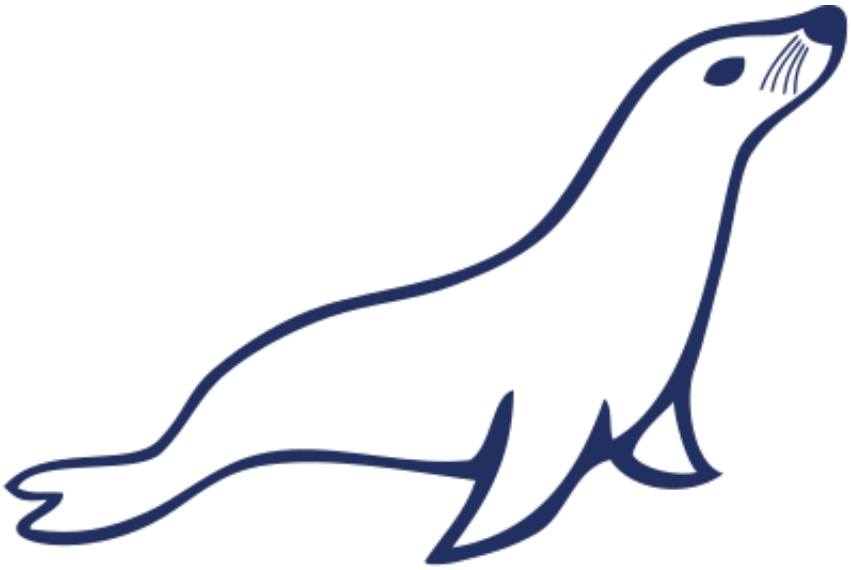

MariaDB ColumnStore C++ API Building Documentation

Release 1.1.1-a6aa245

MariaDB Corporation

November 06, 2017

CONTENTS

CHAPTER

ONE

LICENSING

1.1 Documentation Content

The mcsapi documentation is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

1.2 MariaDB ColumnStore C++ API

The MariaDB ColumnStore C++ API (mcsapi) is licensed under the GNU Lesser General Public License, version 2.1.

CHAPTER

TWO

VERSION HISTORY

This is a version history of C++ API interface changes. It does not include internal fixes and changes.

Version	Changes
1.1.1	• Add `ColumnStoreBulkInsert::isActive()` • Make `ColumnStoreBulkInsert::rollback()` fail without error • Add `ColumnStoreBulkInsert::resetRow()` • `ColumnStoreDateTime::ColumnStoreDateTime()` now uses uint32_t for every parameter • `ColumnStoreSystemCatalog` now uses const for the sub-class strings
1.1.0β	• First beta release

CHAPTER

THREE

BUILDING LIBMCSAPI

Note: CentOS 6 is not currently supported and it is not expected that the API will build on this platform.

libmcsapi uses CMake which is a portable cross-platform build system.

3.1 Pre-requisites

You need the development pacakges for **libuv**, **libxml2** and **snappy** to build mcsapi.

To build the documentation you need **python-sphinx** and **python-sphinx-latex**.

The test suite will use **cppcheck** for additional static code analysis checks if it is installed.

3.1.1 Ubuntu 16.04 (Xenial)

For the main build you need:

```
sudo apt-get install cmake g++ libuv1-dev libxml2-dev libsnappy-dev pkg-config
```

For the documentation:

```
sudo apt-get install python-sphinx texlive-latex-recommended texlive-latex-extra latexmk
```

For the test suite:

```
sudo apt-get install libgtest-dev cppcheck
cd /usr/src/gtest
sudo cmake . -DCMAKE_BUILD_TYPE=RELEASE -DBUILD_SHARED_LIBS=ON
sudo make
sudo mv libg* /usr/lib/
```

3.1.2 Debian 8 (Jessie)

Debian Jessie will only compile if the latest CLang is along with LLVM's libc++, it also requires packages that are not in the main repositories. First of all you need Debian's Jessie backports repository enabled, edit the file `/etc/apt/sources/list` and add the following line:

```
deb http://httpredir.debian.org/debian jessie-backports main contrib non-free
```

Then install the following:

```
sudo apt-get install cmake g++ libuv1-dev libxml2-dev libsnappy-dev pkg-config clang-3.8 libc++-dev
```

Now set the following environment variables so that CLang is used to compile:

```
export CC=clang-3.8
export CXX=clang++-3.8
export CXXFLAGS=-stdlib=libc++
```

For the documentation:

```
sudo apt-get install python-sphinx texlive-latex-recommended texlive-latex-extra latexmk python-pip
sudo pip install python-sphinx
```

For the test suite make sure you still have the exported environment variables above and then do the following in a directory separate from the API:

```
git clone https://github.com/google/googletest
cd googletest
cmake . -DCMAKE_BUILD_TYPE=RELEASE -DBUILD_SHARED_LIBS=ON
make
sudo make install
```

3.1.3 CentOS 7

For the main build you need the following, the devtoolset is because GCC5 minimum is required for full C++11 support:

```
sudo yum install epel-release
sudo yum install cmake libuv-devel libxml2-devel snappy-devel
sudo yum install centos-release-scl
sudo yum install devtoolset-4-gcc*
scl enable devtoolset-4 bash
```

For the documentation:

```
sudo yum install python-sphinx texlive-scheme-full latexmk
```

For the test suite:

```
sudo yum install gtest-devel cppcheck
```

3.1.4 SUSE Enterprise Linux 12

For the main build you need GCC5 minimum. For this example we are using GCC6, you will need the SDK and Toolchain modules enabled in Yast first:

```
sudo zypper install gcc6 gcc6-c++ cmake libxml2-devel snappy-devel

export CC=/usr/bin/gcc-6
export CXX=/usr/bin/g++-6
```

Then in a directory separate from the API:

```
git clone https://github.com/libuv/libuv
cd libuv
./autogen.sh
```

```
./configure
make
sudo make install
```

Unfortunately it is not possible to build the documentation in SUSE Enterprise Linux 12 due to missing LaTeX dependencies.

For the test suite do the following in a directory separate from the API:

```
sudo zypper ar -f http://download.opensuse.org/repositories/devel:/tools/SLE_12_SP3/devel:tools.repo
sudo zypper install cppcheck
git clone https://github.com/google/googletest
cmake . -DCMAKE_BUILD_TYPE=RELEASE -DBUILD_SHARED_LIBS=ON
make
sudo make install
```

3.2 CMake Options

Several options are available when execution CMake by using the following command line:

```
cmake -D<Variable>=<Value>
```

Alternatively you can use one of the CMake GUIs to set the options.

The options are as follows:

Option	Default	Definition
CMAKE_INSTALL_PREFIX	(Platform dependent)	Where to install libmcsapi
CMAKE_BUILD_TYPE	RELWITHDEBINFO	The type of build (Debug, Release or RelWithDebInfo)
TEST_RUNNER	OFF	Build the test suite
BUILD_DOCS	OFF	Build the PDF documentation
RPM	OFF	Build a RPM (and the OS name for the package)
DEB	OFF	Build a DEB (and the OS name for the package)
RUN_CPPCHECK	OFF	Run cppcheck during make test or make all_cppcheck

3.3 Linux / Unix

3.3.1 Requirements

To compile on POSIX based operating systems you need a functioning C++11 compiler (for GCC version 5.0 minimum) and cmake. To compile the documentation you will also need python-sphinx version 1.0 or higher.

3.3.2 Compiling

After running CMake as described above you simple need to run `make` and then `sudo make install`. To run the test suite you can run `make check`.

3.4 Building a Package

To build an RPM or DEB package you first need to specify the OS you want to build for, for example:

```
cmake . -DRPM=centos7
```

or

```
cmake . -DDEB=xenial
```

You should of course add options as above to this as required. Then you can build the package using:

```
make package
```