MySQL Connector/C Developer Guide

Abstract

This manual describes how to install and configure MySQL Connector/C, the C interface for communicating with MySQL servers, and how to use it to develop database applications.

For notes detailing the changes in each release of Connector/C, see MySQL Connector/C Release Notes.

For legal information, see the Legal Notices.

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

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

Document generated on: 2017-05-08 (revision: 52054)

Table of Contents

Preface and Legal Notices

This manual describes how to build and test MySQL Connector/C, the C interface for communicating with MySQL servers.

Legal Notices

Chapter 1 Introduction to Connector/C

Connector/C is a client library that implements the C API for client/server communication. It is a standalone replacement for the MySQL client library shipped with MySQL Server distributions. See MySQL C API Implementations.

To see which platforms are supported, visit MySQL Connector/C downloads.

Reasons to use Connector/C:

- If you need only the client library, Connector/C provides everything required. There is no need to compile or install the MySQL Server package, which is much larger.
- Connector/C does not rely on the MySQL Server release cycle. Bug fixes and new features can be distributed independently of MySQL Server releases.

For documentation of the C API implemented by Connector/C, see MySQL C API.

For notes detailing the changes in each release of Connector/C, see MySQL Connector/C Release Notes.

The following discussion covers these topics:

- Connector/C versions and supported platforms
- Connector/C distribution contents
- Obtaining and installing Connector/C
- Building client programs that use Connector/C

Chapter 2 Connector/C Versions

These versions of Connector/C are available:

- Connector/C 6.1: Based on the C API parts of current MySQL sources and kept up to date with those sources.
- Connector/C 6.0: Created originally from a branch of the MySQL source tree, but now out of date with respect to C API changes in that tree.

Consequently, Connector/C 6.1 is preferred over 6.0. Connector/C 6.1 provides these features not present in 6.0:

- Support for the pluggable authentication framework that enables implementation of authentication methods as plugins. This framework can be used for MySQL native authentication as well as external authentication methods. See Pluggable Authentication.
- Client-side support for the SHA-256, PAM, and Windows native authentication plugins. See SHA-256 Pluggable Authentication, PAM Pluggable Authentication, and Windows Native Pluggable Authentication.

  The older Connector/C 6.0 can connect only to accounts that use native MySQL passwords. If a client program attempts to connect to an account that requires a different authentication method, an “Access denied for user” error occurs.
- Support for connecting to accounts that have expired passwords. See Password Expiration and Sandbox Mode.
- Support for prepared `CALL` statements. This enables client programs to handle stored procedures that produce multiple result sets and to obtain the final value of `OUT` and `INOUT` procedure parameters. See C API Support for Prepared CALL Statements.
- Support for connecting over IPv6. See IPv6 Support.
- Support for binding client programs to a specific IP address at connect time. See mysql_options().
- Support for specifying connection attributes to pass to the server at connect time. See mysql_options(), and mysql_options4().

Chapter 3 Connector/C Distribution Contents

Connector/C 6.1 distributions contain the header, library, and utility files necessary to build MySQL client applications that communicate with MySQL Server using the C API.

Distributions are available in binary and source formats. A binary distribution contains the header, library, and utility components discussed following, compiled and ready for use in writing client programs. A source distribution contains the source files required to produce the same headers, libraries, and utilities included in a binary distribution, but you compile them yourself.

Connector/C distributions include these components:

- A set of `.h` header files that C applications include at compile time. These files are located in the `include` directory.
- Static and dynamic libraries that C applications link to at link time. These libraries are located in the `lib` directory. The library names depend on the library type and platform for which a distribution is built:
  - On Unix (and Unix-like) sytems, the static library is `libmysqlclient.a`. The dynamic library is `libmysqlclient.so` on most Unix systems and `libmysqlclient.dylib` on OS X.
  - On Windows, the static library is `mysqlclient.lib` and the dynamic library is `libmysql.dll`. Windows distributions also include `libmysql.lib`, a static import library needed for using the dynamic library.

    Windows distributions also include a set of debug libraries. These have the same names as the nondebug libraries, but are located in the `lib/debug` library.
- Utilities. Connector/C 6.1 includes the following utilities, located in the `bin` directory. They are the same as in MySQL Server distributions:
  - `mysql_config` displays flags needed to compile C applications to use Connector/C. This utility is a shell script and is included only for Unix systems. See `mysql_config` — Display Options for Compiling Clients.
  - `my_print_defaults` displays the options that are present in option groups within option files. See `my_print_defaults` — Display Options from Option Files.
  - `perror` displays error messages corresponding to error codes. See `perror` — Explain Error Codes.

Connector/C 6.0 distributions are similar to 6.1 distributions, with these exceptions:

- Debug libraries, `my_print_defaults`, and `perror` are not included.
- `mysql_config` is an executable program that is available on all platforms. However, this version of `mysql_config` is more limited than the shell script version in the types of information it can display.

Chapter 4 Installing Connector/C

Table of Contents

Connector/C distributions are available in binary and source formats. Binary distributions are available in native format for many platforms, such as RPM packages for Linux, DMG packages for OS X, and PKG packages for Solaris. Distributions are also available in more generic formats such as Zip archives or compressed `tar` files.

To obtain a distribution, visit Connector/C downloads.

After installing Connector/C, you may need to take additional steps to enable your compiler or linker to find the C API header files and libraries. See Section 4.3, “Postinstallation Steps”.

4.1 Installing Connector/C from a Binary Distribution

Installers in native package formats are available for many Unix and Unix-like systems, and for Windows. Alternatively, you can install using a distribution in a more generic format such as a Zip archive or compressed `tar` file.

You may need to have `root` or administrator privileges to perform the installation operation.

Installing Connector/C on Unix Using Compressed `tar` Files

On Unix and Unix-like systems, a generic Connector/C binary distribution is packaged as a compressed `tar` file, denoted here as *`PACKAGE`*`.tar.gz`. To install a distribution file, unpack it in the intended installation directory using this command:

```
shell> tar zxvf PACKAGE.tar.gz
```

Installing Connector/C on Microsoft Windows

> **Important**
>
> MySQL Connector/C Community requires the Visual C++ Redistributable for Visual Studio 2015 to work on Windows platforms; install it before installing MySQL Connector/C Community.

The simplest and recommended method for installing Connector/C on Windows platforms is to download *MySQL Installer* and let it install and configure all the MySQL products on your system. See MySQL Installer for Windows for details. Those who are not using the *MySQL Installer* can choose between two binary distributions:

- Windows MSI Installer (`.msi` file): To use the MSI Installer, launch it and follow the prompts in the screens it presents to install Connector/C in the location of your choosing.

- Zip archive without installer (`.zip` file): To use a Zip archive, unpack it in the intended installation directory using `WinZip` or another tool that can read `.zip` files.

Installing Connector/C on OS X Using DMG Packages

A OS X native package installer is provided as a DMG (disk image) file. To install a DMG package, double-click the image file, then follow the prompts.

By default, the DMG package installs Connector/C under `/usr/local`, into a dedicated directory that does not conflict with the one used by MySQL Server DMG packages.

Installing Connector/C on Linux Using RPM Packages

There are two Linux RPM packages for Connector/C. Install one or both, depending on the capabilities you require:

- The `shared` RPM contains the shared client library. Install this RPM if you intend to compile or run C API applications that depend on the shared client library.
- The `devel` RPM contains the header files and the static client library. Install this RPM if you intend to compile C API applications.

RPM packages for Connector/C do not include the `perror` or `my_print_defaults` utilities.

A Linux RPM package is provided as a file with an `.rpm` suffix, denoted here as *`PACKAGE`*`.rpm`. To install a given RPM package, use this command:

```
shell> rpm -i PACKAGE.rpm
```

RPM provides a feature to verify the integrity and authenticity of packages before installing them. To learn more about this feature, see Verifying Package Integrity Using MD5 Checksums or GnuPG.

Installing Connector/C on Solaris Using PKG Packages

A Solaris PKG package is provided as a file with a `.pkg.gz` suffix, denoted here as *`PACKAGE`*`.pkg.gz`. To install a PKG package, uncompress it:

```
shell> gunzip PACKAGE.pkg.gz
```

Uncompressing *`PACKAGE`*`.pkg.gz` produces *`PACKAGE`*`.pkg`. Then use `pkgadd` and follow the onscreen prompts:

```
shell> pkgadd -d PACKAGE.pkg
```

By default, the PKG package installs Connector/C under the root path `/opt/mysql`, into a dedicated directory that does not conflict with the one used by MySQL Server PKG packages. You can change only the installation root path using `pkgadd`, which can be used to install MySQL in a different Solaris zone. If you need to install in a specific directory, use a binary `tar` file distribution.

4.2 Installing Connector/C from Source

A Connector/C source distribution is packaged as a compressed `tar` file, Zip archive, or RPM package, denoted here as *`PACKAGE`*`.tar.gz`, *`PACKAGE`*`.zip`, or *`PACKAGE`*`.src.rpm`. A source distribution in `tar` file or Zip archive format can be used on any supported platform. An RPM package source distribution is intended for RPM-based systems such as Linux.

To unpack a compressed `tar` file, use this command in the intended installation directory:

```
shell> tar zxvf PACKAGE.tar.gz
```

After unpacking the distribution, build it using the appropriate instructions for your platform later in this section.

To unpack a Zip archive, use `WinZip` or another tool that can read `.zip` files. After unpacking the distribution, build it using the appropriate instructions for your platform later in this section.

To install an RPM package, use this command to create binary RPM packages that you can install. If you do not have `rpmbuild`, use `rpm` instead.

```
shell> rpmbuild --rebuild --clean PACKAGE.src.rpm
```

The command should produce binary `shared` and `devel` RPM packages and indicate where it placed them. You can install these packages using the instructions in Section 4.1, “Installing Connector/C from a Binary Distribution”.

4.2.1 Installing Connector/C from Source on Unix and Unix-Like Systems

If the native compiler toolset for the target platform is available (for example, SunStudio for Solaris), you can use that for compilation. Alternatively, the GNU toolset can be used on all platforms.

You also need `CMake` 2.6 or newer, which is available from cmake.org.

To build and install the source distribution, use the following procedure:

1. Change location to the top-level directory of the source distribution.
2. Generate the `Makefile`:

   ```
   shell> cmake -G "Unix Makefiles"
   ```

   Or, for a Debug build:

   ```
   shell> cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=Debug
   ```

   By default, the installation location for Connector/C is `/usr/local/mysql`. To change this location, use the `CMAKE_INSTALL_PREFIX` option to specify a different directory when generating the `Makefile`. For example:

   ```
   shell> cmake -G "Unix Makefiles" -DCMAKE_INSTALL_PREFIX=/opt/local/mysql
   ```

   For other `CMake` options that you might find useful, see Other Connector/C Build Options.
3. Build the project:

   ```
   shell> make
   ```
4. As `root`, install the Connector/C headers, libraries, and utilities:

   ```
   ```

```
root-shell> make install
```

4.2.2 Installing Connector/C from Source on Microsoft Windows

To build Connector/C on Microsoft Windows, Visual Studio 8 or 9 is recommended. The Express Edition of Visual Studio and other compilers might work, but are untested.

You also need `CMake` 2.6 or newer, which is available from cmake.org.

To build and install the source distribution, use the following procedure:

1. Set the environment variables for the Visual Studio toolchain. Visual Studio includes a batch file to set these for you, and installs a shortcut in the **Start** menu to open a command prompt with these variables set.

2. Change location to the top-level directory of the source distribution.

3. Generate the `Makefile` by entering the following command in a command-prompt window:

   ```
   shell> cmake -G "Visual Studio 9 2008"
   ```

   For other `CMake` options that you might find useful, see Other Connector/C Build Options.

   The result of the `cmake` command is a project (solution) file, `libmysql.sln`, that you can open with Visual Studio. Alternatively, build from the command line with either of these commands:

   ```
   shell> devenv.com libmysql.sln /build Release
   ```

   ```
   shell> devenv.com libmysql.sln /build RelWithDebInfo
   ```

   For other versions of Visual Studio or for an `nmake`-based build, use the following command to check which generators can be specified with the `-G` option:

   ```
   shell> cmake --help
   ```

   To compile a Debug build, you must set the `CMake` build type so the correct external library versions are used, then compile using the `Debug` solution configuration:

   ```
   shell> cmake -G "Visual Studio 9 2008" -DCMAKE_BUILD_TYPE=Debug
   shell> devenv.com libmysql.sln /build Debug
   ```

   A normal build builds the C API libraries for the `lib` directory. A Debug build additionally builds debug libraries for the `lib/debug` directory. You must use the debug libraries to compile clients built using the debug C runtime.

4. Use the install operation provided by your development environment to install the Connector/C headers, libraries, and utilities. You can also use this `CMake` command:

   ```
   shell> cmake --build . --target INSTALL --config RelWithDebInfo
   ```

4.2.3 Other Connector/C Build Options

The following tables show other options that can be used when building Connector/C from source.

Table 4.1 Build Options for Connector/C 6.1

Build Option	Description
-DWITH_SSL=system	Enable dynamic linking to the system OpenSSL library.
-DWITH_ZLIB=system	Enable dynamic linking to the system Zlib library.

Table 4.2 Build Options for Connector/C 6.0

Build Option	Description
-DWITH_OPENSSL=1	Enable dynamic linking to the system OpenSSL library.
-DWITH_EXTERNAL_ZLIB=1	Enable dynamic linking to the system Zlib library.

4.3 Postinstallation Steps

Connector/C binary `.tar.gz` and `.zip` packages unpack into a directory with a name such as `mysql-connector-c-6.1.0-linux-rhel5-x86-64bit`. If you want to work with a simpler name, rename the directory. On Unix, an alternative is to create a symbolic link with a simpler name:

```
shell> ln -s mysql-connector-c-6.1.0-linux-rhel5-x86-64bit connector-c
```

When you build C applications that use Connector/C, if the compiler or linker have trouble finding the Connector/C header files or libraries, you may need to adjust your development tools or runtime environment. See Building C API Client Programs, and Running C API Client Programs.

4.4 Testing Connector/C

If you build Connector/C from source, you can use the instructions in this section to test it. The details of the test procedure depend on your Connector/C version, except that a running MySQL server instance must be available regardless of version.

To test Connector/C 6.1:

Use the `mysql_client_test` utility in the `tests` directory. For information about the MySQL Test Framework, see the manual available at http://dev.mysql.com/doc/mysqltest/2.0/en/.

To test Connector/C 6.0:

Use the `ctest` command. Before you run the test suite, specify the following environment variables:

- `MYSQL_TEST_HOST`: The host where the MySQL server is running (default `localhost`)
- `MYSQL_TEST_USER`: The user name of the MySQL account to use
- `MYSQL_TEST_PASSWD`: The password of the MySQL account to use
- `MYSQL_TEST_PORT`: The TCP/IP port to connect to
- `MYSQL_TEST_SOCKET`: The socket file to connect to
- `MYSQL_TEST_DB`: The default database to use (default `test`)

To run the test suite, execute `ctest` from the command line:

```
shell> ctest
```

Chapter 5 Building Connector/C Applications

To build C applications that use Connector/C, the connector must be installed. If you need to do that first, see Chapter 4, *Installing Connector/C*.

For instructions on building Connector/C applications, see Building C API Client Programs. To enable your compiler to find the header and library files under the directory where you installed Connector/C, specify the appropriate compile-time options, as indicated in that section.

For binary distributions, the `docs/INFO_BIN` file contains information about the build environment used to compile Connector/C. This may help you select compatible tools for compiling client applications.