a remote copy tool such as scp or rsync, an archiving tool such as zip or tar, or a file system snapshot tool such as dump, providing that your MySQL data files exist on a single file system. If you are replicating only certain databases, copy only those files that relate to those tables. (For InnoDB, all tables in all databases are stored in the system tablespace files, unless you have the innodb_file_per_table option enabled.)

You might want to specifically exclude the following files from your archive:

- Files relating to the mysql database.
- The master.info file.
- The master's binary log files.
- Any relay log files.

To get the most consistent results with a raw data snapshot, shut down the master server during the process, as follows:

1. Acquire a read lock and get the master's status. See Section 2.1.4, “Obtaining the Replication Master Binary Log Coordinates”.
2. In a separate session, shut down the master server:

```
shell> mysqladmin shutdown
```
3. Make a copy of the MySQL data files. The following examples show common ways to do this. You need to choose only one of them:

```
shell> tar cf /tmp/db.tar ./data
shell> zip -r /tmp/db.zip ./data
shell> rsync --recursive ./data /tmp/dbdata
```
4. Restart the master server.

If you are not using InnoDB tables, you can get a snapshot of the system from a master without shutting down the server as described in the following steps:

1. Acquire a read lock and get the master's status. See Section 2.1.4, “Obtaining the Replication Master Binary Log Coordinates”.
2. Make a copy of the MySQL data files. The following examples show common ways to do this. You need to choose only one of them:

```
shell> tar cf /tmp/db.tar ./data
shell> zip -r /tmp/db.zip ./data
shell> rsync --recursive ./data /tmp/dbdata
```
3. In the client where you acquired the read lock, release the lock:

```
mysql> UNLOCK TABLES;
```

Once you have created the archive or copy of the database, copy the files to each slave before starting the slave replication process.
2.1.7 Setting Up Replication with New Master and Slaves

The easiest and most straightforward method for setting up replication is to use new master and slave servers.

You can also use this method if you are setting up new servers but have an existing dump of the databases from a different server that you want to load into your replication configuration. By loading the data into a new master, the data will be automatically replicated to the slaves.

To set up replication between a new master and slave:

1. Configure the MySQL master with the necessary configuration properties. See Section 2.1.1, “Setting the Replication Master Configuration”.
2. Start up the MySQL master.
3. Set up a user. See Section 2.1.3, “Creating a User for Replication”.
4. Obtain the master status information. See Section 2.1.4, “Obtaining the Replication Master Binary Log Coordinates”.
5. On the master, release the read lock:

   mysql> UNLOCK TABLES;

6. On the slave, edit the MySQL configuration. See Section 2.1.2, “Setting the Replication Slave Configuration”.
7. Start up the MySQL slave.
8. Execute a `CHANGE MASTER TO` statement to set the master replication server configuration. See Section 2.1.10, “Setting the Master Configuration on the Slave”.

Perform the slave setup steps on each slave.

Because there is no data to load or exchange on a new server configuration you do not need to copy or import any information.

If you are setting up a new replication environment using the data from a different existing database server, you will now need to run the dump file generated from that server on the new master. The database updates will automatically be propagated to the slaves:

```
shell> mysql -h master < fulldb.dump
```

2.1.8 Setting Up Replication with Existing Data

When setting up replication with existing data, you will need to decide how best to get the data from the master to the slave before starting the replication service.

The basic process for setting up replication with existing data is as follows:

1. With the MySQL master running, create a user to be used by the slave when connecting to the master during replication. See Section 2.1.3, “Creating a User for Replication”.
2. If you have not already configured the `server-id` and enabled binary logging on the master server, you will need to shut it down to configure these options. See Section 2.1.1, “Setting the Replication Master Configuration”.
If you have to shut down your master server, this is a good opportunity to take a snapshot of its databases. You should obtain the master status (see Section 2.1.4, “Obtaining the Replication Master Binary Log Coordinates”) before taking down the master, updating the configuration and taking a snapshot. For information on how to create a snapshot using raw data files, see Section 2.1.6, “Creating a Data Snapshot Using Raw Data Files”.

3. If your master server is already correctly configured, obtain its status (see Section 2.1.4, “Obtaining the Replication Master Binary Log Coordinates”) and then use \texttt{mysqldump} to take a snapshot (see Section 2.1.5, “Creating a Data Snapshot Using \texttt{mysqldump}”) or take a raw snapshot of the live server using the guide in Section 2.1.6, “Creating a Data Snapshot Using Raw Data Files”.

4. Update the configuration of the slave. See Section 2.1.2, “Setting the Replication Slave Configuration”.

5. The next step depends on how you created the snapshot of data on the master.

   If you used \texttt{mysqldump}:
   
   a. Start the slave, using the \texttt{--skip-slave-start} option so that replication does not start.
   
   b. Import the dump file:

   ```shell
   mysql < fulldb.dump
   ```

   If you created a snapshot using the raw data files:
   
   a. Extract the data files into your slave data directory. For example:

   ```shell
   tar xvf dbdump.tar
   ```

   You may need to set permissions and ownership on the files so that the slave server can access and modify them.

   b. Start the slave, using the \texttt{--skip-slave-start} option so that replication does not start.

6. Configure the slave with the replication coordinates from the master. This tells the slave the binary log file and position within the file where replication needs to start. Also, configure the slave with the login credentials and host name of the master. For more information on the \texttt{CHANGE MASTER TO} statement required, see Section 2.1.10, “Setting the Master Configuration on the Slave”.

7. Start the slave threads:

   ```mysql
   START SLAVE;
   ```

After you have performed this procedure, the slave should connect to the master and catch up on any updates that have occurred since the snapshot was taken.

If you have forgotten to set the \texttt{server-id [21]} option for the master, slaves cannot connect to it.

If you have forgotten to set the \texttt{server-id [21]} option for the slave, you get the following error in the slave's error log:

```sql
Warning: You should set server-id to a non-0 value if master_host is set; we will force server id to 2, but this MySQL server will not act as a slave.
```
Introducing Additional Slaves to an Existing Replication Environment

You also find error messages in the slave's error log if it is not able to replicate for any other reason.

Once a slave is replicating, you can find in its data directory one file named `master.info` and another named `relay-log.info`. The slave uses these two files to keep track of how much of the master's binary log it has processed. Do not remove or edit these files unless you know exactly what you are doing and fully understand the implications. Even in that case, it is preferred that you use the `CHANGE MASTER TO` statement to change replication parameters. The slave will use the values specified in the statement to update the status files automatically.

**Note**

The content of `master.info` overrides some of the server options specified on the command line or in `my.cnf`. See Section 2.3, "Replication and Binary Logging Options and Variables", for more details.

A single snapshot of the master suffices for multiple slaves. To set up additional slaves, use the same master snapshot and follow the slave portion of the procedure just described.

### 2.1.9 Introducing Additional Slaves to an Existing Replication Environment

To add another slave to an existing replication configuration, you can do so without stopping the master. Instead, set up the new slave by making a copy of an existing slave, except that you configure the new slave with a different `server-id` value.

To duplicate an existing slave:

1. Shut down the existing slave:

   ```
   shell> mysqladmin shutdown
   ```

2. Copy the data directory from the existing slave to the new slave. You can do this by creating an archive using `tar` or `WinZip`, or by performing a direct copy using a tool such as `cp` or `rsync`. Ensure that you also copy the log files and relay log files.

A common problem that is encountered when adding new replication slaves is that the new slave fails with a series of warning and error messages like these:

```
071118 16:44:10 [Warning] Neither --relay-log nor --relay-log-index were used; so replication may break when this MySQL server acts as a slave and has his hostname changed!! Please use '--relay-log=new_slave_hostname-relay-bin' to avoid this problem.
071118 16:44:10 [ERROR] Failed to open the relay log './old_slave_hostname-relay-bin.003525' (relay_log_pos 22940879)
071118 16:44:10 [ERROR] Could not find target log during relay log initialization
071118 16:44:10 [ERROR] Failed to initialize the master info structure
```

This is due to the fact that, if the `--relay-log` option is not specified, the relay log files contain the host name as part of their file names. (This is also true of the relay log index file if the `--relay-log-index` option is not used. See Section 2.3, "Replication and Binary Logging Options and Variables", for more information about these options.)

To avoid this problem, use the same value for `--relay-log` on the new slave that was used on the existing slave. (If this option was not set explicitly on the existing slave, use `existing_slave_hostname-relay-bin`.) If this is not feasible, copy the existing slave's relay log index file to the new slave and set the `--relay-log-index` option on the new slave to match what was used on the existing slave. (If this option was not set explicitly on the existing slave, use `existing_slave_hostname-relay-bin.index`.) Alternatively—if you have already tried to start
the new slave (after following the remaining steps in this section) and have encountered errors like those described previously—then perform the following steps:

a. If you have not already done so, issue a `STOP SLAVE` on the new slave.

   If you have already started the existing slave again, issue a `STOP SLAVE` on the existing slave as well.

b. Copy the contents of the existing slave’s relay log index file into the new slave’s relay log index file, making sure to overwrite any content already in the file.

c. Proceed with the remaining steps in this section.

3. Copy the `master.info` and `relay-log.info` files from the existing slave to the new slave if they were not located in the data directory. These files hold the current log coordinates for the master’s binary log and the slave’s relay log.

4. Start the existing slave.

5. On the new slave, edit the configuration and give the new slave a unique `server-id` not used by the master or any of the existing slaves.

6. Start the new slave. The slave will use the information in its `master.info` file to start the replication process.

2.1.10 Setting the Master Configuration on the Slave

To set up the slave to communicate with the master for replication, you must tell the slave the necessary connection information. To do this, execute the following statement on the slave, replacing the option values with the actual values relevant to your system:

```sql
mysql> CHANGE MASTER TO
  -> MASTER_HOST='master_host_name',
  -> MASTER_USER='replication_user_name',
  -> MASTER_PASSWORD='replication_password',
  -> MASTER_LOG_FILE='recorded_log_file_name',
  -> MASTER_LOG_POS=recorded_log_position;
```

Note

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

The `CHANGE MASTER TO` statement has other options as well. For example, it is possible to set up secure replication using SSL. For a full list of options, and information about the maximum permissible length for the string-valued options, see `CHANGE MASTER TO Syntax`.

2.2 Replication Formats

Replication works because events written to the binary log are read from the master and then processed on the slave. The events are recorded within the binary log in different formats according to the type of event. The different replication formats used correspond to the binary logging format used when the events were recorded in the master’s binary log. The correlation between binary logging formats and the terms used during replication are:

- Replication capabilities in MySQL originally were based on propagation of SQL statements from master to slave. This is called `statement-based replication` (often abbreviated as `SBR`), which corresponds to
Advantages and Disadvantages of Statement-Based and Row-Based Replication

the standard statement-based binary logging format. In MySQL 5.1.4 and earlier, binary logging and replication used this format exclusively.

- Row-based binary logging logs changes in individual table rows. When used with MySQL replication, this is known as row-based replication (often abbreviated as RBR). In row-based replication, the master writes events to the binary log that indicate how individual table rows are changed.

- As of MySQL 5.1.8, the server can change the binary logging format in real time according to the type of event using mixed-format logging.

  When the mixed format is in effect, statement-based logging is used by default, but automatically switches to row-based logging in particular cases as described later. Replication using the mixed format is often referred to as mixed-based replication or mixed-format replication. For more information, see Mixed Binary Logging Format.

From MySQL 5.1.12 to MySQL 5.1.28, mixed format is the default. Beginning with MySQL 5.1.29, statement-based format is the default.

**MySQL Cluster.** The default binary logging format in all MySQL Cluster NDB 6.x and 7.x releases is ROW. MySQL Cluster Replication always uses row-based replication, and the NDBCLUSTER storage engine is incompatible with statement-based replication. Using NDBCLUSTER sets row-based logging format automatically. See General Requirements for MySQL Cluster Replication, for more information.

Starting with MySQL 5.1.20, when using MIXED format, the binary logging format is determined in part by the storage engine being used and the statement being executed. For more information on mixed-format logging and the rules governing the support of different logging formats, see Mixed Binary Logging Format.

The logging format in a running MySQL server is controlled by setting the binlog_format server system variable. This variable can be set with session or global scope. The rules governing when and how the new setting takes effect are the same as for other MySQL server system variables—setting the variable for the current session lasts only until the end of that session, and the change is not visible to other sessions; setting the variable globally requires a restart of the server to take effect. For more information, see SET Syntax.

There are conditions under which you cannot change the binary logging format at runtime or doing so causes replication to fail. See Setting The Binary Log Format.

You must have the SUPER privilege to set the global binlog_format value. Starting with MySQL 5.1.29, you must have the SUPER privilege to set either the global or session binlog_format value. (Bug #39106)

The statement-based and row-based replication formats have different issues and limitations. For a comparison of their relative advantages and disadvantages, see Section 2.2.1, "Advantages and Disadvantages of Statement-Based and Row-Based Replication".

With statement-based replication, you may encounter issues with replicating stored routines or triggers. You can avoid these issues by using row-based replication instead. For more information, see Binary Logging of Stored Programs.

2.2.1 Advantages and Disadvantages of Statement-Based and Row-Based Replication

Each binary logging format has advantages and disadvantages. For most users, the mixed replication format should provide the best combination of data integrity and performance. If, however, you want to take advantage of the features specific to the statement-based or row-based replication format when
performing certain tasks, you can use the information in this section, which provides a summary of their relative advantages and disadvantages, to determine which is best for your needs.

- Advantages of statement-based replication
- Disadvantages of statement-based replication
- Advantages of row-based replication
- Disadvantages of row-based replication

**Advantages of statement-based replication**

- Proven technology that has existed in MySQL since 3.23.
- Less data written to log files. When updates or deletes affect many rows, this results in much less storage space required for log files. This also means that taking and restoring from backups can be accomplished more quickly.
- Log files contain all statements that made any changes, so they can be used to audit the database.

**Disadvantages of statement-based replication**

- **Statements that are unsafe for SBR.**
  Not all statements which modify data (such as `INSERT`, `DELETE`, `UPDATE`, and `REPLACE` statements) can be replicated using statement-based replication. Any nondeterministic behavior is difficult to replicate when using statement-based replication. Examples of such DML (Data Modification Language) statements include the following:

  - A statement that depends on a UDF or stored program that is nondeterministic, since the value returned by such a UDF or stored program or depends on factors other than the parameters supplied to it. (Row-based replication, however, simply replicates the value returned by the UDF or stored program, so its effect on table rows and data is the same on both the master and slave.) See Section 4.1.12, “Replication of Invoked Features”, for more information.
  
  - `DELETE` and `UPDATE` statements that use a `LIMIT` clause without an `ORDER BY` are nondeterministic. See Section 4.1.16, “Replication and LIMIT”.

  - Statements using any of the following functions cannot be replicated properly using statement-based replication:
    - `LOAD_FILE()`
    - `UUID()`, `UUID_SHORT()`
    - `USER()`
    - `FOUND_ROWS()`
    - `SYSDATE()` (unless both the master and the slave are started with the `--sysdate-is-now` option)
    - `GET_LOCK()`
    - `IS_FREE_LOCK()`
    - `IS_USED_LOCK()`
    - `MASTER_POS_WAIT()`
Advantages and Disadvantages of Statement-Based and Row-Based Replication

- **RAND()**
- **RELEASE_LOCK()**
- **SLEEP()**
- **VERSION()**

However, all other functions are replicated correctly using statement-based replication, including **NOW()** and so forth.

For more information, see Section 4.1.15, “Replication and System Functions”.

Statements that cannot be replicated correctly using statement-based replication are logged with a warning like the one shown here:

```
[Warning] Statement is not safe to log in statement format.
```

A similar warning is also issued to the client in such cases. The client can display it using **SHOW WARNINGS**.

- **INSERT ... SELECT** requires a greater number of row-level locks than with row-based replication.
- **UPDATE** statements that require a table scan (because no index is used in the **WHERE** clause) must lock a greater number of rows than with row-based replication.
- For **InnoDB**: An **INSERT** statement that uses **AUTO_INCREMENT** blocks other nonconflicting **INSERT** statements.
- For complex statements, the statement must be evaluated and executed on the slave before the rows are updated or inserted. With row-based replication, the slave only has to modify the affected rows, not execute the full statement.
- If there is an error in evaluation on the slave, particularly when executing complex statements, statement-based replication may slowly increase the margin of error across the affected rows over time. See Section 4.1.29, “Slave Errors During Replication”.
- Stored functions execute with the same **NOW()** value as the calling statement. However, this is not true of stored procedures.
- Deterministic UDFs must be applied on the slaves.
- Table definitions must be (nearly) identical on master and slave. See Section 4.1.10, “Replication with Differing Table Definitions on Master and Slave”, for more information.

**Advantages of Row-Based Replication**

- All changes can be replicated. This is the safest form of replication.

For MySQL versions earlier than 5.1.14, DDL (Data Definition Language) statements such as **CREATE TABLE** are replicated using statement-based replication, while DML statements, as well as **GRANT** and **REVOKE** statements, are replicated using row-based replication.

In MySQL 5.1.14 and later, the **mysql** database is not replicated. The **mysql** database is instead seen as a node-specific database. Row-based replication is not supported on tables in this database. Instead, statements that would normally update this information—such as **GRANT, REVOKE** and the manipulation
of triggers, stored routines (including stored procedures), and views—are all replicated to slaves using statement-based replication.

For statements such as `CREATE TABLE ... SELECT`, a `CREATE` statement is generated from the table definition and replicated using statement-based format, while the row insertions are replicated using row-based format.

- The technology is the same as in most other database management systems; knowledge about other systems transfers to MySQL.
- Fewer row locks are required on the master, which thus achieves higher concurrency, for the following types of statements:
  - `INSERT ... SELECT`
  - `INSERT` statements with `AUTO_INCREMENT`
  - `UPDATE` or `DELETE` statements with `WHERE` clauses that do not use keys or do not change most of the examined rows.
- Fewer row locks are required on the slave for any `INSERT`, `UPDATE`, or `DELETE` statement.

### Disadvantages of Row-Based Replication

- RBR tends to generate more data that must be logged. To replicate a DML statement (such as an `UPDATE` or `DELETE` statement), statement-based replication writes only the statement to the binary log. By contrast, row-based replication writes each changed row to the binary log. If the statement changes many rows, row-based replication may write significantly more data to the binary log; this is true even for statements that are rolled back. This also means that taking and restoring from backup can require more time. In addition, the binary log is locked for a longer time to write the data, which may cause concurrency problems.
- Deterministic UDFs that generate large `BLOB` values take longer to replicate with row-based replication than with statement-based replication. This is because the `BLOB` column value is logged, rather than the statement generating the data.
- You cannot examine the logs to see what statements were executed, nor can you see on the slave what statements were received from the master and executed.

However, beginning with MySQL 5.1.29, you can see what data was changed using `mysqlbinlog` with the options `--base64-output=DECODE-ROWS` and `--verbose`.

- For tables using the `MyISAM` storage engine, a stronger lock is required on the slave for `INSERT` statements when applying them as row-based events to the binary log than when applying them as statements. This means that concurrent inserts on `MyISAM` tables are not supported when using row-based replication.
- Formerly, when performing a bulk operation that includes nontransactional storage engines, changes were applied as the statement executed. With row-based logging, this meant that the binary log was written while the statement was running. On the master, this does not cause problems with concurrency, because tables are locked until the bulk operation terminates. On the slave server, tables were not locked while the slave applied changes, because the slave did not know that those changes were part of a bulk operation.

In such cases, if you retrieved data from a table on the master (for example, using `SELECT * FROM table_name`), the server waited for the bulk operation to complete before executing the `SELECT` statement, because the table was read-locked. On the slave, the server did not wait (because there was
no lock). This meant that, until the bulk operation on the slave completed, different results were obtained for the same SELECT query on the master and on the slave.

This issue was resolved in MySQL 5.1.24. (Bug #29020)

### 2.2.2 Usage of Row-Based Logging and Replication

Major changes in the replication environment and in the behavior of applications can result from using row-based logging (RBL) or row-based replication (RBR) rather than statement-based logging or replication. This section describes a number of issues known to exist when using row-based logging or replication, and discusses some best practices for taking advantage of row-based logging and replication.

For additional information, see Section 2.2, “Replication Formats”, and Section 2.2.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”.

For information about issues specific to MySQL Cluster Replication (which depends on row-based replication), see Known Issues in MySQL Cluster Replication.

- **RBL, RBR, and temporary tables.** As noted in Section 4.1.23, “Replication and Temporary Tables”, temporary tables are not replicated when using row-based format. When mixed format is in effect, “safe” statements involving temporary tables are logged using statement-based format. For more information, see Section 2.2.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”.

  Temporary tables are not replicated when using row-based format because there is no need. In addition, because temporary tables can be read only from the thread which created them, there is seldom if ever any benefit obtained from replicating them, even when using statement-based format.

- **RBL and the BLACKHOLE storage engine.** Prior to MySQL 5.1.29, DELETE and UPDATE statements for BLACKHOLE tables did not work with RBL. (Bug #38360)

- **RBL and synchronization of nontransactional tables.** When many rows are affected, the set of changes is split into several events; when the statement commits, all of these events are written to the binary log if the statement is an initial nontransactional statement (occurring in the transaction before any transactional statements). When executing on the slave, a table lock is taken on all tables involved, and then the rows are applied in batch mode. (This may or may not be effective, depending on the engine used for the slave’s copy of the table.)

- **Latency and binary log size.** Because RBL writes changes for each row to the binary log, its size can increase quite rapidly. In a replication environment, this can significantly increase the time required to make changes on the slave that match those on the master. You should be aware of the potential for this delay in your applications.

- **Reading the binary log.** mysqlbinlog displays row-based events in the binary log using the BINLOG statement (see BINLOG Syntax). This statement displays an event in printable form, but as a base 64-encoded string the meaning of which is not evident. As of MySQL 5.1.28, when invoked with the --base64-output=DECODE-ROWS and --verbose options, mysqlbinlog formats the contents of the binary log in a manner that is easily human readable. This is helpful when binary log events were written in row-based format if you want to read or recover from a replication or database failure using the contents of the binary log. For more information, see mysqlbinlog Row Event Display.

- **Binary log execution errors and slave_exec_mode.** If slave_exec_mode is IDEMPOTENT, a failure to apply changes from RBL because the original row cannot be found does not trigger an error or cause replication to fail. This means that it is possible that updates are not applied on the slave, so that the master and slave are no longer synchronized. Latency issues and use of nontransactional tables with RBR when slave_exec_mode is IDEMPOTENT can cause the master and slave to diverge even further. For more information about slave_exec_mode, see Server System Variables.
Determination of Safe and Unsafe Statements in Binary Logging

Note

slave_exec_mode=IDEMPOTENT is generally useful only for circular replication or multi-master replication with MySQL Cluster, for which IDEMPOTENT is the default value (see MySQL Cluster Replication).

For other scenarios, setting slave_exec_mode to STRICT is normally sufficient; this is the default value for storage engines other than NDB.

• Lack of binary log checksums. RBL uses no checksums. This means that network, disk, and other errors may not be identified when processing the binary log. To ensure that data is transmitted without network corruption, you may want to consider using SSL, which adds another layer of checksumming, for replication connections. The CHANGE MASTER TO statement has options to enable replication over SSL. See also CHANGE MASTER TO Syntax, for general information about setting up MySQL with SSL.

• Filtering based on server ID not supported. A common practice is to filter out changes on some slaves by using a WHERE clause that includes the relation @@server_id <> id_value clause with UPDATE and DELETE statements, a simple example of such a clause being WHERE @@server_id <> 1. However, this does not work correctly with row-based logging. If you must use the server_id system variable for statement filtering, you must also use --binlog_format=STATEMENT.

• Database-level replication options. The effects of the --replicate-do-db, --replicate-ignore-db, and --replicate-rewrite-db options differ considerably depending on whether row-based or statement-based logging is used. Because of this, it is recommended to avoid database-level options and instead use table-level options such as --replicate-do-table and --replicate-ignore-table. For more information about these options and the impact that your choice of replication format has on how they operate, see Section 2.3, “Replication and Binary Logging Options and Variables”.

• RBL, nontransactional tables, and stopped slaves. When using row-based logging, if the slave server is stopped while a slave thread is updating a nontransactional table, the slave database may reach an inconsistent state. For this reason, it is recommended that you use a transactional storage engine such as InnoDB for all tables replicated using the row-based format.

Use of STOP SLAVE (or STOP SLAVE SQL_THREAD in MySQL 5.1.55 and later) prior to shutting down the slave MySQL server helps prevent such issues from occurring, and is always recommended regardless of the logging format or storage engines employed.

2.2.3 Determination of Safe and Unsafe Statements in Binary Logging

When speaking of the “safeness” of a statement in MySQL Replication, we are referring to whether a statement and its effects can be replicated correctly using statement-based format. If this is true of the statement, we refer to the statement as safe; otherwise, we refer to it as unsafe.

In general, a statement is safe if it deterministic, and unsafe if it is not. However, certain nondeterministic functions are not considered unsafe (see Nondeterministic functions not considered unsafe, later in this section). In addition, statements using results from floating-point math functions—which are hardware-dependent—are always considered unsafe (see Section 4.1.13, “Replication and Floating-Point Values”).

Handling of safe and unsafe statements. A statement is treated differently depending on whether the statement is considered safe, and with respect to the binary logging format (that is, the current value of binlog_format).

• No distinction is made in the treatment of safe and unsafe statements when the binary logging mode is ROW.
If the binary logging format is \textit{MIXED}, statements flagged as unsafe are logged using the row-based format; statements regarded as safe are logged using the statement-based format.

If the binary logging format is \textit{STATEMENT}, statements flagged as being unsafe generate a warning to this effect. (Safe statements are logged normally.)

For more information, see Section 2.2, “Replication Formats”.

\textbf{Statements considered unsafe.}

Statements having the following characteristics are considered unsafe:

- Statements containing system functions that may return a different value on slave. These functions include \texttt{FOUND_ROWS()}, \texttt{GET_LOCK()}, \texttt{IS_FREE_LOCK()}, \texttt{IS_USED_LOCK()}, \texttt{LOAD_FILE()}, \texttt{MASTER_POS_WAIT()}, \texttt{RELEASE_LOCK()}, \texttt{ROW_COUNT()}, \texttt{SESSION_USER()}, \texttt{SLEEP()}, \texttt{SYSDATE()}, \texttt{SYSTEM_USER()}, \texttt{USER()}, \texttt{UUID()}, and \texttt{UUID_SHORT()}.

  In MySQL 5.1.43 and later, \texttt{RAND()} is also treated as unsafe. (Bug #11757207, Bug #49222)

- Nondeterministic functions not considered unsafe. Although these functions are not deterministic, they are treated as safe for purposes of logging and replication: \texttt{CONNECTION_ID()}, \texttt{CURDATE()}, \texttt{CURRENT_DATE()}, \texttt{CURRENT_TIME()}, \texttt{CURRENT_TIMESTAMP()}, \texttt{CURTIME()}, \texttt{LOCALTIME()}, \texttt{LOCALTIMESTAMP()}, \texttt{NOW()}, \texttt{UNIX_TIMESTAMP()}, \texttt{UTC_DATE()}, \texttt{UTC_TIME()}, \texttt{UTC_TIMESTAMP()}, and \texttt{LAST_INSERT_ID()}

  For more information, see Section 4.1.15, “Replication and System Functions”.

- References to system variables. Most system variables are not replicated correctly using the statement-based format. For exceptions, see Mixed Binary Logging Format.

  See Section 4.1.38, “Replication and Variables”.

- UDFs. Since we have no control over what a UDF does, we must assume that it is executing unsafe statements.

- Updates a table having an AUTO_INCREMENT column. This is unsafe because the order in which the rows are updated may differ on the master and the slave.

  In addition, an \texttt{INSERT} into a table that has a composite primary key containing an \texttt{AUTO_INCREMENT} column that is not the first column of this composite key is unsafe.

  For more information, see Section 4.1.1, “Replication and AUTO_INCREMENT”.

- \texttt{INSERT DELAYED} statement. This statement is considered unsafe because the insertion of the rows may interleave with concurrently executing statements.

- \texttt{INSERT ... ON DUPLICATE KEY UPDATE} statements on tables with multiple primary or unique keys. When executed against a table that contains more than one primary or unique key, this statement is considered unsafe, being sensitive to the order in which the storage engine checks the keys, which is not deterministic, and on which the choice of rows updated by the MySQL Server depends.

  Thus, such statements are considered unsafe.

- Updates using \texttt{LIMIT}. The order in which rows are retrieved is not specified.

  See Section 4.1.16, “Replication and LIMIT”.

- Accesses or references log tables. The contents of the system log table may differ between master and slave.
• **Nontransactional operations after transactional operations.** Within a transaction, allowing any nontransactional reads or writes to execute after any transactional reads or writes is considered unsafe.

For more information, see Section 4.1.35, “Replication and Transactions”.

• **Accesses or references self-logging tables.** All reads and writes to self-logging tables are considered unsafe. Within a transaction, any statement following a read or write to self-logging tables is also considered unsafe.

• **LOAD DATA INFILE statements.** Beginning with MySQL 5.1.50, LOAD DATA INFILE is considered unsafe, it causes a warning in statement-based mode, and a switch to row-based format when using mixed-format logging. See Section 4.1.17, “Replication and LOAD DATA INFILE”.

For additional information, see Section 4.1, “Replication Features and Issues”.

### 2.3 Replication and Binary Logging Options and Variables

The next few sections contain information about `mysqld` options and server variables that are used in replication and for controlling the binary log. Options and variables for use on replication masters and replication slaves are covered separately, as are options and variables relating to binary logging. A set of quick-reference tables providing basic information about these options and variables is also included (in the next section following this one).

Of particular importance is the `--server-id` option.

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th><strong>--server-id=</strong>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>server_id</td>
</tr>
<tr>
<td></td>
<td>Variable Scope</td>
</tr>
<tr>
<td></td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Min Value</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Max Value</td>
</tr>
<tr>
<td></td>
<td>4294967295</td>
</tr>
</tbody>
</table>

This option is common to both master and slave replication servers, and is used in replication to enable master and slave servers to identify themselves uniquely. For additional information, see Section 2.3.2, “Replication Master Options and Variables”, and Section 2.3.3, “Replication Slave Options and Variables”.

On the master and each slave, you must use the `--server-id` option to establish a unique replication ID in the range from 1 to $2^{32} - 1$. “Unique”, means that each ID must be different from every other ID in use by any other replication master or slave. Example: `server-id=3`.

If you omit `--server-id`, the default ID is 0, in which case the master refuses connections from all slaves, and slaves refuse to connect to the master. In MySQL 5.1, whether the server ID is set to 0 explicitly or the default is allowed to be used, the server sets the `server_id` system variable to 1; this is a known issue that is fixed in MySQL 5.7.

For more information, see Section 2.1.2, “Setting the Replication Slave Configuration”.

21
2.3.1 Replication and Binary Logging Option and Variable Reference

The following tables list basic information about the MySQL command-line options and system variables applicable to replication and the binary log.

Table 2.1 Summary of Replication options and variables in MySQL 5.1

<table>
<thead>
<tr>
<th>Option or Variable Name</th>
<th>Command Line</th>
<th>System Variable</th>
<th>Status Variable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort-slave-event-count</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Com_change_master</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Com_show_master_status</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Com_show_new_master</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Com_show_slave_hosts</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Com_show_slave_status</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Com_slave_start</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Com_slave_stop</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

DESCRIPTION: Option used by mysql-test for debugging and testing of replication
<table>
<thead>
<tr>
<th>Option or Variable Name</th>
<th>Command Line</th>
<th>System Variable</th>
<th>Status Variable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>disconnect-slave-event-count</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>DESCRIPTION: Count of STOP SLAVE statements</td>
</tr>
<tr>
<td>have_row_based_replication</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>DESCRIPTION: Option used by mysql-test for debugging and testing of replication</td>
</tr>
<tr>
<td>init_slave</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>DESCRIPTION: Shows whether row-based replication is supported</td>
</tr>
<tr>
<td>log-slave-updates</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>DESCRIPTION: Statements that are executed when a slave connects to a master</td>
</tr>
<tr>
<td>log_slave_updates</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>DESCRIPTION: This option tells the slave to log the updates performed by its SQL thread to its own binary log</td>
</tr>
<tr>
<td>master-connect-retry</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>DESCRIPTION: Number of seconds the slave thread will sleep before retrying to connect to the master in case the master goes down or the connection is lost</td>
</tr>
<tr>
<td>master-host</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>DESCRIPTION: Master host name or IP address for replication</td>
</tr>
</tbody>
</table>

23
<table>
<thead>
<tr>
<th>Option or Variable Name</th>
<th>Command Line</th>
<th>System Variable</th>
<th>Status Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>master-password</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: The location and name of the file that remembers the master and where the I/O replication thread is in the master's binary logs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-port</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: The password the slave thread will authenticate with when connecting to master</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-retry-count</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: The port the master is listening on</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-ssl</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Number of tries the slave will make to connect to the master before giving up</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-ssl-ca</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Master SSL CA file; applies only if master-ssl is enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-ssl-capath</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Master SSL CA path; applies only if master-ssl is enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-ssl-cert</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Master SSL certificate file name; applies only if master-ssl is enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-ssl-cipher</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Master SSL cipher; applies only if master-ssl is enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>master-ssl-key</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Option or Variable Name</td>
<td>Command Line</td>
<td>System Variable</td>
<td>Status Variable</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>master-user</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>relay-log</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>relay-log-index</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>relay-log-info-file</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>relay_log_index</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>relay_log_info_file</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>relay_log_purge</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

DESCRIPTION: Master SSL key file name; applies only if master-ssl is enabled

DESCRIPTION: The user name the slave thread will use for authentication when connecting to master. The user must have FILE privilege. If the master user is not set, user test is assumed. The value in master.info will take precedence if it can be read

DESCRIPTION: The location and base name to use for relay logs

DESCRIPTION: The location and name to use for the file that keeps a list of the last relay logs

DESCRIPTION: The location and name of the file that remembers where the SQL replication thread is in the relay logs

DESCRIPTION: The name of the relay log index file.

DESCRIPTION: The name of the file in which the slave records information about the relay logs.

DESCRIPTION: Determines whether relay logs are purged
<table>
<thead>
<tr>
<th>Option or Variable Name</th>
<th>Command Line</th>
<th>System Variable</th>
<th>Status Variable</th>
<th>Notes</th>
<th>Scope</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>replicate-do-db</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Global</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Maximum space to use for all relay logs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-do-table</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Tells the slave SQL thread to restrict replication to the specified database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-ignore-db</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Tells the slave SQL thread not to replicate to the specified database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-ignore-table</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Tells the slave SQL thread not to replicate to the specified table</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-rewrite-db</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Updates to a database with a different name than the original</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-same-server-id</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: In replication, if set to 1, do not skip events having our server id</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-wild-do-table</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DESCRIPTION: Tells the slave thread to restrict replication to the tables that match the specified wildcard pattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>replicate-wild-ignore-table</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Tells the slave thread not to replicate to the tables that match the given wildcard pattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option or Variable Name</td>
<td>Command Line</td>
<td>System Variable</td>
<td>Status Variable</td>
<td>Scope</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>----------------</td>
<td>-------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>report-host</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Global</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Host name or IP of the slave to be reported to the master during slave registration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>report-password</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Global</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: An arbitrary password that the slave server should report to the master. Not the same as the password for the MySQL replication user account</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>report-port</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Global</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Port for connecting to slave reported to the master during slave registration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>report-user</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Global</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: An arbitrary user name that a slave server should report to the master. Not the same as the name used with the MySQL replication user account.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpl_recovery_rank</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Global</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Not used; removed in later versions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rpl_status</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Global</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: The status of fail-safe replication (not implemented)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>show-slave-auth-info</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Show user name and password in SHOW SLAVE HOSTS on this master</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>skip-slave-start</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: If set, slave is not autostarted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave-load-tmpdir</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option or Variable Name</td>
<td>Command Line</td>
<td>System Variable</td>
<td>Status Variable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave_max_allowed_packet</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: The location where the slave should put its temporary files when replicating a LOAD DATA INFILE statement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave_net_timeout</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Global</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Maximum size, in bytes, of a packet that can be sent from a replication master to a slave; overrides max_allowed_packet.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave_skip_errors</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Global</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Number of seconds to wait for more data from a master/slave connection before aborting the read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave_compressed_protocol</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Global</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Tells the slave thread to continue replication when a query returns an error from the provided list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave_exec_mode</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Global</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Allows for switching the slave thread between IDEMPOTENT mode (key and some other errors suppressed) and STRICT mode; STRICT mode is the default, except for MySQL Cluster, where IDEMPOTENT is always used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slave_heartbeat_period</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: The slave's replication heartbeat interval, in seconds</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slave_max_allowed_packet</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Global</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION: Maximum size, in bytes, of a packet that can be sent from a replication master to a slave; overrides max_allowed_packet.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option or Variable Name</td>
<td>Command Line</td>
<td>System Variable</td>
<td>Status Variable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Slave_open_temp_tables</strong></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Slave_retried_transactions</strong></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Slave_running</strong></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>slave_transaction_retries</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>slave_type_conversions</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sql_slave_skip_counter</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>sync_binlog</strong></td>
<td>Yes</td>
<td>Global</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DESCRIPTION: Number of temporary tables that the slave SQL thread currently has open.
DESCRIPTION: The total number of times since startup that the replication slave SQL thread has retried transactions.
DESCRIPTION: The state of this server as a replication slave (slave I/O thread status).
DESCRIPTION: Number of times the slave SQL thread will retry a transaction in case it failed with a deadlock or elapsed lock wait timeout, before giving up and stopping.
DESCRIPTION: Controls type conversion mode on replication slave. Value is a list of zero or more elements from the list: ALL_LOSSY, ALL_NON_LOSSY. Set to an empty string to disallow type conversions between master and slave.
DESCRIPTION: Number of events from the master that a slave server should skip. Not compatible with GTID replication.
DESCRIPTION: Synchronously flush binary log to disk after every #th event.

Section 2.3.2, “Replication Master Options and Variables”, provides more detailed information about options and variables relating to replication master servers. For more information about options and variables relating to replication slaves, see Section 2.3.3, “Replication Slave Options and Variables”.
Table 2.2 Summary of Binary Logging options and variables in MySQL 5.1

<table>
<thead>
<tr>
<th>Option or Variable Name</th>
<th>Command Line</th>
<th>System Variable</th>
<th>Status Variable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>binlog-do-db</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>DESCRIPTION: Limits binary logging to specific databases</td>
</tr>
<tr>
<td>binlog_format</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>binlog-ignore-db</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>binlog-row-event-max-size</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Binlog_cache_disk_use</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>binlog_cache_size</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Binlog_cache_use</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>binlog_direct_non_transactional_updates</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Com_show_binlog_events</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

DESCRIPTION: Causes updates using statement format to nontransactional engines to be written directly to binary log. See documentation before using.
<table>
<thead>
<tr>
<th>Option or Variable Name</th>
<th>Command Line</th>
<th>System Variable</th>
<th>Status Variable</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Option File</td>
<td>Scope</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td><strong>Com_show_binlogs</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>log-bin-use-v1-row-events</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>max-binlog-dump-events</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>max_binlog_cache_size</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>max_binlog_size</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>sporadic-binlog-dump-fail</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**DESCRIPTION:** Count of SHOW BINLOG EVENTS statements

For information about the `sql_log_bin` and `sql_log_off` variables, see Server System Variables.

Section 2.3.4, “Binary Log Options and Variables”, provides more detailed information about options and variables relating to binary logging. For additional general information about the binary log, see The Binary Log.
Replication Master Options and Variables

For a table showing all command-line options, system and status variables used with `mysqld`, see Server Option and Variable Reference.

### 2.3.2 Replication Master Options and Variables

This section describes the server options and system variables that you can use on replication master servers. You can specify the options either on the command line or in an option file. You can specify system variable values using `SET`.

On the master and each slave, you must use the `server-id` option to establish a unique replication ID. For each server, you should pick a unique positive integer in the range from 1 to $2^{32} - 1$, and each ID must be different from every other ID in use by any other replication master or slave. Example: `server-id=3`.

For options used on the master for controlling binary logging, see Section 2.3.4, “Binary Log Options and Variables”.

#### System Variables Used on Replication Masters

The following system variables are used in controlling replication masters:

- `auto_increment_increment`

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Name</th>
<th>auto_increment_increment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Scope</td>
<td>Global, Session</td>
<td></td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Min Value</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Max Value</td>
<td>65535</td>
<td></td>
</tr>
</tbody>
</table>

`auto_increment_increment` and `auto_increment_offset` are intended for use with master-to-master replication, and can be used to control the operation of `AUTO_INCREMENT` columns. Both variables have global and session values, and each can assume an integer value between 1 and 65,535 inclusive. Setting the value of either of these two variables to 0 causes its value to be set to 1 instead. Attempting to set the value of either of these two variables to an integer greater than 65,535 or less than 0 causes its value to be set to 65,535 instead. Attempting to set the value of `auto_increment_increment` or `auto_increment_offset` to a noninteger value gives rise to an error, and the actual value of the variable remains unchanged.

**Note**

`auto_increment_increment` is supported for use with NDB tables beginning with MySQL 5.1.20, MySQL Cluster NDB 6.2.5, and MySQL Cluster NDB 6.3.2. Previously, setting it when using MySQL Cluster tables or MySQL Cluster Replication produced unpredictable results.

These two variables affect `AUTO_INCREMENT` column behavior as follows:
• **auto_increment_increment** controls the interval between successive column values. For example:

```sql
mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 1     |
| auto_increment_offset    | 1     |
+--------------------------+-------+
```

2 rows in set (0.00 sec)

```sql
mysql> CREATE TABLE autoinc1
    -> (col INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
Query OK, 0 rows affected (0.04 sec)
```

```sql
mysql> SET @@auto_increment_increment=10;
Query OK, 0 rows affected (0.00 sec)
```

```sql
mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 10    |
| auto_increment_offset    | 1     |
+--------------------------+-------+
```

2 rows in set (0.01 sec)

```sql
mysql> INSERT INTO autoinc1 VALUES (NULL), (NULL), (NULL), (NULL);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```sql
mysql> SELECT col FROM autoinc1;
+-----+
| col |
+-----+
|   1 |
|  11 |
|  21 |
|  31 |
+-----+
```

4 rows in set (0.00 sec)

• **auto_increment_offset** determines the starting point for the AUTO_INCREMENT column value. Consider the following, assuming that these statements are executed during the same session as the example given in the description for `auto_increment_increment`:

```sql
mysql> SET @@auto_increment_offset=5;
Query OK, 0 rows affected (0.00 sec)
```

```sql
mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 10    |
| auto_increment_offset    | 5     |
+--------------------------+-------+
```

2 rows in set (0.00 sec)

```sql
mysql> CREATE TABLE autoinc2
    -> (col INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
Query OK, 0 rows affected (0.06 sec)
```
If the value of `auto_increment_offset` is greater than that of `auto_increment_increment`, the value of `auto_increment_offset` is ignored.

Should one or both of these variables be changed and then new rows inserted into a table containing an `AUTO_INCREMENT` column, the results may seem counterintuitive because the series of `AUTO_INCREMENT` values is calculated without regard to any values already present in the column, and the next value inserted is the least value in the series that is greater than the maximum existing value in the `AUTO_INCREMENT` column. In other words, the series is calculated like so:

\[ \text{auto_increment_offset} + N \times \text{auto_increment_increment} \]

where \( N \) is a positive integer value in the series \([1, 2, 3, \ldots]\). For example:

```
mysql> SHOW VARIABLES LIKE 'auto_inc%';
+--------------------------+-----+
| Variable_name            | Value|
|--------------------------+-----+
| auto_increment_increment | 10  |
| auto_increment_offset    | 5   |
+--------------------------+-----+
2 rows in set (0.00 sec)

mysql> SELECT col FROM autoinc1;
+----+
| col |
+----+
| 1   |
| 11  |
| 21  |
| 31  |
+----+
4 rows in set (0.00 sec)

mysql> INSERT INTO autoinc1 VALUES (NULL), (NULL), (NULL), (NULL);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT col FROM autoinc1;
+----+
| col |
+----+
| 1   |
| 11  |
| 21  |
| 31  |
| 35  |
| 45  |
| 55  |
| 65  |
+----+
8 rows in set (0.00 sec)
```
The values shown for `auto_increment_increment` and `auto_increment_offset` generate the series \(5 + N \times 10\), that is, [5, 15, 25, 35, 45, ...]. The greatest value present in the `col` column prior to the `INSERT` is 31, and the next available value in the `AUTO_INCREMENT` series is 35, so the inserted values for `col` begin at that point and the results are as shown for the `SELECT` query.

It is not possible to confine the effects of these two variables to a single table, and thus they do not take the place of the sequences offered by some other database management systems; these variables control the behavior of all `AUTO_INCREMENT` columns in all tables on the MySQL server. If the global value of either variable is set, its effects persist until the global value is changed or overridden by setting the session value, or until `mysqld` is restarted. If the local value is set, the new value affects `AUTO_INCREMENT` columns for all tables into which new rows are inserted by the current user for the duration of the session, unless the values are changed during that session.

The default value of `auto_increment_increment` is 1. See Section 4.1.1, “Replication and `AUTO_INCREMENT`”.

- `auto_increment_offset`

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Name</th>
<th>auto_increment_offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Scope</td>
<td>Global, Session</td>
<td></td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>Default</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Min Value</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Max Value</td>
<td>65535</td>
</tr>
</tbody>
</table>

This variable has a default value of 1. For particulars, see the description for `auto_increment_increment`.

Note

`auto_increment_offset` is supported for use with NDB tables beginning with MySQL 5.1.20, MySQL Cluster NDB 6.2.5, and MySQL Cluster NDB 6.3.2. Previously, setting it when using MySQL Cluster tables or MySQL Cluster Replication produced unpredictable results.

### 2.3.3 Replication Slave Options and Variables

#### Startup Options for Replication Slaves

#### System Variables Used on Replication Slaves

This section describes the server options and system variables that apply to slave replication servers. You can specify the options either on the command line or in an option file. Many of the options can be set while the server is running by using the `CHANGE MASTER TO` statement. You can specify system variable values using `SET`. 
Server ID. On the master and each slave, you must use the server-id [21] option to establish a unique replication ID in the range from 1 to $2^{32} - 1$. "Unique" means that each ID must be different from every other ID in use by any other replication master or slave. Example my.cnf file:

```
[mysqld]
server-id=3
```

Important

Certain --master-xxx options are handled in a special way to ensure that the active replication configuration is not inadvertently altered or affected:

- --master-host
- --master-user
- --master-password
- --master-port
- --master-connect-retry
- --master-ssl
- --master-ssl-ca
- --master-ssl-capath
- --master-ssl-cert
- --master-ssl-cipher
- --master-ssl-key

Before MySQL 5.1.17, these options are silently ignored if given unless there is no master.info file. If that file exists, the MySQL server has already previously been configured for replication, so the information in the file is used instead. Because the server gives an existing master.info file precedence over the startup options just described, you might elect not to use startup options for these values at all, and instead to specify the replication parameters associated with them by using the CHANGE MASTER TO statement. See CHANGE MASTER TO Syntax.

Beginning with MySQL 5.1.17, these options are deprecated and have no effect when mysqld is started. If they are used, an appropriate warning is written to the error log. Instead, you must use CHANGE MASTER TO to set the values corresponding to the deprecated options. These options are removed in MySQL 5.5.

The master.info file format in MySQL 5.1 includes as its first line the number of lines in the file. (See Section 5.2, "Replication Relay and Status Logs".) If you upgrade an older server (before MySQL 4.1.1) to a newer version, the new server upgrades the master.info file to the new format automatically when it starts. However, if you downgrade a newer server to an older version, you should remove the first line manually before starting the older server for the first time.

If no master.info file exists when the slave server starts, it uses the values for those options that are specified in option files or on the command line. This occurs when you start the server as a replication
slave for the very first time, when you use `CHANGE MASTER TO`, or when you have run `RESET SLAVE` and then have shut down and restarted the slave.

Because the server uses `master.info` file contents and ignores any startup options that correspond to the values listed in the file, if you start the slave server with different values of the startup options that correspond to values in the file, the different values have no effect. To use different values, the preferred method is to use the `CHANGE MASTER TO` statement to reset the values while the slave is running. Alternatively, you can stop the server, remove the `master.info` file, and restart the server with different option values.

Suppose that you specify this option in your `my.cnf` file to configure a pre-5.1.17 slave server:

```sql
[mysqld]
master-host=some_host
```

The first time you start the server as a replication slave, it reads and uses that option from the `my.cnf` file. The server then records the value in the `master.info` file. The next time you start the server, it reads the master host value from the `master.info` file only and ignores the value in the option file. If you modify the `my.cnf` file to specify a different master host value of `some_other_host`, the change has no effect. Instead, use `CHANGE MASTER TO` to change the value.

This example shows a more extensive use of startup options to configure a pre-5.1.17 slave server:

```sql
[mysqld]
server-id=2
master-host=db-master.mycompany.com
master-port=3306
master-user=pertinax
master-password=freitag
master-connect-retry=60
report-host=db-slave.mycompany.com
```

As of 5.1.17, the `master-xxx` options are ignored and result in warnings in the error log.

### Startup Options for Replication Slaves

The following list describes startup options for controlling replication slave servers. Many of these options can be set while the server is running by using the `CHANGE MASTER TO` statement. Others, such as the `--replicate-*` options, can be set only when the slave server starts. Replication-related system variables are discussed later in this section.

- **--abort-slave-event-count**

  **Command-Line Format**

  `--abort-slave-event-count=#`

  **Permitted Values**

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>Default</td>
<td>0</td>
</tr>
<tr>
<td>Min Value</td>
<td>0</td>
</tr>
</tbody>
</table>

  When this option is set to some positive integer `value` other than 0 (the default) it affects replication behavior as follows: After the slave SQL thread has started, `value` log events are permitted to be executed; after that, the slave SQL thread does not receive any more events, just as if the network connection from the master were cut. The slave thread continues to run, and the output from `SHOW SLAVE STATUS` displays `Yes` in both the `Slave_IO_Running` and the `Slave_SQL_Running` columns, but no further events are read from the relay log.
This option is used internally by the MySQL test suite for replication testing and debugging. It is not intended for use in a production setting.

- **--disconnect-slave-event-count**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--disconnect-slave-event-count=#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type: integer</td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
</tr>
</tbody>
</table>

This option is used internally by the MySQL test suite for replication testing and debugging.

- **--log-slave-updates**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--log-slave-updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name: log_slave_updates</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>No</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type: boolean</td>
</tr>
<tr>
<td></td>
<td>Default: OFF</td>
</tr>
</tbody>
</table>

Normally, a slave does not log to its own binary log any updates that are received from a master server. This option tells the slave to log the updates performed by its SQL thread to its own binary log. For this option to have any effect, the slave must also be started with the **--log-bin** option to enable binary logging. **--log-slave-updates** is used when you want to chain replication servers. For example, you might want to set up replication servers using this arrangement:

```
A -> B -> C
```

Here, **A** serves as the master for the slave **B**, and **B** serves as the master for the slave **C**. For this to work, **B** must be both a master and a slave. You must start both **A** and **B** with **--log-bin** to enable binary logging, and **B** with the **--log-slave-updates** option so that updates received from **A** are logged by **B** to its binary log.

When using MySQL Cluster Replication prior to MySQL Cluster NDB 6.2.16 and MySQL Cluster NDB 6.3.13, records for “empty” epochs—that is, epochs in which no changes to **NDBCLUSTER** data or tables took place—were inserted into the **ndb_apply_status** and **ndb_binlog_index** tables on the slave even when **--log-slave-updates** was disabled (Bug #37472). Beginning with MySQL Cluster NDB 6.3.33, MySQL Cluster NDB 7.0.14, and MySQL Cluster NDB 7.1.3, it is possible to re-enable the older behavior by using the **--ndb-log-empty-epochs** option.

**Note**

The **--ndb-log-empty-epochs** option was first implemented in MySQL Cluster NDB 6.3.21 and MySQL Cluster NDB 6.4.1, but did not work correctly before the versions cited previously.

- **--log-slow-slave-statements**

| Introduced | 5.1.21 |
### Command-Line Format

<table>
<thead>
<tr>
<th><strong>--log-slow-slave-statements</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Permitted Values</strong></th>
<th><strong>Type</strong></th>
<th>boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td></td>
<td>OFF</td>
</tr>
</tbody>
</table>

When the slow query log is enabled, this option enables logging for queries that have taken more than `long_query_time` seconds to execute on the slave.

This option was added in MySQL 5.1.21.

- **--log-warnings[=level]**

<table>
<thead>
<tr>
<th><strong>Command-Line Format</strong></th>
<th><strong>--log-warnings[=#]</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>System Variable</strong></th>
<th><strong>Name</strong></th>
<th>log_warnings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable Scope</strong></td>
<td></td>
<td>Global, Session</td>
</tr>
<tr>
<td><strong>Dynamic Variable</strong></td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Permitted Values</strong></th>
<th><strong>Type</strong></th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Min Value</strong></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td><strong>Max Value</strong></td>
<td></td>
<td>4294967295</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Permitted Values</strong></th>
<th><strong>Type</strong></th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>Min Value</strong></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td><strong>Max Value</strong></td>
<td></td>
<td>18446744073709547520</td>
</tr>
</tbody>
</table>

This option causes a server to print more messages to the error log about what it is doing. With respect to replication, the server generates warnings that it succeeded in reconnecting after a network/connection failure, and informs you as to how each slave thread started. This option is enabled (1) by default; to disable it, use **--log-warnings=0**. Aborted connections are not logged to the error log unless the value is greater than 1.

Note that the effects of this option are not limited to replication. It produces warnings across a spectrum of server activities.

- **--master-connect-retry=seconds**

<table>
<thead>
<tr>
<th><strong>Command-Line Format</strong></th>
<th><strong>--master-connect-retry=#</strong></th>
</tr>
</thead>
</table>

<p>| <strong>Deprecated</strong>          | 5.1.17                      |</p>
<table>
<thead>
<tr>
<th><strong>Permitted Values</strong></th>
<th><strong>Type</strong></th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>

The number of seconds that the slave thread sleeps before trying to reconnect to the master in case the master goes down or the connection is lost. The value in the `master.info` file takes precedence.
if it can be read. If not set, the default is 60. Connection retries are not invoked until the slave times out reading data from the master according to the value of `--slave-net-timeout`. The number of reconnection attempts is limited by the `--master-retry-count` option.

This option is deprecated as of MySQL 5.1.17 and is removed in MySQL 5.5.

- `slave-max-allowed-packet=bytes`

<table>
<thead>
<tr>
<th>Introduced</th>
<th>5.1.64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--slave-max-allowed-packet=#</code></td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type: integer</td>
</tr>
<tr>
<td></td>
<td>Default: 1073741824</td>
</tr>
<tr>
<td></td>
<td>Min Value: 1024</td>
</tr>
<tr>
<td></td>
<td>Max Value: 1073741824</td>
</tr>
</tbody>
</table>

In MySQL 5.1.64 and later, this option sets the maximum packet size in bytes for the slave SQL and I/O threads, so that large updates using row-based replication do not cause replication to fail because an update exceeded `max_allowed_packet`. (Bug #12400221, Bug #60926)

The corresponding server variable `slave_max_allowed_packet` always has a value that is a positive integer multiple of 1024; if you set it to some value that is not such a multiple, the value is automatically rounded down to the next highest multiple of 1024. (For example, if you start the server with `--slave-max-allowed-packet=10000`, the value used is 9216; setting 0 as the value causes 1024 to be used.) A truncation warning is issued in such cases.

The maximum (and default) value is 1073741824 (1 GB); the minimum is 1024.

- `--master-host=host_name`

<table>
<thead>
<tr>
<th>Deprecated</th>
<th>5.1.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--master-host=name</code></td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type: string</td>
</tr>
</tbody>
</table>

The host name or IP address of the master replication server. The value in `master.info` takes precedence if it can be read. If no master host is specified, the slave thread does not start.

This option is deprecated as of MySQL 5.1.17 and is removed in MySQL 5.5.

- `--master-info-file=file_name`

| Command-Line Format | `--master-info-file=file_name` |
| Permitted Values | Type: file name |
| | Default: `master.info` |

The name to use for the file in which the slave records information about the master. The default name is `master.info` in the data directory. For information about the format of this file, see Section 5.2.2, “Slave Status Logs”.

- `--master-password=password`
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Deprecated</th>
<th>5.1.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--master-password=name</code></td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type: string</td>
</tr>
</tbody>
</table>

The password of the account that the slave thread uses for authentication when it connects to the master. The value in the master.info file takes precedence if it can be read. If not set, an empty password is assumed.

This option is deprecated as of MySQL 5.1.17 and is removed in MySQL 5.5.

- `--master-port=port_number`

<table>
<thead>
<tr>
<th>Deprecated</th>
<th>5.1.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--master-port=#</code></td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type: integer</td>
</tr>
<tr>
<td>Default</td>
<td>3306</td>
</tr>
</tbody>
</table>

The TCP/IP port number that the master is listening on. The value in the master.info file takes precedence if it can be read. If not set, the compiled-in setting is assumed (normally 3306).

This option is deprecated as of MySQL 5.1.17 and is removed in MySQL 5.5.

- `--master-retry-count=count`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th><code>--master-retry-count=#</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values (32-bit platforms)</td>
<td>Type: integer</td>
</tr>
<tr>
<td>Default</td>
<td>86400</td>
</tr>
<tr>
<td>Min Value</td>
<td>0</td>
</tr>
<tr>
<td>Max Value</td>
<td>4294967295</td>
</tr>
</tbody>
</table>

| Permitted Values (64-bit platforms) | Type: integer |
| Default              | 86400 |
| Min Value            | 0 |
| Max Value            | 18446744073709551615 |

The number of times that the slave tries to connect to the master before giving up. Reconnects are attempted at intervals set by the `--master-connect-retry` option (or the MASTER_CONNECT_RETRY option of the CHANGE MASTER TO statement) and reconnects are triggered when data reads by the slave time out according to the `--slave-net-timeout` option. The default value is 86400. A value of 0 means “infinite”; the slave attempts to connect forever.

- `--master-ssl, --master-ssl-ca=file_name, --master-ssl-capath=directory_name, --master-ssl-cert=file_name, --master-ssl-cipher=cipher_list, --master-ssl-key=file_name`
These options are used for setting up a secure replication connection to the master server using SSL. Their meanings are the same as the corresponding --ssl, --ssl-ca, --ssl-capath, --ssl-cert, --ssl-cipher, --ssl-key options that are described in SSL Command Options. The values in the master.info file take precedence if they can be read.

These options are deprecated as of MySQL 5.1.17 and are removed in MySQL 5.5.

- **--master-user=user_name**

  | Deprecated | 5.1.17 |
  | Command-Line Format | --master-user=name |
  | Permitted Values | Type | string |
  | | Default | test |

  The user name of the account that the slave thread uses for authentication when it connects to the master. This account must have the REPLICATION SLAVE privilege. The value in the master.info file takes precedence if it can be read. If the master user name is not set, the name test is assumed.

  This option is deprecated as of MySQL 5.1.17 and is removed in MySQL 5.5.

- **--max-relay-log-size=size**

  | Command-Line Format | --max_relay_log_size=# |
  | System Variable | Name | max_relay_log_size |
  | | Variable Scope | Global |
  | | Dynamic Variable | Yes |
  | Permitted Values | Type | integer |
  | | Default | 0 |
  | | Min Value | 0 |
  | | Max Value | 1073741824 |

  The size at which the server rotates relay log files automatically. If this value is nonzero, the relay log is rotated automatically when its size exceeds this value. If this value is zero (the default), the size at which relay log rotation occurs is determined by the value of max_binlog_size. For more information, see Section 5.2.1, “The Slave Relay Log”.

- **--relay-log=file_name**

  | Command-Line Format | --relay-log=file_name |
  | System Variable | Name | relay_log |
  | | Variable Scope | Global |
  | | Dynamic Variable | No |
The base name for the relay log. The default base name is `host_name-relay-bin`. The server writes the file in the data directory unless the base name is given with a leading absolute path name to specify a different directory. The server creates relay log files in sequence by adding a numeric suffix to the base name.

Due to the manner in which MySQL parses server options, if you specify this option, you must supply a value; the default base name is used only if the option is not actually specified. If you use the `--relay-log` option without specifying a value, unexpected behavior is likely to result; this behavior depends on the other options used, the order in which they are specified, and whether they are specified on the command line or in an option file. For more information about how MySQL handles server options, see Specifying Program Options.

If you specify this option, the value specified is also used as the base name for the relay log index file. You can override this behavior by specifying a different relay log index file base name using the `--relay-log-index` option.

You may find the `--relay-log` option useful in performing the following tasks:

- Creating relay logs whose names are independent of host names.
- If you need to put the relay logs in some area other than the data directory because your relay logs tend to be very large and you do not want to decrease `max_relay_log_size`.
- To increase speed by using load-balancing between disks.
- `--relay-log-index=file_name`

### Permitted Values

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>file name</th>
</tr>
</thead>
</table>

The name to use for the relay log index file. The default name is `host_name-relay-bin.index` in the data directory, where `host_name` is the name of the slave server.

Due to the manner in which MySQL parses server options, if you specify this option, you must supply a value; the default base name is used only if the option is not actually specified. If you use the `--relay-log-index` option without specifying a value, unexpected behavior is likely to result; this behavior depends on the other options used, the order in which they are specified, and whether they are specified on the command line or in an option file. For more information about how MySQL handles server options, see Specifying Program Options.

If you specify this option, the value specified is also used as the base name for the relay logs. You can override this behavior by specifying a different relay log file base name using the `--relay-log` option.

- `--relay-log-info-file=file_name`

### System Variable

<table>
<thead>
<tr>
<th>Name</th>
<th>Global</th>
<th>relay_log_index</th>
</tr>
</thead>
</table>

### Command-Line Format

--relay-log-index=file_name

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th><code>--relay-log-index=file_name</code></th>
</tr>
</thead>
</table>
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td>relay-log.info</td>
</tr>
</tbody>
</table>

The name to use for the file in which the slave records information about the relay logs. The default name is `relay-log.info` in the data directory. For information about the format of this file, see Section 5.2.2, “Slave Status Logs”.

- `--relay-log-purge={0|1}`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--relay_log_purge</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name relay_log_purge</td>
</tr>
<tr>
<td></td>
<td>Variable Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td>TRUE</td>
</tr>
</tbody>
</table>

Disable or enable automatic purging of relay logs as soon as they are no longer needed. The default value is 1 (enabled). This is a global variable that can be changed dynamically with `SET GLOBAL relay_log_purge = N`.

- `--relay-log-space-limit=size`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--relay_log_space_limit=#</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name relay_log_space_limit</td>
</tr>
<tr>
<td></td>
<td>Variable Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (32-bit platforms)</th>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Min Value</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Max Value</td>
<td></td>
<td>4294967295</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (64-bit platforms)</th>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Min Value</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Max Value</td>
<td></td>
<td>18446744073709547520</td>
</tr>
</tbody>
</table>

This option places an upper limit on the total size in bytes of all relay logs on the slave. A value of 0 means “no limit.” This is useful for a slave server host that has limited disk space. When the limit is reached, the I/O thread stops reading binary log events from the master server until the SQL thread has caught up and deleted some unused relay logs. Note that this limit is not absolute: There are cases where the SQL thread needs more events before it can delete relay logs. In that case, the I/O thread
Replication Slave Options and Variables

exceeds the limit until it becomes possible for the SQL thread to delete some relay logs because not doing so would cause a deadlock. You should not set `--relay-log-space-limit` to less than twice the value of `--max-relay-log-size` (or `--max-binlog-size` if `--max-relay-log-size` is 0). In that case, there is a chance that the I/O thread waits for free space because `--relay-log-space-limit` is exceeded, but the SQL thread has no relay log to purge and is unable to satisfy the I/O thread. This forces the I/O thread to ignore `--relay-log-space-limit` temporarily.

- `--replicate-do-db=db_name`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--replicate-do-db=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>

The effects of this option depend on whether statement-based or row-based replication is in use.

**Statement-based replication.** Tell the slave SQL thread to restrict replication to statements where the default database (that is, the one selected by `USE`) is `db_name`. To specify more than one database, use this option multiple times, once for each database; however, doing so does not replicate cross-database statements such as `UPDATE some_db.some_table SET foo='bar'` while a different database (or no database) is selected.**

**Warning**

To specify multiple databases you **must** use multiple instances of this option. Because database names can contain commas, if you supply a comma separated list then the list will be treated as the name of a single database.

An example of what does not work as you might expect when using statement-based replication: If the slave is started with `--replicate-do-db=sales` and you issue the following statements on the master, the `UPDATE` statement is **not** replicated:

````
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

The main reason for this “check just the default database” behavior is that it is difficult from the statement alone to know whether it should be replicated (for example, if you are using multiple-table `DELETE` statements or multiple-table `UPDATE` statements that act across multiple databases). It is also faster to check only the default database rather than all databases if there is no need.

**Row-based replication.** Tells the slave SQL thread to restrict replication to database `db_name`. Only tables belonging to `db_name` are changed; the current database has no effect on this. Suppose that the slave is started with `--replicate-do-db=sales` and row-based replication is in effect, and then the following statements are run on the master:

````
USE prices;
UPDATE sales.february SET amount=amount+100;
```

The `february` table in the `sales` database on the slave is changed in accordance with the `UPDATE` statement; this occurs whether or not the `USE` statement was issued. However, issuing the following statements on the master has no effect on the slave when using row-based replication and `--replicate-do-db=sales`:

````
USE prices;
UPDATE prices.march SET amount=amount-25;
```
Even if the statement `USE prices` were changed to `USE sales`, the `UPDATE` statement's effects would still not be replicated.

Another important difference in how `--replicate-do-db` is handled in statement-based replication as opposed to row-based replication occurs with regard to statements that refer to multiple databases. Suppose that the slave is started with `--replicate-do-db=db1`, and the following statements are executed on the master:

```sql
USE db1;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

If you are using statement-based replication, then both tables are updated on the slave. However, when using row-based replication, only `table1` is affected on the slave; since `table2` is in a different database, `table2` on the slave is not changed by the `UPDATE`. Now suppose that, instead of the `USE db1` statement, a `USE db4` statement had been used:

```sql
USE db4;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

In this case, the `UPDATE` statement would have no effect on the slave when using statement-based replication. However, if you are using row-based replication, the `UPDATE` would change `table1` on the slave, but not `table2`—in other words, only tables in the database named by `--replicate-do-db` are changed, and the choice of default database has no effect on this behavior.

If you need cross-database updates to work, use `--replicate-wild-do-table=db_name.%` instead. See Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

**Note**

This option affects replication in the same manner that `--binlog-do-db` affects binary logging, and the effects of the replication format on how `--replicate-do-db` affects replication behavior are the same as those of the logging format on the behavior of `--binlog-do-db`.

Beginning with MySQL 5.1.35, this option has no effect on `BEGIN`, `COMMIT`, or `ROLLBACK` statements. (Bug #43263)

- `--replicate-ignore-db=db_name`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th><code>--replicate-ignore-db=name</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type string</td>
</tr>
</tbody>
</table>

As with `--replicate-do-db`, the effects of this option depend on whether statement-based or row-based replication is in use.

**Statement-based replication.** Tells the slave SQL thread not to replicate any statement where the default database (that is, the one selected by `USE`) is `db_name`.

**Row-based replication.** Tells the slave SQL thread not to update any tables in the database `db_name`. The default database has no effect.

When using statement-based replication, the following example does not work as you might expect. Suppose that the slave is started with `--replicate-ignore-db=sales` and you issue the following statements on the master:
USE prices;
UPDATE sales.january SET amount=amount+1000;

The `UPDATE` statement is replicated in such a case because `--replicate-ignore-db` applies only to the default database (determined by the `USE` statement). Because the `sales` database was specified explicitly in the statement, the statement has not been filtered. However, when using row-based replication, the `UPDATE` statement's effects are not propagated to the slave, and the slave's copy of the `sales.january` table is unchanged; in this instance, `--replicate-ignore-db=sales` causes all changes made to tables in the master's copy of the `sales` database to be ignored by the slave.

To specify more than one database to ignore, use this option multiple times, once for each database. Because database names can contain commas, if you supply a comma separated list then the list will be treated as the name of a single database.

You should not use this option if you are using cross-database updates and you do not want these updates to be replicated. See Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

If you need cross-database updates to work, use `--replicate-wild-ignore-table=db_name.%` instead. See Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

**Note**

This option affects replication in the same manner that `--binlog-ignore-db` affects binary logging, and the effects of the replication format on how `--replicate-ignore-db` affects replication behavior are the same as those of the logging format on the behavior of `--binlog-ignore-db`.

Beginning with MySQL 5.1.35, this option has no effect on `BEGIN`, `COMMIT`, or `ROLLBACK` statements. (Bug #43263)

**•--replicate-do-table=**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--replicate-do-table=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
</tbody>
</table>

Tells the slave SQL thread to restrict replication to the specified table. To specify more than one table, use this option multiple times, once for each table. This works for both cross-database updates and default database updates, in contrast to `--replicate-do-db`. See Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

This option affects only statements that apply to tables. It does not affect statements that apply only to other database objects, such as stored routines. To filter statements operating on stored routines, use one or more of the `--replicate-*/-db` options.

**•--replicate-ignore-table=**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--replicate-ignore-table=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
</tbody>
</table>

Tells the slave SQL thread not to replicate any statement that updates the specified table, even if any other tables might be updated by the same statement. To specify more than one table to ignore, use
this option multiple times, once for each table. This works for cross-database updates, in contrast to `--replicate-ignore-db`. See Section 5.3, "How Servers Evaluate Replication Filtering Rules".

This option affects only statements that apply to tables. It does not affect statements that apply only to other database objects, such as stored routines. To filter statements operating on stored routines, use one or more of the `--replicate-*-db` options.

- `--replicate-rewrite-db=from_name->to_name`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>Permitted Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--replicate-rewrite-db=old_name-&gt;new_name</code></td>
<td>Type: string</td>
</tr>
</tbody>
</table>

Tells the slave to translate the default database (that is, the one selected by `USE`) to `to_name` if it was `from_name` on the master. Only statements involving tables are affected (not statements such as `CREATE DATABASE`, `DROP DATABASE`, and `ALTER DATABASE`), and only if `from_name` is the default database on the master. To specify multiple rewrites, use this option multiple times. The server uses the first one with a `from_name` value that matches. The database name translation is done before the `--replicate-` rules are tested.

Statements in which table names are qualified with database names when using this option do not work with table-level replication filtering options such as `--replicate-do-table`. Suppose we have a database named `a` on the master, one named `b` on the slave, each containing a table `t`, and have started the master with `--replicate-rewrite-db='a->b'`. At a later point in time, we execute `DELETE FROM a.t`. In this case, no relevant filtering rule works, for the reasons shown here:

1. `--replicate-do-table=a.t` does not work because the slave has table `t` in database `b`.
2. `--replicate-do-table=b.t` does not match the original statement and so is ignored.
3. `--replicate-do-table=*.*` is handled identically to `--replicate-do-table=a.t`, and thus does not work, either.

Similarly, the `--replication-rewrite-db` option does not work with cross-database updates.

If you use this option on the command line and the “>” character is special to your command interpreter, quote the option value. For example:

```
shell> mysqld --replicate-rewrite-db="olddb->newdb"
```

- `--replicate-same-server-id`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>Permitted Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--replicate-same-server-id</code></td>
<td>Type: boolean</td>
</tr>
<tr>
<td>Default</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

To be used on slave servers. Usually you should use the default setting of 0, to prevent infinite loops caused by circular replication. If set to 1, the slave does not skip events having its own server ID. Normally, this is useful only in rare configurations. Cannot be set to 1 if `--log-slave-updates` is used. By default, the slave I/O thread does not write binary log events to the relay log if they have the slave’s server ID (this optimization helps save disk usage). If you want to use `--replicate-same-server-id`, be sure to start the slave with this option before you make the slave read its own events that you want the slave SQL thread to execute.

- `--replicate-wild-do-table=db_name.tbl_name`
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--replicate-wild-do-table=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type string</td>
</tr>
</tbody>
</table>

Tells the slave thread to restrict replication to statements where any of the updated tables match the specified database and table name patterns. Patterns can contain the “%” and “_” wildcard characters, which have the same meaning as for the LIKE pattern-matching operator. To specify more than one table, use this option multiple times, once for each table. This works for cross-database updates. See Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

This option applies to tables, views, and triggers. It does not apply to stored procedures and functions, or events. To filter statements operating on the latter objects, use one or more of the --replicate--*-db options.

Example: `--replicate-wild-do-table=foo%.bar%` replicates only updates that use a table where the database name starts with `foo` and the table name starts with `bar`.

If the table name pattern is `%`, it matches any table name and the option also applies to database-level statements (CREATE DATABASE, DROP DATABASE, and ALTER DATABASE). For example, if you use `--replicate-wild-do-table=foo%.%`, database-level statements are replicated if the database name matches the pattern `foo`.

To include literal wildcard characters in the database or table name patterns, escape them with a backslash. For example, to replicate all tables of a database that is named `my_own%db`, but not replicate tables from the `my1ownAABCdb` database, you should escape the “_” and “%” characters like this: `--replicate-wild-do-table=my\_own\%db`. If you use the option on the command line, you might need to double the backslashes or quote the option value, depending on your command interpreter. For example, with the bash shell, you would need to type `--replicate-wild-do-table=my\_own\%db`.

• `--replicate-wild-ignore-table=db_name.tbl_name`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--replicate-wild-ignore-table=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type string</td>
</tr>
</tbody>
</table>

Tells the slave thread not to replicate a statement where any table matches the given wildcard pattern. To specify more than one table to ignore, use this option multiple times, once for each table. This works for cross-database updates. See Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

Example: `--replicate-wild-ignore-table=foo%.bar%` does not replicate updates that use a table where the database name starts with `foo` and the table name starts with `bar`.

For information about how matching works, see the description of the --replicate-wild-do-table option. The rules for including literal wildcard characters in the option value are the same as for --replicate-wild-ignore-table as well.

• `--report-host=host_name`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--report-host=host_name</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>System Variable (&gt;= 5.1.24)</th>
<th>Name</th>
<th>report_host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Scope</td>
<td>Global</td>
</tr>
</tbody>
</table>

49
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Dynamic Variable</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
</tbody>
</table>

The host name or IP address of the slave to be reported to the master during slave registration. This value appears in the output of `SHOW SLAVE HOSTS` on the master server. Leave the value unset if you do not want the slave to register itself with the master. Note that it is not sufficient for the master to simply read the IP address of the slave from the TCP/IP socket after the slave connects. Due to NAT and other routing issues, that IP may not be valid for connecting to the slave from the master or other hosts.

- `--report-password=password`

Command-Line Format

| --report-password=name |

System Variable (>= 5.1.24)

| Name | report_password |
| Variable Scope | Global |
| Dynamic Variable | No |
| Permitted Values | Type | string |

An arbitrary account password to be reported by the slave to the master during slave registration. This value appears in the output of `SHOW SLAVE HOSTS` on the master server if the `--show-slave-auth-info` option is given.

Although the name of this option might imply otherwise, `--report-password` is not connected to the MySQL user privilege system and so is not necessarily (or even likely to be) the same as the password for the MySQL replication user account.

- `--report-port=slave_port_num`

Command-Line Format

| --report-port=## |

System Variable (>= 5.1.24)

| Name | report_port |
| Variable Scope | Global |
| Dynamic Variable | No |
| Permitted Values | Type | integer |
| Default | 3306 |
| Min Value | 0 |
| Max Value | 65535 |

The TCP/IP port number for connecting to the slave, to be reported to the master during slave registration. Set this only if the slave is listening on a nondefault port or if you have a special tunnel from the master or other clients to the slave. If you are not sure, do not use this option.

- `--report-user=user_name`
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--report-user=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable (&gt;= 5.1.24)</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Variable</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
</tr>
<tr>
<td></td>
<td>Type</td>
</tr>
</tbody>
</table>

The account user name of the slave to be reported to the master during slave registration. This value appears in the output of SHOW SLAVE HOSTS on the master server if the --show-slave-auth-info option is given.

Although the name of this option might imply otherwise, --report-user is not connected to the MySQL user privilege system and so is not necessarily (or even likely to be) the same as the name of the MySQL replication user account.

- --show-slave-auth-info

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--show-slave-auth-info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td>Default</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Display slave user names and passwords in the output of SHOW SLAVE HOSTS on the master server for slaves started with the --report-user and --report-password options.

- --skip-slave-start

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--skip-slave-start</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td>Default</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Tells the slave server not to start the slave threads when the server starts. To start the threads later, use a START SLAVE statement.

- --slave_compressed_protocol={0|1}

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave_compressed_protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Yes</td>
</tr>
<tr>
<td>Permit Values</td>
<td>Type</td>
</tr>
<tr>
<td>Default</td>
<td>OFF</td>
</tr>
</tbody>
</table>

If this option is set to 1, use compression for the slave/master protocol if both the slave and the master support it. The default is 0 (no compression).

- --slave-load-tmpdir=dir_name
Replication Slave Options and Variables

---

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave-load-tmpdir=dir_name</th>
</tr>
</thead>
</table>

**System Variable**

<table>
<thead>
<tr>
<th>Name</th>
<th>slave_load_tmpdir</th>
</tr>
</thead>
</table>

**Variable Scope**

| Global |

**Permitted Values**

<table>
<thead>
<tr>
<th>Type</th>
<th>directory name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>/tmp</td>
</tr>
</tbody>
</table>

The name of the directory where the slave creates temporary files. This option is by default equal to the value of the `tmpdir` system variable. When the slave SQL thread replicates a `LOAD DATA INFILE` statement, it extracts the file to be loaded from the relay log into temporary files, and then loads these into the table. If the file loaded on the master is huge, the temporary files on the slave are huge, too. Therefore, it might be advisable to use this option to tell the slave to put temporary files in a directory located in some file system that has a lot of available space. In that case, the relay logs are huge as well, so you might also want to use the `--relay-log` option to place the relay logs in that file system.

The directory specified by this option should be located in a disk-based file system (not a memory-based file system) because the temporary files used to replicate `LOAD DATA INFILE` must survive machine restarts. The directory also should not be one that is cleared by the operating system during the system startup process.

• **--slave-net-timeout=seconds**

---

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave-net-timeout=#</th>
</tr>
</thead>
</table>

**System Variable**

<table>
<thead>
<tr>
<th>Name</th>
<th>slave_net_timeout</th>
</tr>
</thead>
</table>

**Variable Scope**

| Global |

**Permitted Values**

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>3600</td>
</tr>
<tr>
<td>Min Value</td>
<td>1</td>
</tr>
</tbody>
</table>

The number of seconds to wait for more data from the master before the slave considers the connection broken, aborts the read, and tries to reconnect. The first retry occurs immediately after the timeout. The interval between retries is controlled by the `MASTER_CONNECT_RETRY` option for the `CHANGE MASTER TO` statement or `--master-connect-retry` option, and the number of reconnection attempts is limited by the `--master-retry-count` option. The default is 3600 seconds (one hour).

• **--slave-skip-errors=[err_code1, err_code2, ...|all]**

*(MySQL Cluster NDB 7.0.33 and later; MySQL Cluster NDB 7.1.22 and later:) --slave-skip-errors=[err_code1, err_code2, ...|all|ddl_exist_errors]*

---

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave-skip-errors=name</th>
</tr>
</thead>
</table>

**System Variable**

<table>
<thead>
<tr>
<th>Name</th>
<th>slave_skip_errors</th>
</tr>
</thead>
</table>

52
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Global Scope</th>
<th>Dynamic Variable</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>[list of error codes]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>all</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.61-ndb-7.0.33,5.1.61-ndb-7.1.22)</th>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>[list of error codes]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>all</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ddl_exist_errors</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Normally, replication stops when an error occurs on the slave. This gives you the opportunity to resolve the inconsistency in the data manually. This option tells the slave SQL thread to continue replication when a statement returns any of the errors listed in the option value.

Do not use this option unless you fully understand why you are getting errors. If there are no bugs in your replication setup and client programs, and no bugs in MySQL itself, an error that stops replication should never occur. Indiscriminate use of this option results in slaves becoming hopelessly out of synchrony with the master, with you having no idea why this has occurred.

**Note**

Prior to MySQL 5.1.35, this option had no effect with row-based logging. (Bug #39393)

For error codes, you should use the numbers provided by the error message in your slave error log and in the output of `SHOW SLAVE STATUS`. Errors, Error Codes, and Common Problems, lists server error codes.

You can also (but should not) use the very nonrecommended value of all to cause the slave to ignore all error messages and keeps going regardless of what happens. Needless to say, if you use all, there are no guarantees regarding the integrity of your data. Please do not complain (or file bug reports) in this case if the slave's data is not anywhere close to what it is on the master. **You have been warned.**

For MySQL Cluster Replication in MySQL Cluster NDB 7.0 beginning with version 7.0.33 and MySQL Cluster NDB 7.1 beginning with version 7.1.22, an additional shorthand value `ddl_exist_errors` is supported for use with the enhanced failover mechanism which is implemented in that and later version of MySQL Cluster NDB 7.2. This value is equivalent to the error code list `1007, 1008, 1050, 1051, 1054, 1060, 1061, 1068, 1094, 1146`. **This value is not supported by the mysql binary included with the MySQL Server 5.1 distribution.** (Bug #11762277, Bug #54854) For more information, see Implementing Failover with MySQL Cluster Replication.

Examples:

```
--slave-skip-errors=1062,1053
```
System Variables Used on Replication Slaves

The following list describes system variables for controlling replication slave servers. They can be set at server startup and some of them can be changed at runtime using `SET`. Server options used with replication slaves are listed earlier in this section.

- **init_slave**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--init-slave=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>init_slave</td>
</tr>
<tr>
<td></td>
<td>Variable Scope</td>
</tr>
<tr>
<td></td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>

This variable is similar to `init_connect`, but is a string to be executed by a slave server each time the SQL thread starts. The format of the string is the same as for the `init_connect` variable.

**Note**
The SQL thread sends an acknowledgment to the client before it executes `init_slave`. Therefore, it is not guaranteed that `init_slave` has been executed when `START SLAVE` returns. See `START SLAVE Syntax`, for more information.

- **relay_log**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--relay-log=file_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>relay_log</td>
</tr>
<tr>
<td></td>
<td>Variable Scope</td>
</tr>
<tr>
<td></td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>file name</td>
</tr>
</tbody>
</table>

The name of the relay log file.

- **relay_log_index**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--relay-log-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>relay_log_index</td>
</tr>
<tr>
<td></td>
<td>Variable Scope</td>
</tr>
<tr>
<td></td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable</td>
</tr>
<tr>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>file name</td>
</tr>
</tbody>
</table>
Replication Slave Options and Variables

| Default | *host_name*-relay-bin.index |

The name of the relay log index file. The default name is `host_name-relay-bin.index` in the data directory, where `host_name` is the name of the slave server.

- **relay_log_info_file**

  | Command-Line Format | --relay-log-info-file=file_name |
  | System Variable | Name | relay_log_info_file |
  | Variable Scope | Global |
  | Dynamic Variable | No |
  | Permitted Values | Type | file name |
  | | Default | relay-log.info |

The name of the file in which the slave records information about the relay logs. The default name is `relay-log.info` in the data directory.

- **rpl_recovery_rank**

  This variable is unused, and is removed in MySQL 5.6.

- **slave_compressed_protocol**

  | Command-Line Format | --slave_compressed_protocol |
  | System Variable | Name | slave_compressed_protocol |
  | Variable Scope | Global |
  | Dynamic Variable | Yes |
  | Permitted Values | Type | boolean |
  | | Default | OFF |

Whether to use compression of the slave/master protocol if both the slave and the master support it.

- **slave_exec_mode**

  | Introduced | 5.1.23-ndb-6.2.14, 5.1.23-ndb-6.3.11, 5.1.24 |
  | Command-Line Format | --slave-exec-mode=mode |
  | System Variable | Name | slave_exec_mode |
  | Variable Scope | Global |
  | Dynamic Variable | Yes |
  | Permitted Values | Type | enumeration |
  | | Default | STRICT (ALL) |
  | | Default | IDEMPOTENT (NDB) |
Replication Slave Options and Variables

| Valid Values | IDEMPOTENT | STRICT |

Controls whether IDEMPOTENT or STRICT mode is used in replication conflict resolution and error checking. IDEMPOTENT mode causes suppression of duplicate-key and no-key-found errors.

This mode is needed for multi-master replication, circular replication, and some other special replication scenarios for MySQL Cluster Replication. (See MySQL Cluster Replication: Multi-Master and Circular Replication, and MySQL Cluster Replication Conflict Resolution, for more information.) The `mysqld` supplied with MySQL Cluster ignores any value explicitly set for `slave_exec_mode`, and always treats it as IDEMPOTENT.

In MySQL Server 5.1, STRICT mode is the default value. This should not be changed; currently, IDEMPOTENT mode is supported only by NDB.

- **slave_max_allowed_packet**

<table>
<thead>
<tr>
<th>Introduced</th>
<th>5.1.64</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>Yes</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Min Value</td>
</tr>
<tr>
<td></td>
<td>Max Value</td>
</tr>
</tbody>
</table>

In MySQL 5.1.64 and later, this variable sets the maximum packet size for the slave SQL and I/O threads, so that large updates using row-based replication do not cause replication to fail because an update exceeded `max_allowed_packet`.

This global variable always has a value that is a positive integer multiple of 1024; if you set it to some value that is not, the value is rounded down to the next highest multiple of 1024 for it is stored or used; setting `slave_max_allowed_packet` to 0 causes 1024 to be used. (A truncation warning is issued in all such cases.) The default and maximum value is 1073741824 (1 GB); the minimum is 1024.

`slave_max_allowed_packet` can also be set at startup, using the `--slave-max-allowed-packet` option.

- **slave_load_tmpdir**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave-load-tmpdir=dir_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>No</td>
</tr>
</tbody>
</table>
Replication Slave Options and Variables

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>directory name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td>/tmp</td>
</tr>
</tbody>
</table>

The name of the directory where the slave creates temporary files for replicating `LOAD DATA INFILE` statements.

- **slave_net_timeout**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave-net-timeout=#</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name: <code>slave_net_timeout</code></td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td></td>
<td>3600</td>
</tr>
<tr>
<td>Min Value</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

The number of seconds to wait for more data from a master/slave connection before aborting the read. Before MySQL 5.1.41, this timeout applies only to TCP/IP connections, not to connections made through Unix socket files, named pipes, or shared memory.

- **slave_skip_errors**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave-skip-errors=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name: <code>slave_skip_errors</code></td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.61-ndb-7.0.33,5.1.61-ndb-7.1.22)</th>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid Values</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[list of error codes]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (5.1.61-5.1.7,5.1.61-ndb-7.1.22)</th>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid Values</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[list of error codes]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ddl_exist_errors</td>
<td></td>
</tr>
</tbody>
</table>

Normally, replication stops when an error occurs on the slave. This gives you the opportunity to resolve the inconsistency in the data manually. This variable tells the slave SQL thread to continue replication when a statement returns any of the errors listed in the variable value.
Replication Slave Options and Variables

- **slave_transaction_retries**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--slave_transaction_retries=#</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>slave_transaction_retries</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Min Value</td>
</tr>
<tr>
<td></td>
<td>Max Value</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Min Value</td>
</tr>
<tr>
<td></td>
<td>Max Value</td>
</tr>
</tbody>
</table>

If a replication slave SQL thread fails to execute a transaction because of an InnoDB deadlock or because the transaction's execution time exceeded InnoDB's `innodb_lock_wait_timeout` or NDBCLUSTER's `TransactionDeadlockDetectionTimeout` or `TransactionInactiveTimeout`, it automatically retries `slave_transaction_retries` times before stopping with an error. The default value is 10.

- **slave_type_conversions**

| Introduced | 5.1.44-ndb-6.3.33, 5.1.44-ndb-7.0.14, 5.1.44-ndb-7.1.3 |
| Command-Line Format | --slave_type_conversions=set |
| System Variable | Name                          |
|                  | slave_type_conversions        | Global |
|                  | Dynamic Variable              | No     |
| Permitted Values | Type                          | set |
|                  | Default                       |       |
|                  | Valid Values                  | ALL_LOSSY |
|                  |                               | ALL_NON_LOSSY |

Controls the type conversion mode in effect on the slave when using MySQL Cluster Replication. Its value is a comma-delimited set of zero or more elements from the list: ALL_LOSSY, ALL_NON_LOSSY. Set this variable to an empty string to disallow type conversions between the master and the slave. Changes require a restart of the slave to take effect.
For additional information on type conversion modes applicable to MySQL Cluster replication attribute promotion and demotion, see Attribute promotion and demotion (MySQL Cluster).

This variable was added in MySQL Cluster NDB 6.3.33, 7.0.14, and 7.1.3.

- **sql_slave_skip_counter**

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Name</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Scope</td>
<td>sql_slave_skip_counter</td>
<td></td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Permitted Values</td>
<td></td>
<td>integer</td>
</tr>
</tbody>
</table>

The number of events from the master that a slave server should skip.

**Important**

If skipping the number of events specified by setting this variable would cause the slave to begin in the middle of an event group, the slave continues to skip until it finds the beginning of the next event group and begins from that point. For more information, see SET GLOBAL sql_slave_skip_counter Syntax.

### 2.3.4 Binary Log Options and Variables

#### Startup Options Used with Binary Logging

#### System Variables Used with Binary Logging

You can use the `mysqld` options and system variables that are described in this section to affect the operation of the binary log as well as to control which statements are written to the binary log. For additional information about the binary log, see The Binary Log. For additional information about using MySQL server options and system variables, see Server Command Options, and Server System Variables.

#### Startup Options Used with Binary Logging

The following list describes startup options for enabling and configuring the binary log. System variables used with binary logging are discussed later in this section.

- **--binlog-row-event-max-size=N**

<table>
<thead>
<tr>
<th>Introduced</th>
<th>5.1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--binlog-row-event-max-size=#</code></td>
</tr>
<tr>
<td>Permitted Values (32-bit platforms)</td>
<td>Type: integer</td>
</tr>
<tr>
<td>Default</td>
<td>1024</td>
</tr>
<tr>
<td>Min Value</td>
<td>256</td>
</tr>
<tr>
<td>Max Value</td>
<td>4294967295</td>
</tr>
<tr>
<td>Permitted Values (64-bit platforms)</td>
<td>Type: integer</td>
</tr>
<tr>
<td>Default</td>
<td>1024</td>
</tr>
</tbody>
</table>
### Binary Log Options and Variables

<table>
<thead>
<tr>
<th>Min Value</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Value</td>
<td>18446744073709547520</td>
</tr>
</tbody>
</table>

Specify the maximum size of a row-based binary log event, in bytes. Rows are grouped into events smaller than this size if possible. The value should be a multiple of 256. The default is 1024. See Section 2.2, “Replication Formats”. This option was added in MySQL 5.1.5.

- **--log-bin[=base_name]**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--log-bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>No</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
</tbody>
</table>

Enable binary logging. The server logs all statements that change data to the binary log, which is used for backup and replication. See The Binary Log.

The option value, if given, is the base name for the log sequence. The server creates binary log files in sequence by adding a numeric suffix to the base name. It is recommended that you specify a base name (see Known Issues in MySQL, for the reason). Otherwise, MySQL uses `host_name-bin` as the base name.

Setting this option causes the `log_bin` system variable to be set to ON (or 1), and not to the base name. This is a known issue; see Bug #19614 for more information.

- **--log-bin-index[=file_name]**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--log-bin-index=file_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
</tbody>
</table>

The index file for binary log file names. See The Binary Log. If you omit the file name, and if you did not specify one with --log-bin, MySQL uses `host_name-bin.index` as the file name.

- **--log-bin-trust-function-creators[={0|1}]**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--log-bin-trust-function-creators</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Variable</td>
<td>Name</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>Yes</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>Type</td>
</tr>
<tr>
<td>Default</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
This option sets the corresponding `log_bin_trust_function_creators` system variable. If no argument is given, the option sets the variable to 1. `log_bin_trust_function_creators` affects how MySQL enforces restrictions on stored function and trigger creation. See Binary Logging of Stored Programs.

**Note**

Previously, this option was known as `--log-bin-trust-routine-creators`, which is now deprecated.

- `--log-bin-use-v1-row-events[=0|1]`

<table>
<thead>
<tr>
<th>Introduced</th>
<th>5.1.56-ndb-7.0.27, 5.1.56-ndb-7.1.16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td>`--log-bin-use-v1-row-events[=0</td>
</tr>
<tr>
<td>System Variable</td>
<td>Name <code>log_bin_use_v1_row_events</code></td>
</tr>
<tr>
<td></td>
<td>Variable Scope Global</td>
</tr>
<tr>
<td></td>
<td>Dynamic No</td>
</tr>
<tr>
<td>Permitted Values (&gt;=</td>
<td>Type boolean</td>
</tr>
<tr>
<td>5.1.56-ndb-7.0.27)</td>
<td>Default 1</td>
</tr>
<tr>
<td>Permitted Values (&gt;=</td>
<td>Type boolean</td>
</tr>
<tr>
<td>5.1.56-ndb-7.1.16)</td>
<td>Default 1</td>
</tr>
</tbody>
</table>

Version 2 binary log row events are available beginning with MySQL Cluster NDB 7.0.27 and MySQL Cluster NDB 7.1.16; however, Version 2 events cannot be read by previous MySQL Cluster releases. Setting `--log-bin-use-v1-row-events` to 0 causes mysqld to use Version 2 binary log events. Setting this option to 1 (the default) causes mysqld to write the binary log using Version 1 logging events, which is the only version of binary log events used in previous releases, and thus produce binary logs that can be read by older slaves.

The value used for this option can be obtained from the read-only `log_bin_use_v1_row_events` system variable.

`--log-bin-use-v1-row-events` is chiefly of interest when setting up replication conflict detection and resolution using `NDB$EPOCH_TRANS()` as the conflict detection function, which requires Version 2 binary log row events. Thus, this option and `--ndb-log-transaction-id` are not compatible.

**Note**

MySQL Cluster NDB 7.2.1 and later use Version 2 binary log events by default (and so the default value for this option changes to 0 in those versions). You should keep this mind when planning upgrades for setups using MySQL Cluster Replication.

`--log-bin-use-v1-row-events` is not supported in mainline MySQL Server 5.1 releases.

For more information, see MySQL Cluster Replication Conflict Resolution.

**Statement selection options.** The options in the following list affect which statements are written to the binary log, and thus sent by a replication master server to its slaves. There are also options for slave
servers that control which statements received from the master should be executed or ignored. For details, see Section 2.3.3, “Replication Slave Options and Variables”.

- **--binlog-do-db=db_name**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--binlog-do-db=name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type string</td>
</tr>
</tbody>
</table>

This option affects binary logging in a manner similar to the way that **--replicate-do-db** affects replication.

The effects of this option depend on whether the statement-based or row-based logging format is in use, in the same way that the effects of **--replicate-do-db** depend on whether statement-based or row-based replication is in use. You should keep in mind that the format used to log a given statement may not necessarily be the same as that indicated by the value of **binlog_format**. For example, DDL statements such as **CREATE TABLE** and **ALTER TABLE** are always logged as statements, without regard to the logging format in effect, so the following statement-based rules for **--binlog-do-db** always apply in determining whether or not the statement is logged.

**Statement-based logging.** Only those statements are written to the binary log where the default database (that is, the one selected by **USE**) is **db_name**. To specify more than one database, use this option multiple times, once for each database; however, doing so does **not** cause cross-database statements such as **UPDATE some_db.some_table** SET foo='bar' to be logged while a different database (or no database) is selected.

**Warning**

To specify multiple databases you **must** use multiple instances of this option. Because database names can contain commas, the list will be treated as the name of a single database if you supply a comma-separated list.

An example of what does not work as you might expect when using statement-based logging: If the server is started with **--binlog-do-db=sales** and you issue the following statements, the **UPDATE** statement is **not** logged:

```
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

The main reason for this “just check the default database” behavior is that it is difficult from the statement alone to know whether it should be replicated (for example, if you are using multiple-table **DELETE** statements or multiple-table **UPDATE** statements that act across multiple databases). It is also faster to check only the default database rather than all databases if there is no need.

Another case which may not be self-evident occurs when a given database is replicated even though it was not specified when setting the option. If the server is started with **--binlog-do-db=sales**, the following **UPDATE** statement is logged even though **prices** was not included when setting **--binlog-do-db**:

```
USE sales;
UPDATE prices.discounts SET percentage = percentage + 10;
```

Because **sales** is the default database when the **UPDATE** statement is issued, the **UPDATE** is logged.
**Row-based logging.** Logging is restricted to database `db_name`. Only changes to tables belonging to `db_name` are logged; the default database has no effect on this. Suppose that the server is started with `--binlog-do-db=sales` and row-based logging is in effect, and then the following statements are executed:

```sql
USE prices;
UPDATE sales.febuary SET amount=amount+100;
```

The changes to the `febuary` table in the `sales` database are logged in accordance with the `UPDATE` statement; this occurs whether or not the `USE` statement was issued. However, when using the row-based logging format and `--binlog-do-db=sales`, changes made by the following `UPDATE` are not logged:

```sql
USE prices;
UPDATE prices.march SET amount=amount-25;
```

Even if the `USE prices` statement were changed to `USE sales`, the `UPDATE` statement's effects would still not be written to the binary log.

Another important difference in `--binlog-do-db` handling for statement-based logging as opposed to the row-based logging occurs with regard to statements that refer to multiple databases. Suppose that the server is started with `--binlog-do-db=(db1)`, and the following statements are executed:

```sql
USE db1;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

If you are using statement-based logging, the updates to both tables are written to the binary log. However, when using the row-based format, only the changes to `table1` are logged; `table2` is in a different database, so it is not changed by the `UPDATE`. Now suppose that, instead of the `USE db1` statement, a `USE db4` statement had been used:

```sql
USE db4;
UPDATE db1.table1 SET col1 = 10, db2.table2 SET col2 = 20;
```

In this case, the `UPDATE` statement is not written to the binary log when using statement-based logging. However, when using row-based logging, the change to `table1` is logged, but not that to `table2`—in other words, only changes to tables in the database named by `--binlog-do-db` are logged, and the choice of default database has no effect on this behavior.

- **--binlog-ignore-db=db_name**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th><code>--binlog-ignore-db=name</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td><code>string</code></td>
</tr>
</tbody>
</table>

This option affects binary logging in a manner similar to the way that `--replicate-ignore-db` affects replication.

The effects of this option depend on whether the statement-based or row-based logging format is in use, in the same way that the effects of `--replicate-ignore-db` depend on whether statement-based or row-based replication is in use. You should keep in mind that the format used to log a given statement may not necessarily be the same as that indicated by the value of `binlog_format`. For example, DDL statements such as `CREATE TABLE` and `ALTER TABLE` are always logged as statements, without
regard to the logging format in effect, so the following statement-based rules for --binlog-ignore-db always apply in determining whether or not the statement is logged.

**Statement-based logging.**  Tells the server to not log any statement where the default database (that is, the one selected by `USE`) is *db_name*.

Prior to MySQL 5.1.70, this option caused any statements containing fully qualified table names not to be logged if there was no default database specified (that is, when `SELECT DATABASE()` returned `NULL`). In MySQL 5.1.70 and later, when there is no default database, no --binlog-ignore-db options are applied, and such statements are always logged. (Bug #11829838, Bug #60188)

**Row-based format.**  Tells the server not to log updates to any tables in the database *db_name*. The current database has no effect.

When using statement-based logging, the following example does not work as you might expect. Suppose that the server is started with `--binlog-ignore-db=sales` and you issue the following statements:

```
USE prices;
UPDATE sales.january SET amount=amount+1000;
```

The `UPDATE` statement *is* logged in such a case because --binlog-ignore-db applies only to the default database (determined by the `USE` statement). Because the `sales` database was specified explicitly in the statement, the statement has not been filtered. However, when using row-based logging, the `UPDATE` statement's effects are *not* written to the binary log, which means that no changes to the `sales.january` table are logged; in this instance, `--binlog-ignore-db=sales` causes all changes made to tables in the master's copy of the `sales` database to be ignored for purposes of binary logging.

To specify more than one database to ignore, use this option multiple times, once for each database. Because database names can contain commas, the list will be treated as the name of a single database if you supply a comma-separated list.

You should not use this option if you are using cross-database updates and you do not want these updates to be logged.

**Testing and debugging options.**  The following binary log options are used in replication testing and debugging. They are not intended for use in normal operations.

- **--max-binlog-dump-events=N**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--max-binlog-dump-events=#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type: integer</td>
</tr>
<tr>
<td></td>
<td>Default: 0</td>
</tr>
</tbody>
</table>

This option is used internally by the MySQL test suite for replication testing and debugging.

- **--sporadic-binlog-dump-fail**

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>--sporadic-binlog-dump-fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permitted Values</td>
<td>Type: boolean</td>
</tr>
<tr>
<td></td>
<td>Default: FALSE</td>
</tr>
</tbody>
</table>

This option is used internally by the MySQL test suite for replication testing and debugging.
System Variables Used with Binary Logging

The following list describes system variables for controlling binary logging. They can be set at server startup and some of them can be changed at runtime using `SET`. Server options used to control binary logging are listed earlier in this section. For information about the `sql_log_bin` and `sql_log_off` variables, see Server System Variables.

- `binlog_cache_size`

<table>
<thead>
<tr>
<th>Command-Line Format</th>
<th>Name</th>
<th>Scope</th>
<th>Dynamic Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--binlog_cache_size=#</code></td>
<td><code>binlog_cache_size</code></td>
<td>Global</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (32-bit platforms)</th>
<th>Type</th>
<th>Default</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>integer</td>
<td>32768</td>
<td>4096</td>
<td>4294967295</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (64-bit platforms)</th>
<th>Type</th>
<th>Default</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>integer</td>
<td>32768</td>
<td>4096</td>
<td>18446744073709547520</td>
</tr>
</tbody>
</table>

The size of the cache to hold the SQL statements for the binary log during a transaction. A binary log cache is allocated for each client if the server supports any transactional storage engines and if the server has the binary log enabled (`--log-bin` option). If you often use large, multiple-statement transactions, you can increase this cache size to get better performance. The `Binlog_cache_use` and `Binlog_cache_disk_use` status variables can be useful for tuning the size of this variable. See The Binary Log.

- `binlog_direct_non_transactional_updates`

<table>
<thead>
<tr>
<th>Introduced</th>
<th>Command-Line Format</th>
<th>System Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.44</td>
<td><code>--binlog_direct_non_transactional_updates[=value]</code></td>
<td><code>binlog_direct_non_transactional_updates</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable Scope</th>
<th>Dynamic Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global, Session</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Due to concurrency issues, a slave can become inconsistent when a transaction contains updates to both transactional and nontransactional tables. MySQL tries to preserve causality among these statements by writing nontransactional statements to the transaction cache, which is flushed upon commit. However, problems arise when modifications done to nontransactional tables on behalf of a transaction become immediately visible to other connections because these changes may not be written immediately into the binary log.

Beginning with MySQL 5.1.44, the `binlog_direct_non_transactional_updates` variable offers one possible workaround to this issue. By default, this variable is disabled. Enabling `binlog_direct_non_transactional_updates` causes updates to nontransactional tables to be written directly to the binary log, rather than to the transaction cache.

`binlog_direct_non_transactional_updates works only for statements that are replicated using the statement-based binary logging format; that is, it works only when the value of `binlog_format` is `STATEMENT`, or when `binlog_format` is `MIXED` and a given statement is being replicated using the statement-based format. This variable has no effect when the binary log format is `ROW`, or when `binlog_format` is set to `MIXED` and a given statement is replicated using the row-based format.

### Important

Before enabling this variable, you must make certain that there are no dependencies between transactional and nontransactional tables; an example of such a dependency would be the statement `INSERT INTO myisam_table SELECT * FROM innodb_table`. Otherwise, such statements are likely to cause the slave to diverge from the master.

- `binlog_format`

<table>
<thead>
<tr>
<th>Introduced</th>
<th>5.1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--binlog-format=format</code></td>
</tr>
<tr>
<td>System Variable (&gt;= 5.1.8)</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Variable Scope</td>
</tr>
<tr>
<td></td>
<td>Dynamic Variable</td>
</tr>
<tr>
<td>Permitted Values (&gt;= 5.1.5, &lt;= 5.1.7)</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Valid Values</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Permitted Values (&gt;= 5.1.8, &lt;= 5.1.11)</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Valid Values</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Permitted Values (&gt;= 5.1.12, &lt;= 5.1.28)</td>
<td>Type</td>
</tr>
<tr>
<td></td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Valid Values</td>
</tr>
<tr>
<td>Permitted Values (&gt;= 5.1.29)</td>
<td>Type</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Default</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>Valid Values</td>
<td>STATEMENT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.14-ndb-6.1.0, &lt;= 5.1.15-ndb-6.1.23)</th>
<th>Type</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>MIXED</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>ROW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.16-ndb-6.2.0, &lt;= 5.1.34-ndb-6.2.18)</th>
<th>Type</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>MIXED</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>ROW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.19-ndb-6.3.0, &lt;= 5.1.73-ndb-6.3.54)</th>
<th>Type</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>MIXED</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>ROW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.32-ndb-7.0.4, &lt;= 5.1.73-ndb-7.0.42)</th>
<th>Type</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>MIXED</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>ROW</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permitted Values (&gt;= 5.1.39-ndb-7.1.0, &lt;= 5.1.51-ndb-7.2.0)</th>
<th>Type</th>
<th>enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>MIXED</td>
<td></td>
</tr>
<tr>
<td>Valid Values</td>
<td>ROW</td>
<td></td>
</tr>
</tbody>
</table>

This variable sets the binary logging format, and can be any one of STATEMENT, ROW, or MIXED. See Section 2.2, “Replication Formats”. `binlog_format` is set by the `--binlog-format` option at startup, or by the `binlog_format` variable at runtime.

**Note**

While you can change the logging format at runtime, it is **not** recommended that you change it while replication is ongoing. This is due in part to the fact...
that slaves do not honor the master’s `binlog_format` setting; a given MySQL Server can change only its own logging format.

The startup variable was added in MySQL 5.1.5, and the runtime variable in MySQL 5.1.8. MIXED was added in MySQL 5.1.8.

`STATEMENT` was used by default prior to MySQL 5.1.12; in MySQL 5.1.12, the default was changed to MIXED. In MySQL 5.1.29, the default was changed back to `STATEMENT`.

You must have the SUPER privilege to set the global `binlog_format` value. Starting with MySQL 5.1.29, you must have the SUPER privilege to set either the global or session `binlog_format` value. (Bug #39106)

The rules governing when changes to this variable take effect and how long the effect lasts are the same as for other MySQL server system variables. See SET Syntax, for more information.

When MIXED is specified, statement-based replication is used, except for cases where only row-based replication is guaranteed to lead to proper results. For example, this happens when statements contain user-defined functions (UDF) or the UUID() function. An exception to this rule is that MIXED always uses statement-based replication for stored functions and triggers.

There are exceptions when you cannot switch the replication format at runtime:

- From within a stored function or a trigger.
- If the NDBCLUSTER storage engine is enabled.
- If the session is currently in row-based replication mode and has open temporary tables.

Trying to switch the format in those cases results in an error.

Before MySQL 5.1.8, switching to row-based replication format would implicitly set `--log-bin-trust-function-creators=1` and `--innodb_locks_unsafe_for_binlog`. This does not occur for MySQL 5.1.8 and later.

The binary log format affects the behavior of the following server options:

- `--replicate-do-db`
- `--replicate-ignore-db`
- `--binlog-do-db`
- `--binlog-ignore-db`

These effects are discussed in detail in the descriptions of the individual options.

- `log_bin`

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Dynamic Variable</td>
<td>No</td>
</tr>
</tbody>
</table>
Whether the binary log is enabled. If the `--log-bin` option is used, then the value of this variable is **ON**; otherwise it is **OFF**. This variable reports only on the status of binary logging (enabled or disabled); it does not actually report the value to which `--log-bin` is set.

See The Binary Log.

- **log_bin_use_v1_row_events**

<table>
<thead>
<tr>
<th>Name</th>
<th>log_bin_use_v1_row_events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduced</td>
<td>5.1.56-ndb-7.0.27, 5.1.56-ndb-7.1.16</td>
</tr>
<tr>
<td>Command-Line Format</td>
<td>`--log-bin-use-v1-row-events[={0</td>
</tr>
<tr>
<td>System Variable</td>
<td>log_bin_use_v1_row_events</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>&gt;= 5.1.56-ndb-7.0.27</td>
</tr>
<tr>
<td>Default</td>
<td>1</td>
</tr>
<tr>
<td>Permitted Values</td>
<td>&gt;= 5.1.56-ndb-7.1.16</td>
</tr>
<tr>
<td>Default</td>
<td>1</td>
</tr>
</tbody>
</table>

  Shows whether Version 2 binary logging, available beginning with MySQL Cluster NDB 7.0.27 and MySQL Cluster NDB 7.1.16, is in use. A value of 1 shows that the server is writing the binary log using Version 1 logging events (the only version of binary log events used in previous releases), and thus producing a binary log that can be read by older slaves. 0 indicates that Version 2 binary log events are in use.

  This variable is read-only. To switch between Version 1 and Version 2 binary event binary logging, it is necessary to restart `mysqld` with the `--log-bin-use-v1-row-events` option.

  Other than when performing upgrades of MySQL Cluster Replication, `--log-bin-use-v1-row-events` is chiefly of interest when setting up replication conflict detection and resolution using NDB `$EPOCH_TRANS()`, which requires Version 2 binary row event logging. Thus, this option and `--ndb-log-transaction-id` are not compatible.

  **Note**

  MySQL Cluster NDB 7.2.1 and later use Version 2 binary log row events by default (and so the default value for this variable changes to 0 in those versions). You should keep this mind when planning upgrades for setups using MySQL Cluster Replication.

  This variable is not supported in mainline MySQL Server 5.1.

  For more information, see MySQL Cluster Replication Conflict Resolution.

- **log_slave_updates**

<table>
<thead>
<tr>
<th>Name</th>
<th>log_slave_updates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command-Line Format</td>
<td><code>--log-slave-updates</code></td>
</tr>
<tr>
<td>System Variable</td>
<td>log_slave_updates</td>
</tr>
<tr>
<td>Variable Scope</td>
<td>Global</td>
</tr>
</tbody>
</table>
Binary Log Options and Variables

<table>
<thead>
<tr>
<th>Dynamic Variable</th>
<th>Permitted Values</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>boolean</td>
<td>FALSE</td>
<td></td>
</tr>
</tbody>
</table>

Whether updates received by a slave server from a master server should be logged to the slave's own binary log. Binary logging must be enabled on the slave for this variable to have any effect. See Section 2.3, “Replication and Binary Logging Options and Variables”.

- **max_binlog_cache_size**

  **Command-Line Format**

  `--max_binlog_cache_size=#`

  **System Variable**

  - **Name**: `max_binlog_cache_size`
  - **Variable Scope**: Global
  - **Dynamic Variable**: Yes

  **Permitted Values (>= 5.1.36)**

  - **Type**: integer
  - **Default**: 18446744073709547520
  - **Min Value**: 4096
  - **Max Value**: 18446744073709547520

  **Permitted Values (32-bit platforms, <= 5.1.35)**

  - **Type**: integer
  - **Default**: 4294967295
  - **Min Value**: 4096
  - **Max Value**: 4294967295

  **Permitted Values (64-bit platforms, <= 5.1.35)**

  - **Type**: integer
  - **Default**: 18446744073709547520
  - **Min Value**: 4096
  - **Max Value**: 18446744073709547520

  If a multiple-statement transaction requires more than this many bytes of memory, the server generates a **Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage** error. The minimum value is 4096. The maximum and default values are 4GB on 32-bit platforms and 16EB (exabytes) on 64-bit platforms. As of MySQL 5.1.36, the maximum value is 16EB (exabytes) on all platforms. The maximum recommended value is 4GB; this is due to the fact that MySQL currently cannot work with binary log positions greater than 4GB.

  In MySQL 5.1, a change in **max_binlog_cache_size** takes immediate effect for all active sessions.

- **max_binlog_size**

  **Command-Line Format**

  `--max_binlog_size=#`
If a write to the binary log causes the current log file size to exceed the value of this variable, the server rotates the binary logs (closes the current file and opens the next one). The minimum value is 4096 bytes. The maximum and default value is 1GB.

A transaction is written in one chunk to the binary log, so it is never split between several binary logs. Therefore, if you have big transactions, you might see binary log files larger than `max_binlog_size`. If `max_relay_log_size` is 0, the value of `max_binlog_size` applies to relay logs as well.

- `sync_binlog`

For 32-bit platforms:

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>Default</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>integer</td>
<td>0</td>
<td>0</td>
<td>4294967295</td>
</tr>
</tbody>
</table>

For 64-bit platforms:

<table>
<thead>
<tr>
<th>Permitted Values</th>
<th>Type</th>
<th>Default</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>integer</td>
<td>0</td>
<td>0</td>
<td>18446744073709547520</td>
</tr>
</tbody>
</table>

If the value of this variable is greater than 0, the MySQL server synchronizes its binary log to disk (using `fdatasync()`) after every `sync_binlog` writes to the binary log. There is one write to the binary log per statement if autocommit is enabled, and one write per transaction otherwise. The default value of `sync_binlog` is 0, which does no synchronizing to disk—in this case, the server relies on the operating system to flush the binary log's contents from time to time as for any other file. A value of 1 is the safest choice because in the event of a crash you lose at most one statement or transaction from the binary log.
Common Replication Administration Tasks

2.4 Common Replication Administration Tasks

Once replication has been started it should execute without requiring much regular administration. Depending on your replication environment, you will want to check the replication status of each slave periodically, daily, or even more frequently.

2.4.1 Checking Replication Status

The most common task when managing a replication process is to ensure that replication is taking place and that there have been no errors between the slave and the master. The primary statement for this is `SHOW SLAVE STATUS`, which you must execute on each slave:

```sql
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
Slave_IO_State: Waiting for master to send event
    Master_Host: master1
    Master_User: root
    Master_Port: 3306
    Connect_Retry: 60
    Master_Log_File: mysql-bin.000004
    Read_Master_Log_Pos: 931
    Relay_Log_File: slave1-relay-bin.000056
    Relay_Log_Pos: 950
    Slave_IO_Running: Yes
    Slave_SQL_Running: Yes
    Replicate_Do_DB:
    Replicate_Ignore_DB:
    Replicate_Do_Table:
    Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
    Replicate_Wild_Ignore_Table:
    Last_Errno: 0
    Last_Error:
    Skip_Counter: 0
    Exec_Master_Log_Pos: 931
    Relay_Log_Space: 1365
    Until_Condition: None
    Until_Log_File: 
    Until_Log_Pos: 0
    Master_SSL_Allowed: No
    Master_SSL_CA_File: 
    Master_SSL_CA_Path: 
    Master_SSL_Cert: 
    Master_SSLCipher: 
    Master_SSL_Cipher: 
    Seconds_Behind_Master: 0
    Master_SSL_Verify_Server_Cert: No
    Last_IO_Errno: 0
    Last_IO_Error: 
    Last_SQL_Errno: 0
    Last_SQL_Error: 
```

The key fields from the status report to examine are:

- **Slave_IO_State**: The current status of the slave. See Replication Slave I/O Thread States, and Replication Slave SQL Thread States, for more information.
• **Slave_IO_Running**: Whether the I/O thread for reading the master's binary log is running. Normally, you want this to be **Yes** unless you have not yet started replication or have explicitly stopped it with `STOP SLAVE`.

• **Slave_SQL_Running**: Whether the SQL thread for executing events in the relay log is running. As with the I/O thread, this should normally be **Yes**.

• **Last_IO_Error, Last_SQL_Error**: The last errors registered by the I/O and SQL threads when processing the relay log. Ideally these should be blank, indicating no errors.

• **Seconds_Behind_Master**: The number of seconds that the slave SQL thread is behind processing the master binary log. A high number (or an increasing one) can indicate that the slave is unable to handle events from the master in a timely fashion.

A value of 0 for **Seconds_Behind_Master** can usually be interpreted as meaning that the slave has caught up with the master, but there are some cases where this is not strictly true. For example, this can occur if the network connection between master and slave is broken but the slave I/O thread has not yet noticed this—that is, **slave_net_timeout** has not yet elapsed.

It is also possible that transient values for **Seconds_Behind_Master** may not reflect the situation accurately. When the slave SQL thread has caught up on I/O, **Seconds_Behind_Master** displays 0; but when the slave I/O thread is still queuing up a new event, **Seconds_Behind_Master** may show a large value until the SQL thread finishes executing the new event. This is especially likely when the events have old timestamps; in such cases, if you execute `SHOW SLAVE STATUS` several times in a relatively short period, you may see this value change back and forth repeatedly between 0 and a relatively large value.

Several pairs of fields provide information about the progress of the slave in reading events from the master binary log and processing them in the relay log:

• (**Master_Log_file, Read_Master_Log_Pos**): Coordinates in the master binary log indicating how far the slave I/O thread has read events from that log.

• (**Relay_Master_Log_File, Exec_Master_Log_Pos**): Coordinates in the master binary log indicating how far the slave SQL thread has executed events received from that log.

• (**Relay_Log_File, Relay_Log_Pos**): Coordinates in the slave relay log indicating how far the slave SQL thread has executed the relay log. These correspond to the preceding coordinates, but are expressed in slave relay log coordinates rather than master binary log coordinates.

On the master, you can check the status of connected slaves using `SHOW PROCESSLIST` to examine the list of running processes. Slave connections have **Binlog Dump** in the **Command** field:

```
mysql> SHOW PROCESSLIST \G;
*************************** 4. row ***************************
    Id: 10
    User: root
    Host: slave1:58371
    db: NULL
   Command: Binlog Dump
     Time: 777
   State: Has sent all binlog to slave; waiting for binlog to be updated
    Info: NULL
```

Because it is the slave that drives the replication process, very little information is available in this report.

For slaves that were started with the **--report-host** option and are connected to the master, the `SHOW SLAVE HOSTS` statement on the master shows basic information about the slaves. The output includes the ID of the slave server, the value of the **--report-host** option, the connecting port, and master ID:
2.4.2 Pausing Replication on the Slave

You can stop and start the replication of statements on the slave using the `STOP SLAVE` and `START SLAVE` statements.

To stop processing of the binary log from the master, use `STOP SLAVE`:

```
mysql> STOP SLAVE;
```

When replication is stopped, the slave I/O thread stops reading events from the master binary log and writing them to the relay log, and the SQL thread stops reading events from the relay log and executing them. You can pause the I/O or SQL thread individually by specifying the thread type:

```
mysql> STOP SLAVE IO_THREAD;
mysql> STOP SLAVE SQL_THREAD;
```

To start execution again, use the `START SLAVE` statement:

```
mysql> START SLAVE;
```

To start a particular thread, specify the thread type:

```
mysql> START SLAVE IO_THREAD;
mysql> START SLAVE SQL_THREAD;
```

For a slave that performs updates only by processing events from the master, stopping only the SQL thread can be useful if you want to perform a backup or other task. The I/O thread will continue to read events from the master but they are not executed. This makes it easier for the slave to catch up when you restart the SQL thread.

Stopping only the I/O thread enables the events in the relay log to be executed by the SQL thread up to the point where the relay log ends. This can be useful when you want to pause execution to catch up with events already received from the master, when you want to perform administration on the slave but also ensure that it has processed all updates to a specific point. This method can also be used to pause event receipt on the slave while you conduct administration on the master. Stopping the I/O thread but permitting the SQL thread to run helps ensure that there is not a massive backlog of events to be executed when replication is started again.
Chapter 3 Replication Solutions

Table of Contents

3.1 Using Replication for Backups ................................................................. 75
   3.1.1 Backing Up a Slave Using mysqldump ........................................... 76
   3.1.2 Backing Up Raw Data from a Slave ............................................... 76
   3.1.3 Backing Up a Master or Slave by Making It Read Only .................. 77
3.2 Using Replication with Different Master and Slave Storage Engines ............ 79
3.3 Using Replication for Scale-Out ............................................................ 80
3.4 Replicating Different Databases to Different Slaves ................................. 81
3.5 Improving Replication Performance ....................................................... 83
3.6 Switching Masters During Failover ......................................................... 84
3.7 Setting Up Replication Using SSL ......................................................... 86

Replication can be used in many different environments for a range of purposes. This section provides general notes and advice on using replication for specific solution types.

For information on using replication in a backup environment, including notes on the setup, backup procedure, and files to back up, see Section 3.1, “Using Replication for Backups”.

For advice and tips on using different storage engines on the master and slaves, see Section 3.2, “Using Replication with Different Master and Slave Storage Engines”.

Using replication as a scale-out solution requires some changes in the logic and operation of applications that use the solution. See Section 3.3, “Using Replication for Scale-Out”.

For performance or data distribution reasons, you may want to replicate different databases to different replication slaves. See Section 3.4, “Replicating Different Databases to Different Slaves”

As the number of replication slaves increases, the load on the master can increase and lead to reduced performance (because of the need to replicate the binary log to each slave). For tips on improving your replication performance, including using a single secondary server as a replication master, see Section 3.5, “Improving Replication Performance”.

For guidance on switching masters, or converting slaves into masters as part of an emergency failover solution, see Section 3.6, “Switching Masters During Failover”.

To secure your replication communication, you can use SSL to encrypt the communication channel. For step-by-step instructions, see Section 3.7, “Setting Up Replication Using SSL”.

3.1 Using Replication for Backups

To use replication as a backup solution, replicate data from the master to a slave, and then back up the data slave. The slave can be paused and shut down without affecting the running operation of the master, so you can produce an effective snapshot of “live” data that would otherwise require the master to be shut down.

How you back up a database depends on its size and whether you are backing up only the data, or the data and the replication slave state so that you can rebuild the slave in the event of failure. There are therefore two choices:

• If you are using replication as a solution to enable you to back up the data on the master, and the size of your database is not too large, the mysqldump tool may be suitable. See Section 3.1.1, “Backing Up a Slave Using mysqldump”.
Backing Up a Slave Using mysqldump

• For larger databases, where mysqldump would be impractical or inefficient, you can back up the raw data files instead. Using the raw data files option also means that you can back up the binary and relay logs that will enable you to recreate the slave in the event of a slave failure. For more information, see Section 3.1.2, “Backing Up Raw Data from a Slave”.

Another backup strategy, which can be used for either master or slave servers, is to put the server in a read-only state. The backup is performed against the read-only server, which then is changed back to its usual read/write operational status. See Section 3.1.3, “Backing Up a Master or Slave by Making It Read Only”.

3.1.1 Backing Up a Slave Using mysqldump

Using mysqldump to create a copy of a database enables you to capture all of the data in the database in a format that enables the information to be imported into another instance of MySQL Server (see mysqldump — A Database Backup Program). Because the format of the information is SQL statements, the file can easily be distributed and applied to running servers in the event that you need access to the data in an emergency. However, if the size of your data set is very large, mysqldump may be impractical.

When using mysqldump, you should stop replication on the slave before starting the dump process to ensure that the dump contains a consistent set of data:

1. Stop the slave from processing requests. You can stop replication completely on the slave using mysqladmin:

   shell> mysqladmin stop-slave

   Alternatively, you can stop only the slave SQL thread to pause event execution:

   shell> mysql -e 'STOP SLAVE SQL_THREAD;'

   This enables the slave to continue to receive data change events from the master’s binary log and store them in the relay logs using the I/O thread, but prevents the slave from executing these events and changing its data. Within busy replication environments, permitting the I/O thread to run during backup may speed up the catch-up process when you restart the slave SQL thread.

2. Run mysqldump to dump your databases. You may either dump all databases or select databases to be dumped. For example, to dump all databases:

   shell> mysqldump --all-databases > fulldb.dump

3. Once the dump has completed, start slave operations again:

   shell> mysqladmin start-slave

   In the preceding example, you may want to add login credentials (user name, password) to the commands, and bundle the process up into a script that you can run automatically each day.

   If you use this approach, make sure you monitor the slave replication process to ensure that the time taken to run the backup does not affect the slave’s ability to keep up with events from the master. See Section 2.4.1, “Checking Replication Status”. If the slave is unable to keep up, you may want to add another slave and distribute the backup process. For an example of how to configure this scenario, see Section 3.4, “Replicating Different Databases to Different Slaves”.

3.1.2 Backing Up Raw Data from a Slave

To guarantee the integrity of the files that are copied, backing up the raw data files on your MySQL replication slave should take place while your slave server is shut down. If the MySQL server is still running, background tasks may still be updating the database files, particularly those involving storage
engines with background processes such as InnoDB. With InnoDB, these problems should be resolved during crash recovery, but since the slave server can be shut down during the backup process without affecting the execution of the master it makes sense to take advantage of this capability.

To shut down the server and back up the files:

1. Shut down the slave MySQL server:
   
   ```shell
   mysqladmin shutdown
   ```

2. Copy the data files. You can use any suitable copying or archive utility, including cp, tar or WinZip. For example, assuming that the data directory is located under the current directory, you can archive the entire directory as follows:
   
   ```shell
   tar cf /tmp/dbbackup.tar ./data
   ```

3. Start the MySQL server again. Under Unix:
   
   ```shell
   mysqld_safe &
   ```
   
   Under Windows:
   
   ```
   C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld"
   ```

Normally you should back up the entire data directory for the slave MySQL server. If you want to be able to restore the data and operate as a slave (for example, in the event of failure of the slave), then in addition to the slave’s data, you should also back up the slave status files, master.info and relay-log.info, along with the relay log files. These files are needed to resume replication after you restore the slave’s data.

If you lose the relay logs but still have the relay-log.info file, you can check it to determine how far the SQL thread has executed in the master binary logs. Then you can use CHANGE MASTER TO with the MASTER_LOG_FILE and MASTER_LOG_POS options to tell the slave to re-read the binary logs from that point. This requires that the binary logs still exist on the master server.

If your slave is replicating LOAD DATA INFILE statements, you should also back up any SQL_LOAD-* files that exist in the directory that the slave uses for this purpose. The slave needs these files to resume replication of any interrupted LOAD DATA INFILE operations. The location of this directory is the value of the --slave-load-tmpdir option. If the server was not started with that option, the directory location is the value of the tmpdir system variable.

### 3.1.3 Backing Up a Master or Slave by Making It Read Only

It is possible to back up either master or slave servers in a replication setup by acquiring a global read lock and manipulating the read_only system variable to change the read-only state of the server to be backed up:

1. Make the server read-only, so that it processes only retrievals and blocks updates.
2. Perform the backup.
3. Change the server back to its normal read/write state.

**Note**

The instructions in this section place the server to be backed up in a state that is safe for backup methods that get the data from the server, such as mysqldump (see `mysqldump — A Database Backup Program`). You should not attempt to use
these instructions to make a binary backup by copying files directly because the server may still have modified data cached in memory and not flushed to disk.

These instructions also require MySQL 5.1.15 or higher. For earlier versions, setting `read_only` does not block while table locks or outstanding transactions are pending, so data changes can still occur during the backup operation and produce inconsistent backup results.

The following instructions describe how to do this for a master server and for a slave server. For both scenarios discussed here, suppose that you have the following replication setup:

- A master server M1
- A slave server S1 that has M1 as its master
- A client C1 connected to M1
- A client C2 connected to S1

In either scenario, the statements to acquire the global read lock and manipulate the `read_only` variable are performed on the server to be backed up and do not propagate to any slaves of that server.

**Scenario 1: Backup with a Read-Only Master**

Put the master M1 in a read-only state by executing these statements on it:

```sql
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

While M1 is in a read-only state, the following properties are true:

- Requests for updates sent by C1 to M1 will block because the server is in read-only mode.
- Requests for query results sent by C1 to M1 will succeed.
- Making a backup on M1 is safe.
- Making a backup on S1 is not safe. This server is still running, and might be processing the binary log or update requests coming from client C2

While M1 is read only, perform the backup. For example, you can use `mysqldump`.

After the backup operation on M1 completes, restore M1 to its normal operational state by executing these statements:

```sql
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

Although performing the backup on M1 is safe (as far as the backup is concerned), it is not optimal for performance because clients of M1 are blocked from executing updates.

This strategy applies to backing up a master server in a replication setup, but can also be used for a single server in a nonreplication setting.

**Scenario 2: Backup with a Read-Only Slave**

Put the slave S1 in a read-only state by executing these statements on it:
Using Replication with Different Master and Slave Storage Engines

mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;

While S1 is in a read-only state, the following properties are true:

- The master M1 will continue to operate, so making a backup on the master is not safe.
- The slave S1 is stopped, so making a backup on the slave S1 is safe.

These properties provide the basis for a popular backup scenario: Having one slave busy performing a backup for a while is not a problem because it does not affect the entire network, and the system is still running during the backup. In particular, clients can still perform updates on the master server, which remains unaffected by backup activity on the slave.

While S1 is read only, perform the backup. For example, you can use mysql dump.

After the backup operation on S1 completes, restore S1 to its normal operational state by executing these statements:

mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;

After the slave is restored to normal operation, it again synchronizes to the master by catching up with any outstanding updates from the binary log of the master.

3.2 Using Replication with Different Master and Slave Storage Engines

It does not matter for the replication process whether the source table on the master and the replicated table on the slave use different engine types. In fact, the system variables storage_engine and table_type are not replicated.

This provides a number of benefits in the replication process in that you can take advantage of different engine types for different replication scenarios. For example, in a typical scale-out scenario (see Section 3.3, “Using Replication for Scale-Out”), you want to use InnoDB tables on the master to take advantage of the transactional functionality, but use MyISAM on the slaves where transaction support is not required because the data is only read. When using replication in a data-logging environment you may want to use the Archive storage engine on the slave.

Configuring different engines on the master and slave depends on how you set up the initial replication process:

- If you used mysqldump to create the database snapshot on your master, you could edit the dump file text to change the engine type used on each table.

Another alternative for mysqldump is to disable engine types that you do not want to use on the slave before using the dump to build the data on the slave. For example, you can add the --skip-innodb option on your slave to disable the InnoDB engine. If a specific engine does not exist for a table to be created, MySQL will use the default engine type, usually MyISAM. (This requires that the NO_ENGINE_SUBSTITUTION SQL mode is not enabled.) If you want to disable additional engines in this way, you may want to consider building a special binary to be used on the slave that only supports the engines you want.

- If you are using raw data files (a binary backup) to set up the slave, you will be unable to change the initial table format. Instead, use ALTER TABLE to change the table types after the slave has been started.
• For new master/slave replication setups where there are currently no tables on the master, avoid specifying the engine type when creating new tables.

If you are already running a replication solution and want to convert your existing tables to another engine type, follow these steps:

1. Stop the slave from running replication updates:

   ```
   mysql> STOP SLAVE;
   ```

   This will enable you to change engine types without interruptions.

2. Execute an `ALTER TABLE ... ENGINE='engine_type'` for each table to be changed.

3. Start the slave replication process again:

   ```
   mysql> START SLAVE;
   ```

   Although the `storage_engine` and `table_type` variables are not replicated, be aware that `CREATE TABLE` and `ALTER TABLE` statements that include the engine specification will be correctly replicated to the slave. For example, if you have a CSV table and you execute:

   ```
   mysql> ALTER TABLE csvtable Engine='MyISAM';
   ```

   The above statement will be replicated to the slave and the engine type on the slave will be converted to `MyISAM`, even if you have previously changed the table type on the slave to an engine other than CSV. If you want to retain engine differences on the master and slave, you should be careful to use the `storage_engine` variable on the master when creating a new table. For example, instead of:

   ```
   mysql> CREATE TABLE tablea (columna int) Engine=MyISAM;
   ```

   Use this format:

   ```
   mysql> SET storage_engine=MyISAM;
   mysql> CREATE TABLE tablea (columna int);
   ```

   When replicated, the `storage_engine` variable will be ignored, and the `CREATE TABLE` statement will execute on the slave using the slave's default engine.

### 3.3 Using Replication for Scale-Out

You can use replication as a scale-out solution; that is, where you want to split up the load of database queries across multiple database servers, within some reasonable limitations.

Because replication works from the distribution of one master to one or more slaves, using replication for scale-out works best in an environment where you have a high number of reads and low number of writes/updates. Most Web sites fit into this category, where users are browsing the Web site, reading articles, posts, or viewing products. Updates only occur during session management, or when making a purchase or adding a comment/message to a forum.

Replication in this situation enables you to distribute the reads over the replication slaves, while still enabling your web servers to communicate with the replication master when a write is required. You can see a sample replication layout for this scenario in Figure 3.1, "Using Replication to Improve Performance During Scale-Out".
If the part of your code that is responsible for database access has been properly abstracted/modularized, converting it to run with a replicated setup should be very smooth and easy. Change the implementation of your database access to send all writes to the master, and to send reads to either the master or a slave. If your code does not have this level of abstraction, setting up a replicated system gives you the opportunity and motivation to clean it up. Start by creating a wrapper library or module that implements the following functions:

- `safe_writer_connect()`
- `safe_reader_connect()`
- `safe_reader_statement()`
- `safe_writer_statement()`

`safe_` in each function name means that the function takes care of handling all error conditions. You can use different names for the functions. The important thing is to have a unified interface for connecting for reads, connecting for writes, doing a read, and doing a write.

Then convert your client code to use the wrapper library. This may be a painful and scary process at first, but it pays off in the long run. All applications that use the approach just described are able to take advantage of a master/slave configuration, even one involving multiple slaves. The code is much easier to maintain, and adding troubleshooting options is trivial. You need modify only one or two functions; for example, to log how long each statement took, or which statement among those issued gave you an error.

If you have written a lot of code, you may want to automate the conversion task by using the `replace` utility that comes with standard MySQL distributions, or write your own conversion script. Ideally, your code uses consistent programming style conventions. If not, then you are probably better off rewriting it anyway, or at least going through and manually regularizing it to use a consistent style.

### 3.4 Replicating Different Databases to Different Slaves

There may be situations where you have a single master and want to replicate different databases to different slaves. For example, you may want to distribute different sales data to different departments to help spread the load during data analysis. A sample of this layout is shown in Figure 3.2, "Using Replication to Replicate Databases to Separate Replication Slaves".
Replicating Different Databases to Different Slaves

Figure 3.2 Using Replication to Replicate Databases to Separate Replication Slaves

You can achieve this separation by configuring the master and slaves as normal, and then limiting the binary log statements that each slave processes by using the `--replicate-wild-do-table` configuration option on each slave.

**Important**

You should not use `--replicate-do-db` for this purpose when using statement-based replication, since statement-based replication causes this option's affects to vary according to the database that is currently selected. This applies to mixed-format replication as well, since this enables some updates to be replicated using the statement-based format.

However, it should be safe to use `--replicate-do-db` for this purpose if you are using row-based replication only, since in this case the currently selected database has no effect on the option's operation.

For example, to support the separation as shown in Figure 3.2, “Using Replication to Replicate Databases to Separate Replication Slaves”, you should configure each replication slave as follows, before executing `START SLAVE`:

- Replication slave 1 should use `--replicate-wild-do-table=databaseA.%`.
- Replication slave 2 should use `--replicate-wild-do-table=databaseB.%`.
- Replication slave 3 should use `--replicate-wild-do-table=databaseC.%`.

Each slave in this configuration receives the entire binary log from the master, but executes only those events from the binary log that apply to the databases and tables included by the `--replicate-wild-do-table` option in effect on that slave.

If you have data that must be synchronized to the slaves before replication starts, you have a number of choices:

- Synchronize all the data to each slave, and delete the databases, tables, or both that you do not want to keep.
- Use `mysqldump` to create a separate dump file for each database and load the appropriate dump file on each slave.
- Use a raw data file dump and include only the specific files and databases that you need for each slave.

**Note**

This does not work with InnoDB databases unless you use `innodb_file_per_table`.  

82
3.5 Improving Replication Performance

As the number of slaves connecting to a master increases, the load, although minimal, also increases, as each slave uses a client connection to the master. Also, as each slave must receive a full copy of the master binary log, the network load on the master may also increase and create a bottleneck.

If you are using a large number of slaves connected to one master, and that master is also busy processing requests (for example, as part of a scale-out solution), then you may want to improve the performance of the replication process.

One way to improve the performance of the replication process is to create a deeper replication structure that enables the master to replicate to only one slave, and for the remaining slaves to connect to this primary slave for their individual replication requirements. A sample of this structure is shown in Figure 3.3, “Using an Additional Replication Host to Improve Performance”.

Figure 3.3 Using an Additional Replication Host to Improve Performance

For this to work, you must configure the MySQL instances as follows:

- Master 1 is the primary master where all changes and updates are written to the database. Binary logging should be enabled on this machine.

- Master 2 is the slave to the Master 1 that provides the replication functionality to the remainder of the slaves in the replication structure. Master 2 is the only machine permitted to connect to Master 1. Master 2 also has binary logging enabled, and the `--log-slave-updates` option so that replication instructions from Master 1 are also written to Master 2's binary log so that they can then be replicated to the true slaves.

- Slave 1, Slave 2, and Slave 3 act as slaves to Master 2, and replicate the information from Master 2, which actually consists of the upgrades logged on Master 1.

The above solution reduces the client load and the network interface load on the primary master, which should improve the overall performance of the primary master when used as a direct database solution.

If your slaves are having trouble keeping up with the replication process on the master, there are a number of options available:

- If possible, put the relay logs and the data files on different physical drives. To do this, use the `--relay-log` option to specify the location of the relay log.

- If the slaves are significantly slower than the master, you may want to divide up the responsibility for replicating different databases to different slaves. See Section 3.4, “Replicating Different Databases to Different Slaves”.

- If your master makes use of transactions and you are not concerned about transaction support on your slaves, use MyISAM or another nontransactional engine on the slaves. See Section 3.2, “Using Replication with Different Master and Slave Storage Engines”.

For this to work, you must configure the MySQL instances as follows:
Switching Masters During Failover

- If your slaves are not acting as masters, and you have a potential solution in place to ensure that you can bring up a master in the event of failure, then you can switch off `--log-slave-updates`. This prevents “dumb” slaves from also logging events they have executed into their own binary log.

### 3.6 Switching Masters During Failover

There is in MySQL 5.1 no official solution for providing failover between master and slaves in the event of a failure. Instead, you must set up a master and one or more slaves; then, you need to write an application or script that monitors the master to check whether it is up, and instructs the slaves and applications to change master in case of failure. This section discusses some of the issues encountered when setting up failover in this fashion.

**Note**

The MySQL Utilities include a `mysqlfailover` tool that provides failover capability using GTIDs, support for which requires MySQL 5.6 or later. For more information, see `mysqlfailover — Automatic replication health monitoring and failover`, and `Replication with Global Transaction Identifiers`.

You can tell a slave to change to a new master using the `CHANGE MASTER TO` statement. The slave does not check whether the databases on the master are compatible with those on the slave; it simply begins reading and executing events from the specified coordinates in the new master's binary log. In a failover situation, all the servers in the group are typically executing the same events from the same binary log file, so changing the source of the events should not affect the structure or integrity of the database, provided that you exercise care in making the change.

Slaves should be run with the `--log-bin` option and without `--log-slave-updates`. In this way, the slave is ready to become a master without restarting the slave `mysqld`. Assume that you have the structure shown in Figure 3.4, “Redundancy Using Replication, Initial Structure”.

**Figure 3.4 Redundancy Using Replication, Initial Structure**

In this diagram, the MySQL Master holds the master database, the MySQL Slave hosts are replication slaves, and the Web Client machines are issuing database reads and writes. Web clients that issue only reads (and would normally be connected to the slaves) are not shown, as they do not need to switch to a new server in the event of failure. For a more detailed example of a read/write scale-out replication structure, see Section 3.3, “Using Replication for Scale-Out”.

84
Each MySQL Slave (Slave 1, Slave 2, and Slave 3) is a slave running with --log-bin and without --log-slave-updates. Because updates received by a slave from the master are not logged in the binary log unless --log-slave-updates is specified, the binary log on each slave is empty initially. If for some reason MySQL Master becomes unavailable, you can pick one of the slaves to become the new master. For example, if you pick Slave 1, all Web Clients should be redirected to Slave 1, which writes the updates to its binary log. Slave 2 and Slave 3 should then replicate from Slave 1.

The reason for running the slave without --log-slave-updates is to prevent slaves from receiving updates twice in case you cause one of the slaves to become the new master. If Slave 1 has --log-slave-updates enabled, it writes any updates that it receives from Master in its own binary log. This means that, when Slave 2 changes from Master to Slave 1 as its master, it may receive updates from Slave 1 that it has already received from Master.

Make sure that all slaves have processed any statements in their relay log. On each slave, issue STOP SLAVE IO_THREAD, then check the output of SHOW PROCESSLIST until you see Has read all relay log. When this is true for all slaves, they can be reconfigured to the new setup. On the slave Slave 1 being promoted to become the master, issue STOP SLAVE and RESET MASTER.

On the other slaves Slave 2 and Slave 3, use STOP SLAVE and CHANGE MASTER TO MASTER_HOST='Slave1' (where 'Slave1' represents the real host name of Slave 1). To use CHANGE MASTER TO, add all information about how to connect to Slave 1 from Slave 2 or Slave 3 (user, password, port). When issuing the CHANGE MASTER TO statement in this, there is no need to specify the name of the Slave 1 binary log file or log position to read from, since the first binary log file and position 4, are the defaults. Finally, execute START SLAVE on Slave 2 and Slave 3.

Once the new replication setup is in place, you need to tell each Web Client to direct its statements to Slave 1. From that point on, all updates statements sent by Web Client to Slave 1 are written to the binary log of Slave 1, which then contains every update statement sent to Slave 1 since Master died.

The resulting server structure is shown in Figure 3.5, "Redundancy Using Replication, After Master Failure".

Figure 3.5 Redundancy Using Replication, After Master Failure
When **Master** becomes available again, you should make it a slave of **Slave 1**. To do this, issue on **Master** the same **CHANGE MASTER TO** statement as that issued on **Slave 2** and **Slave 3** previously. **Master** then becomes a slave of **Slave 1** and picks up the **Web Client** writes that it missed while it was offline.

To make **Master** a master again, use the preceding procedure as if **Slave 1** was unavailable and **Master** was to be the new master. During this procedure, do not forget to run **RESET MASTER** on **Master** before making **Slave 1**, **Slave 2**, and **Slave 3** slaves of **Master**. If you fail to do this, the slaves may pick up stale writes from the **Web Client** applications dating from before the point at which **Master** became unavailable.

You should be aware that there is no synchronization between slaves, even when they share the same master, and thus some slaves might be considerably ahead of others. This means that in some cases the procedure outlined in the previous example might not work as expected. In practice, however, relay logs on all slaves should be relatively close together.

One way to keep applications informed about the location of the master is to have a dynamic DNS entry for the master. With **bind** you can use **nsupdate** to update the DNS dynamically.

### 3.7 Setting Up Replication Using SSL

To use SSL for encrypting the transfer of the binary log required during replication, both the master and the slave must support SSL network connections. If either host does not support SSL connections (because it has not been compiled or configured for SSL), replication through an SSL connection is not possible.

Setting up replication using an SSL connection is similar to setting up a server and client using SSL. You must obtain (or create) a suitable security certificate that you can use on the master, and a similar certificate (from the same certificate authority) on each slave.

For more information on setting up a server and client for SSL connectivity, see [Configuring MySQL to Use Secure Connections](#).

To enable SSL on the master you must create or obtain suitable certificates, and then add the following configuration options to the master's configuration within the **[mysqld]** section of the master's **my.cnf** file:

```sql
[mysqld]
ssl-ca=cacert.pem
ssl-cert=server-cert.pem
ssl-key=server-key.pem
```

The paths to the certificates may be relative or absolute; we recommend that you always use complete paths for this purpose.

The options are as follows:

- **ssl-ca** identifies the Certificate Authority (CA) certificate.
- **ssl-cert** identifies the server public key. This can be sent to the client and authenticated against the CA certificate that it has.
- **ssl-key** identifies the server private key.

On the slave, you have two options available for setting the SSL information. You can either add the slave certificates to the **[client]** section of the slave's **my.cnf** file, or you can explicitly specify the SSL information using the **CHANGE MASTER TO** statement:
• To add the slave certificates using an option file, add the following lines to the [client] section of the slave's my.cnf file:

```
[client]
ssl-ca=cacert.pem
ssl-cert=client-cert.pem
ssl-key=client-key.pem
```

Restart the slave server, using the `--skip-slave-start` option to prevent the slave from connecting to the master. Use `CHANGE MASTER TO` to specify the master configuration, using the `MASTER_SSL` option to enable SSL connectivity:

```
mysql> CHANGE MASTER TO
    -> MASTER_HOST='master_hostname',
    -> MASTER_USER='replicate',
    -> MASTER_PASSWORD='password',
    -> MASTER_SSL=1;
```

• To specify the SSL certificate options using the `CHANGE MASTER TO` statement, append the SSL options:

```
mysql> CHANGE MASTER TO
    -> MASTER_HOST='master_hostname',
    -> MASTER_USER='replicate',
    -> MASTER_PASSWORD='password',
    -> MASTER_SSL=1,
    -> MASTER_SSL_CA = 'ca_file_name',
    -> MASTER_SSL_CAPATH = 'ca_directory_name',
    -> MASTER_SSL_CERT = 'cert_file_name',
    -> MASTER_SSL_KEY = 'key_file_name';
```

After the master information has been updated, start the slave replication process:

```
mysql> START SLAVE;
```

You can use the `SHOW SLAVE STATUS` statement to confirm that the SSL connection was established successfully.

For more information on the `CHANGE MASTER TO` statement, see `CHANGE MASTER TO Syntax`.

If you want to enforce the use of SSL connections during replication, then create a user with the `REPLICATION SLAVE` privilege and use the `REQUIRE SSL` option for that user. For example:

```
mysql> CREATE USER 'repl'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
mysql> GRANT REPLICATION SLAVE ON *.*
    -> TO 'repl'@'%.mydomain.com' REQUIRE SSL;
```

If the account already exists, you can add `REQUIRE SSL` to it with this statement:

```
mysql> GRANT USAGE ON *.*
    -> TO 'repl'@'%.mydomain.com' REQUIRE SSL;
```
Chapter 4 Replication Notes and Tips

Table of Contents

4.1 Replication Features and Issues ................................................................. 89
  4.1.1 Replication and AUTO_INCREMENT ................................................. 90
  4.1.2 Replication and BLACKHOLE Tables ............................................. 91
  4.1.3 Replication and Character Sets .................................................... 91
  4.1.4 Replication and CHECKSUM TABLE ............................................. 92
  4.1.5 Replication of CREATE ... IF NOT EXISTS Statements ..................... 92
  4.1.6 Replication of CREATE TABLE ... SELECT Statements .................... 92
  4.1.7 Replication of CURRENT_USER() ................................................. 93
  4.1.8 Replication of CREATE SERVER, ALTER SERVER, and DROP SERVER ...... 94
  4.1.9 Replication of DROP ... IF EXISTS Statements ................................ 94
  4.1.10 Replication with Differing Table Definitions on Master and Slave ...... 94
  4.1.11 Replication and DIRECTORY Table Options .................................. 100
  4.1.12 Replication of Invoked Features .............................................. 100
  4.1.13 Replication and Floating-Point Values ........................................ 102
  4.1.14 Replication and FLUSH ............................................................... 102
  4.1.15 Replication and System Functions ............................................ 102
  4.1.16 Replication and LIMIT ............................................................... 105
  4.1.17 Replication and LOAD DATA INFILE ........................................... 105
  4.1.18 Replication and the Slow Query Log .......................................... 105
  4.1.19 Replication and REPAIR TABLE ................................................ 105
  4.1.20 Replication and Master or Slave Shutdowns .................................. 105
  4.1.21 Replication and max_allowed_packet ........................................ 106
  4.1.22 Replication and MEMORY Tables ............................................... 106
  4.1.23 Replication and Temporary Tables ........................................... 107
  4.1.24 Replication of the mysql System Database .................................... 108
  4.1.25 Replication and the Query Optimizer ........................................ 108
  4.1.26 Replication and Partitioning .................................................... 108
  4.1.27 Replication and Reserved Words ............................................... 108
  4.1.28 SET PASSWORD and Row-Based Replication ................................ 109
  4.1.29 Slave Errors During Replication .............................................. 109
  4.1.30 Replication of Server-Side Help Tables ...................................... 109
  4.1.31 Replication and Server SQL Mode .............................................. 111
  4.1.32 Replication Retries and Timeouts ............................................. 111
  4.1.33 Replication and TIMESTAMP ..................................................... 112
  4.1.34 Replication and Time Zones ..................................................... 112
  4.1.35 Replication and Transactions .................................................. 112
  4.1.36 Replication and Triggers .......................................................... 113
  4.1.37 Replication and TRUNCATE TABLE ............................................ 114
  4.1.38 Replication and Variables ....................................................... 114
  4.1.39 Replication and Views ............................................................. 116

4.2 Replication Compatibility Between MySQL Versions .................................. 116
4.3 Upgrading a Replication Setup ................................................................. 117
4.4 Troubleshooting Replication ...................................................................... 118
4.5 How to Report Replication Bugs or Problems ........................................... 119
The following sections provide information about what is supported and what is not in MySQL replication, and about specific issues and situations that may occur when replicating certain statements.

Statement-based replication depends on compatibility at the SQL level between the master and slave. In others, successful SBR requires that any SQL features used be supported by both the master and the slave servers. For example, if you use a feature on the master server that is available only in MySQL 5.1 (or later), you cannot replicate to a slave that uses MySQL 5.0 (or earlier).

Such incompatibilities also can occur within a release series when using pre-production releases of MySQL. For example, the `SLEEP()` function is available beginning with MySQL 5.0.12. If you use this function on the master, you cannot replicate to a slave that uses MySQL 5.0.11 or earlier.

For this reason, use Generally Available (GA) releases of MySQL for statement-based replication in a production setting, since we do not introduce new SQL statements or change their behavior within a given release series once that series reaches GA release status.

If you are planning to use statement-based replication between MySQL 5.1 and a previous MySQL release series, it is also a good idea to consult the edition of the MySQL Reference Manual corresponding to the earlier release series for information regarding the replication characteristics of that series.

With MySQL's statement-based replication, there may be issues with replicating stored routines or triggers. You can avoid these issues by using MySQL's row-based replication instead. For a detailed list of issues, see Binary Logging of Stored Programs. For more information about row-based logging and row-based replication, see Binary Logging Formats, and Section 2.2, “Replication Formats”.

For additional information specific to replication and InnoDB, see InnoDB and MySQL Replication. For information relating to replication with MySQL Cluster, see MySQL Cluster Replication.

### 4.1.1 Replication and AUTO_INCREMENT

Statement-based replication of `AUTO_INCREMENT`, `LAST_INSERT_ID()` and `TIMESTAMP` values is done correctly, subject to the following exceptions:

- When using statement-based replication, `AUTO_INCREMENT` columns in tables on the slave must match the same columns on the master; that is, `AUTO_INCREMENT` columns must be replicated to `AUTO_INCREMENT` columns.
  
  This is a known issue which is fixed in MySQL 5.5. (Bug #12669186)

- Prior to MySQL 5.1.12, a stored procedure that uses `LAST_INSERT_ID()` does not replicate properly using statement-based binary logging.

- Prior to MySQL 5.1.12, when a stored routine or trigger caused an `INSERT` into an `AUTO_INCREMENT` column, the generated `AUTO_INCREMENT` value was not written into the binary log, so a different value could in some cases be inserted on the slave.

- An insert into an `AUTO_INCREMENT` column caused by a stored routine or trigger running on a master that uses MySQL 5.0.60 or earlier does not replicate correctly to a slave running MySQL 5.1.12 through 5.1.23 (inclusive). (Bug #33029)

  A statement invoking a trigger or function that causes an update to an `AUTO_INCREMENT` column is not replicated correctly using statement-based replication. Beginning with MySQL 5.1.40, such statements are marked as unsafe. (Bug #45677)

- The `AUTO_INCREMENT` table option was not replicated correctly prior to MySQL 5.1.31. (Bug #41986)
• An INSERT into a table that has a composite primary key that includes an AUTO_INCREMENT column that is not the first column of this composite key is not safe for statement-based logging or replication. Beginning with MySQL 5.1.64, such statements are marked as unsafe. (Bug #11754117, Bug #45670)

This issue does not affect tables using the InnoDB storage engine, since an InnoDB table with an AUTO_INCREMENT column requires at least one key where the auto-increment column is the only or leftmost column.

• Adding an AUTO_INCREMENT column to a table with ALTER TABLE might not produce the same ordering of the rows on the slave and the master. This occurs because the order in which the rows are numbered depends on the specific storage engine used for the table and the order in which the rows were inserted. If it is important to have the same order on the master and slave, the rows must be ordered before assigning an AUTO_INCREMENT number. Assuming that you want to add an AUTO_INCREMENT column to a table t1 that has columns col1 and col2, the following statements produce a new table t2 identical to t1 but with an AUTO_INCREMENT column:

```sql
CREATE TABLE t2 LIKE t1;
ALTER TABLE t2 ADD id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO t2 SELECT * FROM t1 ORDER BY col1, col2;
```

**Important**

To guarantee the same ordering on both master and slave, the ORDER BY clause must name all columns of t1.

The instructions just given are subject to the limitations of CREATE TABLE ... LIKE: Foreign key definitions are ignored, as are the DATA DIRECTORY and INDEX DIRECTORY table options. If a table definition includes any of those characteristics, create t2 using a CREATE TABLE statement that is identical to the one used to create t1, but with the addition of the AUTO_INCREMENT column.

Regardless of the method used to create and populate the copy having the AUTO_INCREMENT column, the final step is to drop the original table and then rename the copy:

```sql
DROP t1;
ALTER TABLE t2 RENAME t1;
```

See also Problems with ALTER TABLE.

### 4.1.2 Replication and BLACKHOLE Tables

The BLACKHOLE storage engine accepts data but discards it and does not store it. When performing binary logging, all inserts to such tables are always logged, regardless of the logging format in use. Updates and deletes are handled differently depending on whether statement based or row based logging is in use.

With the statement based logging format, all statements affecting BLACKHOLE tables are logged, but their effects ignored. When using row-based logging, updates and deletes to such tables are simply skipped—they are not written to the binary log.

For this reason we recommend when you replicate to tables using the BLACKHOLE storage engine that you have the binlog_format server variable set to STATEMENT, and not to either ROW or MIXED.

### 4.1.3 Replication and Character Sets

The following applies to replication between MySQL servers that use different character sets:

• If the master has databases with a character set different from the global character_set_server value, you should design your CREATE TABLE statements so that they do not implicitly rely on the
database default character set. A good workaround is to state the character set and collation explicitly in CREATE TABLE statements.

4.1.4 Replication and CHECKSUM TABLE

CHECKSUM TABLE returns a checksum that is calculated row by row, using a method that depends on the table row storage format, which is not guaranteed to remain the same between MySQL release series. For example, the storage format for VARCHAR changed between MySQL 4.1 and 5.0, so if a 4.1 table is upgraded to MySQL 5.0, the checksum value may change.

4.1.5 Replication of CREATE ... IF NOT EXISTS Statements

This section discusses the rules that are applied when various CREATE ... IF NOT EXISTS statements are replicated.

Previous to MySQL 5.1.38. CREATE DATABASE IF NOT EXISTS was replicated only if the database named in the statement did not exist on the master. CREATE TABLE IF NOT EXISTS was replicated only if the table named in the statement did not exist on the master.

MySQL 5.1.39 and later. Every CREATE DATABASE IF NOT EXISTS statement is replicated, whether or not the database already exists on the master. Similarly, every CREATE TABLE IF NOT EXISTS statement is replicated, whether or not the table already exists on the master. This includes CREATE TABLE IF NOT EXISTS ... LIKE. However, replication of CREATE TABLE IF NOT EXISTS ... SELECT follows somewhat different rules; see Section 4.1.6, “Replication of CREATE TABLE ... SELECT Statements”, for more information.

Replication of CREATE EVENT IF NOT EXISTS. CREATE EVENT IF NOT EXISTS is always replicated, whether or not the event named in the statement already exists on the master. This is true for all MySQL 5.1 releases beginning with MySQL 5.1.6, when this statement was introduced.

See also Bug #45574.

4.1.6 Replication of CREATE TABLE ... SELECT Statements

This section discusses how MySQL replicates CREATE TABLE ... SELECT statements.

These behaviors are not dependent on MySQL version:

• CREATE TABLE ... SELECT always performs an implicit commit (Statements That Cause an Implicit Commit).

• If destination table does not exist, logging occurs as follows. It does not matter whether IF NOT EXISTS is present.
  • STATEMENT or MIXED format: The statement is logged as written.
  • ROW format: The statement is logged as a CREATE TABLE statement followed by a series of insert-row events.
  • If the statement fails, nothing is logged. This includes the case that the destination table exists and IF NOT EXISTS is not given.

When the destination table exists and IF NOT EXISTS is given, MySQL handles the statement in a version-dependent way.

In MySQL 5.1 before 5.1.51 and in MySQL 5.5 before 5.5.6 (this is the original behavior):
4.1.7 Replication of CURRENT_USER()

The following statements support use of the CURRENT_USER() function to take the place of the name of (and, possibly, the host for) an affected user or a definer; in such cases, CURRENT_USER() is expanded where and as needed:

- DROP USER
- RENAME USER
- GRANT
- REVOKE
- CREATE FUNCTION
- CREATE PROCEDURE
- CREATE TRIGGER
- CREATE EVENT
- CREATE VIEW
- ALTER EVENT
- ALTER VIEW
- SET PASSWORD

In all MySQL 5.1 releases, when CURRENT_USER() or CURRENT_USER is used as the definer in any of the statements CREATE FUNCTION, CREATE PROCEDURE, CREATE TRIGGER, CREATE EVENT, CREATE VIEW, or ALTER VIEW when binary logging is enabled, the function reference is expanded before it is written to the binary log, so that the statement refers to the same user on both the master and the slave when the statement is replicated. Beginning with MySQL 5.1.49, CURRENT_USER() or CURRENT_USER is
also expanded prior to being written to the binary log when used in DROP USER, RENAME USER, GRANT, REVOKE, or ALTER EVENT. (Bug #48321)

4.1.8 Replication of CREATE SERVER, ALTER SERVER, and DROP SERVER

In MySQL 5.1, the statements CREATE SERVER, ALTER SERVER, and DROP SERVER are logged only when binlog_format is equal to ROW. These statements are not written to the binary log when using either the statement-based or mixed logging format.

4.1.9 Replication of DROP ... IF EXISTS Statements

The DROP DATABASE IF EXISTS, DROP TABLE IF EXISTS, and DROP VIEW IF EXISTS statements are always replicated, even if the database, table, or view to be dropped does not exist on the master. This is to ensure that the object to be dropped no longer exists on either the master or the slave, once the slave has caught up with the master.

Beginning with MySQL 5.1.33, DROP ... IF EXISTS statements for stored programs (stored procedures and functions, triggers, and events) are also replicated, even if the stored program to be dropped does not exist on the master. (Bug #13684)

4.1.10 Replication with Differing Table Definitions on Master and Slave

Starting with MySQL 5.1.21, source and target tables for replication do not have to be identical. A table on the master can have more or fewer columns than the slave's copy of the table. In addition, corresponding table columns on the master and the slave can use different data types, subject to certain conditions.

Note
Replication between tables which are partitioned differently from one another is not supported. See Section 4.1.26, “Replication and Partitioning”.

In all cases where the source and target tables do not have identical definitions, the database and table names must be the same on both the master and the slave. Additional conditions are discussed, with examples, in the following two sections.

4.1.10.1 Replication with More Columns on Master or Slave

Starting with MySQL 5.1.21, you can replicate a table from the master to the slave such that the master and slave copies of the table have differing numbers of columns, subject to the following conditions:

• Columns common to both versions of the table must be defined in the same order on the master and the slave.
  (This is true even if both tables have the same number of columns.)

• Columns common to both versions of the table must be defined before any additional columns.

This means that executing an ALTER TABLE statement on the slave where a new column is inserted into the table within the range of columns common to both tables causes replication to fail, as shown in the following example:

Suppose that a table t, existing on the master and the slave, is defined by the following CREATE TABLE statement:

```sql
CREATE TABLE t {
  c1 INT,
  ...
}
Replication with Differing Table Definitions on Master and Slave

Suppose that the `ALTER TABLE` statement shown here is executed on the slave:

```sql
ALTER TABLE t ADD COLUMN cnew1 INT AFTER c3;
```

The previous `ALTER TABLE` is permitted on the slave because the columns `c1`, `c2`, and `c3` that are common to both versions of table `t` remain grouped together in both versions of the table, before any columns that differ.

However, the following `ALTER TABLE` statement cannot be executed on the slave without causing replication to break:

```sql
ALTER TABLE t ADD COLUMN cnew2 INT AFTER c2;
```

Replication fails after execution on the slave of the `ALTER TABLE` statement just shown, because the new column `cnew2` comes between columns common to both versions of `t`.

- Each “extra” column in the version of the table having more columns must have a default value.

**Note**

A column’s default value is determined by a number of factors, including its type, whether it is defined with a `DEFAULT` option, whether it is declared as `NULL`, and the server SQL mode in effect at the time of its creation; for more information, see [Data Type Default Values](#).

In addition, when the slave’s copy of the table has more columns than the master’s copy, each column common to the tables must use the same data type in both tables.

**Examples.** The following examples illustrate some valid and invalid table definitions:

**More columns on the master.** The following table definitions are valid and replicate correctly:

```
master> CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
slave> CREATE TABLE t1 (c1 INT, c2 INT);
```

The following table definitions would raise an error because the definitions of the columns common to both versions of the table are in a different order on the slave than they are on the master:

```
master> CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
slave> CREATE TABLE t1 (c2 INT, c1 INT);
```

The following table definitions would also raise an error because the definition of the extra column on the master appears before the definitions of the columns common to both versions of the table:

```
master> CREATE TABLE t1 (c3 INT, c1 INT, c2 INT);
slave> CREATE TABLE t1 (c1 INT, c2 INT);
```

**More columns on the slave.** The following table definitions are valid and replicate correctly:

```
master> CREATE TABLE t1 (c1 INT, c2 INT);
slave> CREATE TABLE t1 (c1 INT, c2 INT, c3 INT);
```
The following definitions raise an error because the columns common to both versions of the table are not defined in the same order on both the master and the slave:

| master> CREATE TABLE t1 (c1 INT, c2 INT); | slave> CREATE TABLE t1 (c2 INT, c1 INT, c3 INT); |

The following table definitions also raise an error because the definition for the extra column in the slave's version of the table appears before the definitions for the columns which are common to both versions of the table:

| master> CREATE TABLE t1 (c1 INT, c2 INT); | slave> CREATE TABLE t1 (c3 INT, c1 INT, c2 INT); |

The following table definitions fail because the slave's version of the table has additional columns compared to the master's version, and the two versions of the table use different data types for the common column c2:

| master> CREATE TABLE t1 (c1 INT, c2 BIGINT); | slave> CREATE TABLE t1 (c1 INT, c2 INT, c3 INT); |

### 4.1.10.2 Replication of Columns Having Different Data Types

Corresponding columns on the master's and the slave's copies of the same table ideally should have the same data type. However, beginning with MySQL 5.1.21, this is not always strictly enforced, as long as certain conditions are met.

All other things being equal, it is always possible to replicate from a column of a given data type to another column of the same type and same size or width, where applicable, or larger. For example, you can replicate from a `CHAR(10)` column to another `CHAR(10)`, or from a `CHAR(10)` column to a `CHAR(25)` column without any problems. In certain cases, it also possible to replicate from a column having one data type (on the master) to a column having a different data type (on the slave); when the data type of the master's version of the column is promoted to a type that is the same size or larger on the slave, this is known as *attribute promotion*.

Attribute promotion can be used with both statement-based and row-based replication, and is not dependent on the storage engine used by either the master or the slave. However, the choice of logging format does have an effect on the type conversions that are permitted; the particulars are discussed later in this section.

| Important | Whether you use statement-based or row-based replication, the slave's copy of the table cannot contain more columns than the master's copy if you wish to employ attribute promotion. |

**Statement-based replication.** When using statement-based replication, a simple rule of thumb to follow is, “If the statement run on the master would also execute successfully on the slave, it should also replicate successfully”. In other words, if the statement uses a value that is compatible with the type of a given column on the slave, the statement can be replicated. For example, you can insert any value that fits in a `TINYINT` column into a `BIGINT` column as well; it follows that, even if you change the type of a `TINYINT` column in the slave's copy of a table to `BIGINT`, any insert into that column on the master that succeeds should also succeed on the slave, since it is impossible to have a legal `TINYINT` value that is large enough to exceed a `BIGINT` column.
When using statement-based replication, `AUTO_INCREMENT` columns must be the same on both the master and the slave; otherwise, updates may be applied to the wrong table on the slave. This is a known issue which is fixed in MySQL 5.5. (Bug #12669186)

**Note**

The restrictions described in the next few paragraphs do not apply to MySQL Cluster Replication, beginning with MySQL Cluster NDB 6.3.33, 7.0.14, and 7.1.3. These and later versions of MySQL Cluster support attribute promotion and demotion (such as `TINYINT` to `BIGINT`, and `VARCHAR(50)` to `CHAR(25)`, respectively) in replication; see [Attribute promotion and demotion (MySQL Cluster)](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-replication-attribute-promotion-demotion.html), for more information.

**Row-based replication.** For row-based replication, the case is not so simple, due to the fact that changes rather than statements are replicated, and these changes are transmitted from master to slave using formats that do not always map directly to MySQL server column data types. For example, with row-based binary logging, you cannot replicate between different `INT` subtypes, such as from `TINYINT` to `BIGINT`, because changes to columns of these types are represented differently from one another in the binary log when using row-based logging. However, you can replicate from `BLOB` to `TEXT` using row-based replication because changes to `BLOB` and `TEXT` columns are represented using the same format in the binary log.

Supported conversions for attribute promotion when using row-based replication are shown in the following table:

<table>
<thead>
<tr>
<th>From (Master)</th>
<th>To (Slave)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARY</td>
<td>CHAR</td>
</tr>
<tr>
<td>BLOB</td>
<td>TEXT</td>
</tr>
<tr>
<td>CHAR</td>
<td>BINARY</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>NUMERIC</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>TEXT</td>
<td>BLOB</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>VARBINARY</td>
</tr>
</tbody>
</table>

**Note**

In all cases, the size or width of the column on the slave must be equal to or greater than that of the column on the master. For example, you can replicate from a `CHAR(10)` column on the master to a column that uses `BINARY(10)` or `BINARY(25)` on the slave, but you cannot replicate from a `CHAR(10)` column on the master to a `BINARY(5)` column on the slave.

Any unique index (including primary keys) having a prefix must use a prefix of the same length on both master and slave; in such cases, differing prefix lengths are disallowed. It is possible to use a nonunique index whose prefix length differs between master and slave, but this can cause serious performance issues, particularly when the prefix used on the master is longer. This is due to the fact that 2 unique prefixes of a given length may no longer be unique at a shorter length; for example, the words `catalogue` and `catamount` have the 5-character prefixes `catal` and `catam`, respectively, but share the same 4-character prefix (`cata`). This can lead to queries that use such indexes executing less efficiently on the slave, when
Replication with Differing Table Definitions on Master and Slave

A shorter prefix is employed in the slave's definition of the same index than on the master.

For DECIMAL and NUMERIC columns, both the mantissa (M) and the number of decimals (D) must be the same size or larger on the slave as compared with the master. For example, replication from a NUMERIC(5,4) to a DECIMAL(6,4) works, but not from a NUMERIC(5,4) to a DECIMAL(5,3).

MySQL (except for MySQL Cluster) does not support attribute promotion of any of the following data types to or from any other data type when using row-based replication:

- INT (including TINYINT, SMALLINT, MEDIUMINT, BIGINT).
  Promotion between INT subtypes—for example, from SMALLINT to BIGINT—is also not supported.
- SET or ENUM.
- FLOAT or DOUBLE.
- All of the data types relating to dates, times, or both: DATE, TIME, DATETIME, TIMESTAMP, and YEAR.

Attribute promotion and demotion (MySQL Cluster). Beginning with MySQL Cluster NDB 6.3.33, 7.0.14, and 7.1.3, MySQL Cluster Replication supports attribute promotion and demotion between smaller data types and larger types. It is also possible to specify whether or not to permit lossy (truncated) or non-lossy conversions of demoted column values, as explained later in this section.

Lossy and non-lossy conversions. In the event that the target type cannot represent the value being inserted, a decision must be made on how to handle the conversion. If we permit the conversion but truncate (or otherwise modify) the source value to achieve a “fit” in the target column, we make what is known as a lossy conversion. A conversion which does not require truncation or similar modifications to fit the source column value in the target column is a non-lossy conversion.

Type conversion modes (slave_type_conversions variable). The setting of the slave_type_conversions global server variable controls the type conversion mode used on the slave. This variable takes a set of values from the following table, which shows the effects of each mode on the slave's type-conversion behavior:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL_LOSSY</td>
<td>In this mode, type conversions that would mean loss of information are permitted.</td>
</tr>
<tr>
<td></td>
<td>This does not imply that non-lossy conversions are permitted, merely that only cases requiring either lossy conversions or no conversion at all are permitted: for example, enabling only this mode permits an INT column to be converted to TINYINT (a lossy conversion), but not a TINYINT column to an INT column (non-lossy). Attempting the latter conversion in this case would cause replication to stop with an error on the slave.</td>
</tr>
<tr>
<td>ALL_NON_LOSSY</td>
<td>This mode permits conversions that do not require truncation or other special handling of the source value; that is, it permits conversions where the target type has a wider range than the source type.</td>
</tr>
<tr>
<td></td>
<td>Setting this mode has no bearing on whether lossy conversions are permitted; this is controlled with the ALL_LOSSY mode. If only ALL_NON_LOSSY is set, but not ALL_LOSSY, then attempting a</td>
</tr>
</tbody>
</table>
Replication with Differing Table Definitions on Master and Slave

## Mode | Effect
--- | ---
 | conversion that would result in the loss of data (such as INT to TINYINT, or CHAR(25) to VARCHAR(20)) causes the slave to stop with an error.
ALL_LOSSY, ALL_NON_LOSSY | When this mode is set, all supported type conversions are permitted, whether or not they are lossy conversions.
[empty] | When slave_type_conversions is not set, no attribute promotion or demotion is permitted; this means that all columns in the source and target tables must be of the same types.

This mode is the default.

Changing the type conversion mode requires restarting the slave with the new slave_type_conversions setting.

**Supported conversions.** Supported conversions between different but similar data types are shown in the following list:

- Between any of the integer types TINYINT, SMALLINT, MEDIUMINT, INT, and BIGINT.

  This includes conversions between the signed and unsigned versions of these types.

  Lossy conversions are made by truncating the source value to the maximum (or minimum) permitted by the target column. For insuring non-lossy conversions when going from unsigned to signed types, the target column must be large enough to accommodate the range of values in the source column. For example, you can demote TINYINT UNSIGNED non-lossily to SMALLINT, but not to TINYINT.

- Between any of the decimal types DECIMAL, FLOAT, DOUBLE, and NUMERIC.

  FLOAT to DOUBLE is a non-lossy conversion; DOUBLE to FLOAT can only be handled lossily. A conversion from DECIMAL \( (M, D) \) to DECIMAL \((M', D')\) where \( D' \geq D \) and \((M'-D') \geq (M-D)\) are non-lossy; for any case where \( M' < M, D' < D \), or both, only a lossy conversion can be made.

  For any of the decimal types, if a value to be stored cannot be fit in the target type, the value is rounded down according to the rounding rules defined for the server elsewhere in the documentation. See Rounding Behavior, for information about how this is done for decimal types.

- Between any of the string types CHAR, VARCHAR, and TEXT, including conversions between different widths.

  Conversion of a CHAR, VARCHAR, or TEXT to a CHAR, VARCHAR, or TEXT column the same size or larger is never lossy. Lossy conversion is handled by inserting only the first \( N \) characters of the string on the slave, where \( N \) is the width of the target column.

  **Important**
  Replication between columns using different character sets is not supported.

- Between any of the binary data types BINARY, VARBINARY, and BLOB, including conversions between different widths.

  Conversion of a BINARY, VARBINARY, or BLOB to a BINARY, VARBINARY, or BLOB column the same size or larger is never lossy. Lossy conversion is handled by inserting only the first \( N \) bytes of the string on the slave, where \( N \) is the width of the target column.

- Between any 2 BIT columns of any 2 sizes.
When inserting a value from a BIT(M) column into a BIT(M') column, where M' > M, the most significant bits of the BIT(M') columns are cleared (set to zero) and the M bits of the BIT(M) value are set as the least significant bits of the BIT(M') column.

When inserting a value from a source BIT(M) column into a target BIT(M') column, where M' < M, the maximum possible value for the BIT(M') column is assigned; in other words, an “all-set” value is assigned to the target column.

Conversions between types not in the previous list are not permitted.

4.1.11 Replication and DIRECTORY Table Options

If a DATA DIRECTORY or INDEX DIRECTORY table option is used in a CREATE TABLE statement on the master server, the table option is also used on the slave. This can cause problems if no corresponding directory exists in the slave host file system or if it exists but is not accessible to the slave server. This can be overridden by using the NO_DIR_IN_CREATE server SQL mode on the slave, which causes the slave to ignore the DATA DIRECTORY and INDEX DIRECTORY table options when replicating CREATE TABLE statements. The result is that MyISAM data and index files are created in the table’s database directory.

For more information, see Server SQL Modes.

4.1.12 Replication of Invoked Features

Replication of invoked features such as user-defined functions (UDFs) and stored programs (stored procedures and functions, triggers, and events) was re-implemented in MySQL 5.1.18 to provide the following characteristics:

• The effects of the feature are always replicated.

• The following statements are replicated using statement-based replication:
  • CREATE EVENT
  • ALTER EVENT
  • DROP EVENT
  • CREATE PROCEDURE
  • DROP PROCEDURE
  • CREATE FUNCTION
  • DROP FUNCTION
  • CREATE TRIGGER
  • DROP TRIGGER

However, the effects of features created, modified, or dropped using these statements are replicated using row-based replication.

Note

Attempting to replicate invoked features using statement-based replication produces the warning Statement may not be safe to log in statement format. (Prior to MySQL 5.1.36, this was Statement is not
Replication of Invoked Features

Safe to log in statement format; see Bug #42415.) For example, trying to replicate a UDF with statement-based replication generates this warning because it currently cannot be determined by the MySQL server whether the UDF is deterministic. If you are absolutely certain that the invoked feature's effects are deterministic, you can safely disregard such warnings.

- In the case of CREATE EVENT and ALTER EVENT:

  - The status of the event is set to SLAVESIDE_DISABLED on the slave regardless of the state specified (this does not apply to DROP EVENT).

  - The master on which the event was created is identified on the slave by its server ID. The ORIGINATOR column in INFORMATION_SCHEMA.EVENTS and the originator column in mysql.event were added to these tables in MySQL 5.1.18 to store this information. See The INFORMATION_SCHEMA EVENTS Table, and SHOW EVENTS Syntax, for more information.

  - The feature implementation resides on the slave in a renewable state so that if the master fails, the slave can be used as the master without loss of event processing.

To determine whether there are any scheduled events on a MySQL server that were created on a different server (that was acting as a replication master), query the INFORMATION_SCHEMA.EVENTS table in a manner similar to what is shown here:

```
SELECT EVENT_SCHEMA, EVENT_NAME
FROM INFORMATION_SCHEMA.EVENTS
WHERE STATUS = 'SLAVESIDE_DISABLED';
```

Alternatively, you can use the SHOW EVENTS statement, like this:

```
SHOW EVENTS
WHERE STATUS = 'SLAVESIDE_DISABLED';
```

When promoting a replication slave having such events to a replication master, you must enable each event using ALTER EVENT event_name ENABLED, where event_name is the name of the event.

If more than one master was involved in creating events on this slave, and you wish to identify events that were created only on a given master having the server ID master_id, modify the previous query on the EVENTS table to include the ORIGINATOR column, as shown here:

```
SELECT EVENT_SCHEMA, EVENT_NAME, ORIGINATOR
FROM INFORMATION_SCHEMA.EVENTS
WHERE STATUS = 'SLAVESIDE_DISABLED'
AND ORIGINATOR = 'master_id'
```

You can employ ORIGINATOR with the SHOW EVENTS statement in a similar fashion:

```
SHOW EVENTS
WHERE STATUS = 'SLAVESIDE_DISABLED'
AND ORIGINATOR = 'master_id'
```

Before enabling events that were replicated from the master, you should disable the MySQL Event Scheduler on the slave (using a statement such as SET GLOBAL event_scheduler = OFF;), run any necessary ALTER EVENT statements, restart the server, then re-enable the Event Scheduler on the slave afterward (using a statement such as SET GLOBAL event_scheduler = ON;).

If you later demote the new master back to being a replication slave, you must disable manually all events enabled by the ALTER EVENT statements. You can do this by storing in a separate table the event names

101
Replication and Floating-Point Values

from the SELECT statement shown previously, or using ALTER EVENT statements to rename the events with a common prefix such as replicated_ to identify them.

If you rename the events, then when demoting this server back to being a replication slave, you can identify the events by querying the EVENTS table, as shown here:

```sql
SELECT CONCAT(EVENT_SCHEMA, '.', EVENT_NAME) AS 'Db.Event'
FROM INFORMATION_SCHEMA.EVENTS
WHERE INSTR(EVENT_NAME, 'replicated_') = 1;
```

4.1.13 Replication and Floating-Point Values

With statement-based replication, values are converted from decimal to binary. Because conversions between decimal and binary representations of them may be approximate, comparisons involving floating-point values are inexact. This is true for operations that use floating-point values explicitly, or that use values that are converted to floating-point implicitly. Comparisons of floating-point values might yield different results on master and slave servers due to differences in computer architecture, the compiler used to build MySQL, and so forth. See Type Conversion in Expression Evaluation, and Problems with Floating-Point Values.

4.1.14 Replication and FLUSH

Some forms of the FLUSH statement are not logged because they could cause problems if replicated to a slave: FLUSH LOGS, FLUSH MASTER, FLUSH SLAVE, and FLUSH TABLES WITH READ LOCK. For a syntax example, see FLUSH Syntax. The FLUSH TABLES, ANALYZE TABLE, OPTIMIZE TABLE, and REPAIR TABLE statements are written to the binary log and thus replicated to slaves. This is not normally a problem because these statements do not modify table data.

However, this behavior can cause difficulties under certain circumstances. If you replicate the privilege tables in the mysql database and update those tables directly without using GRANT, you must issue a FLUSH PRIVILEGES on the slaves to put the new privileges into effect. In addition, if you use FLUSH TABLES when renaming a MyISAM table that is part of a MERGE table, you must issue FLUSH TABLES manually on the slaves. These statements are written to the binary log unless you specify NO_WRITE_TO_BINLOG or its alias LOCAL.

4.1.15 Replication and System Functions

Certain functions do not replicate well under some conditions:

- The USER(), CURRENT_USER() (or CURRENT_USER), UUID(), VERSION(), LOAD_FILE(), and RAND() functions are replicated without change and thus do not work reliably on the slave unless row-based replication is enabled. (See Section 2.2, “Replication Formats”.)

For early implementations of mixed-format logging, stored functions, triggers, and views that use these functions in their body do not replicate reliably in mixed-format logging mode because the logging did not switch from statement-based to row-based format. For example, INSERT INTO t SELECT FROM v, where v is a view that selects UUID() could cause problems. This limitation is lifted in MySQL 5.1.12.

Beginning with MySQL 5.1.23, USER() and CURRENT_USER() are automatically replicated using row-based replication when using MIXED mode, and generate a warning in STATEMENT mode. (Bug #28086) (See also Section 4.1.7, “Replication of CURRENT_USER()”.)

Beginning with MySQL 5.1.42, VERSION() is also automatically replicated using row-based replication when using MIXED mode, and generates a warning in STATEMENT mode. (Bug #47995) Beginning with MySQL 5.1.43, this is also true with regard to the RAND() function. (Bug #49222)
• For `NOW()`, the binary log includes the timestamp. This means that the value *as returned by the call to this function on the master* is replicated to the slave. This can lead to a possibly unexpected result when replicating between MySQL servers in different time zones. Suppose that the master is located in New York, the slave is located in Stockholm, and both servers are using local time. Suppose further that, on the master, you create a table `mytable`, perform an `INSERT` statement on this table, and then select from the table, as shown here:

```sql
mysql> CREATE TABLE mytable (mycol TEXT);
Query OK, 0 rows affected (0.06 sec)
mysql> INSERT INTO mytable VALUES ( NOW() );
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM mytable;
+---------------------+
<table>
<thead>
<tr>
<th>mycol</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-09-01 12:00:00</td>
</tr>
</tbody>
</table>
+---------------------+
1 row in set (0.00 sec)
```

Local time in Stockholm is 6 hours later than in New York; so, if you issue `SELECT NOW()` on the slave at that exact same instant, the value `2009-09-01 18:00:00` is returned. For this reason, if you select from the slave's copy of `mytable` after the `CREATE TABLE` and `INSERT` statements just shown have been replicated, you might expect `mycol` to contain the value `2009-09-01 18:00:00`. However, this is not the case; when you select from the slave's copy of `mytable`, you obtain exactly the same result as on the master:

```sql
mysql> SELECT * FROM mytable;
+---------------------+
<table>
<thead>
<tr>
<th>mycol</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-09-01 12:00:00</td>
</tr>
</tbody>
</table>
+---------------------+
1 row in set (0.00 sec)
```

Unlike `NOW()`, the `SYSDATE()` function is not replication-safe because it is not affected by `SET TIMESTAMP` statements in the binary log and is nondeterministic if statement-based logging is used. This is not a problem if row-based logging is used.

An alternative is to use the `--sysdate-is-now` option to cause `SYSDATE()` to be an alias for `NOW()`. This must be done on the master and the slave to work correctly. In such cases, a warning is still issued by this function, but can safely be ignored as long as `--sysdate-is-now` is used on both the master and the slave.

Beginning with MySQL 5.1.42, `SYSDATE()` is automatically replicated using row-based replication when using `MIXED` mode, and generates a warning in `STATEMENT` mode. (Bug #47995)

See also Section 4.1.34, “Replication and Time Zones”.

• *The following restriction applies to statement-based replication only, not to row-based replication.* The `GET_LOCK()`, `RELEASE_LOCK()`, `IS_FREE_LOCK()`, and `IS_USED_LOCK()` functions that handle user-level locks are replicated without the slave knowing the concurrency context on the master. Therefore, these functions should not be used to insert into a master table because the content on the slave would differ. For example, do not issue a statement such as `INSERT INTO mytable VALUES(GET_LOCK(...))`.

Beginning with MySQL 5.1.42, these functions are automatically replicated using row-based replication when using `MIXED` mode, and generate a warning in `STATEMENT` mode. (Bug #47995)
As a workaround for the preceding limitations when statement-based replication is in effect, you can use the strategy of saving the problematic function result in a user variable and referring to the variable in a later statement. For example, the following single-row `INSERT` is problematic due to the reference to the `UUID()` function:

```
INSERT INTO t VALUES(UUID());
```

To work around the problem, do this instead:

```
SET @my_uuid = UUID();
INSERT INTO t VALUES(@my_uuid);
```

That sequence of statements replicates because the value of `@my_uuid` is stored in the binary log as a user-variable event prior to the `INSERT` statement and is available for use in the `INSERT`.

The same idea applies to multiple-row inserts, but is more cumbersome to use. For a two-row insert, you can do this:

```
SET @my_uuid1 = UUID(); @my_uuid2 = UUID();
INSERT INTO t VALUES(@my_uuid1),(@my_uuid2);
```

However, if the number of rows is large or unknown, the workaround is difficult or impracticable. For example, you cannot convert the following statement to one in which a given individual user variable is associated with each row:

```
INSERT INTO t2 SELECT UUID(), * FROM t1;
```

Within a stored function, `RAND()` replicates correctly as long as it is invoked only once during the execution of the function. (You can consider the function execution timestamp and random number seed as implicit inputs that are identical on the master and slave.)

The `FOUND_ROWS()` and `ROW_COUNT()` functions are not replicated reliably using statement-based replication. A workaround is to store the result of the function call in a user variable, and then use that in the `INSERT` statement. For example, if you wish to store the result in a table named `mytable`, you might normally do so like this:

```
SELECT SQL_CALC_FOUND_ROWS FROM mytable LIMIT 1;
INSERT INTO mytable VALUES( FOUND_ROWS() );
```

However, if you are replicating `mytable`, you should use `SELECT ... INTO`, and then store the variable in the table, like this:

```
SELECT SQL_CALC_FOUND_ROWS INTO @found_rows FROM mytable LIMIT 1;
INSERT INTO mytable VALUES(@found_rows);
```

In this way, the user variable is replicated as part of the context, and applied on the slave correctly.

Beginning with MySQL 5.1.23, these functions are automatically replicated using row-based replication when using `MIXED` mode, and generate a warning in `STATEMENT` mode. (Bug #12092, Bug #30244)

Prior to MySQL 5.1.73, the value of `LAST_INSERT_ID()` was not replicated correctly if any filtering options such as `--replicate-ignore-db` and `--replicate-do-table` were enabled on the slave. (Bug #17234370, BUG# 69861)
4.1.16 Replication and LIMIT

Statement-based replication of **LIMIT** clauses in **DELETE**, **UPDATE**, and **INSERT ... SELECT** statements is unsafe since the order of the rows affected is not defined. (Such statements can be replicated correctly with statement-based replication only if they also contain an **ORDER BY** clause.) Beginning with MySQL 5.1.24, when such a statement is encountered:

- When using **STATEMENT** mode, a warning that the statement is not safe for statement-based replication is now issued.

  When using **STATEMENT** mode, warnings are issued for DML statements containing **LIMIT** even when they also have an **ORDER BY** clause (and so are made deterministic). This is a known issue. (Bug #42851)

- When using **MIXED** mode, the statement is now automatically replicated using row-based mode.

4.1.17 Replication and LOAD DATA INFILE

The **LOAD DATA INFILE** statement was not always replicated correctly to a slave running MySQL 5.1.42 or earlier from a master running MySQL 4.0 or earlier. When using statement-based replication, the **LOAD DATA INFILE** statement **CONCURRENT** option was not replicated. This issue was fixed in MySQL 5.1.43. This issue does not have any impact on **CONCURRENT** option handling when using row-based replication in MySQL 5.1 or later. (Bug #34628)

In MySQL 5.1.52 and later, **LOAD DATA INFILE** is considered unsafe (see Section 2.2.3, “Determination of Safe and Unsafe Statements in Binary Logging”). It causes a warning when using statement-based logging format, and is logged using row-based format when using mixed-format logging.

4.1.18 Replication and the Slow Query Log

Prior to MySQL 5.1.45, replication slaves did not write replicated queries to the slow query log, even if the same queries were written to the slow query log on the master. (Bug #23300)

4.1.19 Replication and REPAIR TABLE

When used on a corrupted or otherwise damaged table, it is possible for the **REPAIR TABLE** statement to delete rows that cannot be recovered. However, any such modifications of table data performed by this statement are not replicated, which can cause master and slave to lose synchronization. For this reason, in the event that a table on the master becomes damaged and you use **REPAIR TABLE** to repair it, you should first stop replication (if it is still running) before using **REPAIR TABLE**, then afterward compare the master's and slave's copies of the table and be prepared to correct any discrepancies manually, before restarting replication.

4.1.20 Replication and Master or Slave Shutdowns

It is safe to shut down a master server and restart it later. When a slave loses its connection to the master, the slave tries to reconnect immediately and retries periodically if that fails. The default is to retry every 60 seconds. This may be changed with the **CHANGE MASTER TO** statement or **--master-connect-retry** option. A slave also is able to deal with network connectivity outages. However, the slave notices the network outage only after receiving no data from the master for **slave_net_timeout** seconds. If your outages are short, you may want to decrease **slave_net_timeout**. See **Server System Variables**.

An unclean shutdown (for example, a crash) on the master side can result in the master binary log having a final position less than the most recent position read by the slave, due to the master binary log file not
Replication and max_allowed_packet

being flushed. This can cause the slave not to be able to replicate when the master comes back up. Setting sync_binlog=1 in the master my.cnf file helps to minimize this problem because it causes the master to flush its binary log more frequently.

Shutting down a slave cleanly is safe because it keeps track of where it left off. However, be careful that the slave does not have temporary tables open; see Section 4.1.23, “Replication and Temporary Tables”. Unclean shutdowns might produce problems, especially if the disk cache was not flushed to disk before the problem occurred:

• For transactions, the slave commits and then updates relay-log.info. If a crash occurs between these two operations, relay log processing will have proceeded further than the information file indicates and the slave will re-execute the events from the last transaction in the relay log after it has been restarted.

• A similar problem can occur if the slave updates relay-log.info but the server host crashes before the write has been flushed to disk. Writes are not forced to disk because the server relies on the operating system to flush the file from time to time.

The fault tolerance of your system for these types of problems is greatly increased if you have a good uninterruptible power supply.

4.1.21 Replication and max_allowed_packet

max_allowed_packet sets an upper limit on the size of any single message between the MySQL server and clients, including replication slaves. If you are replicating large column values (such as might be found in TEXT or BLOB columns) and max_allowed_packet is too small on the master, the master fails with an error, and the slave shuts down the I/O thread. If max_allowed_packet is too small on the slave, this also causes the slave to stop the I/O thread.

Prior to MySQL 5.1.40, Last_IO_Error and Last_IO_Errno in the output of SHOW SLAVE STATUS were not set in the event that replication failed due to exceeding max_allowed_packet (Bug #42914).

Row-based replication currently sends all columns and column values for updated rows from the master to the slave, including values of columns that were not actually changed by the update. This means that, when you are replicating large column values using row-based replication, you must take care to set max_allowed_packet large enough to accommodate the largest row in any table to be replicated, even if you are replicating updates only, or you are inserting only relatively small values.

4.1.22 Replication and MEMORY Tables

When a master server shuts down and restarts, its MEMORY tables become empty. To replicate this effect to slaves, the first time that the master uses a given MEMORY table after startup, it logs an event that notifies slaves that the table must be emptied by writing a DELETE statement for that table to the binary log.

When a slave server shuts down and restarts, its MEMORY tables become empty. This causes the slave to be out of synchrony with the master and may lead to other failures or cause the slave to stop:

• Row-format updates and deletes received from the master may fail with Can't find record in 'memory_table'.

• Statements such as INSERT INTO ... SELECT FROM memory_table may insert a different set of rows on the master and slave.

The safe way to restart a slave that is replicating MEMORY tables is to first drop or delete all rows from the MEMORY tables on the master and wait until those changes have replicated to the slave. Then it is safe to restart the slave.
An alternative restart method may apply in some cases. When `binlog_format=ROW`, you can prevent the slave from stopping if you set `slave_exec_mode=IDEMPOTENT` before you start the slave again. This allows the slave to continue to replicate, but its MEMORY tables will still be different from those on the master. This can be okay if the application logic is such that the contents of MEMORY tables can be safely lost (for example, if the MEMORY tables are used for caching). `slave_exec_mode=IDEMPOTENT` applies globally to all tables, so it may hide other replication errors in non-MEMORY tables.

(The method just described is not applicable in MySQL Cluster, where `slave_exec_mode` is always IDEMPOTENT, and cannot be changed.)

The size of MEMORY tables is limited by the value of the `max_heap_table_size` system variable, which is not replicated (see Section 4.1.38, “Replication and Variables”). A change in `max_heap_table_size` takes effect for MEMORY tables that are created or updated using `ALTER TABLE ... ENGINE = MEMORY` or `TRUNCATE TABLE` following the change, or for all MEMORY tables following a server restart. If you increase the value of this variable on the master without doing so on the slave, it becomes possible for a table on the master to grow larger than its counterpart on the slave, leading to inserts that succeed on the master but fail on the slave with `Table is full` errors. This is a known issue (Bug #48666). In such cases, you must set the global value of `max_heap_table_size` on the slave as well as on the master, then restart replication. It is also recommended that you restart both the master and slave MySQL servers, to insure that the new value takes complete (global) effect on each of them.

See The MEMORY Storage Engine, for more information about MEMORY tables.

### 4.1.23 Replication and Temporary Tables

The discussion in the following paragraphs does not apply when `binlog_format=ROW` because, in that case, temporary tables are not replicated; this means that there are never any temporary tables on the slave to be lost in the event of an unplanned shutdown by the slave. The remainder of this section applies only when using statement-based or mixed-format replication. Loss of replicated temporary tables on the slave can be an issue, whenever `binlog_format` is `STATEMENT` or `MIXED`, for statements involving temporary tables that can be logged safely using statement-based format. For more information about row-based replication and temporary tables, see RBL, RBR, and temporary tables.

**Safe slave shutdown when using temporary tables.** Temporary tables are replicated except in the case where you stop the slave server (not just the slave threads) and you have replicated temporary tables that are open for use in updates that have not yet been executed on the slave. If you stop the slave server, the temporary tables needed by those updates are no longer available when the slave is restarted. To avoid this problem, do not shut down the slave while it has temporary tables open. Instead, use the following procedure:

1. Issue a `STOP SLAVE SQL_THREAD` statement.
2. Use `SHOW STATUS` to check the value of the `Slave_open_temp_tables` variable.
3. If the value is not 0, restart the slave SQL thread with `START SLAVE SQL_THREAD` and repeat the procedure later.
4. When the value is 0, issue a `mysqladmin shutdown` command to stop the slave.

**Temporary tables and replication options.** By default, all temporary tables are replicated; this happens whether or not there are any matching `--replicate-do-db`, `--replicate-do-table`, or `--replicate-wild-do-table` options in effect. However, the `--replicate-ignore-table` and `--replicate-wild-ignore-table` options are honored for temporary tables.

A recommended practice when using statement-based or mixed-format replication is to designate a prefix for exclusive use in naming temporary tables that you do not want replicated, then employ a --
Replication of the mysql System Database

replicate-wild-ignore-table option to match that prefix. For example, you might give all such tables names beginning with norep (such as norepmytable, norepyourtable, and so on), then use --replicate-wild-ignore-table=norep% to prevent them from being replicated.

4.1.24 Replication of the mysql System Database

MySQL 5.1.14 and later. Data modification statements made to tables in the mysql database are replicated according to the value of binlog_format; if this value is MIXED, these statement are replicated using the row-based format. However, statements that would normally update this information indirectly—such GRANT, REVOKE, and statements manipulating triggers, stored routines, and views—are replicated to slaves using statement-based replication.

MySQL 5.1.13 and earlier. User privileges are replicated only if the mysql database is replicated. That is, the GRANT, REVOKE, SET PASSWORD, CREATE USER, and DROP USER statements take effect on the slave only if the replication setup includes the mysql database.

4.1.25 Replication and the Query Optimizer

It is possible for the data on the master and slave to become different if a statement is written in such a way that the data modification is nondeterministic; that is, left up the query optimizer. (In general, this is not a good practice, even outside of replication.) Examples of nondeterministic statements include DELETE or UPDATE statements that use LIMIT with no ORDER BY clause; see Section 4.1.16, “Replication and LIMIT“, for a detailed discussion of these.

4.1.26 Replication and Partitioning

Replication is supported between partitioned tables as long as they use the same partitioning scheme and otherwise have the same structure except where an exception is specifically allowed (see Section 4.1.10, “Replication with Differing Table Definitions on Master and Slave“).

Replication between tables having different partitioning is generally not supported. This because statements (such as ALTER TABLE ... DROP PARTITION) acting directly on partitions in such cases may produce different results on master and slave. In the case where a table is partitioned on the master but not on the slave, any statements operating on partitions on the master's copy of the slave fail on the slave. When the slave's copy of the table is partitioned but the master's copy is not, statements acting on partitions cannot be run on the master without causing errors there.

Due to these dangers of causing replication to fail entirely (on account of failed statements) and of inconsistencies (when the result of a partition-level SQL statement produces different results on master and slave), we recommend that insure that the partitioning of any tables to be replicated from the master is matched by the slave's versions of these tables.

4.1.27 Replication and Reserved Words

You can encounter problems when you attempt to replicate from an older master to a newer slave and you make use of identifiers on the master that are reserved words in the newer MySQL version running on the slave. An example of this is using a table column named range on a 5.0 master that is replicating to a 5.1 or higher slave because RANGE is a reserved word beginning in MySQL 5.1. Replication can fail in such cases with Error 1064 You have an error in your SQL syntax.... even if a database or table named using the reserved word or a table having a column named using the reserved word is excluded from replication. This is due to the fact that each SQL event must be parsed by the slave prior to execution, so that the slave knows which database object or objects would be affected; only after the event is parsed can the slave apply any filtering rules defined by --replicate-do-db, --replicate-do-table, --replicate-ignore-db, and --replicate-ignore-table.
To work around the problem of database, table, or column names on the master which would be regarded as reserved words by the slave, do one of the following:

- Use one or more `ALTER TABLE` statements on the master to change the names of any database objects where these names would be considered reserved words on the slave, and change any SQL statements that use the old names to use the new names instead.

- In any SQL statements using these database object names, write the names as quoted identifiers using backtick characters (``).

For listings of reserved words by MySQL version, see Reserved Words, in the MySQL Server Version Reference. For identifier quoting rules, see Schema Object Names.

### 4.1.28 SET PASSWORD and Row-Based Replication

Row-based replication of `SET PASSWORD` statements from a MySQL 5.1 master to a MySQL 5.5 slave did not work correctly prior to MySQL 5.1.53 on the master and MySQL 5.5.7 on the slave (see Bug #57098, Bug #57357).

### 4.1.29 Slave Errors During Replication

If a statement produces the same error (identical error code) on both the master and the slave, the error is logged, but replication continues.

If a statement produces different errors on the master and the slave, the slave SQL thread terminates, and the slave writes a message to its error log and waits for the database administrator to decide what to do about the error. This includes the case that a statement produces an error on the master or the slave, but not both. To address the issue, connect to the slave manually and determine the cause of the problem. `SHOW SLAVE STATUS` is useful for this. Then fix the problem and run `START SLAVE`. For example, you might need to create a nonexistent table before you can start the slave again.

If this error code validation behavior is not desirable, some or all errors can be masked out (ignored) with the `--slave-skip-errors` option.

For nontransactional storage engines such as `MyISAM`, it is possible to have a statement that only partially updates a table and returns an error code. This can happen, for example, on a multiple-row insert that has one row violating a key constraint, or if a long update statement is killed after updating some of the rows. If that happens on the master, the slave expects execution of the statement to result in the same error code. If it does not, the slave SQL thread stops as described previously.

If you are replicating between tables that use different storage engines on the master and slave, keep in mind that the same statement might produce a different error when run against one version of the table, but not the other, or might cause an error for one version of the table, but not the other. For example, since `MyISAM` ignores foreign key constraints, an `INSERT` or `UPDATE` statement accessing an `InnoDB` table on the master might cause a foreign key violation but the same statement performed on a `MyISAM` version of the same table on the slave would produce no such error, causing replication to stop.

### 4.1.30 Replication of Server-Side Help Tables

The server maintains tables in the `mysql` database that store information for the `HELP` statement (see `HELP Syntax`). These tables can be loaded manually as described at Server-Side Help.

Help table content is derived from the MySQL Reference Manual. There are versions of the manual specific to each MySQL release series, so help content is specific to each series as well. Normally, you load a version of help content that matches the server version. This has implications for replication. For
example, you would load MySQL 5.5 help content into a MySQL 5.5 master server, but not necessarily replicate that content to a MySQL 5.6 slave server for which 5.6 help content is more appropriate.

This section describes how to manage help table content upgrades when your servers participate in replication. Server versions are one factor in this task. Another is that help table structure may differ between the master and the slave.

Assume that help content is stored in a file named `fill_help_tables.sql`. In MySQL distributions, this file is located under the `share` or `share/mysql` directory, and the most recent version is always available for download from http://dev.mysql.com/doc/index-other.html.

To upgrade help tables, using the following procedure. Connection parameters are not shown for the `mysql` commands discussed here; in all cases, connect to the server using an account such as `root` that has privileges for modifying tables in the `mysql` database.

1. Upgrade your servers by running `mysql_upgrade`, first on the slaves and then on the master. This is the usual principle of upgrading slaves first.

2. Decide whether you want to replicate help table content from the master to its slaves. If not, load the content on the master and each slave individually. Otherwise, check for and resolve any incompatibilities between help table structure on the master and its slaves, then load the content into the master and let it replicate to the slaves.

More detail about these two methods of loading help table content follows.

### Loading Help Table Content Without Replication to Slaves

To load help table content without replication, run this command on the master and each slave individually, using a `fill_help_tables.sql` file containing content appropriate to the server version (enter the command on one line):

```
mysql --init-command="SET sql_log_bin=0"
  mysql < fill_help_tables.sql
```

Use the `--init-command` option on each server, including the slaves, in case a slave also acts as a master to other slaves in your replication topology. The `SET` statement suppresses binary logging. After the command has been run on each server to be upgraded, you are done.

### Loading Help Table Content With Replication to Slaves

If you do want to replicate help table content, check for help table incompatibilities between your master and its slaves. The `url` column in the `help_category` and `help_topic` tables was originally `CHAR(128)`, but is `TEXT` in newer MySQL versions to accommodate longer URLs. To check help table structure, use this statement:

```
SELECT TABLE_NAME, COLUMN_NAME, COLUMN_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'mysql'
AND COLUMN_NAME = 'url';
```

For tables with the old structure, the statement produces this result:

```
+---------------+-------------+-------------+
<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>COLUMN_NAME</th>
<th>COLUMN_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>help_category</td>
<td>url</td>
<td>char(128)</td>
</tr>
</tbody>
</table>
```
For tables with the new structure, the statement produces this result:

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>COLUMN_NAME</th>
<th>COLUMN_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>help_category</td>
<td>url</td>
<td>text</td>
</tr>
<tr>
<td>help_topic</td>
<td>url</td>
<td>text</td>
</tr>
</tbody>
</table>

If the master and slave both have the old structure or both have the new structure, they are compatible and you can replicate help table content by executing this command on the master:

```bash
mysql mysql < fill_help_tables.sql
```

The table content will load into the master, then replicate to the slaves.

If the master and slave have incompatible help tables (one server has the old structure and the other has the new), you have a choice between not replicating help table content after all, or making the table structures compatible so that you can replicate the content.

- If you decide not to replicate the content after all, upgrade the master and slaves individually using `mysql` with the `--init-command` option, as described previously.

- If instead you decide to make the table structures compatible, upgrade the tables on the server that has the old structure. Suppose that your master server has the old table structure. Upgrade its tables to the new structure manually by executing these statements (binary logging is disabled here to prevent replication of the changes to the slaves, which already have the new structure):

  ```sql
  SET sql_log_bin=0;
  ALTER TABLE mysql.help_category ALTER COLUMN url TEXT;
  ALTER TABLE mysql.help_topic ALTER COLUMN url TEXT;
  
  Then run this command on the master:
  
  ```bash
  mysql mysql < fill_help_tables.sql
  ```

  The table content will load into the master, then replicate to the slaves.

### 4.1.31 Replication and Server SQL Mode

Using different server SQL mode settings on the master and the slave may cause the same `INSERT` statements to be handled differently on the master and the slave, leading the master and slave to diverge. For best results, you should always use the same server SQL mode on the master and on the slave. This advice applies whether you are using statement-based or row-based replication.

If you are replicating partitioned tables, using different SQL modes on the master and the slave is likely to cause issues. At a minimum, this is likely to cause the distribution of data among partitions to be different in the master’s and slave’s copies of a given table. It may also cause inserts into partitioned tables that succeed on the master to fail on the slave.

For more information, see [Server SQL Modes](#).

### 4.1.32 Replication Retries and Timeouts
The global system variable `slave_transaction_retries` affects replication as follows: If the slave SQL thread fails to execute a transaction because of an InnoDB deadlock or because it exceeded the `InnoDB innodb_lock_wait_timeout` value, or the NDBCLUSTER `TransactionDeadlockDetectionTimeout` or `TransactionInactiveTimeout` value, the slave automatically retries the transaction `slave_transaction_retries` times before stopping with an error. The default value is 10. The total retry count can be seen in the output of `SHOW STATUS`; see Server Status Variables.

### 4.1.33 Replication and TIMESTAMP

Older versions of MySQL (prior to 4.1) differed significantly in several ways in their handling of the `TIMESTAMP` data type from what is supported in MySQL versions 5.1 and newer; these include syntax extensions which are deprecated in MySQL 5.1, and that no longer supported in MySQL 5.5. This can cause problems (including replication failures) when replicating between MySQL Server versions, if you are using columns that are defined using the old `TIMESTAMP(N)` syntax. See Changes Affecting Upgrades to 5.1, for more information about the differences, how they can impact MySQL replication, and what you can do if you encounter such problems.

### 4.1.34 Replication and Time Zones

The same system time zone should be set for both master and slave. Otherwise, statements depending on the local time on the master are not replicated properly, such as statements that use the `NOW()` or `FROM_UNIXTIME()` functions. You can set the time zone in which MySQL server runs by using the `--timezone=timezone_name` option of the `mysqld_safe` script or by setting the `TZ` environment variable. See also Section 4.1.15, “Replication and System Functions”.

`CONVERT_TZ(...,...,@@session.time_zone)` is properly replicated only if both master and slave are running MySQL 5.0.4 or newer.

### 4.1.35 Replication and Transactions

**Mixing transactional and nontransactional statements within the same transaction.** In general, you should avoid transactions that update both transactional and nontransactional tables in a replication environment. You should also avoid using any statement that accesses both transactional (or temporary) and nontransactional tables and writes to any of them.

As of MySQL 5.1.44, the server uses these rules for binary logging:

- If the initial statements in a transaction are nontransactional, they are written to the binary log immediately. The remaining statements in the transaction are cached and not written to the binary log until the transaction is committed. (If the transaction is rolled back, the cached statements are written to the binary log only if they make nontransactional changes that cannot be rolled back. Otherwise, they are discarded.)

- For statement-based logging, logging of nontransactional statements is affected by the `binlog_direct_non_transactional_updates` system variable. When this variable is `OFF` (the default), logging is as just described. When this variable is `ON`, logging occurs immediately for nontransactional statements occurring anywhere in the transaction (not just initial nontransactional statements). Other statements are kept in the transaction cache and logged when the transaction commits. `binlog_direct_non_transactional_updates` has no effect for row-format or mixed-format binary logging.

**Transactional, nontransactional, and mixed statements.** To apply those rules, the server considers a statement nontransactional if it changes only nontransactional tables, and transactional if it changes only transactional tables. A statement that changes both
nontransactional and transactional tables is considered “mixed”. Mixed statements, like transactional
statements, are cached and logged when the transaction commits.

**Note**

A mixed statement is unrelated to mixed binary logging format.

Before MySQL 5.1.44, the rules for binary logging are similar to those just described, except that there is no `binlog_direct_non_transactional_updates` system variable to affect logging of transactional statements. Thus, the server immediately logs only the initial nontransactional statements in a transaction and caches the rest until commit time.

Before MySQL 5.1.31, the effect of the rules differs because the definition of transactional statement is different: In these earlier versions, a statement is nontransactional if the first changes it makes change nontransactional tables, transactional if the first changes it makes change transactional tables. “First” applies in the sense that a statement may have several effects if it involves such things as triggers, stored functions, or multiple-table updates. A mixed statement that changes both nontransactional and transactional tables is handled as nontransactional or transactional depending on the type of changes it makes first.

In situations where transactions mix updates to transactional and nontransactional tables, the order of statements in the binary log is correct, and all needed statements are written to the binary log even in case of a `ROLLBACK`. However, when a second connection updates the nontransactional table before the first connection transaction is complete, statements can be logged out of order because the second connection update is written immediately after it is performed, regardless of the state of the transaction being performed by the first connection.

**Using different storage engines on master and slave.** It is possible to replicate transactional tables on the master using nontransactional tables on the slave. For example, you can replicate an `InnoDB` master table as a `MyISAM` slave table. However, if you do this, there are problems if the slave is stopped in the middle of a `BEGIN ... COMMIT` block because the slave restarts at the beginning of the `BEGIN` block.

Beginning with MySQL 5.1.48, it is also safe to replicate transactions from `MyISAM` tables on the master to transactional tables—such as tables that use the `InnoDB` storage engine—on the slave. In such cases (beginning with MySQL 5.1.48), an `AUTOCOMMIT=1` statement issued on the master is replicated, thus enforcing `AUTOCOMMIT` mode on the slave.

When the storage engine type of the slave is nontransactional, transactions on the master that mix updates of transactional and nontransactional tables should be avoided because they can cause inconsistency of the data between the master transactional table and the slave nontransactional table. That is, such transactions can lead to master storage engine-specific behavior with the possible effect of replication going out of synchrony. MySQL does not issue a warning about this currently, so extra care should be taken when replicating transactional tables from the master to nontransactional tables on the slaves.

Beginning with MySQL Cluster NDB 6.2.14 and MySQL 5.1.24, every transaction (including `autocommit` transactions) is recorded in the binary log as though it starts with a `BEGIN` statement, and ends with either a `COMMIT` or a `ROLLBACK` statement. However, this does not apply to nontransactional changes; any statements affecting tables using a nontransactional storage engine such as `MyISAM` are regarded for this purpose as nontransactional, even when `autocommit` is enabled. (Bug #26395)

### 4.1.36 Replication and Triggers

With statement-based replication, triggers executed on the master also execute on the slave. With row-based replication, triggers executed on the master do not execute on the slave. Instead, the row changes on the master resulting from trigger execution are replicated and applied on the slave.
This behavior is by design. If under row-based replication the slave applied the triggers as well as the row changes caused by them, the changes would in effect be applied twice on the slave, leading to different data on the master and the slave.

If you want triggers to execute on both the master and the slave—perhaps because you have different triggers on the master and slave—you must use statement-based replication. However, to enable slave-side triggers, it is not necessary to use statement-based replication exclusively. It is sufficient to switch to statement-based replication only for those statements where you want this effect, and to use row-based replication the rest of the time.

Before MySQL 5.1.31, a trigger that was defined on a transactional table but that updated a nontransactional table could cause updates on the transactional table to be replicated before they were actually committed on the master, and not be rolled back correctly on the slave if they were rolled back on the master. (Bug #40116) See also Section 4.1.35, “Replication and Transactions”.

A statement invoking a trigger (or function) that causes an update to an AUTO_INCREMENT column is not replicated correctly using statement-based replication. Beginning with MySQL 5.1.40, such statements are marked as unsafe. (Bug #45677)

4.1.37 Replication and TRUNCATE TABLE

TRUNCATE TABLE is normally regarded as a DML statement, and so would be expected to be logged and replicated using row-based format when the binary logging mode is ROW or MIXED. However this caused issues when logging or replicating, in STATEMENT or MIXED mode, tables that used transactional storage engines such as InnoDB when the transaction isolation level was READ COMMITTED or READ UNCOMMITTED, which precludes statement-based logging.

Beginning with MySQL 5.1.32, TRUNCATE TABLE is treated for purposes of logging and replication as DDL rather than DML so that it can be logged and replicated as a statement. However, the effects of the statement as applicable to InnoDB and other transactional tables on replication slaves still follow the rules described in TRUNCATE TABLE Syntax governing such tables. (Bug #36763)

4.1.38 Replication and Variables

System variables are not replicated correctly when using STATEMENT mode, except for the following variables when they are used with session scope:

- auto_increment_increment
- auto_increment_offset
- character_set_client
- character_set_connection
- character_set_database
- character_set_server
- collation_connection
- collation_database
- collation_server
- foreign_key_checks
- identity
Replication and Variables

- last_insert_id
- lc_time_names
- pseudo_thread_id
- sql_auto_is_null
- time_zone
- timestamp
- unique_checks

When MIXED mode is used, the variables in the preceding list, when used with session scope, cause a
switch from statement-based to row-based logging. See Mixed Binary Logging Format.

sql_mode is also replicated except for the NO_DIR_IN_CREATE mode; the slave always preserves
its own value for NO_DIR_IN_CREATE, regardless of changes to it on the master. This is true for all
replication formats.

However, when mysqlbinlog parses a SET @@sql_mode = mode statement, the full mode value,
including NO_DIR_IN_CREATE, is passed to the receiving server. For this reason, replication of such a
statement may not be safe when STATEMENT mode is in use.

The storage_engine system variable is not replicated, regardless of the logging mode; this is intended
to facilitate replication between different storage engines.

The read_only system variable is not replicated. In addition, the enabling this variable has different
effects with regard to temporary tables, table locking, and the SET PASSWORD statement in different
MySQL versions.

The max_heap_table_size system variable is not replicated. Increasing the value of this variable on
the master without doing so on the slave can lead eventually to Table is full errors on the slave when
trying to execute INSERT statements on a MEMORY table on the master that is thus permitted to grow larger
than its counterpart on the slave. For more information, see Section 4.1.22, “Replication and MEMORY
Tables”.

In statement-based replication, session variables are not replicated properly when used in statements that
update tables. For example, the following sequence of statements will not insert the same data on the
master and the slave:

```
SET max_join_size=1000;
INSERT INTO mytable VALUES(@@max_join_size);
```

This does not apply to the common sequence:

```
SET time_zone=...;
INSERT INTO mytable VALUES(CONVERT_TZ(..., ..., @@time_zone));
```

Replication of session variables is not a problem when row-based replication is being used, in which case,
session variables are always replicated safely. See Section 2.2, “Replication Formats”.

In MySQL 5.1.20 and later (and in MySQL 5.0.46 and later in MySQL 5.0, for backward compatibility), the
following session variables are written to the binary log and honored by the replication slave when parsing
the binary log, regardless of the logging format:

- sql_mode
Replication and Views

- foreign_key_checks
- unique_checks
- character_set_client
- collation_connection
- collation_database
- collation_server
- sql_auto_is_null

**Important**

Even though session variables relating to character sets and collations are written to the binary log, replication between different character sets is not supported.

It is strongly recommended that you always use the same setting for the `lower_case_table_names` system variable on both master and slave. In particular, when a case-sensitive file system is used, setting this variable to 1 on the slave, but to a different value on the master, can cause two types of problems: Names of databases are not converted to lowercase; in addition, when using row-based replication names of tables are also not converted. Either of these problems can cause replication to fail. This is a known issue, which is fixed in MySQL 5.6.

### 4.1.39 Replication and Views

Views are always replicated to slaves. Views are filtered by their own name, not by the tables they refer to. This means that a view can be replicated to the slave even if the view contains a table that would normally be filtered out by `replication-ignore-table` rules. Care should therefore be taken to ensure that views do not replicate table data that would normally be filtered for security reasons.

Replication from a table to a same-named view is supported using statement-based logging, but not when using row-based logging. In MySQL 5.1.69 and later, trying to do so when row-based logging is in effect causes an error. (Bug #11752707, Bug #43975)

### 4.2 Replication Compatibility Between MySQL Versions

MySQL supports replication from one release series to the next higher release series. For example, you can replicate from a master running MySQL 5.0 to a slave running MySQL 5.1, from a master running MySQL 5.1 to a slave running MySQL 5.5, and so on.

However, one may encounter difficulties when replicating from an older master to a newer slave if the master uses statements or relies on behavior no longer supported in the version of MySQL used on the slave.

The use of more than two MySQL Server versions is not supported in replication setups involving multiple masters, regardless of the number of master or slave MySQL servers. This restriction applies not only to release series, but to version numbers within the same release series as well. For example, if you are using a chained or circular replication setup, you cannot use MySQL 5.1.31, MySQL 5.1.32, and MySQL 5.1.34 concurrently, although you could use any two of these releases together.

**Important**

It is strongly recommended to use the most recent release available within a given MySQL release series because replication (and other) capabilities are continually
Upgrading a Replication Setup

being improved. It is also recommended to upgrade masters and slaves that use early releases of a release series of MySQL to GA (production) releases when the latter become available for that release series.

Replication from newer masters to older slaves may be possible, but is generally not supported. This is due to a number of factors:

• **Binary log format changes.** The binary log format can change between major releases. While we attempt to maintain backward compatibility, this is not always possible. For example, the binary log format implemented in MySQL 5.0 changed considerably from that used in previous versions, especially with regard to handling of character sets, `LOAD DATA INFILE`, and time zones. This means that replication from a MySQL 5.0 (or later) master to a MySQL 4.1 (or earlier) slave is generally not supported.

  This also has significant implications for upgrading replication servers; see Section 4.3, “Upgrading a Replication Setup”, for more information.

• **Use of row-based replication.** Row-based replication was implemented in MySQL 5.1.5, so you cannot replicate using row-based replication from any MySQL 5.1 or later master to a slave older than MySQL 5.1.5.

  For more information about row-based replication, see Section 2.2, “Replication Formats”.

• **SQL incompatibilities.** You cannot replicate from a newer master to an older slave using statement-based replication if the statements to be replicated use SQL features available on the master but not on the slave.

  However, if both the master and the slave support row-based replication, and there are no data definition statements to be replicated that depend on SQL features found on the master but not on the slave, you can use row-based replication to replicate the effects of data modification statements even if the DDL run on the master is not supported on the slave.

  For more information on potential replication issues, see Section 4.1, “Replication Features and Issues”.

### 4.3 Upgrading a Replication Setup

When you upgrade servers that participate in a replication setup, the procedure for upgrading depends on the current server versions and the version to which you are upgrading.

This section applies to upgrading replication from older versions of MySQL to MySQL 5.1. A 4.0 server should be 4.0.3 or newer.

When you upgrade a master to 5.1 from an earlier MySQL release series, you should first ensure that all the slaves of this master are using the same 5.1.x release. If this is not the case, you should first upgrade the slaves. To upgrade each slave, shut it down, upgrade it to the appropriate 5.1.x version, restart it, and restart replication. The 5.1 slave is able to read the old relay logs written prior to the upgrade and to execute the statements they contain. Relay logs created by the slave after the upgrade are in 5.1 format.

After the slaves have been upgraded, shut down the master, upgrade it to the same 5.1.x release as the slaves, and restart it. The 5.1 master is able to read the old binary logs written prior to the upgrade and to send them to the 5.1 slaves. The slaves recognize the old format and handle it properly. Binary logs created by the master subsequent to the upgrade are in 5.1 format. These too are recognized by the 5.1 slaves.

In other words, when upgrading to MySQL 5.1, the slaves must be MySQL 5.1 before you can upgrade the master to 5.1. Note that downgrading from 5.1 to older versions does not work so simply: You must ensure
that any 5.1 binary log or relay log has been fully processed, so that you can remove it before proceeding with the downgrade.

Downgrading a replication setup to a previous version cannot be done once you have switched from statement-based to row-based replication, and after the first row-based statement has been written to the binlog. See Section 2.2, “Replication Formats”.

Some upgrades may require that you drop and re-create database objects when you move from one MySQL series to the next. For example, collation changes might require that table indexes be rebuilt. Such operations, if necessary, will be detailed at Changes Affecting Upgrades to 5.1. It is safest to perform these operations separately on the slaves and the master, and to disable replication of these operations from the master to the slave. To achieve this, use the following procedure:

1. Stop all the slaves and upgrade them. Restart them with the --skip-slave-start option so that they do not connect to the master. Perform any table repair or rebuilding operations needed to re-create database objects, such as use of REPAIR TABLE or ALTER TABLE, or dumping and reloading tables or triggers.

2. Disable the binary log on the master. To do this without restarting the master, execute a SET sql_log_bin = 0 statement. Alternatively, stop the master and restart it without the --log-bin option. If you restart the master, you might also want to disallow client connections. For example, if all clients connect using TCP/IP, use the --skip-networking option when you restart the master.

3. With the binary log disabled, perform any table repair or rebuilding operations needed to re-create database objects. The binary log must be disabled during this step to prevent these operations from being logged and sent to the slaves later.

4. Re-enable the binary log on the master. If you set sql_log_bin to 0 earlier, execute a SET sql_log_bin = 1 statement. If you restarted the master to disable the binary log, restart it with --log-bin, and without --skip-networking so that clients and slaves can connect.

5. Restart the slaves, this time without the --skip-slave-start option.

4.4 Troubleshooting Replication

If you have followed the instructions but your replication setup is not working, the first thing to do is check the error log for messages. Many users have lost time by not doing this soon enough after encountering problems.

If you cannot tell from the error log what the problem was, try the following techniques:

- Verify that the master has binary logging enabled by issuing a SHOW MASTER STATUS statement. If logging is enabled, Position is nonzero. If binary logging is not enabled, verify that you are running the master with the --log-bin option.

- Verify that the master and slave both were started with the --server-id option and that the ID value is unique on each server.

- Verify that the slave is running. Use SHOW SLAVE STATUS to check whether the Slave_IO_Running and Slave_SQL_Running values are both Yes. If not, verify the options that were used when starting the slave server. For example, --skip-slave-start prevents the slave threads from starting until you issue a START SLAVE statement.

- If the slave is running, check whether it established a connection to the master. Use SHOW PROCESSLIST, find the I/O and SQL threads and check their State column to see what they display. See Section 5.1, “Replication Implementation Details”. If the I/O thread state says Connecting to master, check the following:
• Verify the privileges for the user being used for replication on the master.

• Check that the host name of the master is correct and that you are using the correct port to connect to the master. The port used for replication is the same as used for client network communication (the default is 3306). For the host name, ensure that the name resolves to the correct IP address.

• Check that networking has not been disabled on the master or slave. Look for the `skip-networking` option in the configuration file. If present, comment it out or remove it.

• If the master has a firewall or IP filtering configuration, ensure that the network port being used for MySQL is not being filtered.

• Check that you can reach the master by using `ping` or `traceroute/tracert` to reach the host.

• If the slave was running previously but has stopped, the reason usually is that some statement that succeeded on the master failed on the slave. This should never happen if you have taken a proper snapshot of the master, and never modified the data on the slave outside of the slave thread. If the slave stops unexpectedly, it is a bug or you have encountered one of the known replication limitations described in Section 4.1, “Replication Features and Issues”. If it is a bug, see Section 4.5, “How to Report Replication Bugs or Problems”, for instructions on how to report it.

• If a statement that succeeded on the master refuses to run on the slave, try the following procedure if it is not feasible to do a full database resynchronization by deleting the slave’s databases and copying a new snapshot from the master:

  1. Determine whether the affected table on the slave is different from the master table. Try to understand how this happened. Then make the slave’s table identical to the master’s and run `START SLAVE`.

  2. If the preceding step does not work or does not apply, try to understand whether it would be safe to make the update manually (if needed) and then ignore the next statement from the master.

  3. If you decide that the slave can skip the next statement from the master, issue the following statements:

     ```sql
     mysql> SET GLOBAL sql_slave_skip_counter = N;
     mysql> START SLAVE;
     ```

     The value of \( N \) should be 1 if the next statement from the master does not use `AUTO_INCREMENT` or `LAST_INSERT_ID()`. Otherwise, the value should be 2. The reason for using a value of 2 for statements that use `AUTO_INCREMENT` or `LAST_INSERT_ID()` is that they take two events in the binary log of the master.

     See also `SET GLOBAL sql_slave_skip_counter Syntax`.

  4. If you are sure that the slave started out perfectly synchronized with the master, and that no one has updated the tables involved outside of the slave thread, then presumably the discrepancy is the result of a bug. If you are running the most recent version of MySQL, please report the problem. If you are running an older version, try upgrading to the latest production release to determine whether the problem persists.

4.5 How to Report Replication Bugs or Problems

When you have determined that there is no user error involved, and replication still either does not work at all or is unstable, it is time to send us a bug report. We need to obtain as much information as possible
from you to be able to track down the bug. Please spend some time and effort in preparing a good bug report.

If you have a repeatable test case that demonstrates the bug, please enter it into our bugs database using the instructions given in How to Report Bugs or Problems. If you have a “phantom” problem (one that you cannot duplicate at will), use the following procedure:

1. Verify that no user error is involved. For example, if you update the slave outside of the slave thread, the data goes out of synchrony, and you can have unique key violations on updates. In this case, the slave thread stops and waits for you to clean up the tables manually to bring them into synchrony. *This is not a replication problem. It is a problem of outside interference causing replication to fail.*

2. Run the slave with the `--log-slave-updates` and `--log-bin` options. These options cause the slave to log the updates that it receives from the master into its own binary logs.

3. Save all evidence before resetting the replication state. If we have no information or only sketchy information, it becomes difficult or impossible for us to track down the problem. The evidence you should collect is:
   
   - All binary log files from the master
   - All binary log files from the slave
   - The output of `SHOW MASTER STATUS` from the master at the time you discovered the problem
   - The output of `SHOW SLAVE STATUS` from the slave at the time you discovered the problem
   - Error logs from the master and the slave

4. Use `mysqlbinlog` to examine the binary logs. The following should be helpful to find the problem statement. `log_file` and `log_pos` are the `Master_Log_File` and `Read_Master_Log_Pos` values from `SHOW SLAVE STATUS`.

```
shell> mysqlbinlog --start-position=log_pos log_file | head
```

After you have collected the evidence for the problem, try to isolate it as a separate test case first. Then enter the problem with as much information as possible into our bugs database using the instructions at How to Report Bugs or Problems.
Chapter 5 Replication Implementation

Table of Contents

5.1 Replication Implementation Details ................................................................. 121
5.2 Replication Relay and Status Logs ................................................................. 123
  5.2.1 The Slave Relay Log ............................................................................. 123
  5.2.2 Slave Status Logs .............................................................................. 124
5.3 How Servers Evaluate Replication Filtering Rules ........................................... 126
  5.3.1 Evaluation of Database-Level Replication and Binary Logging Options ........ 127
  5.3.2 Evaluation of Table-Level Replication Options ..................................... 129
  5.3.3 Replication Rule Application ............................................................... 131

Replication is based on the master server keeping track of all changes to its databases (updates, deletes, and so on) in its binary log. The binary log serves as a written record of all events that modify database structure or content (data) from the moment the server was started. Typically, SELECT statements are not recorded because they modify neither database structure nor content.

Each slave that connects to the master requests a copy of the binary log. That is, it pulls the data from the master, rather than the master pushing the data to the slave. The slave also executes the events from the binary log that it receives. This has the effect of repeating the original changes just as they were made on the master. Tables are created or their structure modified, and data is inserted, deleted, and updated according to the changes that were originally made on the master.

Because each slave is independent, the replaying of the changes from the master's binary log occurs independently on each slave that is connected to the master. In addition, because each slave receives a copy of the binary log only by requesting it from the master, the slave is able to read and update the copy of the database at its own pace and can start and stop the replication process at will without affecting the ability to update to the latest database status on either the master or slave side.

For more information on the specifics of the replication implementation, see Section 5.1, “Replication Implementation Details”.

Masters and slaves report their status in respect of the replication process regularly so that you can monitor them. See Examining Thread Information, for descriptions of all replicated-related states.

The master binary log is written to a local relay log on the slave before it is processed. The slave also records information about the current position with the master's binary log and the local relay log. See Section 5.2, “Replication Relay and Status Logs”.

Database changes are filtered on the slave according to a set of rules that are applied according to the various configuration options and variables that control event evaluation. For details on how these rules are applied, see Section 5.3, “How Servers Evaluate Replication Filtering Rules”.

5.1 Replication Implementation Details

MySQL replication capabilities are implemented using three threads, one on the master server and two on the slave:

- **Binlog dump thread.** The master creates a thread to send the binary log contents to a slave when the slave connects. This thread can be identified in the output of SHOW PROCESSLIST on the master as the Binlog Dump thread.
Replication Implementation Details

The binary log dump thread acquires a lock on the master's binary log for reading each event that is to be sent to the slave. As soon as the event has been read, the lock is released, even before the event is sent to the slave.

- **Slave I/O thread.** When a `START SLAVE` statement is issued on a slave server, the slave creates an I/O thread, which connects to the master and asks it to send the updates recorded in its binary logs.

  The slave I/O thread reads the updates that the master's Binlog Dump thread sends (see previous item) and copies them to local files that comprise the slave's relay log.

  The state of this thread is shown as Slave_IO_running in the output of `SHOW SLAVE STATUS` or as Slave_running in the output of `SHOW STATUS`.

- **Slave SQL thread.** The slave creates an SQL thread to read the relay log that is written by the slave I/O thread and execute the events contained therein.

In the preceding description, there are three threads per master/slave connection. A master that has multiple slaves creates one binary log dump thread for each currently connected slave, and each slave has its own I/O and SQL threads.

A slave uses two threads to separate reading updates from the master and executing them into independent tasks. Thus, the task of reading statements is not slowed down if statement execution is slow. For example, if the slave server has not been running for a while, its I/O thread can quickly fetch all the binary log contents from the master when the slave starts, even if the SQL thread lags far behind. If the slave stops before the SQL thread has executed all the fetched statements, the I/O thread has at least fetched everything so that a safe copy of the statements is stored locally in the slave's relay logs, ready for execution the next time that the slave starts. This enables the master server to purge its binary logs sooner because it no longer needs to wait for the slave to fetch their contents.

The `SHOW PROCESSLIST` statement provides information that tells you what is happening on the master and on the slave regarding replication. For information on master states, see Replication Master Thread States. For slave states, see Replication Slave I/O Thread States, and Replication Slave SQL Thread States.

The following example illustrates how the three threads show up in the output from `SHOW PROCESSLIST`.

On the master server, the output from `SHOW PROCESSLIST` looks like this:

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row ***************************
    Id: 2
    User: root
    Host: localhost:32931
    db: NULL
  Command: Binlog Dump
    Time: 94
    State: Has sent all binlog to slave; waiting for binlog to be updated
    Info: NULL
```

Here, thread 2 is a Binlog Dump replication thread that services a connected slave. The State information indicates that all outstanding updates have been sent to the slave and that the master is waiting for more updates to occur. If you see no Binlog Dump threads on a master server, this means that replication is not running; that is, no slaves are currently connected.

On a slave server, the output from `SHOW PROCESSLIST` looks like this:
Replication Relay and Status Logs

The **State** information indicates that thread 10 is the I/O thread that is communicating with the master server, and thread 11 is the SQL thread that is processing the updates stored in the relay logs. At the time that `SHOW PROCESSLIST` was run, both threads were idle, waiting for further updates.

The value in the **Time** column can show how late the slave is compared to the master. See MySQL 5.1 FAQ: Replication. If sufficient time elapses on the master side without activity on the Binlog Dump thread, the master determines that the slave is no longer connected. As for any other client connection, the timeouts for this depend on the values of `net_write_timeout` and `net_retry_count`; for more information about these, see Server System Variables.

The **SHOW SLAVE STATUS** statement provides additional information about replication processing on a slave server. See Section 2.4.1, “Checking Replication Status”.

### 5.2 Replication Relay and Status Logs

During replication, a slave server creates several logs that hold the binary log events relayed from the master to the slave, and to record information about the current status and location within the relay log. There are three types of logs used in the process, listed here:

- **The **relay log** consists** of the events read from the binary log of the master and written by the slave I/O thread. Events in the relay log are executed on the slave as part of the SQL thread.

- **The **master info log** contains status and current configuration information for the slave's connection to the master. This log holds information on the master host name, login credentials, and coordinates indicating how far the slave has read from the master's binary log.

- **The **relay log info log** holds status information about the execution point within the slave's relay log.

### 5.2.1 The Slave Relay Log

The relay log, like the binary log, consists of a set of numbered files containing events that describe database changes, and an index file that contains the names of all used relay log files.

The term “relay log file” generally denotes an individual numbered file containing database events. The term “relay log” collectively denotes the set of numbered relay log files plus the index file.

Relay log files have the same format as binary log files and can be read using `mysqlbinlog` (see `mysqlbinlog — Utility for Processing Binary Log Files`).
By default, relay log file names have the form `host_name-relay-bin.nnnnnn` in the data directory, where `host_name` is the name of the slave server host and `nnnnnn` is a sequence number. Successive relay log files are created using successive sequence numbers, beginning with 000001. The slave uses an index file to track the relay log files currently in use. The default relay log index file name is `host_name-relay-bin.index` in the data directory.

The default relay log file and relay log index file names can be overridden with, respectively, the `--relay-log` and `--relay-log-index` server options (see Section 2.3, “Replication and Binary Logging Options and Variables”).

If a slave uses the default host-based relay log file names, changing a slave's host name after replication has been set up can cause replication to fail with the errors `Failed to open the relay log` and `Could not find target log during relay log initialization`. This is a known issue (see Bug #2122). If you anticipate that a slave's host name might change in the future (for example, if networking is set up on the slave such that its host name can be modified using DHCP), you can avoid this issue entirely by using the `--relay-log` and `--relay-log-index` options to specify relay log file names explicitly when you initially set up the slave. This will make the names independent of server host name changes.

If you encounter the issue after replication has already begun, one way to work around it is to stop the slave server, prepend the contents of the old relay log index file to the new one, and then restart the slave. On a Unix system, this can be done as shown here:

```
shell> cat new_relay_log_name.index >> old_relay_log_name.index
shell> mv old_relay_log_name.index new_relay_log_name.index
```

A slave server creates a new relay log file under the following conditions:

- Each time the I/O thread starts.
- When the logs are flushed; for example, with `FLUSH LOGS` or `mysqladmin flush-logs`.
- When the size of the current relay log file becomes “too large,” determined as follows:
  - If the value of `max_relay_log_size` is greater than 0, that is the maximum relay log file size.
  - If the value of `max_relay_log_size` is 0, `max_binlog_size` determines the maximum relay log file size.

The SQL thread automatically deletes each relay log file as soon as it has executed all events in the file and no longer needs it. There is no explicit mechanism for deleting relay logs because the SQL thread takes care of doing so. However, `FLUSH LOGS` rotates relay logs, which influences when the SQL thread deletes them.

### 5.2.2 Slave Status Logs

A replication slave server creates two logs. By default, these logs are files named `master.info` and `relay-log.info` and created in the data directory. The names and locations of these files can be changed by using the `--master-info-file` and `--relay-log-info-file` options, respectively. See Section 2.3, “Replication and Binary Logging Options and Variables”.

The two status logs contain information like that shown in the output of the `SHOW SLAVE STATUS` statement, which is discussed in SQL Statements for Controlling Slave Servers. Because the status logs are stored on disk, they survive a slave server’s shutdown. The next time the slave starts up, it reads the two logs to determine how far it has proceeded in reading binary logs from the master and in processing its own relay logs.
The master info log should be protected because it contains the password for connecting to the master. See Passwords and Logging.

The slave I/O thread updates the master info log. The following table shows the correspondence between the lines in the `master.info` file and the columns displayed by `SHOW SLAVE STATUS`.

<table>
<thead>
<tr>
<th>Line in master.info</th>
<th>SHOW SLAVE STATUS Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Number of lines in the file</td>
</tr>
<tr>
<td>2</td>
<td>Master_Log_File</td>
<td>The name of the master binary log currently being read from the master</td>
</tr>
<tr>
<td>3</td>
<td>Read_Master_Log_Pos</td>
<td>The current position within the master binary log that have been read from the master</td>
</tr>
<tr>
<td>4</td>
<td>Master_Host</td>
<td>The host name of the master</td>
</tr>
<tr>
<td>5</td>
<td>Master_User</td>
<td>The user name used to connect to the master</td>
</tr>
<tr>
<td>6</td>
<td>Password (not shown by SHOW SLAVE STATUS)</td>
<td>The password used to connect to the master</td>
</tr>
<tr>
<td>7</td>
<td>Master_Port</td>
<td>The network port used to connect to the master</td>
</tr>
<tr>
<td>8</td>
<td>Connect_Retry</td>
<td>The period (in seconds) that the slave will wait before trying to reconnect to the master</td>
</tr>
<tr>
<td>9</td>
<td>Master_SSL_Allowed</td>
<td>Indicates whether the server supports SSL connections</td>
</tr>
<tr>
<td>10</td>
<td>Master_SSL_CA_File</td>
<td>The file used for the Certificate Authority (CA) certificate</td>
</tr>
<tr>
<td>11</td>
<td>Master_SSL_CA_Path</td>
<td>The path to the Certificate Authority (CA) certificates</td>
</tr>
<tr>
<td>12</td>
<td>Master_SSL_Cert</td>
<td>The name of the SSL certificate file</td>
</tr>
<tr>
<td>13</td>
<td>Master_SSL_CIPHER</td>
<td>The list of possible ciphers used in the handshake for the SSL connection</td>
</tr>
<tr>
<td>14</td>
<td>Master_SSL_Key</td>
<td>The name of the SSL key file</td>
</tr>
<tr>
<td>15</td>
<td>Master_SSL_Verify_Server_Cert</td>
<td>Whether to verify the server certificate (added in MySQL 5.1.18)</td>
</tr>
</tbody>
</table>

The slave SQL thread updates the relay log info log. The following table shows the correspondence between the lines in the `relay-log.info` file and the columns displayed by `SHOW SLAVE STATUS`.

<table>
<thead>
<tr>
<th>Line in relay-log.info</th>
<th>SHOW SLAVE STATUS Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relay_Log_File</td>
<td>The name of the current relay log file</td>
</tr>
<tr>
<td>2</td>
<td>Relay_Log_Pos</td>
<td>The current position within the relay log file; events up to this position have been executed on the slave database</td>
</tr>
<tr>
<td>3</td>
<td>Relay_Master_Log_File</td>
<td>The name of the master binary log file from which the events in the relay log file were read</td>
</tr>
</tbody>
</table>
How Servers Evaluate Replication Filtering Rules

<table>
<thead>
<tr>
<th>Line in relay-log.info</th>
<th>SHOW SLAVE STATUS Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Exec_Master_Log_Pos</td>
<td>The equivalent position within the master’s binary log file of events that have already been executed</td>
</tr>
</tbody>
</table>

The contents of the relay-log.info file and the states shown by the SHOW SLAVE STATUS statement might not match if the relay-log.info file has not been flushed to disk. Ideally, you should only view relay-log.info on a slave that is offline (that is, mysqld is not running). For a running system, SHOW SLAVE STATUS should be used.

When you back up the slave’s data, you should back up these two status logs, along with the relay log files. The status logs are needed to resume replication after you restore the data from the slave. If you lose the relay logs but still have the relay log info log, you can check it to determine how far the SQL thread has executed in the master binary logs. Then you can use CHANGE MASTER TO with the MASTER_LOG_FILE and MASTER_LOG_POS options to tell the slave to re-read the binary logs from that point. Of course, this requires that the binary logs still exist on the master.

5.3 How Servers Evaluate Replication Filtering Rules

If a master server does not write a statement to its binary log, the statement is not replicated. If the server does log the statement, the statement is sent to all slaves and each slave determines whether to execute it or ignore it.

On the master, you can control which databases to log changes for by using the --binlog-do-db and --binlog-ignore-db options to control binary logging. For a description of the rules that servers use in evaluating these options, see Section 5.3.1, “Evaluation of Database-Level Replication and Binary Logging Options”. You should not use these options to control which databases and tables are replicated. Instead, use filtering on the slave to control the events that are executed on the slave.

On the slave side, decisions about whether to execute or ignore statements received from the master are made according to the --replicate-* options that the slave was started with. (See Section 2.3, “Replication and Binary Logging Options and Variables”.)

In the simplest case, when there are no --replicate-* options, the slave executes all statements that it receives from the master. Otherwise, the result depends on the particular options given.

Database-level options (--replicate-do-db, --replicate-ignore-db) are checked first; see Section 5.3.1, “Evaluation of Database-Level Replication and Binary Logging Options”, for a description of this process. If no database-level options are used, option checking proceeds to any table-level options that may be in use (see Section 5.3.2, “Evaluation of Table-Level Replication Options”, for a discussion of these). If one or more database-level options are used but none are matched, the statement is not replicated.

To make it easier to determine what effect an option set will have, it is recommended that you avoid mixing “do” and “ignore” options, or wildcard and nonwildcard options. An example of the latter that may have unintended effects is the use of --replicate-do-db and --replicate-wild-do-table together, where --replicate-wild-do-table uses a pattern for the database name that matches the name given for --replicate-do-db. Suppose a replication slave is started with --replicate-do-db=dbx --replicate-wild-do-table=db%.t1. Then, suppose that on the master, you issue the statement CREATE DATABASE dbx. Although you might expect it, this statement is not replicated because it does not reference a table named t1.

If any --replicate-rewrite-db options were specified, they are applied before the --replicate-* filtering rules are tested.
Evaluation of Database-Level Replication and Binary Logging Options

Note

Database-level filtering options are case-sensitive on platforms supporting case sensitivity in filenames, whereas table-level filtering options are not (regardless of platform). This is true regardless of the value of the `lower_case_table_names` system variable. These are known issues which are fixed in MySQL 5.5.

5.3.1 Evaluation of Database-Level Replication and Binary Logging Options

When evaluating replication options, the slave begins by checking to see whether there are any `--replicate-do-db` or `--replicate-ignore-db` options that apply. When using `--binlog-do-db` or `--binlog-ignore-db`, the process is similar, but the options are checked on the master.

With statement-based replication, the default database is checked for a match. With row-based replication, the database where data is to be changed is the database that is checked. Regardless of the binary logging format, checking of database-level options proceeds as shown in the following diagram.

![Diagram](image)

The steps involved are listed here:

1. Are there any `--replicate-do-db` options?
Evaluation of Database-Level Replication and Binary Logging Options

- **Yes.** Do any of them match the database?
- **Yes.** Execute the statement and exit.
- **No.** Ignore the statement and exit.
- **No.** Continue to step 2.

2. Are there any `--replicate-ignore-db` options?
   - **Yes.** Do any of them match the database?
     - **Yes.** Ignore the statement and exit.
     - **No.** Continue to step 3.
   - **No.** Continue to step 3.

3. Proceed to checking the table-level replication options, if there are any. For a description of how these options are checked, see Section 5.3.2, “Evaluation of Table-Level Replication Options”.

   **Important**
   
   A statement that is still permitted at this stage is not yet actually executed. The statement is not executed until all table-level options (if any) have also been checked, and the outcome of that process permits execution of the statement.

For binary logging, the steps involved are listed here:

1. Are there any `--binlog-do-db` or `--binlog-ignore-db` options?
   - **Yes.** Continue to step 2.
   - **No.** Log the statement and exit.

2. Is there a default database (has any database been selected by `USE`)?
   - **Yes.** Continue to step 3.
   - **No.** Ignore the statement and exit.

3. There is a default database. Are there any `--binlog-do-db` options?
   - **Yes.** Do any of them match the database?
     - **Yes.** Log the statement and exit.
     - **No.** Ignore the statement and exit.
   - **No.** Continue to step 4.

4. Do any of the `--binlog-ignore-db` options match the database?
   - **Yes.** Ignore the statement and exit.
   - **No.** Log the statement and exit.
Important

For statement-based logging, an exception is made in the rules just given for the CREATE DATABASE, ALTER DATABASE, and DROP DATABASE statements. In those cases, the database being created, altered, or dropped replaces the default database when determining whether to log or ignore updates.

--binlog-do-db can sometimes mean “ignore other databases”. For example, when using statement-based logging, a server running with only --binlog-do-db=sales does not write to the binary log statements for which the default database differs from sales. When using row-based logging with the same option, the server logs only those updates that change data in sales.

5.3.2 Evaluation of Table-Level Replication Options

The slave checks for and evaluates table options only if either of the following two conditions is true:

- No matching database options were found.
- One or more database options were found, and were evaluated to arrive at an “execute” condition according to the rules described in the previous section (see Section 5.3.1, “Evaluation of Database-Level Replication and Binary Logging Options”).

First, as a preliminary condition, the slave checks whether statement-based replication is enabled. If so, and the statement occurs within a stored function, the slave executes the statement and exits. If row-based replication is enabled, the slave does not know whether a statement occurred within a stored function on the master, so this condition does not apply.

Note

For statement-based replication, replication events represent statements (all changes making up a given event are associated with a single SQL statement); for row-based replication, each event represents a change in a single table row (thus a single statement such as UPDATE mytable SET mycol = 1 may yield many row-based events). When viewed in terms of events, the process of checking table options is the same for both row-based and statement-based replication.

Having reached this point, if there are no table options, the slave simply executes all events. If there are any --replicate-do-table or --replicate-wild-do-table options, the event must match one of these if it is to be executed; otherwise, it is ignored. If there are any --replicate-ignore-table or --replicate-wild-ignore-table options, all events are executed except those that match any of these options. This process is illustrated in the following diagram.
The following steps describe this evaluation in more detail:
1. Are there any table options?
   • Yes. Continue to step 2.
   • No. Execute the event and exit.

2. Are there any \texttt{--replicate-do-table} options?
   • Yes. Does the table match any of them?
     • Yes. Execute the event and exit.
     • No. Continue to step 3.
   • No. Continue to step 3.

3. Are there any \texttt{--replicate-ignore-table} options?
   • Yes. Does the table match any of them?
     • Yes. Ignore the event and exit.
     • No. Continue to step 4.
   • No. Continue to step 4.

4. Are there any \texttt{--replicate-wild-do-table} options?
   • Yes. Does the table match any of them?
     • Yes. Execute the event and exit.
     • No. Continue to step 5.
   • No. Continue to step 5.

5. Are there any \texttt{--replicate-wild-ignore-table} options?
   • Yes. Does the table match any of them?
     • Yes. Ignore the event and exit.
     • No. Continue to step 6.
   • No. Continue to step 6.

6. Are there any \texttt{--replicate-do-table} or \texttt{--replicate-wild-do-table} options?
   • Yes. Ignore the event and exit.
   • No. Execute the event and exit.

5.3.3 Replication Rule Application

This section provides additional explanation and examples of usage for different combinations of replication filtering options.

Some typical combinations of replication filter rule types are given in the following table:
### Replication Rule Application

<table>
<thead>
<tr>
<th>Condition (Types of Options)</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td><em><em>No --replicate-</em> options at all:</em>*</td>
<td>The slave executes all events that it receives from the master.</td>
</tr>
<tr>
<td><strong>--replicate-*-db options, but no table options:</strong></td>
<td>The slave accepts or ignores events using the database options. It executes all events permitted by those options because there are no table restrictions.</td>
</tr>
<tr>
<td><strong>--replicate-*-table options, but no database options:</strong></td>
<td>All events are accepted at the database-checking stage because there are no database conditions. The slave executes or ignores events based solely on the table options.</td>
</tr>
<tr>
<td><strong>A combination of database and table options:</strong></td>
<td>The slave accepts or ignores events using the database options. Then it evaluates all events permitted by those options according to the table options. This can sometimes lead to results that seem counterintuitive, and that may be different depending on whether you are using statement-based or row-based replication; see the text for an example.</td>
</tr>
</tbody>
</table>

A more complex example follows, in which we examine the outcomes for both statement-based and row-based settings.

Suppose that we have two tables `mytbl1` in database `db1` and `mytbl2` in database `db2` on the master, and the slave is running with the following options (and no other replication filtering options):

```sql
replicate-ignore-db = db1
replicate-do-table  = db2.tbl2
```

Now we execute the following statements on the master:

```sql
USE db1;
INSERT INTO db2.tbl2 VALUES (1);
```

The results on the slave vary considerably depending on the binary log format, and may not match initial expectations in either case.

**Statement-based replication.** The `USE` statement causes `db1` to be the default database. Thus the `--replicate-ignore-db` option matches, and the `INSERT` statement is ignored. The table options are not checked.

**Row-based replication.** The default database has no effect on how the slave reads database options when using row-based replication. Thus, the `USE` statement makes no difference in how the `--replicate-ignore-db` option is handled: the database specified by this option does not match the database where the `INSERT` statement changes data, so the slave proceeds to check the table options. The table specified by `--replicate-do-table` matches the table to be updated, and the row is inserted.
Appendix A Licenses for Third-Party Components

Table of Contents

A.1 ANTLR 3 License ........................................................................................................... 135
A.2 dtoa.c License ............................................................................................................. 135
A.3 Editline Library (libedit) License ................................................................................ 136
A.4 FindGTest.cmake License ........................................................................................... 138
A.5 Fred Fish’s Dbug Library License .................................................................................. 139
A.6 getarg License .............................................................................................................. 140
A.7 GNU General Public License Version 2.0, June 1991 .................................................... 140
A.8 GNU Lesser General Public License Version 2.1, February 1999 ................................. 146
A.9 GNU Libtool License ..................................................................................................... 154
A.10 GNU Readline License ............................................................................................... 154
A.11 Google Controlling Master Thread I/O Rate Patch License ........................................ 155
A.12 Google Perf tools (TCMalloc utility) License .............................................................. 155
A.13 Google SMP Patch License ........................................................................................ 156
A.14 lib_sql.cc License ....................................................................................................... 157
A.15 libevent License ......................................................................................................... 157
A.16 Linux-PAM License ..................................................................................................... 159
A.17 md5 (Message-Digest Algorithm 5) License ................................................................. 160
A.18 memcached License ..................................................................................................... 160
A.19 nt_servc (Windows NT Service class library) License ................................................ 161
A.20 OpenPAM License ....................................................................................................... 161
A.21 Paramiko License ....................................................................................................... 161
A.22 Percona Multiple I/O Threads Patch License ............................................................... 162
A.23 RegEX-Spencer Library License ................................................................................... 162
A.24 RFC 3174 - US Secure Hash Algorithm 1 (SHA1) License ....................................... 163
A.25 Richard A. O'Keefe String Library License ............................................................... 163
A.26 SHA-1 in C License ..................................................................................................... 164
A.27 zlib License ................................................................................................................. 164

The following is a list of the libraries we have included with the MySQL Server source and components used to test MySQL. We are thankful to all individuals that have created these. Some of the components require that their licensing terms be included in the documentation of products that include them. Cross references to these licensing terms are given with the applicable items in the list.

- Bjorn Benson

  For his safe_malloc (memory checker) package which is used in when you build MySQL using one of the BUILD/compile-*debug scripts or by manually setting the -DSAFEMALLOC flag.

- GroupLens Research Project

  The MySQL Quality Assurance team would like to acknowledge the use of the MovieLens Data Sets (10 million ratings and 100,000 tags for 10681 movies by 71567 users) to help test MySQL products and to thank the GroupLens Research Project at the University of Minnesota for making the data sets available.

MySQL 5.1

- Section A.2, “dtoa.c License”
- Section A.3, “Editline Library (libedit) License”
MySQL Cluster 7.1

- Section A.4, “FindGTest.cmake License”
- Section A.5, “Fred Fish’s Dbug Library License”
- Section A.6, “getarg License”
- Section A.7, “GNU General Public License Version 2.0, June 1991”
- Section A.9, “GNU Libtool License”
- Section A.10, “GNU Readline License”
- Section A.11, “Google Controlling Master Thread I/O Rate Patch License”
- Section A.13, “Google SMP Patch License”
- Section A.14, “lib_sql.cc License”
- Section A.17, “md5 (Message-Digest Algorithm 5) License”
- Section A.19, “nt_servc (Windows NT Service class library) License”
- Section A.22, “Percona Multiple I/O Threads Patch License”
- Section A.23, “RegEX-Spencer Library License”
- Section A.24, “RFC 3174 - US Secure Hash Algorithm 1 (SHA1) License”
- Section A.25, “Richard A. O’Keefe String Library License”
- Section A.27, “zlib License”

MySQL Cluster 7.1

- Section A.1, “ANTLR 3 License”
- Section A.2, “dtoa.c License”
- Section A.3, “Editline Library (libedit) License”
- Section A.4, “FindGTest.cmake License”
- Section A.5, “Fred Fish’s Dbug Library License”
- Section A.6, “getarg License”
- Section A.7, “GNU General Public License Version 2.0, June 1991”
- Section A.8, “GNU Lesser General Public License Version 2.1, February 1999”
- Section A.9, “GNU Libtool License”
- Section A.10, “GNU Readline License”
- Section A.11, “Google Controlling Master Thread I/O Rate Patch License”
- Section A.12, “Google Perftools (TCMalloc utility) License”
- Section A.13, “Google SMP Patch License”
- Section A.14, “lib_sql.cc License”
- Section A.15, “libevent License”
• Section A.16, “Linux-PAM License”
• Section A.17, "md5 (Message-Digest Algorithm 5) License"
• Section A.18, “memcached License”
• Section A.19, “nt_servc (Windows NT Service class library) License”
• Section A.20, “OpenPAM License”
• Section A.21, “Paramiko License”
• Section A.22, “Percona Multiple I/O Threads Patch License”
• Section A.23, “RegEX-Spencer Library License”
• Section A.25, “Richard A. O'Keefe String Library License”
• Section A.26, “SHA-1 in C License”
• Section A.27, “zlib License”

A.1 ANTLR 3 License

The following software may be included in this product:

ANTLR 3

ANTLR 3 License
[The BSD License]
Copyright (c) 2003-2007, Terence Parr
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in
  the documentation and/or other materials provided with the
  distribution.
* Neither the name of the author nor the names of its contributors
  may be used to endorse or promote products derived from this
  software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

A.2(dtoa.c License

The following software may be included in this product:
dtoa.c

The author of this software is David M. Gay.

Copyright (c) 1991, 2000, 2001 by Lucent Technologies.

Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.

THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

A.3 Editline Library (libedit) License

The following software may be included in this product:

Editline Library (libedit)

Some files are:

Copyright (c) 1992, 1993
The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Christos Zoulas of Cornell University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Some files are:
Copyright (c) 2001 The NetBSD Foundation, Inc.
All rights reserved.

This code is derived from software contributed to The NetBSD Foundation
by Anthony Mallet.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the
   above copyright notice, this list of conditions
   and the following disclaimer.
2. Redistributions in binary form must reproduce the
   above copyright notice, this list of conditions and the
   following disclaimer in the documentation and/or
   other materials provided with the distribution.

This software is provided by the NetBSD Foundation, Inc.
and contributors "AS IS" and any express or implied
warranties, including, but not limited to, the implied
warranties of merchantability and fitness for a
particular purpose are disclaimed. In no event shall
the Foundation or contributors be liable for any
direct, indirect, incidental, special, exemplary, or
consequential damages (including, but not limited to,
procurement of substitute goods or services; loss of
use, data, or profits; or business interruption) however
caused and on any theory of liability, whether in
contract, strict liability, or tort (including negligence
or otherwise) arising in any way out of the use of this
software, even if advised of the possibility of such
damage.

Some files are:

Copyright (c) 1997 The NetBSD Foundation, Inc.
All rights reserved.

This code is derived from software contributed to The NetBSD Foundation
by Jaromir Dolecek.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the
   above copyright notice, this list of conditions
   and the following disclaimer.
2. Redistributions in binary form must reproduce
   the above copyright notice, this list of conditions
   and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

This software is provided by the NetBSD Foundation, Inc.
and contributors "AS IS" and any express or implied
warranties, including, but not limited to, the implied
warranties of merchantability and fitness for a
particular purpose are disclaimed. In no event shall
the Foundation or contributors be liable for any
direct, indirect, incidental, special, exemplary, or
consequential damages (including, but not limited to,
procurement of substitute goods or services; loss of
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Some files are:

Copyright (c) 1998 Todd C. Miller <Todd.Miller@courtesan.com>

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND TODD C. MILLER DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL TODD C. MILLER BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

A.4 FindGTest.cmake License

The following software may be included in this product:

FindGTest.cmake helper script (part of CMake)

Copyright 2009 Kitware, Inc.
Copyright 2009 Philip Lowman
Copyright 2009 Daniel Blezek

Distributed under the OSI-approved BSD License (the "License"); see accompanying file Copyright.txt for details.

This software is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the License for more information.

(To distributed this file outside of CMake, substitute the full License text for the above reference.)

Thanks to Daniel Blezek for the GTEST_ADD_TESTS code

Text of Copyright.txt mentioned above:

CMake - Cross Platform Makefile Generator
Copyright 2000-2009 Kitware, Inc., Insight Software Consortium
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

* Neither the names of Kitware, Inc., the Insight Software Consortium,
  nor the names of their contributors may be used to endorse or promote
  products derived from this software without specific prior written
  permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.5 Fred Fish's Dbug Library License

The following software may be included in this product:

Fred Fish's Dbug Library

NOTICE

Copyright Abandoned, 1987, Fred Fish

This previously copyrighted work has been placed into the public
domain by the author and may be freely used for any purpose,
private or commercial.

Because of the number of inquiries I was receiving about the use
of this product in commercially developed works I have decided to
simply make it public domain to further its unrestricted use. I
specifically would be most happy to see this material become a
part of the standard Unix distributions by AT&T and the Berkeley
Computer Science Research Group, and a standard part of the GNU
system from the Free Software Foundation.

I would appreciate it, as a courtesy, if this notice is left in
all copies and derivative works. Thank you.

The author makes no warranty of any kind with respect to this
A.6 getarg License

The following software may be included in this product:

getarg Function (getarg.h, getarg.c files)

Copyright (c) 1997 – 2000 Kungliga Tekniska Högskolan
(Royal Institute of Technology, Stockholm, Sweden).
All rights reserved.

Redistribution and use in source and binary forms, with
or without modification, are permitted provided that the
following conditions are met:

1. Redistributions of source code must retain the above
copyright notice, this list of conditions and the
following disclaimer.
2. Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following
disclaimer in the documentation and/or other materials
provided with the distribution.
3. Neither the name of the Institute nor the names of its
contributors may be used to endorse or promote products
derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.

A.7 GNU General Public License Version 2.0, June 1991

The following applies to all products licensed under the GNU General
Public License, Version 2.0: You may not use the identified files
except in compliance with the GNU General Public License, Version
2.0 (the "License.") You may obtain a copy of the License at
http://www.gnu.org/licenses/gpl-2.0.txt. A copy of the license is
also reproduced below. Unless required by applicable law or agreed
to in writing, software distributed under the License is distributed
on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied. See the License for the specific language
governing permissions and limitations under the License.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim
copies of this license document, but changing it is not
allowed.

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it. (Some other Free Software Foundation software is covered by
the GNU Lesser General Public License instead.) You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not
price. Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have. You must make sure that they, too, receive or can get the
source code. And you must show them these terms so they know their
rights.

We protect your rights with two steps: (1) copyright the software,
and (2) offer you this license which gives you legal permission to
copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software. If the software is modified by someone else and passed on,
we want its recipients to know that what they have is not the original,
so that any problems introduced by others will not reflect on the
original authors' reputations.

Finally, any free program is threatened constantly by software
patents. We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary. To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and
modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License. The "Program", below,
refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification"). Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all. For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded. In such case, this License incorporates
the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new
versions of the General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation. If the Program does not specify a
version number of this License, you may choose any version ever
published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the
author to ask for permission. For software which is copyrighted by the
Free Software Foundation, write to the Free Software Foundation; we
sometimes make exceptions for this. Our decision will be guided by the
two goals of preserving the free status of all derivatives of our free
software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS
WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it
does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version
2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.

The hypothetical commands 'show w' and 'show c' should show the
appropriate parts of the General Public License. Of course, the
commands you use may be called something other than 'show w' and
'show c'; they could even be mouse-clicks or menu items--whatever
suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

A.8 GNU Lesser General Public License Version 2.1, February 1999

The following applies to all products licensed under the GNU Lesser General Public License, Version 2.1: You may not use the identified files except in compliance with the GNU Lesser General Public License, Version 2.1 (the "License"). You may obtain a copy of the License at http://www.gnu.org/licenses/lgpl-2.1.html. A copy of the license is also reproduced below. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.
Although the Lesser General Public License is less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) The modified work must itself be a software library.

   b) You must cause the files modified to carry prominent notices
stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no
charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a
table of data to be supplied by an application program that uses
the facility, other than as an argument passed when the facility
is invoked, then you must make a good faith effort to ensure that,
in the event an application does not supply such function or
table, the facility still operates, and performs whatever part of
its purpose remains meaningful.

(For example, a function in a library to compute square roots has
a purpose that is entirely well-defined independent of the
application. Therefore, Subsection 2d requires that any
application-supplied function or table used by this function must
be optional: if the application does not supply it, the square
root function must still compute square roots.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
 distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public
License instead of this License to a given copy of the Library. To do
this, you must alter all the notices that refer to this License, so
that they refer to the ordinary GNU General Public License, version 2,
instead of to this License. (If a newer version than version 2 of the
ordinary GNU General Public License has appeared, then you can specify
that version instead if you wish.) Do not make any other change in
these notices.

Once this change is made in a given copy, it is irreversible for
that copy, so the ordinary GNU General Public License applies to all
subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of
the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or
derivative of it, under Section 2) in object code or executable form
under the terms of Sections 1 and 2 above provided that you accompany
it with the complete corresponding machine-readable source code, which
must be distributed under the terms of Sections 1 and 2 above on a
medium customarily used for software interchange.

If distribution of object code is made by offering access to copy
from a designated place, then offering equivalent access to copy the
source code from the same place satisfies the requirement to
5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is
interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY
15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
That's all there is to it!

A.9 GNU Libtool License

The following software may be included in this product:

GNU Libtool (The GNU Portable Library Tool)

If you are receiving a copy of the Oracle software in source code, you are also receiving a copy of two files (ltmain.sh and ltd1.h) generated by the GNU Libtool in source code. If you received the Oracle software under a license other than a commercial (non-GPL) license, then the terms of the Oracle license do NOT apply to these files from GNU Libtool; they are licensed under the following licenses, separately from the Oracle programs you receive.

Oracle elects to use GNU General Public License version 2 (GPL) for any software where a choice of GPL or GNU Lesser/Library General Public License (LGPL) license versions are made available with the language indicating that GPL/LGPL or any later version may be used, or where a choice of which version of the GPL/LGPL is applied is unspecified.

From GNU Libtool:

ltmain.sh - Provide generalized library-building support services.
NOTE: Changing this file will not affect anything until you rerun configure.
Originally by Gordon Matzigkeit, 1996

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

As a special exception to the GNU General Public License, if you distribute this file as part of a program that contains a configuration script generated by Autoconf, you may include it under the same distribution terms that you use for the rest of that program.

This component is licensed under Section A.7, “GNU General Public License Version 2.0, June 1991”

A.10 GNU Readline License

The following software may be included in this product:
GNU Readline Library

With respect to MySQL Server/Cluster software licensed under GNU General Public License, you are receiving a copy of the GNU Readline Library in source code. The terms of any Oracle license that might accompany the Oracle programs do NOT apply to the GNU Readline Library; it is licensed under the following license, separately from the Oracle programs you receive. Oracle elects to use GNU General Public License version 2 (GPL) for any software where a choice of GPL license versions are made available with the language indicating that GPLv2 or any later version may be used, or where a choice of which version of the GPL is applied is unspecified.

This component is licensed under Section A.7, "GNU General Public License Version 2.0, June 1991"

A.11 Google Controlling Master Thread I/O Rate Patch License

The following software may be included in this product:

Google Controlling master thread I/O rate patch

Copyright (c) 2009, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of the Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.12 Google Perftools (TCMalloc utility) License

The following software may be included in this product:

Google Perftools (TCMalloc utility)

Copyright (c) 1998-2006, Google Inc.
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.13 Google SMP Patch License

The following software may be included in this product:

Google SMP Patch

Google SMP patch
Copyright (c) 2008, Google Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of the Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
A.14 lib_sql.cc License

The following software may be included in this product:

**lib_sql.cc**

Copyright (c) 2000
SWsoft company

This material is provided "as is", with absolutely no warranty expressed or implied. Any use is at your own risk.

Permission to use or copy this software for any purpose is hereby granted without fee, provided the above notices are retained on all copies. Permission to modify the code and to distribute modified code is granted, provided the above notices are retained, and a notice that the code was modified is included with the above copyright notice.

This code was modified by the MySQL team.

A.15 libevent License

The following software may be included in this product:

**libevent**

Copyright (c) 2000-2007 Niels Provos <provos@citi.umich.edu>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

==
Parts developed by Adam Langley
==

==
log.c
Based on err.c, which was adapted from OpenBSD libc *err*warncode.

Copyright (c) 2005 Nick Mathewson
Copyright (c) 2000 Dug Song
Copyright (c) 1993 The Regents of the University of California.
libevent License

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==

min_heap.h

Copyright (c) 2006 Maxim Yegorushkin
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

==

==

win32.c

Copyright 2000-2002 Niels Provos
Copyright 2003 Michael A. Davis
All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

A.16 Linux-PAM License

The following software may be included in this product:

Linux-PAM (pam-devel, Pluggable authentication modules for Linux)

Copyright Theodore Ts'o, 1996. All rights reserved.

(For the avoidance of doubt, Oracle uses and distributes this component under the terms below and elects not to do so under the GPL even though the GPL is referenced as an option below.)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, and the entire permission notice in its entirety, including the disclaimer of warranties.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

ALTERNATIVELY, this product may be distributed under the terms of the GNU Public License, in which case the provisions of the GPL are required INSTEAD OF the above restrictions. (This clause is necessary due to a potential bad interaction between the GPL and the restrictions contained in a BSD-style copyright.)

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
A.17 md5 (Message-Digest Algorithm 5) License

The following software may be included in this product:

md5 (Message-Digest Algorithm 5)

This code implements the MD5 message-digest algorithm. The algorithm is due to Ron Rivest. This code was written by Colin Plumb in 1993, no copyright is claimed. This code is in the public domain; do with it what you wish.

Equivalent code is available from RSA Data Security, Inc. This code has been tested against that, and is equivalent, except that you don't need to include two pages of legalese with every copy.

The code has been modified by Mikael Ronstroem to handle calculating a hash value of a key that is always a multiple of 4 bytes long. Word 0 of the calculated 4-word hash value is returned as the hash value.

A.18 memcached License

The following software may be included in this product:

memcached

Copyright (c) 2003, Danga Interactive, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the Danga Interactive nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
A.19 nt_servc (Windows NT Service class library) License

The following software may be included in this product:

nt_servc (Windows NT Service class library)

Windows NT Service class library
Copyright Abandoned 1998 Irena Pancirov - Irnet Snc
This file is public domain and comes with NO WARRANTY of any kind

A.20 OpenPAM License

The following software may be included in this product:

OpenPAM

Copyright (c) 2002-2003 Networks Associates Technology, Inc.
Copyright (c) 2004-2007 Dag-Erling Sørgrav
All rights reserved.

This software was developed for the FreeBSD Project by
ThinkSec AS and Network Associates Laboratories, the
Security Research Division of Network Associates, Inc.
under DARPA/SPAWAR contract N66001-01-C-8035 ("CBoss"),
as part of the DARPA CHATS research program.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions are met:

1. Redistributions of source code must retain the above
   copyright notice, this list of conditions and the
   following disclaimer.
2. Redistributions in binary form must reproduce the
   above copyright notice, this list of conditions and
   the following disclaimer in the documentation and/or
   other materials provided with the distribution.
3. The name of the author may not be used to endorse or
   promote products derived from this software without
   specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN
NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.21 Paramiko License

The following software may be included in this product:
This component is licensed under Section A.8, “GNU Lesser General Public License Version 2.1, February 1999”.

A.22 Percona Multiple I/O Threads Patch License

The following software may be included in this product:

Percona Multiple I/O threads patch

Copyright (c) 2008, 2009 Percona Inc
All rights reserved.

Redistribution and use of this software in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Percona Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission of Percona Inc.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.23 RegEX-Spencer Library License

The following software may be included in this product: Henry Spencer's Regular-Expression Library (RegEX-Spencer)

Copyright 1992, 1993, 1994 Henry Spencer. All rights reserved.
This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

**A.24 RFC 3174 - US Secure Hash Algorithm 1 (SHA1) License**

The following software may be included in this product:

RFC 3174 - US Secure Hash Algorithm 1 (SHA1)

---

The Richard O'Keefe String Library is subject to the following notice:

These files are in the public domain. This includes getopt.c, which is the work of Henry Spencer, University of Toronto Zoology, who says of it "None of this software is derived from Bell software. I had no access to the source for Bell's versions at the time I wrote it. This software is hereby explicitly placed in the public domain."
It may be used for any purpose on any machine by anyone." I would greatly prefer it if "my" material received no military use.

The t ctype.h file is subject to the following notice:

Copyright (C) 1998, 1999 by Pruet Boonma, all rights reserved.
Copyright (C) 1998 by Theppitak Karoonboonyanan, all rights reserved.

Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies.

Smaphan Raruenrom and Pruet Boonma makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

A.26 SHA-1 in C License

The following software may be included in this product:

SHA-1 in C

SHA-1 in C
By Steve Reid <steve@edmweb.com>
100% Public Domain

A.27 zlib License

The following software may be included in this product:

zlib

Oracle gratefully acknowledges the contributions of Jean-loup Gailly and Mark Adler in creating the zlib general purpose compression library which is used in this product.

zlib.h -- interface of the 'zlib' general purpose compression library
Copyright (C) 1995-2004 Jean-loup Gailly and Mark Adler

zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.3, July 18th, 2005
Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.5, April 19th, 2010
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly jloup@gzip.org
Mark Adler madler@alumni.caltech.edu