

Building MySQL from Source

Abstract

This is the Building MySQL from Source extract from the MySQL 5.7 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.

Document generated on: 2018-06-22 (revision: 57811)

Table of Contents

Preface and Legal Notices	v
1 Installing MySQL from Source	1
2 Installing MySQL Using a Standard Source Distribution	3
3 Installing MySQL Using a Development Source Tree	9
4 MySQL Source-Configuration Options	11
5 Dealing with Problems Compiling MySQL	33

Preface and Legal Notices

This is the Building MySQL from Source extract from the MySQL 5.7 Reference Manual.

Licensing information—MySQL 5.7. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.7, see the [MySQL 5.7 Commercial Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL 5.7, see the [MySQL 5.7 Community Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 7.5. This product may include third-party software, used under license. If you are using a *Commercial* release of NDB Cluster 7.5, see the [MySQL NDB Cluster 7.5 Commercial Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of NDB Cluster 7.5, see the [MySQL NDB Cluster 7.5 Community Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 7.6. If you are using a Developer Preview release of NDB Cluster 7.6, see the [MySQL NDB Cluster 7.6 Community Release License Information User Manual](#) for licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 1997, 2018, 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.

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

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

Documentation Accessibility

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

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Chapter 1 Installing MySQL from Source

Building MySQL from the source code enables you to customize build parameters, compiler optimizations, and installation location. For a list of systems on which MySQL is known to run, see <http://www.mysql.com/support/supportedplatforms/database.html>.

Before you proceed with an installation from source, check whether Oracle produces a precompiled binary distribution for your platform and whether it works for you. We put a great deal of effort into ensuring that our binaries are built with the best possible options for optimal performance. Instructions for installing binary distributions are available in [Installing MySQL on Unix/Linux Using Generic Binaries](#).

Warning

Building MySQL with nonstandard options may lead to reduced functionality, performance, or security.

Source Installation Methods

There are two methods for installing MySQL from source:

- Use a standard MySQL source distribution. To obtain a standard distribution, see [How to Get MySQL](#). For instructions on building from a standard distribution, see [Chapter 2, Installing MySQL Using a Standard Source Distribution](#).

Standard distributions are available as compressed `tar` files, Zip archives, or RPM packages. Distribution files have names of the form `mysql-VERSION.tar.gz`, `mysql-VERSION.zip`, or `mysql-VERSION.rpm`, where `VERSION` is a number like `5.7.24`. File names for source distributions can be distinguished from those for precompiled binary distributions in that source distribution names are generic and include no platform name, whereas binary distribution names include a platform name indicating the type of system for which the distribution is intended (for example, `pc-linux-i686` or `winx64`).

- Use a MySQL development tree. For information on building from one of the development trees, see [Chapter 3, Installing MySQL Using a Development Source Tree](#).

Source Installation System Requirements

Installation of MySQL from source requires several development tools. Some of these tools are needed no matter whether you use a standard source distribution or a development source tree. Other tool requirements depend on which installation method you use.

To install MySQL from source, the following system requirements must be satisfied, regardless of installation method:

- `CMake`, which is used as the build framework on all platforms. `CMake` can be downloaded from <http://www.cmake.org>.
- A good `make` program. Although some platforms come with their own `make` implementations, it is highly recommended that you use GNU `make` 3.75 or higher. It may already be available on your system as `gmake`. GNU `make` is available from <http://www.gnu.org/software/make/>.
- A working ANSI C++ compiler. See the description of the `FORCE_UNSUPPORTED_COMPILER` option for some guidelines.
- The Boost C++ libraries are required to build MySQL (but not to use it). Boost 1.59.0 must be installed. To obtain Boost and its installation instructions, visit [the official site](#). After Boost is installed, tell the build system where the Boost files are located by defining the `WITH_BOOST` option when you invoke `CMake`. For example:

```
shell> cmake . -DWITH_BOOST=/usr/local/boost_1_59_0
```

Adjust the path as necessary to match your installation.

- The [ncurses](#) library.
- Sufficient free memory. If you encounter problems such as “internal compiler error” when compiling large source files, it may be that you have too little memory. If compiling on a virtual machine, try increasing the memory allocation.
- Perl is needed if you intend to run test scripts. Most Unix-like systems include Perl. On Windows, you can use a version such as ActiveState Perl.

To install MySQL from a standard source distribution, one of the following tools is required to unpack the distribution file:

- For a `.tar.gz` compressed `tar` file: GNU [gunzip](#) to uncompress the distribution and a reasonable `tar` to unpack it. If your `tar` program supports the `z` option, it can both uncompress and unpack the file.

GNU `tar` is known to work. The standard `tar` provided with some operating systems is not able to unpack the long file names in the MySQL distribution. You should download and install GNU `tar`, or if available, use a preinstalled version of GNU `tar`. Usually this is available as `gnutar`, `gtar`, or as `tar` within a GNU or Free Software directory, such as `/usr/sfw/bin` or `/usr/local/bin`. GNU `tar` is available from <http://www.gnu.org/software/tar/>.

- For a `.zip` Zip archive: [WinZip](#) or another tool that can read `.zip` files.
- For an `.rpm` RPM package: The `rpmbuild` program used to build the distribution unpacks it.

To install MySQL from a development source tree, the following additional tools are required:

- The Git revision control system is required to obtain the development source code. The [GitHub Help](#) provides instructions for downloading and installing Git on different platforms. MySQL officially joined GitHub in September, 2014. For more information about MySQL's move to GitHub, refer to the announcement on the MySQL Release Engineering blog: [MySQL on GitHub](#)
- `bison` 2.1 or higher, available from <http://www.gnu.org/software/bison/>. (Version 1 is no longer supported.) Use the latest version of `bison` where possible; if you experience problems, upgrade to a later version, rather than revert to an earlier one.

`bison` is available from <http://www.gnu.org/software/bison/>. `bison` for Windows can be downloaded from <http://gnuwin32.sourceforge.net/packages/bison.htm>. Download the package labeled “Complete package, excluding sources”. On Windows, the default location for `bison` is the `C:\Program Files\GnuWin32` directory. Some utilities may fail to find `bison` because of the space in the directory name. Also, Visual Studio may simply hang if there are spaces in the path. You can resolve these problems by installing into a directory that does not contain a space; for example `C:\GnuWin32`.

- On Solaris Express, `m4` must be installed in addition to `bison`. `m4` is available from <http://www.gnu.org/software/m4/>.

Note

If you have to install any programs, modify your `PATH` environment variable to include any directories in which the programs are located. See [Setting Environment Variables](#).

If you run into problems and need to file a bug report, please use the instructions in [How to Report Bugs or Problems](#).

Chapter 2 Installing MySQL Using a Standard Source Distribution

To install MySQL from a standard source distribution:

1. Verify that your system satisfies the tool requirements listed at [Chapter 1, *Installing MySQL from Source*](#).
2. Obtain a distribution file using the instructions in [How to Get MySQL](#).
3. Configure, build, and install the distribution using the instructions in this section.
4. Perform postinstallation procedures using the instructions in [Postinstallation Setup and Testing](#).

In MySQL 5.7, [CMake](#) is used as the build framework on all platforms. The instructions given here should enable you to produce a working installation. For additional information on using [CMake](#) to build MySQL, see [How to Build MySQL Server with CMake](#).

If you start from a source RPM, use the following command to make a binary RPM that you can install. If you do not have `rpmbuild`, use `rpm` instead.

```
shell> rpmbuild --rebuild --clean MySQL-VERSION.src.rpm
```

The result is one or more binary RPM packages that you install as indicated in [Installing MySQL on Linux Using RPM Packages from Oracle](#).

The sequence for installation from a compressed `tar` file or Zip archive source distribution is similar to the process for installing from a generic binary distribution (see [Installing MySQL on Unix/Linux Using Generic Binaries](#)), except that it is used on all platforms and includes steps to configure and compile the distribution. For example, with a compressed `tar` file source distribution on Unix, the basic installation command sequence looks like this:

```
# Preconfiguration setup
shell> groupadd mysql
shell> useradd -r -g mysql -s /bin/false mysql
# Beginning of source-build specific instructions
shell> tar zxvf mysql-VERSION.tar.gz
shell> cd mysql-VERSION
shell> mkdir bld
shell> cd bld
shell> cmake ..
shell> make
shell> make install
# End of source-build specific instructions
# Postinstallation setup
shell> cd /usr/local/mysql
shell> mkdir mysql-files
shell> chown mysql:mysql mysql-files
shell> chmod 750 mysql-files
shell> bin/mysqld --initialize --user=mysql
shell> bin/mysql_ssl_rsa_setup
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

A more detailed version of the source-build specific instructions is shown following.

Note

The procedure shown here does not set up any passwords for MySQL accounts. After following the procedure, proceed to [Postinstallation Setup and Testing](#), for postinstallation setup and testing.

Perform Preconfiguration Setup

On Unix, set up the `mysql` user and group that will be used to run and execute the MySQL server and own the database directory. For details, see [Creating a `mysql` System User and Group](#), in [Installing MySQL on Unix/Linux Using Generic Binaries](#). Then perform the following steps as the `mysql` user, except as noted.

Obtain and Unpack the Distribution

Pick the directory under which you want to unpack the distribution and change location into it.

Obtain a distribution file using the instructions in [How to Get MySQL](#).

Unpack the distribution into the current directory:

- To unpack a compressed `tar` file, `tar` can uncompress and unpack the distribution if it has `z` option support:

```
shell> tar zxvf mysql-VERSION.tar.gz
```

If your `tar` does not have `z` option support, use `gunzip` to unpack the distribution and `tar` to unpack it:

```
shell> gunzip < mysql-VERSION.tar.gz | tar xvf -
```

Alternatively, `CMake` can uncompress and unpack the distribution:

```
shell> cmake -E tar zxvf mysql-VERSION.tar.gz
```

- To unpack a Zip archive, use [WinZip](#) or another tool that can read `.zip` files.

Unpacking the distribution file creates a directory named `mysql-VERSION`.

Configure the Distribution

Change location into the top-level directory of the unpacked distribution:

```
shell> cd mysql-VERSION
```

Build outside of the source tree to keep the tree clean. If the top-level source directory is named `mysql-src` under your current working directory, you can build in a directory named `bld` at the same level. Create the directory and go there:

```
shell> mkdir bld
shell> cd bld
```

Configure the build directory. The minimum configuration command includes no options to override configuration defaults:

```
shell> cmake ../mysql-src
```

The build directory needs not be outside the source tree. For example, you can build in a directory named `bld` under the top-level source tree. To do this, starting with `mysql-src` as your current working directory, create the directory `bld` and then go there:

```
shell> mkdir bld
shell> cd bld
```

Configure the build directory. The minimum configuration command includes no options to override configuration defaults:

```
shell> cmake ..
```

If you have multiple source trees at the same level (for example, to build multiple versions of MySQL), the second strategy can be advantageous. The first strategy places all build directories at the same level, which requires that you choose a unique name for each. With the second strategy, you can use the same name for the build directory within each source tree. The following instructions assume this second strategy.

On Windows, specify the development environment. For example, the following commands configure MySQL for 32-bit or 64-bit builds, respectively:

```
shell> cmake .. -G "Visual Studio 12 2013"
shell> cmake .. -G "Visual Studio 12 2013 Win64"
```

On macOS, to use the Xcode IDE:

```
shell> cmake .. -G Xcode
```

When you run `cmake`, you might want to add options to the command line. Here are some examples:

- `-DBUILD_CONFIG=mysql_release`: Configure the source with the same build options used by Oracle to produce binary distributions for official MySQL releases.
- `-DCMAKE_INSTALL_PREFIX=dir_name`: Configure the distribution for installation under a particular location.
- `-DCPACK_MONOLITHIC_INSTALL=1`: Cause `make package` to generate a single installation file rather than multiple files.
- `-DWITH_DEBUG=1`: Build the distribution with debugging support.

For a more extensive list of options, see [Chapter 4, MySQL Source-Configuration Options](#).

To list the configuration options, use one of the following commands:

```
shell> cmake .. -L # overview
shell> cmake .. -LH # overview with help text
shell> cmake .. -LAH # all params with help text
shell> ccmake .. # interactive display
```

If `CMake` fails, you might need to reconfigure by running it again with different options. If you do reconfigure, take note of the following:

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts up, it looks for that file and reads its contents if it exists, on the assumption that the information is still correct. That assumption is invalid when you reconfigure.
- Each time you run `CMake`, you must run `make` again to recompile. However, you may want to remove old object files from previous builds first because they were compiled using different configuration options.

To prevent old object files or configuration information from being used, run these commands in the build directory on Unix before re-running `CMake`:

```
shell> make clean
shell> rm CMakeCache.txt
```

Or, on Windows:

```
shell> devenv MySQL.sln /clean
shell> del CMakeCache.txt
```

If you are going to send mail to a MySQL mailing list to ask for configuration assistance, first check the files in the `CMakeFiles` directory for useful information about the failure. To file a bug report, please use the instructions in [How to Report Bugs or Problems](#).

Build the Distribution

On Unix:

```
shell> make
shell> make VERBOSE=1
```

The second command sets `VERBOSE` to show the commands for each compiled source.

Use `gmake` instead on systems where you are using GNU `make` and it has been installed as `gmake`.

On Windows:

```
shell> devenv MySQL.sln /build RelWithDebInfo
```

If you have gotten to the compilation stage, but the distribution does not build, see [Chapter 5, Dealing with Problems Compiling MySQL](#), for help. If that does not solve the problem, please enter it into our bugs database using the instructions given in [How to Report Bugs or Problems](#). If you have installed the latest versions of the required tools, and they crash trying to process our configuration files, please report that also. However, if you get a `command not found` error or a similar problem for required tools, do not report it. Instead, make sure that all the required tools are installed and that your `PATH` variable is set correctly so that your shell can find them.

Install the Distribution

On Unix:

```
shell> make install
```

This installs the files under the configured installation directory (by default, `/usr/local/mysql`). You might need to run the command as `root`.

To install in a specific directory, add a `DESTDIR` parameter to the command line:

```
shell> make install DESTDIR="/opt/mysql"
```

Alternatively, generate installation package files that you can install where you like:

```
shell> make package
```

This operation produces one or more `.tar.gz` files that can be installed like generic binary distribution packages. See [Installing MySQL on Unix/Linux Using Generic Binaries](#). If you run `CMake` with `-DCPACK_MONOLITHIC_INSTALL=1`, the operation produces a single file. Otherwise, it produces multiple files.

On Windows, generate the data directory, then create a `.zip` archive installation package:

```
shell> devenv MySQL.sln /build RelWithDebInfo /project initial_database
```

```
shell> devenv MySQL.sln /build RelWithDebInfo /project package
```

You can install the resulting `.zip` archive where you like. See [Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive](#).

Perform Postinstallation Setup

The remainder of the installation process involves setting up the configuration file, creating the core databases, and starting the MySQL server. For instructions, see [Postinstallation Setup and Testing](#).

Note

The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in [Postinstallation Setup and Testing](#).

Chapter 3 Installing MySQL Using a Development Source Tree

This section describes how to install MySQL from the latest development source code, which is hosted on [GitHub](#). To obtain the MySQL Server source code from this repository hosting service, you can set up a local MySQL Git repository.

On [GitHub](#), MySQL Server and other MySQL projects are found on the [MySQL](#) page. The MySQL Server project is a single repository that contains branches for several MySQL series.

MySQL officially joined GitHub in September, 2014. For more information about MySQL's move to GitHub, refer to the announcement on the MySQL Release Engineering blog: [MySQL on GitHub](#)

Prerequisites for Installing from Development Source

To install MySQL from a development source tree, your system must satisfy the tool requirements outlined in [Chapter 1, Installing MySQL from Source](#).

Setting Up a MySQL Git Repository

To set up a MySQL Git repository on your machine, use this procedure:

1. Clone the MySQL Git repository to your machine. The following command clones the MySQL Git repository to a directory named `mysql-server`. The initial download will take some time to complete, depending on the speed of your connection.

```
~$ git clone https://github.com/mysql/mysql-server.git
Cloning into 'mysql-server'...
remote: Counting objects: 1035465, done.
remote: Total 1035465 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (1035465/1035465), 437.48 MiB | 5.10 MiB/s, done.
Resolving deltas: 100% (855607/855607), done.
Checking connectivity... done.
Checking out files: 100% (21902/21902), done.
```

2. When the clone operation completes, the contents of your local MySQL Git repository appear similar to the following:

```
~$ cd mysql-server
~/mysql-server$ ls
BUILD          COPYING          libmysqld       regex           unittest
BUILD-CMAKE    dbug             libservices     scripts         VERSION
client         Docs             man             sql            vio
cmake          extra           mysql-test     sql-common     win
CMakeLists.txt include          mysys          storage        zlib
cmd-line-utils INSTALL-SOURCE  packaging      strings
config.h.cmake INSTALL-WIN-SOURCE plugin          support-files
configure.cmake libmysql        README         tests
```

3. Use the `git branch -r` command to view the remote tracking branches for the MySQL repository.

```
~/mysql-server$ git branch -r
origin/5.5
origin/5.6
origin/5.7
origin/HEAD -> origin/5.7
origin/cluster-7.2
origin/cluster-7.3
origin/cluster-7.4
```

4. To view the branches that are checked out in your local repository, issue the `git branch` command. When you cloned the MySQL Git repository, the MySQL 5.7 branch was checked out automatically. The asterisk identifies the 5.7 branch as the active branch.

```
~/mysql-server$ git branch
```

```
* 5.7
```

5. To check out a different MySQL branch, run the `git checkout` command, specifying the branch name. For example, to check out the MySQL 5.5 branch:

```
~/mysql-server$ git checkout 5.5
Branch 5.5 set up to track remote branch 5.5 from origin.
Switched to a new branch '5.5'
```

6. Run `git branch` to verify that the MySQL 5.5 branch is present. MySQL 5.5, which is the last branch you checked out, is marked by an asterisk indicating that it is the active branch.

```
~/mysql-server$ git branch
* 5.5
  5.7
```

7. Use the `git checkout` command to switch between branches. For example:

```
~/mysql-server$ git checkout 5.7
```

8. To obtain changes made after your initial setup of the MySQL Git repository, switch to the branch you want to update and issue the `git pull` command:

```
~/mysql-server$ git checkout 5.7
~/mysql-server$ git pull
```

To examine the commit history, use the `git log` option:

```
~/mysql-server$ git log
```

You can also browse commit history and source code on the GitHub [MySQL](#) site.

If you see changes or code that you have a question about, send an email to the MySQL [internals](#) mailing list. See [MySQL Mailing Lists](#). For information about contributing a patch, see [Contributing to MySQL Server](#).

9. After you have cloned the MySQL Git repository and have checked out the branch you want to build, you can build MySQL Server from the source code. Instructions are provided in [Chapter 2, Installing MySQL Using a Standard Source Distribution](#), except that you skip the part about obtaining and unpacking the distribution.

Be careful about installing a build from a distribution source tree on a production machine. The installation command may overwrite your live release installation. If you already have MySQL installed and do not want to overwrite it, run `CMake` with values for the `CMAKE_INSTALL_PREFIX`, `MYSQL_TCP_PORT`, and `MYSQL_UNIX_ADDR` options different from those used by your production server. For additional information about preventing multiple servers from interfering with each other, see [Running Multiple MySQL Instances on One Machine](#).

Play hard with your new installation. For example, try to make new features crash. Start by running `make test`. See [The MySQL Test Suite](#).

Chapter 4 MySQL Source-Configuration Options

The `CMake` program provides a great deal of control over how you configure a MySQL source distribution. Typically, you do this using options on the `CMake` command line. For information about options supported by `CMake`, run either of these commands in the top-level source directory:

```
shell> cmake . -LH
shell> ccmake .
```

You can also affect `CMake` using certain environment variables. See [MySQL Program Environment Variables](#).

The following table shows the available `CMake` options. In the `Default` column, `PREFIX` stands for the value of the `CMAKE_INSTALL_PREFIX` option, which specifies the installation base directory. This value is used as the parent location for several of the installation subdirectories.

Table 4.1 MySQL Source-Configuration Option Reference (CMake)

Formats	Description	Default	Introduced	Removed
<code>BUILD_CONFIG</code>	Use same build options as official releases			
<code>CMAKE_BUILD_TYPE</code>	Type of build to produce	<code>RelWithDebInfo</code>		
<code>CMAKE_CXX_FLAGS</code>	Flags for C++ Compiler			
<code>CMAKE_C_FLAGS</code>	Flags for C Compiler			
<code>CMAKE_INSTALL_PREFIX</code>	Installation base directory	<code>/usr/local/mysql</code>		
<code>COMPILATION_COMMENT</code>	Comment about compilation environment			
<code>CPACK_MONOLITHIC_INSTALL</code>	Whether package build produces single file	<code>OFF</code>		
<code>DEFAULT_CHARSET</code>	The default server character set	<code>latin1</code>		
<code>DEFAULT_COLLATION</code>	The default server collation	<code>latin1_swedish_ci</code>		
<code>DISABLE_PSI_COND</code>	Exclude Performance Schema condition instrumentation	<code>OFF</code>	5.7.3	
<code>DISABLE_PSI_FILE</code>	Exclude Performance Schema file instrumentation	<code>OFF</code>	5.7.3	
<code>DISABLE_PSI_IDLE</code>	Exclude Performance Schema idle instrumentation	<code>OFF</code>	5.7.3	
<code>DISABLE_PSI_MEMORY</code>	Exclude Performance Schema memory instrumentation	<code>OFF</code>	5.7.3	
<code>DISABLE_PSI_METADATA</code>	Exclude Performance Schema metadata instrumentation	<code>OFF</code>	5.7.3	
<code>DISABLE_PSI_MUTEX</code>	Exclude Performance Schema mutex instrumentation	<code>OFF</code>	5.7.3	
<code>DISABLE_PSI_PS</code>	Exclude the performance schema prepared statements	<code>OFF</code>	5.7.3	

Formats	Description	Default	Introduced	Removed
DISABLE_PSI_RWLOCK	Exclude Performance Schema rwlock instrumentation	OFF	5.7.3	
DISABLE_PSI_SOCKET	Exclude Performance Schema socket instrumentation	OFF	5.7.3	
DISABLE_PSI_SP	Exclude Performance Schema stored program instrumentation	OFF	5.7.3	
DISABLE_PSI_STAGE	Exclude Performance Schema stage instrumentation	OFF	5.7.3	
DISABLE_PSI_STATEMENT	Exclude Performance Schema statement instrumentation	OFF	5.7.3	
DISABLE_PSI_STATEMENT_DIGEST	Exclude Performance Schema statements_digest instrumentation	OFF	5.7.3	
DISABLE_PSI_TABLE	Exclude Performance Schema table instrumentation	OFF	5.7.3	
DISABLE_PSI_THREAD	Exclude the performance schema thread instrumentation	OFF	5.7.3	
DISABLE_PSI_TRANSACTION	Exclude the performance schema transaction instrumentation	OFF	5.7.3	
DOWNLOAD_BOOST	Whether to download the Boost library	OFF	5.7.5	
DOWNLOAD_BOOST_TIMEOUT	Timeout in seconds for downloading the Boost library	600	5.7.6	
ENABLED_LOCAL_INFILE	Whether to enable LOCAL for LOAD DATA INFILE	OFF		
ENABLED_PROFILING	Whether to enable query profiling code	ON		
ENABLE_DEBUG_SYNC	Whether to enable Debug Sync support	ON		5.7.1
ENABLE_DOWNLOADS	Whether to download optional files	OFF		
ENABLE_DTRACE	Whether to include DTrace support			
ENABLE_GCOV	Whether to include gcov support			
ENABLE_GPROF	Enable gprof (optimized Linux builds only)	OFF		
FORCE_UNSUPPORTED_COMPILER	Whether to permit unsupported compiler	OFF	5.7.5	
IGNORE_AIO_CHECK	With - DBUILD_CONFIG=mysql_release, ignore libaio check	OFF		

Formats	Description	Default	Introduced	Removed
INNODB_PAGE_ATOMIC_REF	Enable or disable atomic page reference counting	ON	5.7.4	5.7.5
INSTALL_BINDIR	User executables directory	PREFIX/bin		
INSTALL_DOCDIR	Documentation directory	PREFIX/docs		
INSTALL_DOCREADMEDIR	README file directory	PREFIX		
INSTALL_INCLUDEDIR	Header file directory	PREFIX/include		
INSTALL_INFODIR	Info file directory	PREFIX/docs		
INSTALL_LAYOUT	Select predefined installation layout	STANDALONE		
INSTALL_LIBDIR	Library file directory	PREFIX/lib		
INSTALL_MANDIR	Manual page directory	PREFIX/man		
INSTALL_MYSQLKEYRINGDIR	Directory for keyring_file plugin data file	platform specific	5.7.11	
INSTALL_MYSQLSHAREDIR	Shared data directory	PREFIX/share		
INSTALL_MYSQLTESTDIR	mysql-test directory	PREFIX/mysql-test		
INSTALL_PKGCONFIGDIR	Directory for mysqlclient.pc pkg-config file	INSTALL_LIBDIR/pkgconfig	5.7.9	
INSTALL_PLUGINDIR	Plugin directory	PREFIX/lib/plugin		
INSTALL_SBINDIR	Server executable directory	PREFIX/bin		
INSTALL_SCRIPTDIR	Scripts directory	PREFIX/scripts		
INSTALL_SECURE_FILE_PRIV	secure_file_priv default value	platform specific	5.7.6	
INSTALL_SECURE_FILE_PRIV_LIBMYSQLD	secure_file_priv default value for libmysqld		5.7.8	
INSTALL_SHAREDIR	aclocal/mysql.m4 installation directory	PREFIX/share		
INSTALL_SQLBENCHDIR	sql-bench directory	PREFIX		5.7.8
INSTALL_SUPPORTFILES	Extra support files directory	PREFIX/support-files		
MAX_INDEXES	Maximum indexes per table	64	5.7.1	
MUTEX_TYPE	InnoDB mutex type	event	5.7.2	
MYSQLX_TCP_PORT	TCP/IP port number used by X Plugin	33060	5.7.17	
MYSQLX_UNIX_ADDR	Unix socket file used by X Plugin	/tmp/mysqlx.sock	5.7.15	
MYSQL_DATADIR	Data directory			
MYSQL_MAINTAINER_MODE	Whether to enable MySQL maintainer-specific development environment	OFF		
MYSQL_PROJECT_NAME	Windows/OS X project name	MySQL		
MYSQL_TCP_PORT	TCP/IP port number	3306		
MYSQL_UNIX_ADDR	Unix socket file	/tmp/mysql.sock		

Formats	Description	Default	Introduced	Removed
ODBC_INCLUDES	ODBC includes directory			
ODBC_LIB_DIR	ODBC library directory			
OPTIMIZER_TRACE	Whether to support optimizer tracing			
REPRODUCIBLE_BUILD	Take extra care to create a build result independent of build location and time		5.7.19	
SUNPRO_CXX_LIBRARY	Client link library on Solaris 10+		5.7.5	
SYSCONFDIR	Option file directory			
SYSTEMD_PID_DIR	Directory for PID file under systemd	<code>/var/run/mysqld</code>	5.7.6	
SYSTEMD_SERVICE_NAME	Name of MySQL service under systemd	<code>mysqld</code>	5.7.6	
TMPDIR	tmpdir default value		5.7.4	
WIN_DEBUG_NO_INLINE	Whether to disable function inlining	<code>OFF</code>	5.7.6	
WITHOUT_SERVER	Do not build the server	<code>OFF</code>		
WITHOUT_XXX_STORAGE_ENGINE	Exclude storage engine xxx from build			
WITH_ASAN	Enable AddressSanitizer	<code>OFF</code>	5.7.3	
WITH_ASAN_SCOPE	Enable AddressSanitizer - fsanitize-address-use-after-scope Clang flag	<code>OFF</code>	5.7.21	
WITH_AUTHENTICATION_LDAP	Whether to report error if LDAP authentication plugins cannot be built	<code>OFF</code>	5.7.19	
WITH_AUTHENTICATION_PAM	Build PAM authentication plugin	<code>OFF</code>		
WITH_AWS_SDK	Location of Amazon Web Services software development kit		5.7.19	
WITH_BOOST	The location of the Boost library sources		5.7.5	
WITH_CLIENT_PROTOCOL_TRACING	Build client-side protocol tracing framework	<code>ON</code>	5.7.2	
WITH_CURL	Location of curl library		5.7.19	
WITH_DEBUG	Whether to include debugging support	<code>OFF</code>		
WITH_DEFAULT_COMPILER_OPTIONS	Whether to use default compiler options	<code>ON</code>		
WITH_DEFAULT_FEATURE_SET	Whether to use default feature set	<code>ON</code>		
WITH_EDITLINE	Which libedit/editline library to use	<code>bundled</code>	5.7.2	
WITH_EMBEDDED_SERVER	Whether to build embedded server	<code>OFF</code>		

Formats	Description	Default	Introduced	Removed
WITH_EMBEDDED_SHARED_LIBRARY	Whether to build a shared embedded server library	OFF	5.7.4	
WITH_EXTRA_CHARSETS	Which extra character sets to include	all		
WITH_INNOODB_EXTRA_DEBUG	Whether to include extra debugging support for InnoDB.	OFF	5.7.2	
WITH_INNOODB_MEMCACHED	Whether to generate memcached shared libraries.	OFF		
WITH_KEYRING_TEST	Build the keyring test program	OFF	5.7.11	
WITH_LIBEVENT	Which libevent library to use	bundled		
WITH_LIBWRAP	Whether to include libwrap (TCP wrappers) support	OFF		
WITH_LZ4	Type of LZ4 library support	bundled	5.7.14	
WITH_MECAB	Compiles MeCab		5.7.6	
WITH_MSAN	Enable MemorySanitizer	OFF	5.7.4	
WITH_MSVCRT_DEBUG	Enable Visual Studio CRT memory leak tracing	OFF	5.7.6	
WITH_NDBCLUSTER	Build the NDB storage engine; alias for WITH_NDBCLUSTER_STORAGE_ENGINE	ON		
WITH_NDBCLUSTER_STORAGE_ENGINE	Build the NDB storage engine	ON		
WITH_NDB_TEST	Include NDB API test programs.	OFF		
WITH_NUMA	Set NUMA memory allocation policy		5.7.17	
WITH_PROTOBUF	Which Protocol Buffers package to use	bundled	5.7.12	
WITH_RAPID	Whether to build rapid development cycle plugins	ON	5.7.12	
WITH_SSL	Type of SSL support	bundled		
WITH_SYSTEMD	Enable installation of systemd support files	OFF	5.7.6	
WITH_TEST_TRACE_PLUGIN	Build test protocol trace plugin	OFF	5.7.2	
WITH_UBSAN	Enable Undefined Behavior Sanitizer	OFF	5.7.6	
WITH_UNIT_TESTS	Compile MySQL with unit tests	ON		
WITH_UNIXODBC	Enable unixODBC support	OFF		
WITH_VALGRIND	Whether to compile in Valgrind header files	OFF		
WITH_ZLIB	Type of zlib support	bundled		
WITH_XXX_STORAGE_ENGINE	Compile storage engine xxx statically into server			

The following sections provide more information about CMake options.

- [General Options](#)
- [Installation Layout Options](#)
- [Storage Engine Options](#)
- [Feature Options](#)
- [Compiler Flags](#)
- [CMake Options for Compiling NDB Cluster](#)

For boolean options, the value may be specified as 1 or `ON` to enable the option, or as 0 or `OFF` to disable the option.

Many options configure compile-time defaults that can be overridden at server startup. For example, the `CMAKE_INSTALL_PREFIX`, `MYSQL_TCP_PORT`, and `MYSQL_UNIX_ADDR` options that configure the default installation base directory location, TCP/IP port number, and Unix socket file can be changed at server startup with the `--basedir`, `--port`, and `--socket` options for `mysqld`. Where applicable, configuration option descriptions indicate the corresponding `mysqld` startup option.

General Options

- `-DBUILD_CONFIG=mysql_release`

This option configures a source distribution with the same build options used by Oracle to produce binary distributions for official MySQL releases.

- `-DCMAKE_BUILD_TYPE=type`

The type of build to produce:

- `RelWithDebInfo`: Enable optimizations and generate debugging information. This is the default MySQL build type.
- `Debug`: Disable optimizations and generate debugging information. This build type is also used if the `WITH_DEBUG` option is enabled. That is, `-DWITH_DEBUG=1` has the same effect as `-DCMAKE_BUILD_TYPE=Debug`.
- `-DCPACK_MONOLITHIC_INSTALL=bool`

This option affects whether the `make package` operation produces multiple installation package files or a single file. If disabled, the operation produces multiple installation package files, which may be useful if you want to install only a subset of a full MySQL installation. If enabled, it produces a single file for installing everything.

Installation Layout Options

The `CMAKE_INSTALL_PREFIX` option indicates the base installation directory. Other options with names of the form `INSTALL_XXX` that indicate component locations are interpreted relative to the prefix and their values are relative pathnames. Their values should not include the prefix.

- `-DCMAKE_INSTALL_PREFIX=dir_name`

The installation base directory.

This value can be set at server startup with the `--basedir` option.

- `-DINSTALL_BINDIR=dir_name`

Where to install user programs.

- `-DINSTALL_DOCDIR=dir_name`

Where to install documentation.

- `-DINSTALL_DOCREADMEDIR=dir_name`

Where to install `README` files.

- `-DINSTALL_INCLUDEDIR=dir_name`

Where to install header files.

- `-DINSTALL_INFODIR=dir_name`

Where to install Info files.

- `-DINSTALL_LAYOUT=name`

Select a predefined installation layout:

- `STANDALONE`: Same layout as used for `.tar.gz` and `.zip` packages. This is the default.
- `RPM`: Layout similar to RPM packages.
- `SVR4`: Solaris package layout.
- `DEB`: DEB package layout (experimental).

You can select a predefined layout but modify individual component installation locations by specifying other options. For example:

```
shell> cmake . -DINSTALL_LAYOUT=SVR4 -DMYSQL_DATADIR=/var/mysql/data
```

The `INSTALL_LAYOUT` value determines the default value of the `secure_file_priv`, `keyring_encrypted_file_data`, and `keyring_file_data` system variables. See the descriptions of those variables in [Server System Variables](#), and [Keyring System Variables](#).

- `-DINSTALL_LIBDIR=dir_name`

Where to install library files.

- `-DINSTALL_MANDIR=dir_name`

Where to install manual pages.

- `-DINSTALL_MYSQLKEYRINGDIR=dir_path`

The default directory to use as the location of the `keyring_file` plugin data file. The default value is platform specific and depends on the value of the `INSTALL_LAYOUT` CMake option; see the description of the `keyring_file_data` system variable in [Server System Variables](#).

This option was added in MySQL 5.7.11.

- `-DINSTALL_MYSQLSHAREDIR=dir_name`

Where to install shared data files.

- `-DINSTALL_MYSQLTESTDIR=dir_name`

Where to install the `mysql-test` directory. To suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_MYSQLTESTDIR=`).

- `-DINSTALL_PKGCONFIGDIR=dir_name`

The directory in which to install the `mysqlclient.pc` file for use by `pkg-config`. The default value is `INSTALL_LIBDIR/pkgconfig`, unless `INSTALL_LIBDIR` ends with `/mysql`, in which case that is removed first.

- `-DINSTALL_PLUGINDIR=dir_name`

The location of the plugin directory.

This value can be set at server startup with the `--plugin_dir` option.

- `-DINSTALL_SBINDIR=dir_name`

Where to install the `mysqld` server.

- `-DINSTALL_SCRIPTDIR=dir_name`

Where to install `mysql_install_db`.

- `-DINSTALL_SECURE_FILE_PRIVDIR=dir_name`

The default value for the `secure_file_priv` system variable. The default value is platform specific and depends on the value of the `INSTALL_LAYOUT CMake` option; see the description of the `secure_file_priv` system variable in [Server System Variables](#).

To set the value for the `libmysqld` embedded server, use `INSTALL_SECURE_FILE_PRIV_EMBEDDEDIR`.

- `-DINSTALL_SECURE_FILE_PRIV_EMBEDDEDIR=dir_name`

The default value for the `secure_file_priv` system variable, for the `libmysqld` embedded server.

Note

The `libmysqld` embedded server library is deprecated as of MySQL 5.7.19 and will be removed in MySQL 8.0.

- `-DINSTALL_SHAREDIR=dir_name`

Where to install `aclocal/mysql.m4`.

- `-DINSTALL_SQLBENCHDIR=dir_name`

Where to install the `sql-bench` directory. To suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_SQLBENCHDIR=`).

The `sql-bench` directory is no longer included in MySQL distributions, so the `INSTALL_SQLBENCHDIR=` option is removed as well.

- `-DINSTALL_SUPPORTFILESDIR=dir_name`

Where to install extra support files.

- `-DMYSQL_DATADIR=dir_name`

The location of the MySQL data directory.

This value can be set at server startup with the `--datadir` option.

- `-DODBC_INCLUDES=dir_name`

The location of the ODBC includes directory, and may be used while configuring Connector/ODBC.

- `-DODBC_LIB_DIR=dir_name`

The location of the ODBC library directory, and may be used while configuring Connector/ODBC.

- `-DSYSCONFDIR=dir_name`

The default `my.cnf` option file directory.

This location cannot be set at server startup, but you can start the server with a given option file using the `--defaults-file=file_name` option, where `file_name` is the full path name to the file.

- `-DSYSTEMD_PID_DIR=dir_name`

The name of the directory in which to create the PID file when MySQL is managed by systemd. The default is `/var/run/mysqld`; this might be changed implicitly according to the `INSTALL_LAYOUT` value.

This option is ignored unless `WITH_SYSTEMD` is enabled.

- `-DSYSTEMD_SERVICE_NAME=name`

The name of the MySQL service to use when MySQL is managed by systemd. The default is `mysqld`; this might be changed implicitly according to the `INSTALL_LAYOUT` value.

This option is ignored unless `WITH_SYSTEMD` is enabled.

- `-DTMPDIR=dir_name`

The default location to use for the `tmpdir` system variable. If unspecified, the value defaults to `P_tmpdir` in `<stdio.h>`.

Storage Engine Options

Storage engines are built as plugins. You can build a plugin as a static module (compiled into the server) or a dynamic module (built as a dynamic library that must be installed into the server using the `INSTALL PLUGIN` statement or the `--plugin-load` option before it can be used). Some plugins might not support static or dynamic building.

The `InnoDB`, `MyISAM`, `MERGE`, `MEMORY`, and `CSV` engines are mandatory (always compiled into the server) and need not be installed explicitly.

To compile a storage engine statically into the server, use `-DWITH_engine_STORAGE_ENGINE=1`. Some permissible `engine` values are `ARCHIVE`, `BLACKHOLE`, `EXAMPLE`, `FEDERATED`, `NDB` or `NDBCLUSTER` (`NDB`), and `PARTITION` (partitioning support). Examples:

```
-DWITH_ARCHIVE_STORAGE_ENGINE=1
-DWITH_BLACKHOLE_STORAGE_ENGINE=1
```

Note

`WITH_NDBCLUSTER_STORAGE_ENGINE` is supported only when building NDB Cluster using the NDB Cluster sources. It cannot be used to enable clustering support in other MySQL source trees or distributions. In NDB Cluster source distributions, it is enabled by default. See [Building NDB Cluster from Source on Linux](#), and [Compiling and Installing NDB Cluster from Source on Windows](#), for more information.

Note

It is not possible to compile without Performance Schema support. If it is desired to compile without particular types of instrumentation, that can be done with the following CMake options:

```
DISABLE_PSI_COND
DISABLE_PSI_FILE
DISABLE_PSI_IDLE
DISABLE_PSI_MEMORY
DISABLE_PSI_METADATA
DISABLE_PSI_MUTEX
DISABLE_PSI_PS
DISABLE_PSI_RWLOCK
DISABLE_PSI_SOCKET
DISABLE_PSI_SP
DISABLE_PSI_STAGE
DISABLE_PSI_STATEMENT
DISABLE_PSI_STATEMENT_DIGEST
DISABLE_PSI_TABLE
DISABLE_PSI_THREAD
DISABLE_PSI_TRANSACTION
```

For example, to compile without mutex instrumentation, configure MySQL using the `-DDISABLE_PSI_MUTEX=1` option.

To exclude a storage engine from the build, use `-DWITH_engine_STORAGE_ENGINE=0`. Examples:

```
-DWITH_EXAMPLE_STORAGE_ENGINE=0
-DWITH_FEDERATED_STORAGE_ENGINE=0
-DWITH_PARTITION_STORAGE_ENGINE=0
```

It is also possible to exclude a storage engine from the build using `-DWITHOUT_engine_STORAGE_ENGINE=1` (but `-DWITH_engine_STORAGE_ENGINE=0` is preferred). Examples:

```
-DWITHOUT_EXAMPLE_STORAGE_ENGINE=1
-DWITHOUT_FEDERATED_STORAGE_ENGINE=1
-DWITHOUT_PARTITION_STORAGE_ENGINE=1
```

If neither `-DWITH_engine_STORAGE_ENGINE` nor `-DWITHOUT_engine_STORAGE_ENGINE` are specified for a given storage engine, the engine is built as a shared module, or excluded if it cannot be built as a shared module.

Feature Options

- `-DCOMPILATION_COMMENT=string`

A descriptive comment about the compilation environment.

- `-DDEFAULT_CHARSET=charset_name`

The server character set. By default, MySQL uses the `latin1` (cp1252 West European) character set.

`charset_name` may be one of `binary`, `armscii8`, `ascii`, `big5`, `cp1250`, `cp1251`, `cp1256`, `cp1257`, `cp850`, `cp852`, `cp866`, `cp932`, `dec8`, `eucjpm`s, `euckr`, `gb2312`, `gbk`, `geostd8`, `greek`, `hebrew`, `hp8`, `keybcs2`, `koi8r`, `koi8u`, `latin1`, `latin2`, `latin5`, `latin7`, `macce`, `macroman`, `sjis`, `swe7`, `tis620`, `ucs2`, `ujis`, `utf8`, `utf8mb4`, `utf16`, `utf16le`, `utf32`. The permissible character sets are listed in the `cmake/character_sets.cmake` file as the value of `CHARSETS_AVAILABLE`.

This value can be set at server startup with the `--character_set_server` option.

- `-DDEFAULT_COLLATION=collation_name`

The server collation. By default, MySQL uses `latin1_swedish_ci`. Use the `SHOW COLLATION` statement to determine which collations are available for each character set.

This value can be set at server startup with the `--collation_server` option.

- `-DDISABLE_PSI_COND=bool`

Whether to exclude the Performance Schema condition instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_FILE=bool`

Whether to exclude the Performance Schema file instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_IDLE=bool`

Whether to exclude the Performance Schema idle instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_MEMORY=bool`

Whether to exclude the Performance Schema memory instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_METADATA=bool`

Whether to exclude the Performance Schema metadata instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_MUTEX=bool`

Whether to exclude the Performance Schema mutex instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_RWLOCK=bool`

Whether to exclude the Performance Schema rwlock instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_SOCKET=bool`

Whether to exclude the Performance Schema socket instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_SP=bool`

Whether to exclude the Performance Schema stored program instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_STAGE=bool`

Whether to exclude the Performance Schema stage instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_STATEMENT=bool`

Whether to exclude the Performance Schema statement instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_STATEMENT_DIGEST=bool`

Whether to exclude the Performance Schema `statement_digest` instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_TABLE=bool`

Whether to exclude the Performance Schema table instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_PS=bool`

Exclude the performance schema prepared statements instances instrumentation. The default is `OFF` (include).

- `-DDISABLE_PSI_THREAD=bool`

Exclude the performance schema thread instrumentation. The default is `OFF` (include).

Only disable threads when building without any instrumentation, because other instrumentations have a dependency on threads.

- `-DDISABLE_PSI_TRANSACTION=bool`

Exclude the performance schema transaction instrumentation. The default is `OFF` (include).

- `-DDOWNLOAD_BOOST=bool`

Whether to download the Boost library. The default is `OFF`.

See the `WITH_BOOST` option for additional discussion about using Boost.

- `-DDOWNLOAD_BOOST_TIMEOUT=seconds`

The timeout in seconds for downloading the Boost library. The default is 600 seconds.

See the `WITH_BOOST` option for additional discussion about using Boost.

- `-DENABLE_DEBUG_SYNC=bool`

Note

As of MySQL 5.7.18, `ENABLE_DEBUG_SYNC` is removed and enabling `WITH_DEBUG` enables Debug Sync.

Whether to compile the Debug Sync facility into the server. This facility is used for testing and debugging. This option is enabled by default, but has no effect unless MySQL is configured with debugging enabled. If debugging is enabled and you want to disable Debug Sync, use `-DENABLE_DEBUG_SYNC=0`.

When compiled in, Debug Sync is disabled by default at runtime. To enable it, start `mysqld` with the `--debug-sync-timeout=N` option, where `N` is a timeout value greater than 0. (The default value is 0, which disables Debug Sync.) `N` becomes the default timeout for individual synchronization points.

Sync debug checking for the `InnoDB` storage engine is available when debugging support is compiled in using the `WITH_DEBUG` option.

For a description of the Debug Sync facility and how to use synchronization points, see [MySQL Internals: Test Synchronization](#).

- `-DENABLE_DOWNLOADS=bool`

Whether to download optional files. For example, with this option enabled, `CMake` downloads the Google Test distribution that is used by the test suite to run unit tests.

- `-DENABLE_DTRACE=bool`

Whether to include support for DTrace probes. For information about DTrace, see [Tracing mysqld Using DTrace](#)

This option is deprecated because support for DTrace is deprecated in MySQL 5.7 and is removed in MySQL 8.0.

- `-DENABLE_GCOV=bool`

Whether to include gcov support (Linux only).

- `-DENABLE_GPROF=bool`

Whether to enable `gprof` (optimized Linux builds only).

- `-DENABLED_LOCAL_INFILE=bool`

This option controls the compiled-in default `LOCAL` capability for the MySQL client library. Clients that make no explicit arrangements therefore have `LOCAL` capability disabled or enabled according to the `ENABLED_LOCAL_INFILE` setting specified at MySQL build time.

By default, the client library in MySQL binary distributions is compiled with `ENABLED_LOCAL_INFILE` enabled. If you compile MySQL from source, configure it with `ENABLED_LOCAL_INFILE` disabled or enabled based on whether clients that make no explicit arrangements should have `LOCAL` capability disabled or enabled, respectively.

`ENABLED_LOCAL_INFILE` controls the default for client-side `LOCAL` capability. For the server, the `local_infile` system variable controls server-side `LOCAL` capability. To explicitly cause the server to refuse or permit `LOAD DATA LOCAL` statements (regardless of how client programs and libraries are configured at build time or runtime), start `mysqld` with `local_infile` disabled or enabled, respectively. `local_infile` can also be set at runtime. See [Security Issues with LOAD DATA LOCAL](#).

- `-DENABLED_PROFILING=bool`

Whether to enable query profiling code (for the `SHOW PROFILE` and `SHOW PROFILES` statements).

- `-DFORCE_UNSUPPORTED_COMPILER=bool`

By default, `CMake` checks for minimum versions of supported compilers: Visual Studio 2013 (Windows); GCC 4.4 or Clang 3.3 (Linux); Developer Studio 12.5 (Solaris server); Developer Studio 12.2 or GCC 4.4 (Solaris client library); Clang 3.3 (macOS), Clang 3.3 (FreeBSD). To disable this check, use `-DFORCE_UNSUPPORTED_COMPILER=ON`.

- `-DIGNORE_AIO_CHECK=bool`

If the `-DBUILD_CONFIG=mysql_release` option is given on Linux, the `libaio` library must be linked in by default. If you do not have `libaio` or do not want to install it, you can suppress the check for it by specifying `-DIGNORE_AIO_CHECK=1`.

- `-DINNODB_PAGE_ATOMIC_REF_COUNT=bool`

This option is obsolete.

- `-DMAX_INDEXES=num`

The maximum number of indexes per table. The default is 64. The maximum is 255. Values smaller than 64 are ignored and the default of 64 is used.

- `-DMYSQL_MAINTAINER_MODE=bool`

Whether to enable a MySQL maintainer-specific development environment. If enabled, this option causes compiler warnings to become errors.

- `-DMUTEX_TYPE=type`

The mutex type used by `InnoDB`. Options include:

- `event`: Use event mutexes. This is the default value and the original `InnoDB` mutex implementation.

- `sys`: Use POSIX mutexes on UNIX systems. Use `CRITICAL_SECTION` objects on Windows, if available.
- `futex`: Use Linux futexes instead of condition variables to schedule waiting threads.
- `-DMYSQLX_TCP_PORT=port_num`

The port number on which X Plugin listens for TCP/IP connections. The default is 33060.

This value can be set at server startup with the `--mysqlx-port` option.
- `-DMYSQLX_UNIX_ADDR=file_name`

The Unix socket file path on which the server listens for X Plugin socket connections. This must be an absolute path name. The default is `/tmp/mysqlx.sock`.

This value can be set at server startup with the `--mysqlx-socket` option.
- `-DMYSQL_PROJECT_NAME=name`

For Windows or macOS, the project name to incorporate into the project file name.
- `-DMYSQL_TCP_PORT=port_num`

The port number on which the server listens for TCP/IP connections. The default is 3306.

This value can be set at server startup with the `--port` option.
- `-DMYSQL_UNIX_ADDR=file_name`

The Unix socket file path on which the server listens for socket connections. This must be an absolute path name. The default is `/tmp/mysql.sock`.

This value can be set at server startup with the `--socket` option.
- `-DOPTIMIZER_TRACE=bool`

Whether to support optimizer tracing. See [MySQL Internals: Tracing the Optimizer](#).
- `-DREPRODUCIBLE_BUILD=bool`

For builds on Linux systems, this option controls whether to take extra care to create a build result independent of build location and time.

This option was added in MySQL 5.7.19.
- `-DWIN_DEBUG_NO_INLINE=bool`

Whether to disable function inlining on Windows. The default is off (inlining enabled).
- `-DWITH_ASAN=bool`

Whether to enable the AddressSanitizer, for compilers that support it. The default is off.
- `-DWITH_ASAN_SCOPE=bool`

Whether to enable the AddressSanitizer `-fsanitize-address-use-after-scope` Clang flag for use-after-scope detection. The default is off. To use this option, `-DWITH_ASAN` must also be enabled.
- `-DWITH_AUTHENTICATION_LDAP=bool`

Whether to report an error if the LDAP authentication plugins cannot be built:

- If this option is disabled (the default), the LDAP plugins are built if the required header files and libraries are found. If they are not, [CMake](#) displays a note about it.
- If this option is enabled, a failure to find the required header file and libraries causes [CMake](#) to produce an error, preventing the server from being built.

For information about LDAP authentication, see [LDAP Pluggable Authentication](#). This option was added in MySQL 5.7.19.

- `-DWITH_AUTHENTICATION_PAM=bool`

Whether to build the PAM authentication plugin, for source trees that include this plugin. (See [PAM Pluggable Authentication](#).) If this option is specified and the plugin cannot be compiled, the build fails.

- `-DWITH_AWS_SDK=path_name`

The location of the Amazon Web Services software development kit.

This option was added in MySQL 5.7.19.

- `-DWITH_BOOST=path_name`

The Boost library is required to build MySQL. These [CMake](#) options enable control over the library source location, and whether to download it automatically:

- `-DWITH_BOOST=path_name` specifies the Boost library directory location. It is also possible to specify the Boost location by setting the `BOOST_ROOT` or `WITH_BOOST` environment variable.

As of MySQL 5.7.11, `-DWITH_BOOST=system` is permitted and indicates that the correct version of Boost is installed on the compilation host in the standard location. In this case, the installed version of Boost is used rather than any version included with a MySQL source distribution.

- `-DDOWNLOAD_BOOST=bool` specifies whether to download the Boost source if it is not present in the specified location. The default is `OFF`.
- `-DDOWNLOAD_BOOST_TIMEOUT=seconds` the timeout in seconds for downloading the Boost library. The default is 600 seconds.

For example, if you normally build MySQL placing the object output in the `bld` subdirectory of your MySQL source tree, you can build with Boost like this:

```
mkdir bld
cd bld
cmake .. -DDOWNLOAD_BOOST=ON -DWITH_BOOST=$HOME/my_boost
```

This causes Boost to be downloaded into the `my_boost` directory under your home directory. If the required Boost version is already there, no download is done. If the required Boost version changes, the newer version is downloaded.

If Boost is already installed locally and your compiler finds the Boost header files on its own, it may not be necessary to specify the preceding [CMake](#) options. However, if the version of Boost required by MySQL changes and the locally installed version has not been upgraded, you may have build problems. Using the [CMake](#) options should give you a successful build.

With the above settings that allow Boost download into a specified location, when the required Boost version changes, you need to remove the `bld` folder, recreate it, and perform the `cmake` step again. Otherwise, the new Boost version might not get downloaded, and compilation might fail.

- `-DWITH_CLIENT_PROTOCOL_TRACING=bool`

Whether to build the client-side protocol tracing framework into the client library. By default, this option is enabled.

For information about writing protocol trace client plugins, see [Writing Protocol Trace Plugins](#).

See also the `WITH_TEST_TRACE_PLUGIN` option.

- `-DWITH_CURL=curl_type`

The location of the `curl` library. `curl_type` can be `system` (use the system `curl` library) or a path name to the `curl` library.

This option was added in MySQL 5.7.19.

- `-DWITH_DEBUG=bool`

Whether to include debugging support.

Configuring MySQL with debugging support enables you to use the `--debug="d,parser_debug"` option when you start the server. This causes the Bison parser that is used to process SQL statements to dump a parser trace to the server's standard error output. Typically, this output is written to the error log.

Sync debug checking for the `InnoDB` storage engine is defined under `UNIV_DEBUG` and is available when debugging support is compiled in using the `WITH_DEBUG` option. When debugging support is compiled in, the `innodb_sync_debug` configuration option can be used to enable or disable `InnoDB` sync debug checking.

As of MySQL 5.7.18, enabling `WITH_DEBUG` also enables Debug Sync. For a description of the Debug Sync facility and how to use synchronization points, see [MySQL Internals: Test Synchronization](#).

- `-DWITH_DEFAULT_FEATURE_SET=bool`

Whether to use the flags from `cmake/build_configurations/feature_set.cmake`.

- `-DWITH_EDITLINE=value`

Which `libedit/editline` library to use. The permitted values are `bundled` (the default) and `system`.

`WITH_EDITLINE` replaces `WITH_LIBEDIT`, which has been removed.

- `-DWITH_EMBEDDED_SERVER=bool`

Whether to build the `libmysqld` embedded server library.

Note

The `libmysqld` embedded server library is deprecated as of MySQL 5.7.17 and will be removed in MySQL 8.0.

- `-DWITH_EMBEDDED_SHARED_LIBRARY=bool`

Whether to build a shared `libmysqld` embedded server library.

Note

The `libmysqld` embedded server library is deprecated as of MySQL 5.7.17 and will be removed in MySQL 8.0.

- `-DWITH_EXTRA_CHARSETS=name`

Which extra character sets to include:

- `all`: All character sets. This is the default.
- `complex`: Complex character sets.
- `none`: No extra character sets.
- `-DWITH_INNODB_EXTRA_DEBUG=bool`

Whether to include extra InnoDB debugging support.

Enabling `WITH_INNODB_EXTRA_DEBUG` turns on extra InnoDB debug checks. This option can only be enabled when `WITH_DEBUG` is enabled.

- `-DWITH_INNODB_MEMCACHED=bool`

Whether to generate memcached shared libraries (`libmemcached.so` and `innodb_engine.so`).

- `-DWITH_KEYRING_TEST=bool`

Whether to build the test program that accompanies the `keyring_file` plugin. The default is `OFF`. Test file source code is located in the `plugin/keyring/keyring-test` directory.

This option was added in MySQL 5.7.11.

- `-DWITH_LIBEVENT=string`

Which `libevent` library to use. Permitted values are `bundled` (default), `system`, and `yes`. If you specify `system` or `yes`, the system `libevent` library is used if present. If the system library is not found, the bundled `libevent` library is used. The `libevent` library is required by InnoDB memcached.

- `-DWITH_LIBWRAP=bool`

Whether to include `libwrap` (TCP wrappers) support.

- `-DWITH_LZ4=lz4_type`

The `WITH_LZ4` indicates the source of `zlib` support:

- `bundled`: Use the `LZ4` library bundled with the distribution. This is the default.
- `system`: Use the system `LZ4` library. If `WITH_LZ4` is set to this value, the `lz4_decompress` utility is not built. In this case, the system `lz4` command can be used instead.
- `-DWITH_MSAN=bool`

Whether to enable MemorySanitizer, for compilers that support it. The default is off.

For this option to have an effect if enabled, all libraries linked to MySQL must also have been compiled with the option enabled.

- `-DWITH_MECAB={disabled|system|path_name}`

Use this option to compile the MeCab parser. If you have installed MeCab to its default installation directory, set `-DWITH_MECAB=system`. The `system` option applies to MeCab installations performed from source or from binaries using a native package management utility. If you installed MeCab to a custom installation directory, specify the path to the MeCab installation. For example, `-DWITH_MECAB=/opt/mecab`. If the `system` option does not work, specifying the MeCab installation path should work in all cases.

For related information, see [MeCab Full-Text Parser Plugin](#).

- `-DWITH_MSCRT_DEBUG=bool`

Whether to enable Visual Studio CRT memory leak tracing. The default is `OFF`.

- `-DWITH_NUMA=bool`

Explicitly set the NUMA memory allocation policy. `CMake` sets the default `WITH_NUMA` value based on whether the current platform has `NUMA` support. For platforms without `NUMA` support, `CMake` behaves as follows:

- With no `NUMA` option (the normal case), `CMake` continues normally, producing only this warning: `NUMA library missing or required version not available`
- With `-DWITH_NUMA=ON`, `CMake` aborts with this error: `NUMA library missing or required version not available`

This option was added in MySQL 5.7.17.

- `-DWITH_PROTOBUF=protobuf_type`

Which Protocol Buffers package to use. `protobuf_type` can be one of the following values:

- `bundled`: Use the package bundled with the distribution. This is the default.
- `system`: Use the package installed on the system.

Other values are ignored, with a fallback to `bundled`.

This option was added in MySQL 5.7.12.

- `-DWITH_RAPID=bool`

Whether to build the rapid development cycle plugins. When enabled, a `rapid` directory is created in the build tree containing these plugins. When disabled, no `rapid` directory is created in the build tree. The default is `ON`, unless the `rapid` directory is removed from the source tree, in which case the default becomes `OFF`. This option was added in MySQL 5.7.12.

- `-DWITH_SSL={ssl_type|path_name}`

The type of SSL support to include or the path name to the OpenSSL installation to use.

- `ssl_type` can be one of the following values:
 - `yes`: Use the system OpenSSL library if present, else the library bundled with the distribution.
 - `bundled`: Use the SSL library bundled with the distribution. This is the default.
 - `system`: Use the system OpenSSL library.
- `path_name` is the path name to the OpenSSL installation to use. This can be preferable to using the `ssl_type` value of `system` because it can prevent `CMake` from detecting and using an older or incorrect OpenSSL version installed on the system. (Another permitted way to do the same thing is to set `WITH_SSL` to `system` and set the `CMAKE_PREFIX_PATH` option to `path_name`.)

For information about using SSL support, see [Using Encrypted Connections](#).

- `-DWITH_SYSTEMD=bool`

Whether to enable installation of systemd support files. By default, this option is disabled. When enabled, systemd support files are installed, and scripts such as `mysqld_safe` and the System

V initialization script are not installed. On platforms where systemd is not available, enabling `WITH_SYSTEMD` results in an error from CMake.

For more information about using systemd, see [Managing MySQL Server with systemd](#). That section also includes information about specifying options previously specified in `[mysqld_safe]` option groups. Because `mysqld_safe` is not installed when systemd is used, such options must be specified another way.

- `-DWITH_TEST_TRACE_PLUGIN=bool`

Whether to build the test protocol trace client plugin (see [Using the Test Protocol Trace Plugin](#)). By default, this option is disabled. Enabling this option has no effect unless the `WITH_CLIENT_PROTOCOL_TRACING` option is enabled. If MySQL is configured with both options enabled, the `libmysqlclient` client library is built with the test protocol trace plugin built in, and all the standard MySQL clients load the plugin. However, even when the test plugin is enabled, it has no effect by default. Control over the plugin is afforded using environment variables; see [Using the Test Protocol Trace Plugin](#).

Note

Do *not* enable the `WITH_TEST_TRACE_PLUGIN` option if you want to use your own protocol trace plugins because only one such plugin can be loaded at a time and an error occurs for attempts to load a second one. If you have already built MySQL with the test protocol trace plugin enabled to see how it works, you must rebuild MySQL without it before you can use your own plugins.

For information about writing trace plugins, see [Writing Protocol Trace Plugins](#).

- `-DWITH_UBSAN=bool`

Whether to enable the Undefined Behavior Sanitizer, for compilers that support it. The default is off.

- `-DWITH_UNIT_TESTS={ON|OFF}`

If enabled, compile MySQL with unit tests. The default is ON unless the server is not being compiled.

- `-DWITH_UNIXODBC=1`

Enables unixODBC support, for Connector/ODBC.

- `-DWITH_VALGRIND=bool`

Whether to compile in the Valgrind header files, which exposes the Valgrind API to MySQL code. The default is `OFF`.

To generate a Valgrind-aware debug build, `-DWITH_VALGRIND=1` normally is combined with `-DWITH_DEBUG=1`. See [Building Debug Configurations](#).

- `-DWITH_ZLIB=zlib_type`

Some features require that the server be built with compression library support, such as the `COMPRESS()` and `UNCOMPRESS()` functions, and compression of the client/server protocol. The `WITH_ZLIB` indicates the source of `zlib` support:

- `bundled`: Use the `zlib` library bundled with the distribution. This is the default.
- `system`: Use the system `zlib` library.

- `-DWITHOUT_SERVER=bool`

Whether to build without the MySQL server. The default is `OFF`, which does build the server.

Compiler Flags

- `-DCMAKE_C_FLAGS="flags"`

Flags for the C Compiler.

- `-DCMAKE_CXX_FLAGS="flags"`

Flags for the C++ Compiler.

- `-DWITH_DEFAULT_COMPILER_OPTIONS=bool`

Whether to use the flags from `cmake/build_configurations/compiler_options.cmake`.

Note

All optimization flags were carefully chosen and tested by the MySQL build team. Overriding them can lead to unexpected results and is done at your own risk.

- `-DSUNPRO_CXX_LIBRARY="lib_name"`

Enable linking against `libCstd` instead of `stlport4` on Solaris 10 or later. This works only for client code because the server depends on C++98. Example usage:

```
cmake -DWITHOUT_SERVER=1 -DSUNPRO_CXX_LIBRARY=Cstd
```

To specify your own C and C++ compiler flags, for flags that do not affect optimization, use the `CMAKE_C_FLAGS` and `CMAKE_CXX_FLAGS` CMake options.

When providing your own compiler flags, you might want to specify `CMAKE_BUILD_TYPE` as well.

For example, to create a 32-bit release build on a 64-bit Linux machine, do this:

```
shell> mkdir bld
shell> cd bld
shell> cmake .. -DCMAKE_C_FLAGS=-m32 \
             -DCMAKE_CXX_FLAGS=-m32 \
             -DCMAKE_BUILD_TYPE=RelWithDebInfo
```

If you set flags that affect optimization (`-Onumber`), you must set the `CMAKE_C_FLAGS_build_type` and/or `CMAKE_CXX_FLAGS_build_type` options, where `build_type` corresponds to the `CMAKE_BUILD_TYPE` value. To specify a different optimization for the default build type (`RelWithDebInfo`) set the `CMAKE_C_FLAGS_RELWITHDEBINFO` and `CMAKE_CXX_FLAGS_RELWITHDEBINFO` options. For example, to compile on Linux with `-O3` and with debug symbols, do this:

```
shell> cmake .. -DCMAKE_C_FLAGS_RELWITHDEBINFO="-O3 -g" \
             -DCMAKE_CXX_FLAGS_RELWITHDEBINFO="-O3 -g"
```

CMake Options for Compiling NDB Cluster

The following options are for use when building NDB Cluster with the NDB Cluster sources; they are not currently supported when using sources from the MySQL 5.6 Server tree.

- `-DMEMCACHED_HOME=dir_name`

Perform the build using the memcached (version 1.6 or later) installed in the system directory indicated by `dir_name`. Files from this installation that are used in the build include the memcached binary, header files, and libraries, as well as the `memcached_utilities` library and the header file `engine_testapp.h`.

You must leave this option unset when building `ndbmemcache` using the bundled `memcached` sources (`WITH_BUNDLED_MEMCACHED` option); in other words, the bundled sources are used by default).

While additional CMake options—such as for SASL authorization and for providing `dtrace` support—are available for use when compiling `memcached` from external sources, these options are currently not enabled for the `memcached` sources bundled with NDB Cluster.

- `-DWITH_BUNDLED_LIBEVENT={ON|OFF}`

Use the `libevent` included in the NDB Cluster sources when building NDB Cluster with `ndbmemcached` support. Enabled by default. `OFF` causes the system's `libevent` to be used instead.

- `-DWITH_BUNDLED_MEMCACHED={ON|OFF}`

Build the `memcached` sources included in the NDB Cluster source tree, then use the resulting `memcached` server when building the `ndbmemcache` engine. In this case, `make install` places the `memcached` binary in the installation `bin` directory, and the `ndbmemcache` engine shared library file `ndb_engine.so` in the installation `lib` directory.

This option is ON by default.

- `-DWITH_CLASSPATH=path`

Sets the classpath for building NDB Cluster Connector for Java. The default is empty. This option is ignored if `-DWITH_NDB_JAVA=OFF` is used.

- `-DWITH_ERROR_INSERT={ON|OFF}`

Enables error injection in the NDB kernel. For testing only; not intended for use in building production binaries. The default is `OFF`.

- `-DWITH_NDBCLUSTER_STORAGE_ENGINE={ON|OFF}`

Build and link in support for the NDB (`NDBCLUSTER`) storage engine in `mysqld`. The default is `ON`.

- `-DWITH_NDBCLUSTER={ON|OFF}`

This is an alias for `WITH_NDBCLUSTER_STORAGE_ENGINE`.

- `-DWITH_NDBMTD={ON|OFF}`

Build the multithreaded data node executable `ndbmtd`. The default is `ON`.

- `-DWITH_NDB_BINLOG={ON|OFF}`

Enable binary logging by default in the `mysqld` built using this option. `ON` by default.

- `-DWITH_NDB_DEBUG={ON|OFF}`

Enable building the debug versions of the NDB Cluster binaries. `OFF` by default.

- `-DWITH_NDB_JAVA={ON|OFF}`

Enable building NDB Cluster with Java support, including `ClusterJ`.

This option is `ON` by default. If you do not wish to compile NDB Cluster with Java support, you must disable it explicitly by specifying `-DWITH_NDB_JAVA=OFF` when running CMake. Otherwise, if Java cannot be found, configuration of the build fails.

- `-DWITH_NDB_PORT=port`

Causes the NDB Cluster management server (`ndb_mgmd`) that is built to use this *port* by default. If this option is unset, the resulting management server tries to use port 1186 by default.

- `-DWITH_NDB_TEST={ON|OFF}`

If enabled, include a set of NDB API test programs. The default is OFF.

Chapter 5 Dealing with Problems Compiling MySQL

The solution to many problems involves reconfiguring. If you do reconfigure, take note of the following:

- If `CMake` is run after it has previously been run, it may use information that was gathered during its previous invocation. This information is stored in `CMakeCache.txt`. When `CMake` starts up, it looks for that file and reads its contents if it exists, on the assumption that the information is still correct. That assumption is invalid when you reconfigure.
- Each time you run `CMake`, you must run `make` again to recompile. However, you may want to remove old object files from previous builds first because they were compiled using different configuration options.

To prevent old object files or configuration information from being used, run the following commands before re-running `CMake`:

On Unix:

```
shell> make clean
shell> rm CMakeCache.txt
```

On Windows:

```
shell> devenv MySQL.sln /clean
shell> del CMakeCache.txt
```

If you build outside of the source tree, remove and recreate your build directory before re-running `CMake`. For instructions on building outside of the source tree, see [How to Build MySQL Server with CMake](#).

On some systems, warnings may occur due to differences in system include files. The following list describes other problems that have been found to occur most often when compiling MySQL:

- To define which C and C++ compilers to use, you can define the `CC` and `CXX` environment variables. For example:

```
shell> CC=gcc
shell> CXX=g++
shell> export CC CXX
```

To specify your own C and C++ compiler flags, use the `CMAKE_C_FLAGS` and `CMAKE_CXX_FLAGS` `CMake` options. See [Compiler Flags](#).

To see what flags you might need to specify, invoke `mysql_config` with the `--cflags` and `--cxxflags` options.

- To see what commands are executed during the compile stage, after using `CMake` to configure MySQL, run `make VERBOSE=1` rather than just `make`.
- If compilation fails, check whether the `MYSQL_MAINTAINER_MODE` option is enabled. This mode causes compiler warnings to become errors, so disabling it may enable compilation to proceed.
- If your compile fails with errors such as any of the following, you must upgrade your version of `make` to GNU `make`:

```
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
```

Or:

```
make: file `Makefile' line 18: Must be a separator (:
```

Or:

```
pthread.h: No such file or directory
```

Solaris and FreeBSD are known to have troublesome [make](#) programs.

GNU [make](#) 3.75 is known to work.

- The `sql_yacc.cc` file is generated from `sql_yacc.yy`. Normally, the build process does not need to create `sql_yacc.cc` because MySQL comes with a pregenerated copy. However, if you do need to re-create it, you might encounter this error:

```
"sql_yacc.yy", line xxx fatal: default action causes potential...
```

This is a sign that your version of [yacc](#) is deficient. You probably need to install a recent version of [bison](#) (the GNU version of [yacc](#)) and use that instead.

Versions of [bison](#) older than 1.75 may report this error:

```
sql_yacc.yy:#####: fatal error: maximum table size (32767) exceeded
```

The maximum table size is not actually exceeded; the error is caused by bugs in older versions of [bison](#).

For information about acquiring or updating tools, see the system requirements in [Chapter 1, Installing MySQL from Source](#).