

Building MySQL from Source

Abstract

This is the Building MySQL from Source extract from the MySQL 5.5 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: 2019-02-16 (revision: 60964)

Table of Contents

Preface and Legal Notices	v
1 Installing MySQL from Source	1
2 Installing MySQL Using a Standard Source Distribution	5
3 Installing MySQL Using a Development Source Tree	11
4 MySQL Source-Configuration Options	13
5 Dealing with Problems Compiling MySQL	27

Preface and Legal Notices

This is the Building MySQL from Source extract from the MySQL 5.5 Reference Manual.

Licensing information—MySQL 5.5. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.5, see the [MySQL 5.5 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 5.5, see the [MySQL 5.5 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 7.2. This product may include third-party software, used under license. If you are using a *Commercial* release of NDB Cluster 7.2, see the [MySQL NDB Cluster 7.2 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 7.2, see the [MySQL NDB Cluster 7.2 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, 2019, 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 <https://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.

Note

This section describes how to build MySQL from source using [CMake](#). Before MySQL 5.5, source builds used the GNU autotools on Unix-like systems. Source builds on Windows used [CMake](#), but the process was different from that described here. For source-building instructions for older versions of MySQL, see the *MySQL 5.1 Reference Manual*. If you are familiar with autotools but not [CMake](#), you might find these transition instructions helpful: [Autotools to CMake Transition Guide](#)

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 `5.5.64`. 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. GCC 4.2.1 or later, Sun Studio 12 or later, Visual Studio 2008 or later, and many current vendor-supplied compilers are known to work.
- 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 5.5, [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](#).

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> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf
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](#)
- [Obtain and Unpack the Distribution](#)
- [Configure the Distribution](#)
- [Build the Distribution](#)
- [Install the Distribution](#)
- [Perform Postinstallation Setup](#)

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 9 2008"
shell> cmake .. -G "Visual Studio 9 2008 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](#)
- [Setting Up a MySQL Git Repository](#)

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         tests
BUILD-CMAKE   debug          libservices  scripts       unittest
client        Docs           man           sql           VERSION
cmake         extra          mysql-test    sql-bench     vio
CMakeLists.txt include        mysys        sql-common    win
cmd-line-utils INSTALL-SOURCE packaging      storage       zlib
config.h.cmake INSTALL-WIN-SOURCE plugin         strings
configure.cmake libmysql       README        support-files
```

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` again 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
```

The `git checkout` command is also used to switch branches. For example, to make MySQL 5.7 the active branch again, you would run `git checkout 5.7`.

7. 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 5.5
~/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](#).

8. 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:

```
cmake . -LH
ccmake .
```

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

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.

The following sections provide more information about `CMake` options.

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

CMake Option Reference

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		5.5.7	
<code>CMAKE_BUILD_TYPE</code>	Type of build to produce	<code>RelWithDebInfo</code>	5.5.7	
<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>	5.5.8	
<code>COMPILATION_COMMENT</code>	Comment about compilation environment		5.5.7	

CMake Option Reference

Formats	Description	Default	Introduced	Removed
CPACK_MONOLITHIC_INSTALL	Whether package build produces single file	OFF	5.5.7	
DEFAULT_CHARSET	The default server character set	latin1	5.5.7	
DEFAULT_COLLATION	The default server collation	latin1_swedish_ci	5.5.7	
ENABLED_LOCAL_INFILE	Whether to enable LOCAL for LOAD DATA	OFF	5.5.7	
ENABLED_PROFILING	Whether to enable query profiling code	ON	5.5.7	
ENABLE_DEBUG_SYNC	Whether to enable Debug Sync support	ON	5.5.7	5.5.55
ENABLE_DOWNLOADS	Whether to download optional files	OFF	5.5.7	
ENABLE_DTRACE	Whether to include DTrace support		5.5.7	
ENABLE_GCOV	Whether to include gcov support		5.5.14	
IGNORE_AIO_CHECK	With - DBUILD_CONFIG=mysql_release, ignore libaio check	OFF	5.5.9	
INSTALL_BINDIR	User executables directory	PREFIX/bin	5.5.7	
INSTALL_DOCDIR	Documentation directory	PREFIX/docs	5.5.7	
INSTALL_DOCREADMEDIR	README file directory	PREFIX	5.5.7	
INSTALL_INCLUDEDIR	Header file directory	PREFIX/include	5.5.7	
INSTALL_INFODIR	Info file directory	PREFIX/docs	5.5.7	
INSTALL_LAYOUT	Select predefined installation layout	STANDALONE	5.5.7	
INSTALL_LIBDIR	Library file directory	PREFIX/lib	5.5.7	
INSTALL_MANDIR	Manual page directory	PREFIX/man	5.5.7	
INSTALL_MYSQLSHAREDIR	Shared data directory	PREFIX/share	5.5.7	
INSTALL_MYSQLTESTDIR	mysql-test directory	PREFIX/mysql-test	5.5.7	
INSTALL_PLUGINDIR	Plugin directory	PREFIX/lib/plugin	5.5.7	
INSTALL_SBINDIR	Server executable directory	PREFIX/bin	5.5.7	
INSTALL_SCRIPTDIR	Scripts directory	PREFIX/scripts	5.5.7	
INSTALL_SECURE_FILE_PRIV	secure_file_priv default value	platform specific	5.5.53	
INSTALL_SECURE_FILE_PRIV_LIBMYSQLD	secure_file_priv default value for libmysqld		5.5.53	
INSTALL_SHAREDIR	aclocal/mysql.m4 installation directory	PREFIX/share	5.5.7	

Formats	Description	Default	Introduced	Removed
INSTALL_SQLBENCHDIR	sql-bench directory	PREFIX	5.5.7	
INSTALL_SUPPORTFILESDIR	Extra support files directory	PREFIX/support-files	5.5.7	
MEMCACHED_HOME	Path to memcached	[none]	5.5.16-ndb-7.2.2	
MYSQL_DATADIR	Data directory		5.5.7	
MYSQL_MAINTAINER_MODE	Whether to enable MySQL maintainer-specific development environment	OFF	5.5.7	
MYSQL_PROJECT_NAME	Windows/OS X project name	MySQL	5.5.21	
MYSQL_TCP_PORT	TCP/IP port number	3306	5.5.7	
MYSQL_UNIX_ADDR	Unix socket file	/tmp/mysql.sock	5.5.7	
ODBC_INCLUDES	ODBC includes directory			
ODBC_LIB_DIR	ODBC library directory			
REPRODUCIBLE_BUILD	Take extra care to create a build result independent of build location and time		5.5.57	
SYSCONFDIR	Option file directory		5.5.7	
TMPDIR	tmpdir default value		5.5.36	
WITHOUT_xxx_STORAGE_ENGINE	Exclude storage engine xxx from build		5.5.7	
WITH_ASAN	Enable AddressSanitizer	OFF	5.5.35	
WITH_BUNDLED_LIBEVENT	Use bundled libevent when building ndbmemcache	ON	5.5.16-ndb-7.2.2	
WITH_BUNDLED_MEMCACHED	Use bundled memcached when building ndbmemcache	ON	5.5.16-ndb-7.2.2	
WITH_CLASSPATH	Classpath to use when building MySQL Cluster Connector for Java. Default is an empty string.			
WITH_DEBUG	Whether to include debugging support	OFF	5.5.7	
WITH_EMBEDDED_SERVER	Whether to build embedded server	OFF	5.5.7	
WITH_EMBEDDED_SHARED_LIBRARY	Whether to build a shared embedded server library	OFF	5.5.37	
WITH_ERROR_INSERT	Enable error injection in the NDB storage engine. Should not be used for building binaries intended for production.	OFF		
WITH_EXTRA_CHARSETS	Which extra character sets to include	all	5.5.7	

Formats	Description	Default	Introduced	Removed
WITH_LIBEDIT	Use bundled libedit library	ON	5.5.7	
WITH_LIBWRAP	Whether to include libwrap (TCP wrappers) support	OFF	5.5.7	
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_NDBMTD	Build multithreaded data node.	ON		
WITH_NDB_BINLOG	Enable binary logging by default by mysqld.	ON		
WITH_NDB_DEBUG	Produce a debug build for testing or troubleshooting.	OFF		
WITH_NDB_JAVA	Enable building of Java and ClusterJ support. Enabled by default. Supported in MySQL Cluster only.	ON	5.5.27-ndb-7.2.9	
WITH_NDB_PORT	Default port used by a management server built with this option. If this option was not used to build it, the management server's default port is 1186.	[none]		
WITH_NDB_TEST	Include NDB API test programs.	OFF		
WITH_READLINE	Use bundled readline library	OFF	5.5.7	
WITH_SSL	Type of SSL support	bundled	5.5.7	
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	5.5.6	
WITH_ZLIB	Type of zlib support	bundled	5.5.7	
WITH_xxx_STORAGE_ENGINE	Compile storage engine xxx statically into server		5.5.7	

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:

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

- `-DINSTALL_LIBDIR=dir_name`

Where to install library files.

- `-DINSTALL_MANDIR=dir_name`

Where to install manual pages.

- `-DINSTALL_MYSQLSHAREDIR=dir_name`

Where to install shared data files.

- `-DINSTALL_MYSQLTESTDIR=dir_name`

Where to install the `mysql-test` directory. As of MySQL 5.5.32, to suppress installation of this directory, explicitly set the option to the empty value (`-DINSTALL_MYSQLTESTDIR=`).

- `-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](#).

This option was added in MySQL 5.5.53. To set the value for the `libmysqld` embedded server, use `INSTALL_SECURE_FILE_PRIV_EMBEDDED`.

- `-DINSTALL_SECURE_FILE_PRIV_EMBEDDEDIR=dir_name`

The default value for the `secure_file_priv` system variable, for the `libmysqld` embedded server. This option was added in MySQL 5.5.53.

- `-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=`).

- `-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.

- `-DTMPDIR=dir_name`

The default location to use for the `tmpdir` system variable. If unspecified, the value defaults to `P_tmpdir` in `<stdio.h>`. This option was added in MySQL 5.6.16.

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`, `NDBCLUSTER (NDB)`, `PARTITION` (partitioning support), and `PERFSCHEMA` (Performance Schema). Examples:

```
-DWITH_ARCHIVE_STORAGE_ENGINE=1
-DWITH_BLACKHOLE_STORAGE_ENGINE=1
-DWITH_PERFSCHEMA_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 MySQL NDB Cluster 7.2 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.

To exclude a storage engine from the build, use `-DWITHOUT_engine_STORAGE_ENGINE=1`. 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`, `eucjpms`, `euckr`, `gb2312`, `gbk`, `geostd8`, `greek`, `hebrew`, `hp8`, `keybcs2`, `koi8r`, `koi8u`, `latin1`, `latin2`, `latin5`, `latin7`, `macce`, `macroman`, `sjis`, `swe7`, `tis620`, `ucs2`, `ujis`, `utf8`, `utf8mb4`, `utf16`, `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.

- `-DENABLE_DEBUG_SYNC=bool`

Note

As of MySQL 5.5.55, `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.

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](#)

- `-DENABLE_GCOV=bool`

Whether to include gcov support (Linux only).

- `-DISABLED_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](#).

- `-DISABLED_PROFILING=bool`

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

- `-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`.

- `-DMYSQL_MAINTAINER_MODE=bool`

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

- `-DMYSQL_PROJECT_NAME=name`

For Windows or macOS, the project name to incorporate into the project file name. This option was added in MySQL 5.5.21.

- `-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.

- `-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 5.5.55.

- `-DWITH_ASAN=bool`

Whether to enable AddressSanitizer, for compilers that support it. The default is off. This option was added in MySQL 5.5.35.

- `-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.

As of MySQL 5.5.55, 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_EMBEDDED_SERVER=bool`

Whether to build the `libmysqld` embedded server library.

- `-DWITH_EMBEDDED_SHARED_LIBRARY=bool`

Whether to build a shared `libmysqld` embedded server library. This option was added in MySQL 5.5.37.

- `-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_LIBEDIT=bool`

Whether to use the `libedit` library bundled with the distribution.

- `-DWITH_LIBWRAP=bool`

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

- `-DWITH_READLINE=bool`

Whether to use the `readline` library bundled with the distribution.

- `-DWITH_SSL=ssl_type`

The type of SSL support to include, if any:

- `no`: No SSL support. This is the default before MySQL 5.5.56. As of 5.5.56, this is no longer a permitted value and the default is `bundled`.
- `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 as of MySQL 5.5.56.
- `system`: Use the system OpenSSL library.

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

- `-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.

Compiler Flags

- `-DCMAKE_C_FLAGS="flags"`

Flags for the C Compiler.

- `-DCMAKE_CXX_FLAGS="flags"`

Flags for the C++ Compiler.

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:

```
mkdir bld
cd bld
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:

```
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 MySQL NDB Cluster 7.2 or later. These options are supported only with the MySQL NDB Cluster 7.2 and later NDB Cluster sources; they are not supported when using sources from the MySQL 5.5 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).

This option was added in MySQL NDB Cluster 7.2.2.

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 (MySQL NDB Cluster 7.2.2 and later). 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 (MySQL NDB Cluster 7.2.3 and later), 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. In MySQL NDB Cluster 7.2.9 and later, 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 was added in MySQL NDB Cluster 7.2.9, and 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` option.

- 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](#).