MySQL Connector/C++ Release Notes

Abstract

This document contains release notes for the changes in recent releases of MySQL Connector/C++.

For additional MySQL Connector/C++ documentation, see MySQL Connector/C++ 9.4 Developer Guide.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

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

Document generated on: 2025-08-22 (revision: 30454)

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL Connector/C++.

Legal Notices

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 for Accessibility

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.

Changes in MySQL Connector/C++ 9

Changes in MySQL Connector/C++ 9.5.0 (Not yet released, General Availability)

Version 9.5.0 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Connector/C++ 9.4.0 (2025-07-22, General Availability)

Note

These release notes were created with the assistance of MySQL HeatWave GenAI.

Functionality Added or Changed

- The X DevAPI and X DevAPI for C now support configurable read and write timeouts. These timeouts can be set in milliseconds with the following methods:
  - For both APIs, with the connection options `read-timeout` and `write-timeout` in the connection string.
  - For the X DevAPI, with the `SessionOption` enumeration constants `READ_TIMEOUT` and `WRITE_TIMEOUT` in the `mysqlx::Session` or `mysqlx::SessionSettings` constructor.
  - For the X DevAPI for C, with the `MYSQLX_OPT_READ_TIMEOUT` and `MYSQLX_OPT_WRITE_TIMEOUT` enumeration constants or the `OPT_READ_TIMEOUT()` and `OPT_WRITE_TIMEOUT()` macros, using the `mysqlx_session_option_set()` function.

  The timeouts can only be set at the time of connection. See the MySQL Connector/C++ X DevAPI Reference for details. (WL #16924)

Changes in MySQL Connector/C++ 9.3.0 (2025-04-15, General Availability)

- Security Notes
- Bugs Fixed

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for MySQL Server has been updated to version 3.0.16. For more information, see OpenSSL 3.0 Series Release Notes and OpenSSL Security Advisory (11th February 2025). (Bug #37618844)

Bugs Fixed

- On macOS, when using the JDBC API, loading of the authentication plugins failed due to linking issues with the third-party libraries. (Bug #37094209)
- The `mysql-concpp` module used the `NO_CACHE` CMake option, which was not supported for CMake 3.20 and earlier, making the `find_` methods fail. This patch uses an explicit command to clear the cached variables instead, before the calls of the `find_` methods. Thanks to Lenny Wu for contributing the fix. (Bug #117763, Bug #37734620)

Changes in MySQL Connector/C++ 9.2.0 (2025-01-21, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added a new `OPT_WEBAUTHN_DEVICE_NUMBER` connection option to the classic Connector/C++ that is passed to and interpreted by the WebAuthN authentication plugin. It accepts a numeric value that selects the authenticator device to use during WebAuthN authentication. Previously, the first (#0) authentication plugin was always used. (WL #16645)
- Now `find_package(mysql-concpp)` supports a Connector/C++ installation that only contains debug (and not release) builds of the connector libraries. (WL #16321)

Bugs Fixed

- The "Typical" MSI installation profile no longer installs the CMake configuration script. Use the "Custom or "Complete" profile to install development components, which includes headers and the CMake configuration script. (Bug #37179475)

- Uninstalling connector RPM packages emitted errors despite successfully uninstalling the packages. (Bug #37096144)
- In a situation where `PLUGIN_DIR` was not needed (pluggable authentication is not used), a connection could fail without setting `PLUGIN_DIR` if pluggable authentication was used in another connection. (Bug #36929651)
- Exceptions are no longer thrown inside a destructor. (Bug #116569, Bug #37278704)
- The `-devel` RPM package upgrade from 8.4.0 to 9.0.0 failed due to a conflict on `/usr/include/mysql-cppconn/`. This was fixed by removing the associated symlink in the %pretrans scriplet. (Bug #115472, Bug #36795664)

Changes in MySQL Connector/C++ 9.1.0 (2024-10-15, General Availability)

Functionality Added or Changed

- Added *OpenID Connect* support leveraging the new `authentication_openid_connect_client` client-side authentication plugin. *OpenID Connect* functionality is supported by MySQL Enterprise Edition Server 9.1.0 and later.

  The new `OPT_OPENID_TOKEN_FILE` connection option defines a path to a file containing the JWT formatted identity token. (WL #16435)

- The RPM and DEB packages now install a copy of the MySQL client library plugins for the connector. The version of these plugins match the version of the statically linked MySQL client library.

  They are installed to `{libdir}/mysql/libmysqlcppconn{ABI}/plugin/` where `{libdir}` is the system location where packages install libraries. `{ABI}` is the connector's ABI version, which is currently 10.

  The connector installed from RPM and DEB packages use the bundled plugins as needed without requiring the `PLUGIN_DIR` connection option, although the `PLUGIN_DIR` connection option is still available to change the plugin location. Runtime dependencies required by the plugins, such as Kerberos and LDAP libraries, are expected on the system and installed from their own packages. (WL #16458)

Bugs Fixed

- Starting from version 9.1.0, it might be necessary to add the `-lz` option when linking user code with a static connector library on some platforms. This is handled automatically if specifying a build configuration with CMake using the `mysql-concpp` module. (Bug #37116076)
- The zlib sources bundled in the Connector/C++ source tree were upgraded to zlib 1.3.1. (Bug #37069890)
- On Debian-based systems, executing `apt-get install libmysqlcppconnx2` would fail if the `dpkg-dev` package that installs the required `dpkg-architecture` command was not installed on the system. The `dpkg-architecture` dependency was removed. (Bug #36807184)
- Added a new RPM compatibility package (`mysql-connector-c++-compat`) to allow upgrades from versions older than 9.0.0, the version that the ABI version changed from 9 to 10. (Bug #36753748, WL #16462)

Changes in MySQL Connector/C++ 9.0.0 (2024-07-01, General Availability)

- Bugs Fixed

Functionality Added or Changed

- Removed the dependency on the `mysql-client-plugins` package from the `libmysqlcppconnx` package and added it as recommended for the `libmysqlcppconn` package, as `mysql-client-plugins` is required only for using pluggable authentication with the classic client-server protocol. (Bug #36725056)
- This release includes directory and file name changes: On Linux, RPM and DEB development packages now install public headers to /usr/include/mysql-cppconn/ instead of /usr/include/mysql-cppconn-8/. On Windows, the MSI installation suffix was modified from "Connector C++ X.Y/" to "MySQL Connector C++ X.Y/" (where X.Y is major.minor version of the connector) to align with other MySQL products. The major JDBC ABI version changed from 9 to 10.

  A consequence of bumping JDBC ABI version, the classic connector library soname changes to libmysqlcppconn.so.10 while the symlink used in linker invocation remains as libmysqlcppconn.so. On Debian the library package name changes from libmysqlcppconn9 to libmysqlcppconn10. Known limitation: as a consequence of bumping the JDBC ABI version, upgrading the RPM package from version 8.4.0 to 9.0.0 removes the old .so.9 library from the system and this could break code that depends on that old version. This is not a limitation for DEB packages.

  For the X DevAPI packages, the base name changed from libmysqlcppconn8 to libmysqlcppconnx, and the DEB package is renamed from libmysqlcppconn8-2 to libmysqlcppconnx2. The major X DevAPI ABI version remains at 2. (Bug #36723452, WL #16310)
- Added support to the JDBC API for the `VECTOR` data type that was introduced in MySQL Enterprise Server 9.0.0. This also adds the `ResultSet::getVector()` and `PreparedStatement::setVector()` methods, along with the `DataType::VECTOR` constant. (WL #16170)

Bugs Fixed

- The CMake package configuration file (`mysql-concpp-config.cmake`) would not correctly declare the JDBC API library dependency with the MySQL client library. This could lead to broken builds for applications that use the JDBC API library when the connector was built from source with the default settings. (Bug #36298318)

Changes in MySQL Connector/C++ 8

Changes in MySQL Connector/C++ 8.4.0 (2024-04-30, General Availability)

- Security Notes
- Functionality Added or Changed
- Bugs Fixed

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for Connector/C++ has been updated to version 3.0.13. Issues fixed in OpenSSL version 3.0.13 are described at https://openssl-library.org/news/openssl-3.0-notes/. . (Bug #36278302)

Functionality Added or Changed

- Removed the dependency on the `mysql-client-plugins` package from the `libmysqlcppconnx` package and added it as recommended for the `libmysqlcppconn` package, as `mysql-client-plugins` is required only for using pluggable authentication with the classic client-server protocol. (Bug #36725056)

- Expanded the Windows file attributes for packaged executable and DLL files. (WL #16156)
- Removed support for the deprecated `authentication_fido` authentication plugin. Instead, use `authentication_webauthn`. For backward-compatibility, the `Fido_Callback` callback argument remains but invokes WebAuthn authentication. (WL #16154)
- Setting query attributes for executed queries now supports prepared statements. (WL #15968)
- Known limitation of this release: because the mysql_native_password authentication plugin is disabled by default as of MySQL Server 8.4.0, some unit tests may generate errors unless the plugin is enabled.

Bugs Fixed

- A failed connection from an unreachable host would always reference "localhost" in the error message, but now shows the configured host name and port number. (Bug #36383472)
- Building JDBC Connector/C++ from the source in combination with an older version of the MySQL client library that supports reconnect functionality (MySQL 8.3.0 and earlier) now allows `OPT_RECONNECT` to function. Otherwise, the option is ignored.

  Note that MySQL Server 8.4.0 removes reconnect functionality, but to preserve backward compatibility this connector can still set `OPT_RECONNECT` and read its value as before but it has no effect on connection behavior with MySQL Server 8.4.0. (Bug #36316146)
- On Windows using Visual Studio 2022, the connector would not build with the -DBUILD_STATIC=1 configuration option. (Bug #36250741)
- The build system now uses CMake's FindOpenSSL rather than a custom FindSSL module to better function with LibreSSL, and to better handle upcoming OpenSSL versions. The custom FindSSL module is still utilized with CMake 3.8 and earlier. This fix is based on a patch from Sam James, thank you for the contribution. (Bug #110784, Bug #35584977)

Changes in MySQL Connector/C++ 8.3.0 (2024-01-16, General Availability)

Functionality Added or Changed

- Connector/C++ 8.3.0 binary distributions from Oracle are built with the 8.3.0 version of the MySQL client library. MySQL Server 8.3.0 removed auto-reconnect support after deprecating it in versions 8.0.34 and 8.1.0. The related `OPT_RECONNECT` connection option continues to be recognized by the connector, but enabling the option now has no effect on the connection. Applications that use the legacy JDBC API and also enable `OPT_RECONNECT` can:

  - Implement reconnect logic in the application code.
  - Build JDBC Connector/C++ from sources in combination with an older version of the MySQL client library. However, enabling the `OPT_RECONNECT` option does guarantee a successful automatic reconnection to the server if the connection is found to have been lost.

  (WL #15979)
- Improved OpenTelemetry support to propagate context when executing prepared statements. (WL #15959)
- Improved support to import Connector/C++ into CMake projects by allowing `find_package(mysql-concpp)` to enable consuming it. This defines import targets such as `mysql::concpp` that can be linked to executables used by the connector. (WL #15952)

Changes in MySQL Connector/C++ 8.2.0 (2023-10-25, General Availability)

- Packaging Notes

- Functionality Added or Changed

Packaging Notes

- The MSI package definition files have been updated to work with the Windows Installer XML Toolset (WiX) version 4.0.1. Note that the files can no longer be used with previous version 3 of the toolset. (WL #15846)

Functionality Added or Changed

- Improved OpenTelemetry support. This includes adding spans for preparing and executing prepared statements, and adding connection span attributes such as `db.user`. (WL #15808)
- Connector/C++ now supports WebAuthn authentication for client applications using the legacy JBDC API (that is, applications not using X DevAPI or X DevAPI for C). WebAuthn authentication enables users to authenticate to MySQL Server using WebAuthn-aware devices based on either the FIDO or FIDO2 standard. For an overview of the supported client-side authentication plugins and authentication methods, see Authentication Support. (WL #15239)

Changes in MySQL Connector/C++ 8.1.0 (2023-07-18, General Availability)

MySQL Connector/C++ 8.1.0 is a new GA release version that supersedes the 8.0 series, and is recommended for use on production systems. This release can be used against MySQL Server versions 5.7 and later.

- Packaging Notes
- Functionality Added or Changed

Packaging Notes

- ZSTD sources bundled in the Connector/C++ source tree are upgraded to ZSTD 1.5.5. (Bug #35360566)
- The Connector/C++ static library is most effective when the installed package is specific to the platform on which the final application is built. Accordingly, the following static connector libraries are no longer included in the generic Linux packages:
  - `lib64/libmysqlcppconn-static.a`
  - `lib64/libmysqlcppconn8-static.a`

  (WL #15650)

Functionality Added or Changed

- **Important Change:** 32-bit binaries are no longer built as of MySQL 8.1.0.
- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C) on Linux systems and use OpenTelemetry instrumentation, the connector adds query and connection spans to the trace generated by application code and forwards the current OpenTelemetry context to the server. By default, the connector generates spans only when an instrumented application links with the required OpenTelemetry SDK libraries and configures the trace exporter to send trace data to some destination. If the application code does not use instrumentation, then the legacy connector does not use it either.

  Connector/C++ supports a new connection property option, `OPT_OPENTELEMETRY`, which has these values:

  - `OTEL_DISABLED`
  - `OTEL_PREFERRED` (default)

The `OPT_OPENTELEMETRY` option also accepts a Boolean value in which `false` corresponds to `OTEL_DISABLED`. `false` is the only accepted Boolean value for this option; setting it to `true` has no meaning and emits an error. (WL #15625)

Changes in MySQL Connector/C++ 8.0.33 (2023-04-18, General Availability)

Compilation Notes

- To simplify building the legacy JDBC connector with Linux distributions that do not ship static libraries, the `MYSQLCLIENT_STATIC_LINKING` default now is `OFF` (use dynamic linking to the client library). Previously, the default setting was `ON` and the binary distributions from Oracle are still built with static linking. (WL #15466)
- Binary distributions from Oracle are built in C++17 mode. When building Connector/C++ from sources, the compilation now fails if the compiler used does not support C++17. To compile Connector/C++ applications that use X DevAPI (or if the code uses C++17), enable C++17 support in the compiler using the `-std=c++17` option. (WL #15429)
- Connector/C++ now compiles cleanly using Clang on Windows. (WL #15290)

Packaging Notes

- ZSTD sources bundled in the Connector/C++ source tree are upgraded to ZSTD 1.5.0. (Bug #34983529)
- LZ4 sources bundled in the Connector/C++ source tree are upgraded to LZ4 1.9.4. (Bug #34983380)
- ZLIB sources bundled in the Connector/C++ source tree are upgraded to ZLIB 1.2.13 to match the server. (Bug #34888141)
- RapidJSON sources bundled in the Connector/C++ source tree are upgraded to RapidJSON 1.1.0 to match the server. (Bug #34842662)

Pluggable Authentication

- The Connector/C++ implementation of the `authentication_oci_client` plugin (together with `libmysqlclient`) now enables using a security-token file to support ephemeral key-pair authentication when integration with an external identity provider is needed for classic MySQL protocol connections. The Oracle Cloud Infrastructure CLI generates the ephemeral key pair and security token.

  In addition, applications that use the legacy JDBC API now can set the new `OPT_OCI_CLIENT_CONFIG_PROFILE` connection option to specify which profile in the configuration file to use for authentication. It defaults to the `[DEFAULT]` profile. (WL #15481)

Functionality Added or Changed

- The Connector/C++ legacy JDBC connector now identifies itself through the following new connection attributes (in addition to the connection attributes obtained from the `libmysql` client library by default):

- `_connector_version` to indicate the connector version (for example, `8.0.33`).
- `_connector_license` to indicate the license type of the connector, either `GPL-2.0` or `Commercial`.
- `_connector_name` with a constant value of `mysql-connector-cpp`.

The new attributes are not configurable by applications. Instead, the connector sets all of the attribute values, which it determines automatically after connecting to the server. If the `libmysql` client library or MySQL Server version being used lack support for query attributes, then returning related connection-attribute errors is not possible.

For general information about connection attributes, see Performance Schema Connection Attribute Tables. (WL #15425)

Bugs Fixed

- **X DevAPI:** Operations attempting to modify an entire document, for example,

```
coll.modify("_id = 1").set("$", R"({ "name": "bar" })").execute();
```

emitted a server error rather than modifying the document. An error is expected when passing in a string. To modify an entire document, the caller should pass in a `DbDoc()` document. For example:

```
coll.modify("_id = 1").set("$", DbDoc(R"({ "name": "bar" })")).execute();
```

(Bug #35046616)

- **X DevAPI:** When a connection was made to MySQL Router, rather than directly to the server, closing a session using `mysqlx_session_close` was not possible. Now, the connector no longer waits for a reply to the call before closing the session. (Bug #107693, Bug #34338937)
- Minimum `CMake` version required was changed to 3.12 from the previous version, which was deprecated and produced warnings. (Bug #20422957)

Changes in MySQL Connector/C++ 8.0.32 (2023-01-17, General Availability)

Packaging Notes

- Protobuf sources bundled in the Connector/C++ source tree are upgraded to Protobuf 3.19.6 (to reduce compilation time, only the parts needed for Connector/C++ are included.) (WL #15414)
- Connector/C++ now provides generic Linux packages for ARM architecture (64 bit), in addition to the generic Linux packages for Intel architecture (both 32 and 64 bits). All generic Linux packages are built using the GNU C Library version 2.28. (WL #15258)

Pluggable Authentication

- Connector/C++ 8.0.27 implemented support for the SSPI Kerberos library on Windows, which was not capable of acquiring cached credentials previously generated by using `kinit` command. Connector/C++ 8.0.32 also supports GSSAPI through the MIT Kerberos library to add that capability

using the `authentication_kerberos_client` authentication plugin for classic MySQL protocol connections on Windows. The `OPT_AUTHENTICATION_KERBEROS_CLIENT_MODE` connection option can be set to either `SSPI` (default) or `GSSAPI`. For more information, see Kerberos Authentication. (WL #15346)

Security Notes

- This release of Connector/C++ upgrades Cyrus SASL to version 2.1.28, which has been publicly reported as not vulnerable to `CVE-2022-24407`. (Bug #34680980)

Functionality Added or Changed

- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), setting the `OPT_METADATA_INFO_SCHEMA` (`metadataUseInfoSchema`) connection option was essential to the performance of `Information_Schema` in the older versions of MySQL Server. The option is no longer needed for performance. Now, setting `OPT_METADATA_INFO_SCHEMA` has no affect on the behavior of the connector and the option is ignored. (WL #15322)

Bugs Fixed

- **X DevAPI:** A call to close a session while other `Session` objects are in use could return a busy error. (Bug #34704048)
- **X DevAPI:** An attempt to execute a long SQL statement emitted an error when the connection was established using `mysqlx_get_session_from_options()` to connect to MySQL Router and with compression enabled on the connection. (Bug #107694, Bug #34338921)
- A local object move in a return statement could prevent copy elision (or copy omission). Our thanks to Octavio Valle for the patch. (Bug #108652, Bug #34654192)

Changes in MySQL Connector/C++ 8.0.31 (2022-10-11, General Availability)

Compilation Notes

- Connector/C++ now compiles cleanly using Clang for Linux and Solaris. (WL #15065)

Configuration Notes

- Several `CMake` options have been added or updated to enable using external sources (or builds) of third-party components, like compression libraries and the Protobuf compiler, on which Connector/C++ depends. These options permit substituting external source locations at configuration time if required.

  Supported options are:

  - `WITH_BOOST`: The Boost source directory.
  - `WITH_LZ4`: The LZ4 source directory.
  - `WITH_MYSQL`: The MySQL Server source directory.

- `WITH_PROTOBUF`: The Protobuf source directory.
- `WITH_SSL`: The SSL source directory.
- `WITH_ZLIB`: The ZLIB source directory.
- `WITH_ZSTD`: The ZSTD source directory.

Currently, bundled third-party libraries used by connector are linked statically to it. Externally sourced libraries are linked dynamically. Long-standing issues, such as applications that link to the static connector library (`libmysqlcppconn8-static.a`) not being able to also link to a Protobuf library at the same time, now are resolved by building from sources a variant that links Protobuf dynamically.

For more information, see Specifying External Dependencies. (Bug #32117299, WL #15064)

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for Connector/C++ has been updated to version 1.1.1q. Issues fixed in the new OpenSSL version are described at https://www.openssl.org/news/cl111.txt and https://www.openssl.org/news/vulnerabilities.html. (Bug #34414692)

Functionality Added or Changed

- If building the legacy JDBC connector from source, using an additional `git` command to perform submodule initialization is no longer necessary. (WL #15182)

Bugs Fixed

- **X DevAPI:** If an application program called `mysqlx_session_close` after disconnecting from the Internet, an exception from Connector/C++ could cause the application to halt unexpectedly. (Bug #107692, Bug #34338950)
- On Windows, compiler difficulties were encountered because `unistd.h` was used to call `getcwd` rather than using various Windows alternatives. Our thanks to Luis Pinto for the patch. (Bug #108355, Bug #34553226)
- The `libcrypto` library, which `libssl` attempted to link to, was installed in an unexpected directory by the Connector/C++ binary distribution for macOS. This fix ensures that both bundled libraries are installed in the same directory. (Bug #107947, Bug #34417381)

Changes in MySQL Connector/C++ 8.0.30 (2022-07-26, General Availability)

Character Set Support

- The renaming of the `utf8` character set to `utf8mb3` in MySQL 8.0.30—together with new `utf8mb4` collations—generated errors related to collation names. To support the character-set name change and the new collations, several API changes now are implemented.

  For applications using X DevAPI or X DevAPI for C (breaking changes):

- The `CharacterSet::utf8` enumeration constant was renamed to `CharacterSet::utf8mb3`, but the constant value (21) did not change. Compiling an existing application against the new connector produces errors if code refers to the `CharacterSet::utf8` enumeration constant.

  

  Note

  Code that uses `CharacterSet::utf8mb3` without also using the new `utf8mb4` collations can be expected to work with the old connector library, but only after it has been compiled using the new connector header files.

- The name returned by the `characterSetName()` function for the value of `CharacterSet::utf8/utfmb3` constant (21) changed from "utf8" to "utf8mb3".

- Collation names returned by the `CollationInfo::getName()` method for `CollationInfo` members in `Collation<21>` have changed. For example, `Collation<21>::general_ci.getName()` now returns “utf8mb3_general_ci” instead of “utf8_general_ci”.

- Although not a breaking change, all of the new `CollationInfo` members, such as `utf8mb4_bg_0900_ai_ci`, have been added (for example, `Collation<CharacterSet::utf8mb4>::bg_0900_ai_ci`).

For applications using the legacy JDBC API (breaking changes):

- The character-set name returned by the `MySQL_PreparedResultSetMetaData::getColumnCharset()` and `MySQL_ResultSetMetaData::getColumnCharset()` methods for the character set `utf8/utf8mb3` has changed from "utf8" to "utf8mb3".

- Collation names returned by the `MySQL_PreparedResultSetMetaData::getColumnCollation()` and `MySQL_ResultSetMetaData::getColumnCollation()` methods for `utf8/utf8mb3` have changed. For example, the previous collation "utf8_general_ci" is now replaced with "utf8mb3_general_ci".

Existing applications that contain logic for checking the `utf8` character-set name, or one of its collations, can expect errors when used with the new connector. If an application must be compiled with a pre-8.0.30 connector, then all of the character-set and collation comparisons should be guarded with the `MYSQL_CONCPP_VERSION_NUMBER` macro. For usage examples, see Connector/C++ Version Macros.

Each related ABI remains backward compatible, which means that an application built against old connector sources can link correctly to the new connector library. However, the new library now reports different character-set and collation names for `utf8` and this difference might break the code's logic. (Bug #34149700)

Packaging Notes

- Generic Linux packages now are built using the GNU C Library version 2.27 and the new C++ ABI (`_GLIBCXX_USE_CXX11_ABI=1`). For additional information about this change, see Generic Linux Notes. (Bug #33983351)

Functionality Added or Changed

- Connector/C++ now provides the `MYSQL_CONCPP_VERSION_NUMBER` macro in public header files to indicate the current version of the connector. The format of `MYSQL_CONCPP_VERSION_NUMBER` is *XYYZZZZ*, in which:

  - *X* is the major version number (8)

- *YY* is the minor version number (00)
- *ZZZZ* is the micro version number (0030)

With this macro, code that depends on one or more features that were introduced in a specific version (such as 8.0.32) can perform a conditional test in the compilation of portion of a source file. For example:

```
#if MYSQL_CONCPP_VERSION_NUMBER > 8000032
  // use some 8.0.32+ feature
#endif
```

This type of conditional-compilation directive also works when the `MYSQL_CONCPP_VERSION_NUMBER` macro is not defined (with pre-8.0.30 header files), in which case, it is treated as 0. However, using the macro to check against versions earlier than 8.0.30 is unreliable and should be avoided.

For additional usage examples, see Connector/C++ Version Macros. (WL #15081)

- It is now possible to compile Connector/C++ using OpenSSL 3.0. (WL #14819)
- The Protobuf sources bundled with Connector/C++ were updated to Protobuf 3.19.4. To reduce compilation time, this update includes only the parts needed for Connector/C++. (WL #15084)

Bugs Fixed

- A valid query emitted an error when one or more fields were of spatial data type `GEOMETRY` and it was executed using a prepared statement. (Bug #19192707)

Changes in MySQL Connector/C++ 8.0.29 (2022-04-26, General Availability)

- Pluggable Authentication
- Security Notes
- X DevAPI Notes
- Functionality Added or Changed
- Bugs Fixed

Pluggable Authentication

- Connector/C++ now supports authentication to MySQL Server using devices such as smart cards, security keys, and biometric readers. This authentication method is based on the Fast Identity Online (FIDO) standard. To ensure client applications using the legacy JBDC API are notified when a user is expected to interact with the FIDO device, Connector/C++ implements the new `setCallback()` method in the `MySQL_Driver` class that accepts a single callback argument named `Fido_Callback`.

```
class Fido_Callback
{
public:

  Fido_Callback(std::function<void(SQLString)>);

  /**
  * Override this message to receive Fido Action Requests
  */
  virtual void FidoActionRequested(sql::SQLString msg);

};
```

Any connection created by the driver can use the callback, if needed. However, if an application does not set the callback explicitly, `libmysqlclient` determines the behavior by default, which involves printing a message to standard output.

Note

On Windows, the client application must run as administrator. The is a requirement of the `fido2.dll` library, which is used by the `authentication_fido` plugin.

A client application has two options for obtaining a callback from the connector:

- By passing a function or lambda to `Fido_Callback`.

```
driver->setCallBack(Fido_Callback([](SQLString msg) {...}));
```

- By implementing the virtual method `FidoActionRequested`.

```
class MyWindow : public Fido_Callback
{
  void FidoActionRequested(sql::SQLString msg) override;
};

MyWindow window;
driver->setCallBack(window);
```

Setting a new callback always removes the previous callback. To disable the active callback and restore the default behavior, pass `nullptr` as a function callback. Example:

```
driver->setCallBack(Fido_Callback(nullptr));
```

(WL #14878)

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for Connector/C++ has been updated to version 1.1.1n. Issues fixed in the new OpenSSL version are described at https://www.openssl.org/news/cl111.txt and https://www.openssl.org/news/vulnerabilities.html. (Bug #33987637)

X DevAPI Notes

- The Connector/C++ X DevAPI Reference documentation, available at https://dev.mysql.com/doc/index-connectors.html, updated its usage instructions for the `Collection.modify().unset()` operation. The argument to `unset()` is a string to be interpreted as a document path expression (similar to `"$.foo.'bar'"`), rather than a literal field name. If the argument contains special characters (spaces, '`.`', '`$`', and so on), then it is necessary to enclose the field name in quotation marks. For example:

```
Collection.modify(~~).unset(""field name with spaces"")
```

(Bug #33795881)

Functionality Added or Changed

- Connector/C++ supports new aliases for existing TLS/SSL connection options to deliver better alignment among X DevAPI, X DevAPI for C, and the legacy JDBC-based API. This alignment effort ensures that option naming, functionality, and behavior are implemented consistently while also retaining compatibility with the existing options. For example, Connector/C++ now ensures that setting TLS/SSL connection options, along with `ssl-mode=DISABLED`, does not return an error if a client application provides incompatible options, or if the same option is repeated in a connection string or with properties.

Changes that apply to X DevAPI and X DevAPI for C are:

- `tls-version` is added as an alias to the existing `tls-versions` connection option.
- `ssl-capath`, `ssl-crl`, and `ssl-crlpath` options are now implemented with the same functionality as the legacy JDBC API.
- If the same option is repeated, the last option value prevails.

The new aliases for the legacy JDBC API are:

- `ssl-mode` (for the existing `OPT_SSLMODE` option): Preferred security state of a connection to server.
- `ssl-ca` (for the existing `sslCA` option): File that contains a list of trusted SSL Certificate Authorities.
- `ssl-capath` (for the existing `sslCAPath` option): Directory that contains trusted SSL Certificate Authority certificate files.
- `ssl-cert` (for the existing `sslCert` option): File that contains X.509 certificate.
- `ssl-cipher` (for the existing `sslCipher` option): Permissible ciphers for connection encryption.
- `ssl-key` (for the existing `sslKey` option): File that contains X.509 key.
- `ssl-crl` (for the existing `sslCRL` option): File that contains certificate revocation lists.
- `ssl-crlpath` (for the existing `sslCRLPath` option): Directory that contains certificate revocation-list files.
- `tls-version` (for the existing `OPT_TLS_VERSION` option): Permissible TLS protocols for encrypted connections.

When using the legacy JDBC API, the effect of setting an option twice is determined by the client library. In addition, TLS/SSL options are not supported in URI-like strings when using the legacy JDBC API. (WL #14846)

Bugs Fixed

- Bit-value types in aggregate functions could return unexpected values for an application that uses the legacy JDBC API. (Bug #33748725)
- The Connector/C++ classic driver was unable to find authentication plugins unless the `OPT_PLUGIN_DIR` connection option was set explicitly. The driver now uses its shared library to determine the plugin location as a relative path. (Bug #33721056)
- On Windows, when an application using the legacy JDBC API attempted to authenticate a user with a plugin that was unable to find a required library, the process halted rather than emitting an error message. (Bug #33701997)

Changes in MySQL Connector/C++ 8.0.28 (2022-01-18, General Availability)

Deprecation and Removal Notes

- The TLSv1 and TLSv1.1 connection protocols were deprecated in Connector/C++ 8.0.26 and now are removed in this release. The removed values are considered invalid for use with connection options and session settings. Connections can be made using the more-secure TLSv1.2 and TLSv1.3 protocols. TLSv1.3 requires that both the server and Connector/C++ be compiled with OpenSSL 1.1.1 or higher. (WL #14816, WL #14818)

Pluggable Authentication

- Applications that use the legacy JDBC API now can establish connections using multifactor authentication, such that up to three passwords can be specified at connect time. The new `OPT_PASSWORD1`, `OPT_PASSWORD2`, and `OPT_PASSWORD3` connection options are available for specifying the first, second, and third multifactor authentication passwords, respectively. `OPT_PASSWORD1` is an alias for the existing `OPT_PASSWORD` option; if both are given, `OPT_PASSWORD` is ignored. (WL #14658)

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for Connector/C++ has been updated to version 1.1.1l. Issues fixed in the new OpenSSL version are described at https://www.openssl.org/news/cl111.txt and https://www.openssl.org/news/vulnerabilities.html. (Bug #33309902)

Bugs Fixed

- For connections made using X Protocol, getting `BIT(1)` column data by calling `getRawBytes()` returned an empty buffer. `BIT` is treated as an unsigned integer (`uint64_t`), which means applications can get or set the value by using such a type. (Bug #33335148)

Changes in MySQL Connector/C++ 8.0.27 (2021-10-19, General Availability)

- Pluggable Authentication
- Bugs Fixed

Pluggable Authentication

- Applications that use the legacy JDBC API now can establish connections without passwords for accounts that use the `authentication_oci` server-side authentication plugin, provided that the correct configuration entries are available to map to one unique user in a specific Oracle Cloud Infrastructure tenancy. This functionality is not supported for X Protocol connections.

  To ensure correct account mapping, the client-side Oracle Cloud Infrastructure configuration must contain a fingerprint of the API key to use for authentication (`fingerprint` entry) and the location of a PEM file with the private part of the API key (`key_file` entry). Both entries should be specified in the `[DEFAULT]` profile of the configuration file.

  Unless an alternative path to the configuration file is specified with the `OPT_OCI_CONFIG_FILE` connection option, the following default locations are used:

  - `~/.oci/config` on Linux or Posix host types
  - `%HOMEDRIVE%%HOMEPATH%/.oci/config` on Windows host types

  If the MySQL user name is not provided as a connection option, then the operating system user name is substituted. Specifically, if the private key and correct Oracle Cloud Infrastructure configuration are present on the client side, then a connection can be made without giving any options. (WL #14711)

- In Connector/C++ 8.0.26, the capability was introduced for applications that use the legacy JDBC API to establish connections for accounts that use the `authentication_kerberos` server-side authentication plugin, provided that the correct Kerberos tickets are available or can be obtained from Kerberos. That capability was available on client hosts running Linux only. It is now available on client hosts running Windows.

  For more information about Kerberos authentication, see Kerberos Pluggable Authentication. (WL #14682)

Bugs Fixed

- When linking to the static Connector/C++ library on Windows, builds failed unless `Dnsapi.DLL` was added to the linker invocation line explicitly. (Bug #33190431)

Changes in MySQL Connector/C++ 8.0.26 (2021-07-20, General Availability)

Deprecation and Removal Notes

- The TLSv1 and TLSv1.1 connection protocols now are deprecated and support for them is subject to removal in a future Connector/C++ version. (For background, refer to the IETF memo Deprecating TLSv1.0 and TLSv1.1.) It is recommended that connections be made using the more-secure TLSv1.2 and TLSv1.3 protocols. TLSv1.3 requires that both the server and Connector/C++ be compiled with OpenSSL 1.1.1 or higher. (WL #14584)

Pluggable Authentication

- Applications that use the legacy JDBC API now can establish connections for accounts that use the `authentication_kerberos` server-side authentication plugin, provided that the correct Kerberos tickets are available or can be obtained from Kerberos. This capability is available on client hosts running Linux only.

  It is possible to connect to Kerberos-authenticated accounts without giving a user name under these conditions:

  - The user has a Kerberos principal name assigned, a MySQL Kerberos account for that principal name exists, and the user has the required tickets.
  - The default authentication method must be set to the `authentication_kerberos_client` client-side authentication plugin using the `OPT_DEFAULT_AUTH` connection option.

  It is possible to connect without giving a password, provided that the user has the required tickets in the Kerberos cache (for example, created by `kinit` or a similar command).

  If the required tickets are not present in the Kerberos cache and a password was given, Connector/C++ obtains the tickets from Kerberos using that password. If the required tickets are found in the cache, any password given is ignored and the connection might succeed *even if the password is incorrect.*

  For more information about Kerberos authentication, see Kerberos Pluggable Authentication. (WL #14439)

Functionality Added or Changed

- Applications that use the legacy JDBC API now can define query attribute metadata on a per-query basis. without the use of workarounds such as specially formatted comments included in query strings. This capability is implemented using type-specific methods for the `sql::Statement` class:

```
int Statement::setQueryAttrInt(attr_name, int_value);
int Statement::setQueryAttrString(attr_name, str_value);
int Statement::setQueryAttrBoolean(attr_name, bool_value);
```

  Each method takes a string-valued attribute name and an attribute value of the appropriate type. The return value is the attribute number, or 0 if the server does not support query attributes.

  Similar methods are implemented for the `sql::PreparedStatement` class but do nothing because the MySQL client library does not yet support query attributes for prepared statements.

  Attributes defined using the set-attribute methods apply to the next SQL statement sent to the server for execution. If an attribute with a given name is defined multiple times, the last definition applies. Attributes are cleared after the statement executes, or may be cleared explicitly using the `clearAttributes()` method.

  The `mysql_query_attribute_string()` function returns the current value for a given attribute, except that the value of an attribute with an empty name cannot be retrieved.

  Example:

```
std::unique_ptr<sql::Statement> stmt(con->createStatement());

// Set three query attributes for a query without parameters ("SELECT 1"),
// where the attribute types are int, string, and bool:

stmt->setQueryAttrInt("attr1", 200);
stmt->setQueryAttrString("attr2", "string value");
stmt->setQueryAttrBoolean("attr3", true);

// To retrieve the attributes within a query, use the
// mysql_query_attribute_string() function:

stmt->execute("SELECT 1,
              mysql_query_attribute_string('attr1'),
              mysql_query_attribute_string('attr2'),
              mysql_query_attribute_string('attr3')");

// Change an attribute value:

stmt->setQueryAttrInt("attr1", 100);

// Executing the statement here and fetching the result should show the
// changed attribute value.

// Clear the attributes:

stmt->clearAttributes();

// Executing the statement here and fetching the result should show the
// the attributes are no longer present.
```

  For the query-attribute capability to work within Connector/C++ applications, server-side support for query attributes must be enabled. For instructions, and for more information about query-attribute support in general, see Query Attributes. (WL #14216)

Bugs Fixed

- Builds failed when the `-DMYSQLCLIENT_STATIC_BINDING=0` and `-DMYSQLCLIENT_STATIC_LINKING=0` configuration options were used. (Bug #32882344)

- For connection attempts that specified a list of servers to try, the connection timeout value could be twice the correct duration. (Bug #32781963)
- Connector/C++ returned error 0 for connection failures rather than a nonzero error code. (Bug #32695580)
- `sql::Connection:commit()` threw no error if the connection had been dropped. (Bug #23235968)

Changes in MySQL Connector/C++ 8.0.25 (2021-05-11, General Availability)

This release contains no functional changes, and is published to align its version number with that of the MySQL Server 8.0.25 release.

Changes in MySQL Connector/C++ 8.0.24 (2021-04-20, General Availability)

- Connection Management Notes
- Packaging Notes
- Security Notes
- Bugs Fixed

Connection Management Notes

- Previously, for client applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), if the connection to the server was not used within the period specified by the `wait_timeout` system variable and the server closed the connection, the client received no notification of the reason. Typically, the client would see `Lost connection to MySQL server during query` (`CR_SERVER_LOST`) or `MySQL server has gone away` (`CR_SERVER_GONE_ERROR`).

  In such cases, the server now writes the reason to the connection before closing it, and client receives a more informative error message, `The client was disconnected by the server because of inactivity. See wait_timeout and interactive_timeout for configuring this behavior.` (`ER_CLIENT_INTERACTION_TIMEOUT`).

  The previous behavior still applies for client connections to older servers and connections to the server by older clients. (WL #14425)

- For connections made using X Plugin, if client with a connection to a server remains idle (not sending to the server) for longer than the relevant X Plugin timeout setting (read, write, or wait timeout), X Plugin closes the connection. If any of these timeouts occur, the plugin returns a warning notice with the error code `ER_IO_READ_ERROR` to the client application.

  For such connections, X Plugin now also sends a warning notice if a connection is actively closed due to a server shutdown, or by the connection being killed from another client session. In the case of a server shutdown, the warning notice is sent to all authenticated X Protocol clients with open connections, with the `ER_SERVER_SHUTDOWN` error code. In the case of a killed connection, the warning notice is sent to the relevant client with the `ER_SESSION_WAS_KILLED` error code, unless the connection was killed during SQL execution, in which case a fatal error is returned with the `ER_QUERY_INTERRUPTED` error code.

  If connection pooling is used and a connection close notice is received in a session as a result of a server shutdown, all other idle sessions that are connected to the same endpoint are removed from the pool.

  Client applications can use the warning notices to display to users, or to analyze the reason for disconnection and decide whether to attempt reconnection to the same server, or to a different server. (WL #13946)

Packaging Notes

- Connector/C++ packages now include sasl2 modules due to connection failures for accounts that use the `authentication_ldap_sasl` authentication plugin. (Bug #32175836)

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for Connector/C++ has been updated to version 1.1.1k. Issues fixed in the new OpenSSL version are described at https://www.openssl.org/news/cl111.txt and https://www.openssl.org/news/vulnerabilities.html. (Bug #32719727)

  References: See also: Bug #32680637.

Bugs Fixed

- Upon connecting to the server, Connector/C++ executed a number of `SHOW [SESSION] VARIABLES` statements to retrieve system variable values. Such statements involve locking in the server, so they are now avoided in favor of `SELECT @@var_name` statements.

  Additionally, Connector/C++ was trying to fetch the value of the `max_statement_time` system variable, which has been renamed to `max_execution_time`. Connector/C++ now uses the correct variable name, with the result that `getQueryTimeout()` and `setQueryTimeout()` now work properly for both `Statement` and `Prepared Statement` objects. (Bug #28928712, Bug #93201)

- `DatabaseMetaData.getProcedures()` failed when the `metadataUseInfoSchema` connection option was false. (Bug #24371558)

Changes in MySQL Connector/C++ 8.0.23 (2021-01-18, General Availability)

Legacy (JDBC API) Notes

- Previously, to build or run applications that use the legacy JDBC API, it was necessary to have Boost installed. Boost is no longer required for such applications. The API has not changed, so no code changes are required to build applications. However, in consequence of this change, the ABI version has increased from 7 to 9. To run applications, a version of Connector/C++ built with the same ABI must be installed:

  - Applications built using the new ABI require a version of Connector/C++ also built using the new ABI.
  - Applications built using the old ABI require a version of Connector/C++ also built using the old ABI.

  To build the legacy connector itself from source, it is still necessary to have Boost installed. (WL #13983)

Functionality Added or Changed

- All calls that allow a column name, such as `findColumn()`, `getString()`, and `getInt()`, are now case-sensitive. (Bug #30126457, Bug #96398)
- The developer documentation was improved regarding how to decode the bytes received by `mysqlx_get_bytes()`. Thanks to Daniël van Eeden for pointing at the missing documentation. (Bug #29115299, Bug #93641)

- Thanks to Daniël van Eeden, who contributed various corrections to the developer documentation. (Bug #29038157, Bug #93549)
- A dependency on the `mysql-client-plugins` package was removed. This package now is required only on hosts where Connector/C++ applications make connections using commercial MySQL server accounts with LDAP authentication. In that case, additional libraries must also be installed: `cyrus-sasl-scram` for installations that use RPM packages and `libsasl2-modules-gssapi-mit` for installations that use Debian packages. These SASL packages provide the support required to use the SCRAM-SHA-256 and GSSAPI/Kerberos authentication methods for LDAP.

  If Connector/C++ applications do not use LDAP authentication, no additional packages are required. (WL #13881, WL #14250)

Bugs Fixed

- Connector/C++ 8.0 RPM packages could not be installed on a system where MySQL 5.7 RPM packages were installed. (Bug #32142148)
- Establishing a connection using a `ConnectOptionsMap` object could fail due to differences in `std::string` implementations. (Bug #32039929)
- Commercial Connector/C++ RPM packages were missing `provides` information. (Bug #31775733)

Changes in MySQL Connector/C++ 8.0.22 (2020-10-19, General Availability)

Compilation Notes

- Connector/C++ now can be compiled using MinGW on Windows. Thanks to Eric Beuque for the contribution. Note that this enables building on MinGW but does not make MinGW an officially supported platform for Connector/C++. (Bug #31636723, Bug #100248)

Connection Management Notes

- For connections made using X Plugin, Connector/C++ now enables specifying the compression algorithms to be used for connections that use compression. Connection URIs and `SessionSettings` objects permit explicitly specifying the preferred algorithms:
  - URI strings permit a `compression-algorithms` option. The value is an algorithm name, or a list of one or more comma-separated algorithms specified as an array. Examples:

    ```
    mysqlx://user:password@host:port/db?compression-algorithms=lz4
    mysqlx://user:password@host:port/db?compression-algorithms=[lz4,zstd_stream]
    ```

  - `SessionSettings` objects permit a `SessionOption::COMPRESSION_ALGORITHMS` option. The value is a list of one or more comma-separated algorithms. Examples:

    ```
    mysqlx::Session sess(SessionOption::USER, "user_name",
                         SessionOption::PWD, "password",
                         SessionOption::COMPRESSION_ALGORITHMS, "lz4");
    mysqlx::Session sess(SessionOption::USER, "user_name",
                         SessionOption::PWD, "password",
                         SessionOption::COMPRESSION_ALGORITHMS, "lz4,zstd_stream");
    ```

    Alternatively, the algorithms value can be given as a container:

    ```
    std::list<std::string> algorithms = {"lz4","zstd_stream"};
    ```

```
mysqlx::Session sess(SessionOption::USER, "user_name",
                     SessionOption::PWD, "password",
                     SessionOption::COMPRESSION_ALGORITHMS, algorithms);
```

- For X DevAPI for C, there is a new `MYSQLX_OPT_COMPRESSION_ALGORITHMS` option and corresponding `OPT_COMPRESSION_ALGORITHMS` helper macro.

  URI mode follows X DevAPI URI mode:

```
mysqlx_session_t *sess = mysqlx_get_session_from_url(
  "mysqlx://user:password@host:port/db?compression-algorithms=[lz4,zstd_stream]",
   &error);
```

  Option mode follows the string format used for `SessionOption`:

```
mysqlx_session_option_set(opt,
          OPT_HOST("host_name"),
          OPT_USER("user"),
          OPT_PWD("password"),
          OPT_COMPRESSION_ALGORITHMS("lz4,zstd_stream"),
          PARAM_END));
```

These rules apply:

- Permitted algorithm names are `zstd_stream`, `lz4_message`, and `deflate_stream`, and their aliases `zstd`, `lz4`, and `deflate`. Names are case-insensitive. Unknown names are ignored.
- Compression algorithms options permit multiple algorithms, which should be listed in priority order. Options that specify multiple algorithms can mix full algorithm names and aliases.
- If no compression algorithms option is specified, the default is `zstd_stream,lz4_message,deflate_stream`.
- The actual algorithm used is the first of those listed in the compression algorithms option that is also permitted on the server side. However, the option for compression algorithms is subject to the compression mode:
  - If the compression mode is `disabled`, the compression algorithms option is ignored.
  - If the compression mode is `preferred` but no listed algorithm is permitted on the server side, the connection is uncompressed.
  - If the compression mode is `required` but no listed algorithm is permitted on the server side, an error occurs.

See also Connection Compression with X Plugin. (WL #13908, WL #13947)

Legacy (JDBC API) Notes

- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), Connector/C++ binary distributions now include the libraries that provide the client-side LDAP authentication plugins, as well as any dependent libraries required by the plugins. This enables Connector/C++ application programs to connect to MySQL servers using simple LDAP authentication, or SASL LDAP authentication using the `SCRAM-SHA-1` authentication method.

Note

LDAP authentication requires use of a server from a MySQL Enterprise Edition distribution. For more information about the LDAP authentication plugins, see LDAP Pluggable Authentication.

If Connector/C++ was installed from a compressed `tar` file or Zip archive, the application program will need to set the `OPT_PLUGIN_DIR` connection option to the appropriate directory so that the

bundled plugin library can be found. (Alternatively, copy the required plugin library to the default directory expected by the client library.)

Example:

```
sql::ConnectOptionsMap connection_properties;

// To use simple LDAP authentication ...

connection_properties["userName"] = "simple_ldap_user_name";
connection_properties["password"] = "simple_ldap_password";
connection_properties[OPT_ENABLE_CLEARTEXT_PLUGIN]=true;

// To use SASL LDAP authentication using SCRAM-SHA-1 ...

connection_properties["userName"] = "sasl_ldap_user_name";
connection_properties["password"] = "sasl_ldap_scram_password";

// Needed if Connector/C++ was installed from tar file or Zip archive ...

connection_properties[OPT_PLUGIN_DIR] = "${INSTALL_DIR}/lib{64}/plugin";

auto *driver = get_driver_instance();
auto *con = driver->connect(connection_properties);

// Execute statements ...

con->close();
```

(WL #14113)

- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), `LOCAL` data loading capability for the `LOAD DATA` statement previously could be controlled on the client side only by enabling it for all files accessible to the client, or by disabling it altogether. The new `OPT_LOAD_DATA_LOCAL_DIR` option enables restricting `LOCAL` data loading to files located in a designated directory. For example, to set the value at connect time:

  ```
  sql::ConnectOptionsMap opt;
  opt[OPT_HOSTNAME] = "localhost";
  opt[OPT_LOAD_DATA_LOCAL_DIR] = "/tmp";

  sql::Connection *conn = driver->connect(opt);
  ```

  `OPT_LOAD_DATA_LOCAL_DIR` can also be set after connect time:

  ```
  sql::ConnectOptionsMap opt;
  opt[OPT_HOSTNAME] = "localhost";

  sql::Connection *conn = driver->connect(opt);

  //.... some queries / inserts / updates

  std::string path= "/tmp";
  conn->setClientOption(OPT_LOAD_DATA_LOCAL_DIR, path);

  // LOAD LOCAL DATA DIR
  ...

  //Disable LOCAL INFILE by setting to null
  conn->setClientOption(OPT_LOAD_DATA_LOCAL_DIR, nullptr);
  ```

  The `OPT_LOAD_DATA_LOCAL_DIR` option maps onto the `MYSQL_OPT_LOAD_DATA_LOCAL_DIR` option for the `mysql_options()` C API function. For more information, see Security Considerations for LOAD DATA LOCAL. (WL #13884)

Bugs Fixed

- String decoding failed for utf-8 strings that began with a `\xEF` byte-order mark. (Bug #31656092, Bug #100292)
- With the `CLIENT_MULTI_FLAG` option enabled, executing multiple statements in a batch caused the next query to fail with a `Commands out of sync` error. (Bug #31399362)
- For connections made using X Plugin, connections over Unix socket files did not work. (Bug #31329938)
- For connections made using X Plugin, the default compression mode was `DISABLED` rather than `PREFERRED`. (Bug #31173447)

Changes in MySQL Connector/C++ 8.0.21 (2020-07-13, General Availability)

Configuration Notes

- The `CMake` configuration files were revised to work better when Connector/C++ is used as a subproject of application projects. Thanks to Lou Shuai for the contribution.

  This revision does not change the fact that the intended (and supported) usage scenario is to build the connector in a separate project, install it somewhere and then use it in application projects from that installed location (for example, by defining the imported library target in `CMake`). (Bug #31095993, Bug #99093)

JSON Notes

- The `rapidjson` library included with Connector/C++ has been upgraded to the GitHub snapshot of 16 January 2020. (WL #13877)

Security Notes

- For platforms on which OpenSSL libraries are bundled, the linked OpenSSL library for Connector/C++ has been updated to version 1.1.1g. Issues fixed in the new OpenSSL version are described at https://www.openssl.org/news/cl111.txt and https://www.openssl.org/news/vulnerabilities.html. (Bug #31296689)

X DevAPI Notes

- For X DevAPI or X DevAPI for C applications, methods and functions that create or modify collections now accept options to enable validation of a JSON schema that documents must adhere to before they are permitted to be inserted or updated. Schema validation is performed by the server, which returns an error message if a document in a collection does not match the schema definition or if the server does not support validation.

  These new classes are implemented to support validation options:

  - `CollectionOptions`: The base class that has all options. Contains an `Option` enumeration with constants `REUSE` and `VALIDATION`.

- `CollectionValidation`: Handles only subkey validation. Contains a `Level` enumeration with constants `STRICT` and `OFF`, and an `Option` enumeration with constants `SCHEMA` and `LEVEL`.

X DevAPI now has these signatures for `createCollection()` (and `modifyCollection()`):

```
// Accepts the full document
createCollection("name", "JSON_Document");

// DbDoc usage is also permitted
createCollection("name", DbDoc("validation", DbDoc("level","off",...)));

// List of pairs with Option enum constant and value
createCollection(CollectionOptions::REUSE, true,
                 CollectionOptions::VALIDATION, CollectionValidation(...));

// Old REUSE way is also acceptable
createCollection("name", true);

createCollection("name", CollectionValidation(...));

// createCollection also allows a list of pairs of
// CollectionValidation Option enum constant and value
createCollection("name", CollectionOptions::VALIDATION,
                         CollectionValidation::OFF,
                         CollectionValidation::SCHEMA,
                         "Object");
```

X DevAPI for C examples:

These are the possible options for `mysqlx_collection_options_set()`:

```
OPT_COLLECTION_VALIDATION_LEVEL(VALIDATION_OFF)
OPT_COLLECTION_VALIDATION_SCHEMA("Object")
OPT_COLLECTION_REUSE(true)
```

Perform option operations like this:

```
mysqlx_collection_options_t
*options = mysqlx_collection_options_new() //creates collection options object
mysqlx_free(options) //frees collection options
mysqlx_collection_options_set(options,...);
```

Creation and modification functions have these signatures:

```
mysqlx_collection_create_with_options(mysqlx_schema_t *schema,
                                      mysqlx_collection_options_t *options);

mysqlx_collection_create_with_json_options(mysqlx_schema_t *schema,
                                           const char* json_options);

mysqlx_collection_modify_with_options(mysqlx_schema_t *schema,
                                      mysqlx_collection_options_t *options);

mysqlx_collection_modify_with_json_options(mysqlx_schema_t *schema,
                                           const char* json_options);
```

For modifications, the `REUSE` option is not supported and an error occurs if it is used. (WL #13061)

Functionality Added or Changed

- The `MySQL_Connection_Options` enumeration is no longer sensitive to the order in which the underlying options are declared in the C API source. (Bug #30799197)
- Connector/C++ now implements blocking of failed connection-pool endpoints to prevent them from being reused until a timeout period has elapsed. This should reduce average wait time for applications to obtain a connection from the pool in the event that endpoints become temporarily unavailable. (WL #13701)

Bugs Fixed

- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C) on a system that does not have OpenSSL libraries installed, the libraries that are bundled with Connector/C++ could not be found when resolving run-time dependencies of the application. (Bug #31007317)
- For a reply with multiple result sets (such as the result from a stored procedure that executed multiple queries), an error in reply processing logic could incorrectly deregister the reply object after reading the first result set while more result sets from the server were pending, resulting in an application error. (Bug #30989042)

Changes in MySQL Connector/C++ 8.0.20 (2020-04-27, General Availability)

Connection Management Notes

- For connections made using X Plugin, Connector/C++ now provides control over the use of compression to minimize the number of bytes sent over connections to the server. Connection URIs and `SessionSettings` objects permit explicitly specifying a compression option:
  - URI strings permit a `compression` option with permitted values of `DISABLED`, `PREFERRED`, and `REQUIRED` (not case-sensitive). Examples:

    ```
    mysqlx://user:password@host:port/db?compression=DISABLED
    mysqlx://user:password@host:port/db?compression=PREFERRED
    mysqlx://user:password@host:port/db?compression=REQUIRED
    ```

  - `SessionSettings` objects permit a `SessionOption::COMPRESSION` option with permitted values of `CompressionMode::DISABLED`, `CompressionMode::PREFERRED`, and `CompressionMode::REQUIRED`. Example:

    ```
    mysqlx::Session sess(SessionOption::USER, "user_name",
                         SessionOption::PWD, "password",
                         SessionOption::COMPRESSION, CompressionMode::PREFERRED);
    ```

  These rules apply:

  - If compression is disabled, the connection is uncompressed.
  - If compression is preferred, Connector/C++ and the server negotiate to find a compression algorithm they both permit. If no common algorithm is available, the connection is uncompressed. This is the default mode if not specified explicitly.
  - If compression is required, compression algorithm negotiation occurs as for preferred mode, but if no common algorithm is available, the connection request terminates with an error.

  To avoid CPU inefficiency, data packets are not compressed even when compression is enabled unless they exceed a threshold size (currently 1000 bytes; this is subject to change).

  See also Connection Compression with X Plugin. (WL #12150)

Packaging Notes

- Previously, Connector/C++ binary distributions were compatible with projects built using MSVC 2019 (using either dynamic or static connector libraries) or MSVC 2017 (using dynamic connector libraries only). Binary distributions now are also compatible with MSVC 2017 using the static X DevAPI connector library. This means that binary distributions are fully compatible with MSVC 2019, and fully

compatible with MSVC 2017 with the exception of the static legacy (JDBC) connector library. (WL #13729)

Bugs Fixed

- For connections made using X Plugin, the last byte was removed from `DATETIME` values fetched as raw bytes. (Bug #30838230)
- In X DevAPI expressions, Connector/C++ treated the JSON `->>` operator the same as `->`, rather than applying an additional `JSON_UNQUOTE()` operation. (Bug #29870832)
- Comparison of `JSON` values from query results failed due to an extra `\0` character erroneously being added to the end of such values. (Bug #29847865)
- For connections made using X Plugin, warnings sent following result sets were not captured, and were thus unavailable to `getWarnings()`. (Bug #28047970)

Changes in MySQL Connector/C++ 8.0.19 (2020-01-13, General Availability)

Error Handling

- If an application tries to obtain a result set from a statement that does not produce one, an exception occurs. For applications that do not catch such exceptions, Connector/C++ now produces a more informative error message to indicate why the exception occurred. (Bug #28591814, Bug #92263)

Legacy (JDBC API) Notes

- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), it is now possible when creating a new session to specify multiple hosts to be tried until a successful connection is established. A list of hosts can be given in the session creation options.

  The new `OPT_MULTI_HOST` option is disabled by default for backward compatibility, but if enabled in the `ConnectionOptionsMap` parameter passed to `connect()` calls, it permits other map parameters to specify multiple hosts. Examples:

```
sql::ConnectOptionsMap opts;
opts["hostName"]="host1,host2:13001,localhost:13000";
opts["schema"]="test";
opts["OPT_MULTI_HOST"] = true;
opts["userName"]="user";
opts["password"]="password";
driver->connect(opts);
```

```
sql::ConnectOptionsMap opts;
opts["hostName"]="tcp://host1,host2:13001,localhost:13000/test";
opts["OPT_MULTI_HOST"] = true;
opts["userName"]="user";
opts["password"]="password";
driver->connect(opts);
```

```
sql::ConnectOptionsMap opts;
opts["hostName"]="mysql://host1,host2:13001,localhost:13000/test";
opts["OPT_MULTI_HOST"] = true;
opts["userName"]="user";
```

```
opts["password"]="password";
driver->connect(opts);
```

Port values are host specific. If a host is specified without a port number, the default port is used.

These rules apply:

- If `OPT_MULTI_HOST` is disabled and multiple hosts are specified, an error occurs.
- If `OPT_MULTI_HOST` is disabled and a single host that resolves to multiple hosts is specified, the first host is used for backward compatibility.
- If `OPT_MULTI_HOST` is enabled and multiple hosts are specified, one of them is randomly chosen as the connection target. If the target fails, another host is randomly chosen from those that remain. If all targets fail, an error occurs.
- The `hostName` parameter can accept a URI that contains a list of comma-separated hosts. The URI scheme can be `mysql://`, which works like `tcp://`. The URI scheme can also be omitted, so the parameter can be a list of comma-separated hosts.
- The `connect()` syntax that takes URI, user, and password parameters does not permit multiple hosts because in that case `OPT_MULTI_HOST` is disabled.

(WL #13322)

Packaging Notes

- Connector/C++ now is compatible with MSVC 2019, while retaining compatibility with MSVC 2017:
  - Previously, Connector/C++ binary distributions were compatible with projects built using MSVC 2017 or 2015. Binary distributions now are compatible with projects built using MSVC 2019 (using either dynamic or static connector libraries) or MSVC 2017 (using dynamic connector libraries). Building using MSVC 2015 might work, but is not supported.
  - Previously, Connector/C++ source distributions could be built using MSVC 2017 or 2015. Source distributions now can be built using MSVC 2019 or 2017. Building using MSVC 2015 might work, but is not supported.
  - Previously, the MSI installer accepted the Visual C++ Redistributable for Visual Studio 2017 or 2015. The MSI installer now accepts the Visual C++ Redistributable for Visual Studio 2019 or 2017.

  (WL #13563)

X DevAPI Notes

- For X DevAPI or X DevAPI for C applications, Connector/C++ now provides options that enable specifying the permitted TLS protocols and ciphersuites for TLS connection negotiation:
  - TLS protocols must be chosen from this list: TLSv1, TLSv1.1, TLSv1.2, TLSv1.3. (TLSv1.3 requires that both the server and Connector/C++ be compiled with OpenSSL 1.1.1 or higher.)
  - Ciphersuite values must be IANA ciphersuite names.

  TLS protocols and ciphersuites now may be specified in these contexts:

  - Connection strings permit `tls-versions` and `tls-ciphersuites` options. The `tls-versions` value is a list of one or more comma-separated TLS protocol versions. The `tls-ciphersuites` value is a list of one or more comma-separated ciphersuite names. Examples:

    ```
    ...?tls-versions=[TLSv1.3]&...
    ...?tls-versions=[TLSv1.2,TLSv1.3]&...
    ```

```
...?tls-ciphersuites=[
     TLS_DHE_PSK_WITH_AES_128_GCM_SHA256,
     TLS_CHACHA20_POLY1305_SHA256
   ]&...
```

- `SessionSettings` objects permit `TLS_VERSIONS` and `TLS_CIPHERSUITES` options. Each value is either a string containing one or more comma-separated items or a container with strings (that is, any type that can be iterated with a loop that yields string values).

  Example of single string values:

```
Session s(...,
  TLS_VERSIONS, "TLSv1.2,TLSv1.3",
  TLS_CIPHERSUITES,
    "TLS_DHE_PSK_WITH_AES_128_GCM_SHA256,TLS_CHACHA20_POLY1305_SHA256",
...);
```

  Example of string container values:

```
std::list<std::string> tls_versions = {
  "TLSv1.2",
  "TLSv1.3"
};

std::list<std::string> ciphers = {
  "TLS_DHE_PSK_WITH_AES_128_GCM_SHA256",
  "TLS_CHACHA20_POLY1305_SHA256"
};

Session s(...,
  TLS_VERSIONS, tls_versions
  TLS_CIPHERSUITES, ciphers,
...);

Session s(...,
  TLS_VERSIONS, std::vector{"TLSv1.2","TLSv1.3"},
  TLS_CIPHERSUITES, std::vector{"TLS_DHE_PSK_WITH_AES_128_GCM_SHA256", "TLS_CHACHA20_POLY1305_SHA256"},
...);
```

- `mysqlx_session_option_set()` and friends permit `MYSQLX_OPT_TLS_VERSIONS` and `MYSQLX_OPT_TLS_CIPHERSUITES` session option constants, together with the corresponding `OPT_TLS_VERSIONS()` and `OPT_TLS_CIPHERSUITES()` macros. `MYSQLX_OPT_TLS_VERSIONS` and `MYSQLX_OPT_TLS_CIPHERSUITES` accept a string containing one or more comma-separated items. Examples:

```
mysqlx_session_option_set(opts, ...,
  OPT_TLS_VERSIONS("TLSv1.2,TLSv1.3"),
  OPT_TLS_CIPHERSUITES(
    "TLS_DHE_PSK_WITH_AES_128_GCM_SHA256,TLS_CHACHA20_POLY1305_SHA256"
  ),
...)
```

  For more information about TLS protocols and ciphersuites in MySQL, see Encrypted Connection TLS Protocols and Ciphers. (Bug #28964583, Bug #93299, WL #12755)

- For X DevAPI or X DevAPI for C applications, when creating a new connection (given by a connection string or other means), if the connection data contains several target hosts that have no explicit priority assigned, the behavior of the failover logic now is the same as if all those target hosts have the same priority. That is, the next candidate for making a connection is chosen randomly from the remaining available hosts.

  This is a change from previous behavior, where hosts with no explicit priority were assigned implicit decreasing priorities and tried in the same order as listed in the connection data. (WL #13497)

Functionality Added or Changed

- Connector/C++ now supports the use of DNS SRV records to specify multiple hosts:

- Session and session-pool creation accepts a URI scheme of `mysqlx+srv://` that enables the DNS SRV feature in connect strings. Example:

```
mysqlx+srv://_mysql._tcp.host1.example.com/db?options
```

- For X DevAPI, `mysqlx::Session` objects permit a `SessionOption::DNS_SRV` entry to enable use of a DNS SRV record to specify available services. Example:

```
mysqlx::Session sess(
    SessionOption::HOST, "_mysql._tcp.host1.example.com",
    SessionOption::DNS_SRV, true,
    SessionOption::USER, "user",
    SessionOption::PWD, "password");
```

  Similarly, for X DevAPI for C, the `mysqlx_session_option_set()` function permits an `OPT_DNS_SRV()` option in the argument list. Example:

```
mysqlx_session_option_set(opt,
    OPT_HOST("_mysql._tcp.host1.example.com"),
    OPT_DNS_SRV(true)
    OPT_USER("user"),
    OPT_PWD("password"),
    PARAM_END));
```

- For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), connection maps permit an `OPT_DNS_SRV` element. A map should specify the host for SRV lookup as a full lookup name and without a port. Example:

```
sql::ConnectOptionsMap opts;
opts["hostName"] = "_mysql._tcp.host1.example.com";
opts["OPT_DNS_SRV"] = true;
opts["userName"] = "user";
opts["password"] = "password";
driver->connect(opts);
```

  In legacy applications, DNS SRV resolution cannot be enabled in URI connect strings because parameters are not supported in such strings.

(WL #13344, WL #13402)

Bugs Fixed

- Connector/C++ failed to compile using Clang on Linux. (Bug #30450484)
- Connector/C++ set the transaction isolation level to `REPEATABLE READ` at connect time, regardless of the current server setting. (Bug #22038313, Bug #78852, Bug #30294415)

Changes in MySQL Connector/C++ 8.0.18 (2019-10-14, General Availability)

Compilation Notes

- It is now possible to compile Connector/C++ using OpenSSL 1.1. (WL #13162)
- Connector/C++ no longer supports using wolfSSL as an alternative to OpenSSL. All Connector/C++ builds now use OpenSSL. (WL #13343)

Changes in MySQL Connector/C++ 8.0.17 (2019-07-22, General Availability)

Character Set Support

- Connector/C++ now supports the `utf8mb4_0900_bin` collation added for the `utf8mb4` Unicode character set in MySQL 8.0.17. For more information about this collation, see Unicode Character Sets. (WL #13094, WL #13502)

Compilation Notes

- Connector/C++ now compiles cleanly using the C++14 compiler. This includes MSVC 2017. Binary distributions from Oracle are still built in C++11 mode using MSVC 2015 for compatibility reasons. (WL #13133)

Configuration Notes

- The maximum permitted length of host names throughout Connector/C++ has been raised to 255 ASCII characters, up from the previous limit of 60 characters. Applications that expect host names to be a maximum of 60 characters should be adjusted to account for this change. (WL #13092)

Function and Operator Notes

- Connector/C++ now supports the `OVERLAPS` and `NOT OVERLAPS` operators for expressions on JSON arrays or objects:

  ```
  expr OVERLAPS expr
  expr NOT OVERLAPS expr
  ```

  Suppose that a collection has these contents:

  ```
  [{
     "_id": "1",
     "list": [1, 4]
   }, {
     "_id": "2",
     "list": [4, 7]
  }]
  ```

  This operation:

  ```
  auto res = collection.find("[1, 2, 3] OVERLAPS $.list").fields("_id").execute();
  res.fetchAll();
  ```

  Should return:

  ```
  [{ "_id": "1" }]
  ```

  This operation:

  ```
  auto res = collection.find("$.list OVERLAPS [4]").fields("_id").execute();
  res.fetchAll();
  ```

  Should return:

  ```
  [{ "_id": "1" }, { "_id": "2" }]
  ```

  An error occurs if an application uses either operator and the server does not support it. (WL #12721)

X DevAPI Notes

- For index specifications passed to the `Collection::createIndex()` method (for X DevAPI applications) or the `mysqlx_collection_create_index()` function (for X DevAPI for C applications), Connector/C++ now supports indexing array fields. A single index field description can contain a new member name `array` that takes a `Boolean` value. If set to `true`, the field is assumed to contain arrays of elements of the given type. For example:

```
coll.createIndex("idx",
  R"({ "fields": [{ "field": "foo", "type": "INT", "array": true }] })"
);
```

  In addition, the set of possible index field data types (used as values of member `type` in index field descriptions) is extended with type `CHAR(N)`, where the length `N` is mandatory. For example:

```
coll.createIndex("idx",
  R"({ "fields": [{ "field": "foo", "type": "CHAR(10)" }] })"
);
```

  (WL #12151)

Functionality Added or Changed

- Previously, Connector/C++ reported `INT` in result set metadata for all integer result set columns, which required applications to check column lengths to determine particular integer types. The metadata now reports the more-specific `TINYINT`, `SMALLINT`, `MEDIUMINT`, `INT`, and or `BIGINT` types for integer columns. (Bug #29525077)

Bugs Fixed

- Calling a method such as `.fields()` or `.sort()` on existing objects did not overwrite the effects of any previous call. (Bug #29402358)
- When Connector/C++ applications reported connection attributes to the server upon establishing a new connection, some attributes were taken from the host on which Connector/C++ was built, not the host on which the application was being run. Now application host attributes are sent. (Bug #29394723)
- Assignments of the following form on `CollectionFind` objects invoked a copy assignment operator, which was nonoptimal and prevented potential re-execution of statements using prepared statements:

```
find = find.limit(1);
```

  (Bug #29390170)

- Legal constructs of this form failed to compile:

```
for (string id : res.getGeneratedIds()) { ... }
```

  (Bug #29355100)

- During build configuration, `CMake` could report an incorrect OpenSSL version. (Bug #29282948)

Changes in MySQL Connector/C++ 8.0.16 (2019-04-25, General Availability)

- Character Set Support
- Compilation Notes
- Configuration Notes
- Packaging Notes
- Prepared Statement Notes

- X DevAPI Notes
- X DevAPI for C Notes
- Functionality Added or Changed
- Bugs Fixed

Character Set Support

- Connector/C++ supports all Unicode character sets for connections to servers for MySQL 8.0.14 and higher, but previously had Unicode support limited to the `utf8` character set for servers older than MySQL 8.0.14. Connector/C++ now supports all Unicode character sets for older servers, including `utf8mb4`, `utf16`, `utf16le`, `utf32`, and `ucs2`. (Bug #28966038, WL #12196)

Compilation Notes

- Thanks to Daniël van Eeden, who contributed a code change to use the `stdbool.h` header file rather than a `bool` typedef. (Bug #29167120, Bug #93803)
- Thanks to Daniël van Eeden, who contributed a code change to use `lib` instead of `lib64` on 64-bit FreeBSD. (Bug #29167098, Bug #93801)
- Previously, for Connector/C++ applications that used the legacy JDBC API, source files had to use this set of `#include` directives:

  ```
  #include <jdbc/mysql_driver.h>
  #include <jdbc/mysql_connection.h>
  #include <jdbc/cppconn/*.h>
  ```

  Now a single `#include` directive suffices:

  ```
  #include <mysql/jdbc.h>
  ```

  (WL #12786)

Configuration Notes

- Thanks to Daniël van Eeden, who contributed a code change to build the documentation as part of the `all` target if Connector/C++ is configured with `-DWITH_DOC=ON`. (Bug #29167107, Bug #93802)
- Previously, for Connector/C++ 8.0 applications that use the legacy JDBC connector, only static linking to the MySQL client library was supported. The `MYSQLCLIENT_STATIC_LINKING` and `MYSQLCLIENT_STATIC_BINDING` `CMake` options are now available to permit dynamic linking. By default, `MYSQLCLIENT_STATIC_LINKING` is enabled, to use static linking to the client library. Disable this option to use dynamic linking. If `MYSQLCLIENT_STATIC_LINKING` is enabled, `MYSQLCLIENT_STATIC_BINDING` may also be used. If `MYSQLCLIENT_STATIC_BINDING` is enabled (the default), Connector/C++ is linked to the shared MySQL client library. Otherwise, the shared MySQL client library is loaded and mapped at runtime. (WL #12730)
- Connector/C++ 8.0 configuration now requires a minimum `CMake` version of 3.0. (WL #12753)

Packaging Notes

- Connector/C++ debug packages are now available for Linux and Windows. The packages enable symbolic debugging using tools such as `gdb` on Linux and `windbg` on Windows, as well as obtaining symbolic information about connector code locations from application crash dumps. Use of the debug packages requires that you have installed and configured the Connector/C++ sources. (Bug #29117059, Bug #93645, Bug #26128420, Bug #86415, WL #12263)
- For improved GitHub friendliness, Community Connector/C++ source distributions now include a `CONTRIBUTING.md` markdown file that contains guidelines intended to be helpful to contributors. (WL #12791)

- The Protobuf sources bundled in the Connector/C++ source tree were updated to Protobuf 3.6.1. (Only the parts needed for Connector/C++ are included, to reduce compilation time.) (WL #12889)

Prepared Statement Notes

- For X DevAPI and X DevAPI for C, performance for statements that are executed repeatedly (two or more times) is improved by using server-side prepared statements for the second and subsequent executions. This happens internally; applications need take no action and API behavior should be the same as previously. For statements that change, repreparation occurs as needed. Providing different data values or different `OFFSET` or `LIMIT` clause values does not count as a change. Instead, the new values are passed to a new invocation of the previously prepared statement. (WL #12149)

X DevAPI Notes

- For X DevAPI and X DevAPI for C applications, Connector/C++ now supports the ability to send connection attributes (key-value pairs that application programs can pass to the server at connect time). Connector/C++ defines a default set of attributes, which can be disabled or enabled. In addition, applications can specify attributes to be passed in addition to the default attributes. The default behavior is to send the default attribute set.

  - For X DevAPI applications, specify connection attributes as a `connection-attributes` parameter in a connection string, or by using a `SessionOption::CONNECTION_ATTRIBUTES` option for the `SessionSettings` constructor.

    The `connection-attributes` parameter value must be empty (the same as specifying `true`), a `Boolean` value (`true` or `false` to enable or disable the default attribute set), or a list or zero or more `key=value` specifiers separated by commas (to be sent in addition to the default attribute set). Within a list, a missing key value evaluates as an empty string. Examples:

    ```
    "mysqlx://user@host?connection-attributes"
    "mysqlx://user@host?connection-attributes=true"
    "mysqlx://user@host?connection-attributes=false"
    "mysqlx://user@host?connection-attributes=[attr1=val1,attr2,attr3=]"
    "mysqlx://user@host?connection-attributes=[]"
    ```

    The `SessionOption::CONNECTION_ATTRIBUTES` option value must be a `Boolean` value (`true` or `false` to enable or disable the default attribute set), or a `DbDoc` or `JSON` string (to be sent in addition to the default attribute set). Examples:

    ```
    Session sess(..., SessionOption::CONNECTION_ATTRIBUTES, false);
    Session sess(..., SessionOption::CONNECTION_ATTRIBUTES, attr_doc );
    Session sess(..., SessionOption::CONNECTION_ATTRIBUTES,
      R"({ "attr1": "val1", "attr2" : "val2" })"
    );
    ```

  - For X DevAPI for C applications, specify connection attributes using the `OPT_CONNECTION_ATTRIBUTES()` macro for the `mysqlx_session_option_set()` function. The option value must be null (to disable the default attribute set) or a `JSON` string (to be sent in addition to the default attribute set). Examples:

    ```
    mysqlx_session_option_set(opts, OPT_CONNECTION_ATTRIBUTES(nullptr));
    mysqlx_session_option_set(opts,
      OPT_CONNECTION_ATTRIBUTES("{ \"attr1\": \"val1\", \"attr2\" : \"val2\" }")
    );
    ```

  Application-defined attribute names cannot begin with `_` because such names are reserved for internal attributes.

  If connection attributes are not specified in a valid way, an error occurs and the connection attempt fails.

  For general information about connection attributes, see Performance Schema Connection Attribute Tables. (WL #12495)

X DevAPI for C Notes

- The signatures for several X DevAPI for C functions have been changed to enable better error information to be returned to applications by means of a `mysqlx_error_t` handle. These functions are affected:

```
mysqlx_client_t*
mysqlx_get_client_from_url(
  const char *conn_string,
  const char *client_opts,
  mysqlx_error_t **error
)

mysqlx_client_t*
mysqlx_get_client_from_options(
  mysqlx_session_options_t *opt,
  mysqlx_error_t **error
)

mysqlx_session_t*
mysqlx_get_session(
  const char *host, int port,
  const char *user, const char *password,
  const char *database,
  mysqlx_error_t **error
)

mysqlx_session_t*
mysqlx_get_session_from_url(
  const char *conn_string,
  mysqlx_error_t **error
)

mysqlx_session_t*
mysqlx_get_session_from_options(
  mysqlx_session_options_t *opt,
  mysqlx_error_t **error
)

mysqlx_session_t *
mysqlx_get_session_from_client(
  mysqlx_client_t *cli,
  mysqlx_error_t **error
)
```

  The final argument in each case is a `mysqlx_error_t` handle into which Connector/C++ stores error information. If the argument is a null pointer, Connector/C++ ignores it. The application is responsible to free non-null handles by passing them to `mysqlx_free()`.

  The signature for `mysqlx_free()` has also been changed to accept a `void *` argument so that it can accept a handle of any type. Consequently, other type-specific free functions, such as `mysqlx_free_options()`, are no longer needed and are deprecated.

  The preceding modifications change the Connector/C++ API, which has these implications:

  - The modifications change the ABI, so the ABI version is changed from 1 to 2. This changes the connector library names.
  - X DevAPI for C applications to be compiled against the new API must be modified to use the new function signatures. (X DevAPI applications should build without change.)
  - Applications built against the old ABI will not run with the new connector library.
  - The API change and ABI version change do not affect the legacy JDBC interface, so library names for the legacy JDBC connector library do not change and legacy application need not be changed.

- It is possible to install both the old and new libraries. However, installers may remove the old libraries, so they may need to be re-added manually after installing the new libraries.

(WL #11654, WL #12751)

Functionality Added or Changed

- Thanks to Daniël van Eeden, who contributed documentation for the `mysqlx_column_get_collation()` function and various corrections to the developer documentation. (Bug #29123114, Bug #93665, Bug #29115285, Bug #93640, Bug #29122490, Bug #93663)
- Connector/C++ now has improved support for resetting sessions in connection pools. Returning a session to the pool drops session-related objects such as temporary tables, session variables, and transactions, but the connection remains open and authenticated so that reauthentication is not required when the session is reused. (WL #12497)

Bugs Fixed

- Previously, for the `SSL_MODE_VERIFY_IDENTITY` connection option, Connector/C++ checked whether the host name that it used for connecting matched the Common Name value in the certificate but not the Subject Alternative Name value. Now, if used with OpenSSL 1.0.2 or higher, Connector/C++ checks whether the host name matches either the Subject Alternative Name value or the Common Name value in the server certificate. (Bug #28964313, Bug #93301)
- After repeated calls, `mysqlx_get_session_from_client()` could hang. (Bug #28587287)
- The `SessionSettings`/`ClientSettings` iterator implementation was incomplete. (Bug #28502574)

Changes in MySQL Connector/C++ 8.0.15 (2019-02-01, General Availability)

This release contains no functional changes, and is published to align its version number with that of the MySQL Server 8.0.15 release.

Changes in MySQL Connector/C++ 8.0.14 (2019-01-21, General Availability)

- Configuration Notes
- Packaging Notes
- X DevAPI Notes

Configuration Notes

- These `CMake` options have been added to enable more fine-grained specification of installation directories. All are relative to `CMAKE_INSTALL_PREFIX`:
  - `CMAKE_INSTALL_LIBDIR`: Library installation directory.
  - `CMAKE_INSTALL_INCLUDEDIR`: Header file installation directory.
  - `CMAKE_INSTALL_DOCDIR`: Documentation installation directory.

  (Bug #28045358)

Packaging Notes

- Previously, Connector/C++ binary distributions included a `BUILDINFO.txt` file that contained information about the build environment used to produce the distribution. Binary distributions now

include a file named `INFO_BIN` that provides similar information, and an `INFO_SRC` file that provides information about the product version and the source repository from which the distribution was produced. Source distributions include the `INFO_SRC` file only. (WL #12293)

- Connector/C++ now is compatible with MSVC 2017, while retaining compatibility with MSVC 2015:

  - Previously, Connector/C++ binary distributions were compatible with projects built using MSVC 2015. Binary distributions now are compatible with projects built using MSVC 2017 or 2015. DLLs have a `-vs14` suffix in their names to reflect that they are compatible with MSVC 2015, but can also be used in MSVC 2017 projects.

  - Previously, Connector/C++ source distributions could be built using MSVC 2015. Source distributions now can be built using MSVC 2017 or 2015.

  - Previously, the MSI installer accepted the Visual C++ Redistributable for Visual Studio 2015. The MSI installer now accepts the Visual C++ Redistributable for Visual Studio 2017 or 2015.

  (WL #12611)

- Installers for Connector/C++ are now available as Debian packages. See Installing Connector/C++ from a Binary Distribution. (WL #12101)

X DevAPI Notes

- Connector/C++ now provides collection counting methods for applications that use X DevAPI for C:

  - `mysqlx_collection_count()`: The number of documents in a collection without filtering.

    ```
    mysqlx_collection_t *c1 = mysqlx_get_collection(schema, "c1", 1);
    ulong64_t documents;
    mysqlx_collection_count(c1, &documents);
    ```

  - `mysqlx_table_count()`: The number of rows in a table without filtering.

    ```
    mysqlx_table_t *t1 = mysqlx_get_table(schema, "t1", 1);
    ulong64_t rows;
    mysqlx_table_count(t1, &rows);
    ```

  - `mysqlx_get_count()`: The number of remaining cached rows held at the moment. After a row is consumed by a fetch function, the number of cached rows decreases.

    ```
    mysqlx_stmt_t *stmt = mysqlx_sql_new(session, query, strlen(query));
    mysqlx_result_t *res = mysqlx_execute(stmt);

    ulong64_t row_count;
    mysqlx_get_count(res, &row_count);
    ```

    `mysqlx_get_count()` is similar in all respects to `mysqlx_store_result()` except that the behavior differs after fetching rows when reaching zero number of rows in the cache:

    - `mysqlx_get_count()` returns zero through the parameter and finishes with `RESULT_OK`.

    - `mysqlx_store_result()` does not return anything through the parameter (which remains unchanged) and finishes with `RESULT_ERROR`.

  (WL #12496)

Changes in MySQL Connector/C++ 8.0.13 (2018-10-22, General Availability)

- Legacy (JDBC API) Notes
- Packaging Notes
- X DevAPI Notes

- Functionality Added or Changed
- Bugs Fixed

Legacy (JDBC API) Notes

- For connections to the server made using the legacy JDBC API (that is, not made using X DevAPI or X DevAPI for C), the default connection character set is now `utf8mb4` rather than `utf8`. Connections to the server made using X DevAPI or X DevAPI for C continue to use the connection character set determined by the server. (Bug #28204677)

Packaging Notes

- Connector/C++ 32-bit MSI packages are now available for Windows. These 32-bit builds enable use of the legacy JDBC connector. (WL #12262)
- Connector/C++ compressed `tar` file packages are now available for Solaris.

  It is also possible to build Connector/C++ from source on Solaris. For platform-specific build notes, see Building Connector/C++ Applications: Platform-Specific Considerations. (WL #11655)

X DevAPI Notes

- Connector/C++ now provides connection pooling for applications using X Protocol. This capability is based on client objects, a new type of X DevAPI object. A client can be used to create sessions, which take connections from a pool managed by that client. For a complete description, see Connecting to a Single MySQL Server Using Connection Pooling.

  X DevAPI example:

  ```
  using namespace mysqlx;

  Client cli("user:password@host_name/db_name", ClientOption::POOL_MAX_SIZE, 7);
  Session sess = cli.getSession();

  // use sess as before

  cli.close();  // close session sess
  ```

  X DevAPI for C example:

  ```
  char error_buf[255];
  int  error_code;

  mysqlx_client_t *cli
   = mysqlx_get_client_from_url(
       "user:password@host_name/db_name", "{ \"maxSize\": 7 }", error_buf, &error_code
     );
  mysqlx_session_t *sess = mysqlx_get_session_from_client(cli);

  // use sess as before

  mysqlx_close_client(cli);  // close session sess
  ```

  (WL #11929)

- For X DevAPI, a new `connect-timeout` option can be specified in connection strings or URIs to indicate a connection timeout in milliseconds. The `SessionSettings::Options` object supports a new `CONNECT_TIMEOUT` option.

  For X DevAPI for C, the `mysqlx_opt_type_t` constant is `MYSQLX_OPT_CONNECT_TIMEOUT` together with the `OPT_CONNECT_TIMEOUT()` macro.

  If no timeout option is specified, the default is 10000 (10 seconds). A value of 0 disables the timeout. The following examples set the connection timeout to 10 milliseconds:

X DevAPI examples:

```
Session sess("user@host/db?connect-timoeut=10");

Session sess(..., SessionOption::CONNECT_TIMEOUT, 10, ...);

Session sess(
  ...,
  SessionOption::CONNECT_TIMEOUT, std::chrono::milliseconds(10),
  ...
);
```

X DevAPI for C example:

```
mysqlx_session_options_t *opt = mysqlx_session_options_new();
mysqlx_session_option_set(opt, ..., OPT_CONNECT_TIMEOUT(10), ...);
```

(WL #12148)

Functionality Added or Changed

- **JSON:** Connector/C++ now uses RapidJSON for improved performance of operations that involve parsing `JSON` strings. There are no user-visible API changes for X DevAPI or X DevAPI for C. (WL #12292)

Bugs Fixed

- On SLES 15, Connector/C++ installation failed if `libmysqlcppcon7` was already installed. (Bug #28658120)
- Applications that were statically linked to the legacy JDBC connector could encounter a read access violation at exit time due to nondeterministic global destruction order. (Bug #28525266, Bug #91820)
- Configuring with `-DCMAKE_BUILD_TYPE=Release` did not work on Linux. (Bug #28045274)
- Field references in `.having()` expressions could be interpreted incorrectly and produce errors. (Bug #26310713)

Changes in MySQL Connector/C++ 8.0.12 (2018-07-27, General Availability)

Installation Notes

- Because the Microsoft Visual C++ 2017 Redistributable installer deletes the Microsoft Visual C++ 2015 Redistributable registry keys that identify its installation, standalone MySQL MSIs may fail to detect the Microsoft Visual C++ 2015 Redistributable if both it and the Microsoft Visual C++ 2017 Redistributable are installed. The solution is to repair the Microsoft Visual C++ 2017 Redistributable via the Windows Control Panel to recreate the registry keys needed for the runtime detection. Unlike the standalone MSIs, MySQL Installer for Windows contains a workaround for the detection problem. (Bug #28345281, Bug #91542)

Packaging Notes

- An RPM package for installing ARM 64-bit (aarch64) binaries of Connector/C++ on Oracle Linux 7 is now available in the MySQL Yum Repository and for direct download.

Known Limitation for this ARM release: You must enable the Oracle Linux 7 Software Collections Repository (ol7_software_collections) to install this package, and must also adjust the libstdc++7 path. See Yum's Platform Specific Notes for additional details.

- Installers for Connector/C++ are now available in these formats: MSI packages (Windows); RPM packages (Linux); DMG packages (macOS). See Installing Connector/C++ from a Binary Distribution. (WL #11670, WL #11671, WL #11672)

Security Notes

- yaSSL is no longer included in Connector/C++ source distributions. wolfSSL may be used as a functionally equivalent alternative that has a GPLv2-compatible license. In addition, wolfSSL (like OpenSSL) supports the TLSv1.2 protocol, which yaSSL does not.

  To build Connector/C++ using wolfSSL, use the `-DWITH_SSL=path_name CMake` option, where *`path_name`* indicates the location of the wolfSSL sources. For more information, see Source Installation System Prerequisites, and Connector/C++ Source-Configuration Options. (WL #11683)

X DevAPI Notes

- Connector/C++ now supports `NOWAIT` and `SKIP LOCKED` lock contention modes to be used with `lockExclusive()` and `lockShared()` clauses of CRUD find/select operations (see Locking Read Concurrency with NOWAIT and SKIP LOCKED), and a default lock contention mode. The following list names the permitted constants. For each item, the first and second constants apply to X DevAPI and X DevAPI for C, respectively.

  - `LockContention::DEFAULT`, `LOCK_CONTENTION_DEFAULT`: Block the query until existing row locks are released.
  - `LockContention::NOWAIT`, `LOCK_CONTENTION_NOWAIT`: Return an error if the lock cannot be obtained immediately.
  - `LockContention::SKIP_LOCKED`, `LOCK_CONTENTION_SKIP_LOCKED`: Execute the query immediately, excluding from the query items that are locked.

  For X DevAPI and X DevAPI for C applications, lock mode methods accept these lock contention constants as a parameter. For X DevAPI applications, lock mode methods can be called without this parameter, as before; this is equivalent to passing a lock mode of `DEFAULT`.

  For more information, see Working with Locking. (WL #11374)

- Connector/C++ now supports the `SHA256_MEMORY` authentication mechanism for connections using the X Protocol. For X DevAPI applications, `SessionOption::AUTH` supports the new value `AuthMethod::SHA256_MEMORY`. For X DevAPI for C applications, the session option `MYSQLX_OPT_AUTH` supports the new value `MYSQLX_AUTH_SHA256_MEMORY`. These new values request using the `sha256_memory` authentication mechanism when creating a session. (WL #11685)

- To increase compliance with the X DevAPI, these Connector/C++ changes were made:

  - `getAffectedItemsCount()` was moved from `Result` to `Result_common`.
  - `Collection.modify(condition).arrayDelete()` was removed.
  - `getAffectedRowsCount()` was removed. Use `getAffectedItemsCount()` instead.
  - `getWarningCount()` was renamed to `getWarningsCount()`.

Bugs Fixed

- `utf8mb4` character data was handled incorrectly. (Bug #28240202)

- `Session` creation had a memory leak. (Bug #27917942)
- When configuring to build Connector/C++ with the legacy connector, `CMake` did not account for the `MYSQL_CONFIG_EXECUTABLE` option. (Bug #27874173, Bug #90389)
- Improper error handling for unknown hosts when creating a session could result in unexpected application exit. (Bug #27868302)
- The `mysqlx_row_fetch_one()` X DevAPI for C function could fail to return for large result set exceeding the maximum packet size. Now such result sets produce an error. (Bug #27732224)

Changes in MySQL Connector/C++ 8.0.11 (2018-04-19, General Availability)

For MySQL Connector/C++ 8.0.11 and higher, Commercial and Community distributions require the Visual C++ Redistributable for Visual Studio 2015 to work on Windows platforms. The Redistributable is available at the Microsoft Download Center; install it before installing Connector/C++.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Incompatible Change:** When documents without an `_id` attribute are added to a collection, the server now automatically generates IDs for them. The server determines the ID format, which should be considered opaque from the API perspective (they are no longer UUID-based). As before, no `_id` attribute is generated if a document already contains one. User-provided document IDs must not conflict with IDs of other documents in the collection.

  This capability requires a MySQL 8.0 GA server. If the server does not support document ID generation, the document-add operation returns an error indicating that document IDs were missing.

  For X DevAPI, the generated IDs resulting from a document-add operation can be obtained using the new `Result.getGeneratedIds()` method, which returns a list. For X DevAPI for C, the generated IDs can be obtained using the new `mysqlx_fetch_generated_id()` function, which returns IDs one by one for successive calls, until it returns `NULL` to indicate no more generated IDs are available. For both X DevAPI and X DevAPI for C, document IDs specified explicitly in added documents are not returned.

  Incompatibility: The `getGeneratedIds()` method replaces `getDocumentId()` and `getDocumentIds()`, which are now removed. The `mysqlx_fetch_generated_id()` function replaces `mysqlx_fetch_doc_id()`, which is now removed.

  For more information, see Working with Document IDs. (WL #11450)

- A patch operation has been implemented that enables specifying a JSON-like object that describes the changes to apply to documents in a collection.

  For X DevAPI, the `CollectionModify` operation supports a new `patch()` clause for patching documents. For X DevAPI for C, there are two new functions: `mysqlx_collection_modify_patch()` directly executes patching on documents in a collection that satisfy given criteria. `mysqlx_set_modify_patch()` adds a patch operation to a modify statement created with the `mysql_collection_modify_new()` function. (WL #11205)

- For connections to the server made using the legacy JDBC API (that is, not made using X DevAPI or X DevAPI for C), Connector/C++ 8.0 now supports an `OPT_GET_SERVER_PUBLIC_KEY` connection option that enables requesting the RSA public key from the server. For accounts that use the `caching_sha2_password` or `sha256_password` authentication plugin, this key can be used during the connection process for RSA key-pair based password exchange with TLS disabled. This capability requires a MySQL 8.0 GA server, and is supported only for Connector/C++ built using OpenSSL. (WL #11719)

Bugs Fixed

- Single-document methods such as `Collection.replaceOne()` did not accept `expr()` as the document specification, but instead treated it as a plain JSON string. (Bug #27677910)
- Compiling X DevAPI and X DevAPI for C test programs failed with an error. (Bug #27610760)
- Connecting with an incorrect `SSL_CA` value could result in a memory leak. (Bug #27434254)
- For debug builds, specifying a document as `_id` raised an assertion rather than producing an error. (Bug #27433969)

Changes in MySQL Connector/C++ 8.0.8 - 8.0.10 (Skipped version numbers)

There are no release notes for these skipped version numbers.

Changes in MySQL Connector/C++ 8.0.7 (2018-02-26, Release Candidate)

In addition to the new APIs introduced in MySQL Connector/C++ 8.0 (X DevAPI and X DevAPI for C), Connector/C++ now also supports the legacy API based on JDBC4. Applications written against the JDBC4-based API of Connector/C++ 1.1 can be also compiled with Connector/C++ 8.0, which is backward compatible with the earlier version. Such code does not require the X Plugin and can communicate with older versions of the MySQL Server using the legacy protocol. This contrasts with X DevAPI and X DevAPI for C applications, which expect MySQL Server 8.0.

The legacy API is implemented as a separate library with the base name `mysqlcppconn` (as opposed to `mysqlcppconn8`) implementing the new APIs. To build the legacy library, you must configure Connector/C++ using the `-DWITH_JDBC=ON CMake` option. For information about using the legacy API, refer to the documentation at https://dev.mysql.com/doc/connector-cpp/en/connector-cpp-getting-started-examples.html.

Deprecation and Removal Notes

- View and table DDL methods have been removed. It is preferable that SQL statements be used for such operations.

  Removed X DevAPI methods:

  ```
  Schema.createView()
  Schema.alterView()
  Schema.dropView()
  Schema.dropTable()
  ```

  Removed X DevAPI data types:

  ```
  Algorithm
  CheckOption
  SQLSecurity
  ```

  Removed X DevAPI for C functions:

  ```
  mysqlx_view_create
  ```

```
mysqlx_view_create_new
mysqlx_view_modify
mysqlx_view_modify_new
mysqlx_view_replace
mysqlx_view_replace_new
mysqlx_view_drop
mysqlx_table_drop
mysqlx_set_view_algorithm
mysqlx_set_view_security
mysqlx_set_view_definer
mysqlx_set_view_check_option
mysqlx_set_view_columns
```

Removed X DevAPI for C enumerations:

```
mysqlx_view_algorithm_t
mysqlx_view_security_t
mysqlx_view_check_option_t
```

Removed X DevAPI for C macros:

```
VIEW_ALGORITHM()
VIEW_SECURITY()
VIEW_DEFINER()
VIEW_CHECK_OPTION()
VIEW_COLUMNS()
VIEW_OPTION_XXX
```

(WL #11375)

Security Notes

- Connector/C++ now supports the `caching_sha2_password` authentication plugin introduced in MySQL 8.0 (see Caching SHA-2 Pluggable Authentication), with these limitations:

  - For X DevAPI or X DevAPI for C applications, only encrypted (SSL) connections can be used to connect to `cached_sha2_password` accounts. For non-SSL connections, it is not possible to use `cached_sha2_password` accounts.

  - For applications that use the legacy JDBC API (that is, not X DevAPI or X DevAPI for C), it is possible to make connections to `cached_sha2_password` accounts in the following scenario:

    - The connection is unencrypted (`OPT_SSL_MODE` is set to `SSL_MODE_DISABLED`).

    - The server public key is given using the "rsaKey" option and no RSA key exchange is used (`OPT_GET_SERVER_PUBLIC_KEY` is set to false).

    If RSA key exchange is enabled, the connection works.

  (WL #11415)

X DevAPI Notes

- It is now possible to use the `Collection` interface to create and drop indexes on document collections.

  X DevAPI example:

```
coll.createIndex("idx",
  R"({
    "fields": [
      { "field": "$.zip", "type": "TEXT(10)" },
      { "field": "$.count", "type": "INT UNSIGNED" }
    ]
  })"
);
```

```
coll.createIndex("loc",
  R"({
    "type": "SPATIAL",
    "fields": [ { "field": "$.coords", "type": "GEOJSON", "srid": 31287 } ]
  })"
);

coll.dropIndex("idx");
```

X DevAPI for C example:

```
ret = mysqlx_collection_create_index(coll, "idx",
  R"({
    "fields": [
      { "field": "$.zip", "type": "TEXT(10)" },
      { "field": "$.count", "type": "INT UNSIGNED" }
    ]
  })"
)

ret = mysqlx_collecton_create_index(coll, "loc",
  R"({
    "type": "SPATIAL",
    "fields": [ { "field": "$.coords", "type": "GEOJSON", "srid": 31287 } ]
  })"
);

mysqlx_collection_drop_index(coll, "idx");
```

(WL #11231)

- It is now possible to use the `Session` interface to create savepoints inside transactions and roll back a transaction to a given savepoint. This interface supports the operations provided by the `SAVEPOINT`, `ROLLBACK TO SAVEPOINT`, and `RELEASE SAVEPOINT` statements. For more information about these statements, see SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT Statements.

  X DevAPI example:

  ```
  sess.startTransaction();
  string point1 = sess.setSavepoint();
  sess.setSavepoint("point2");
  sess.rollbackTo(point1);         // this also removes savepoint "point2"
  string point3 = sess.setSavepoint();
  sess.releaseSavepoint(point3);  // explicitly remove savepoint
  sess.commitTransaction();
  ```

  X DevAPI for C example:

  ```
  mysqlx_trasaction_begin(sess);
  const char *point1 = mysqlx_savepoint_set(sess,NULL);
  mysqlx_savepoint_set(sess,"point2");
  mysqlx_rollback_to(sess,point1);
  const char *point3 = mysqlx_savepoint_set(sess,NULL);
  mysqlx_sevepoint_release(sess,point3);
  mysqlx_transaction_commit(sess);
  ```

  For more information, see Working with Savepoints. (WL #11229)

Functionality Added or Changed

- Connector/C++ now implements TLS connections using the OpenSSL library. It is possible to build Connector/C++ with OpenSSL or the bundled yaSSL implementation of TLS. This is controlled by the `WITH_SSL CMake` option, which takes these values: `bundled` (build using bundled yaSSL code); `system` (build using system OpenSSL library, with the location as detected by `CMake`); *`path_name`* (build using OpenSSL library installed at the named location). For more information, see “How to build code that uses Connector/C++” in the Connector/C++ X DevAPI Reference, available at https://dev.mysql.com/doc/index-connectors.html. (WL #11376)

Bugs Fixed

- `replaceOne()` and similar methods did not correctly detect document ID mismatches. (Bug #27246854)
- Calling `bind()` twice on the same parameter for complex types resulted in empty values. (Bug #26962725)

Changes in MySQL Connector/C++ 8.0.6 (2017-09-28, Development Milestone)

X DevAPI Notes

- A session now can acquire a lock for documents or rows returned by find or select statements, to prevent the returned values from being changed from other sessions while the lock is held (provided that appropriate isolation levels are used). Locks can be requested several times for a given find or select statement. Only the final request is acted upon. An acquired lock is held until the end of the current transaction.

  For X DevAPI, `CollectionFind` and `TableSelect` implement `.lockExclusive()` and `.lockShared()` methods, which request exclusive or shared locks, respectively, on returned documents or rows. These methods can be called after `.bind()` and before the final `.execute()`.

  For X DevAPI for C, the new `mysqlx_set_locking(stmt, lock)` function can be called to request exclusive or shared locks on returned documents or rows, or to release locks. The *`lock`* parameter can be `ROW_LOCK_EXCLUSIVE`, `ROW_LOCK_SHARED`, or `ROW_LOCK_NONE`. The first two values specify a type of lock to be acquired. `ROW_LOCK_NONE` removes any row locking request from the statement. (WL #10980)

- For X DevAPI, a new `auth` option can be specified in connection strings or URIs to indicate the authentication mechanism. Permitted values are `PLAIN` and `MYSQL41`. The option name and value are not case sensitive. The `SessionSettings::Options` object supports a new `AUTH` enumeration, with the same permitted values.

  For X DevAPI for C, a new `auth` setting can be specified in connection strings or URIs to indicate the authentication mechanism. Permitted values are `PLAIN` and `MYSQL41`. The option name and value are not case sensitive. A new `MYSQLX_OPT_AUTH` constant is recognized by the `mysqlx_options_set()` function, with permitted values `MYSQLX_AUTH_PLAIN` and `MYSQLX_AUTH_MYSQL41`.

  If the authentication mechanism is not specified, it defaults to `PLAIN` for secure (TLS) connections, or `MYSQL41` for insecure connections. For Unix socket connections, the default is `PLAIN`. (WL #10718)

- Boolean expressions used in queries and statements now support a variant of the `IN` operator for which the right hand side operand is any expression that evaluates to an array or document.

  X DevAPI example:

  ```
  coll.find("'car' IN $.toys").execute();
  ```

  X DevAPI for C example:

  ```
  res = mysqlx_collection_find(coll, "'car' IN $.toys");
  ```

  In this form, the `IN` operator is equivalent to the `JSON_CONTAINS()` SQL function. (WL #10979)

- On Unix and Unix-like systems, Unix domain socket files are now supported as a connection transport for X DevAPI or X DevAPI for C connections. The socket file can be given in a connection string or in the session creation options.

  X DevAPI examples:

```
XSession sess("mysqlx://user:password@(/path/to/mysql.sock)/schema");

XSession sess({ SessionSettings::USER, "user",
SessionSettings::PWD, "password,
SessionSettings::SOCKET, "/path/to/mysql.sock"
SessionSettings::DB, "schema" });
```

  X DevAPI for C examples:

```
mysqlx_session_t *sess = mysqlx_get_session_from_url(
  "mysqlx://user:password@(/path/to/mysql.sock)/schema",
  err_buf, &err_code
);

mysqlx_opt_type_t *sess_opt = mysqlx_session_option_new();
mysqlx_session_option_set(sess_opt,
  MYSQLX_OPT_SOCKET, "/path/to/mysql.sock",
  MYSQLX_OPT_USER, "user",
  MYSQLX_OPT_PWD, "password",
  MYSQLX_OPT_DB, "schema");

mysqlx_session_t *sess = mysqlx_get_session_from_options(
  sess_opt, err_buf, &err_code
);
```

  (WL #9953)

Functionality Added or Changed

- These drop API changes were made:

  - `Session::dropTable(schema, table)` and `Session::dropCollection(schema, coll)` were replaced by `Schema::dropTable(table)` and `Schema::dropCollection(coll)`, respectively.

  - `Schema::dropView()` is now a direct-execute method returning `void` rather than `Executable`.

  - All `dropXXX()` methods succeed if the dropped objects do not exist.

  (WL #10787)

- The following `Collection` methods were added: `addOrReplaceOne()`, `getOne()`, `replaceOne()`, and `removeOne()`.

  The `addOrReplaceOne()` and `replaceOne()` methods work only with MySQL 8.0.3 and higher servers. For older servers, they report an error. (WL #10981)

Bugs Fixed

- Creating a TLS session with only the `ssl-ca` option specified could succeed, although it should fail if `ssl-mode` is not also specified. (Bug #26226502)

- `mysqlx_get_node_session_from_options()` could succeed even when a preceding `mysqlx_session_option_set()` failed. (Bug #26188740)

Changes in MySQL Connector/C++ 8.0.5 (2017-07-10, Development Milestone)

MySQL Connectors and other MySQL client tools and applications now synchronize the first digit of their version number with the (latest) MySQL server version they support. For example, MySQL Connector/C++ 8.0.12 would be designed to support all features of MySQL server version 8 (or earlier). This change makes it easy and intuitive to decide which client version to use for which server version.

Connector/C++ 8.0.5 is the first release to use the new numbering. It is the successor to Connector/C++ 2.0.4.

Character Set Support

- Connector/C++ now supports MySQL servers configured to use `utf8mb4` as the default character set.

  Currently, Connector/C++ works only with UTF-8 and ASCII default character sets (`utf8`, `utf8mb4`, and `ascii`). If a user creates a table with text columns that use a non-UTF-8 character set, and this column holds a string with non-ASCII characters, errors will occur for attempts to access that string (for example, in a query result). On the other hand, if strings consist only of ASCII characters, correct result are obtained regardless of the character set. Also, it is always possible to obtain the raw bytes of the column value, for any character set. (WL #10769)

Deprecation and Removal Notes

- The `NodeSession` class has been renamed to `Session`, and the `XSession` class has been removed. (WL #10785)

X DevAPI Notes

- For X DevAPI or X DevAPI for C applications, when creating a new session, multiple hosts can be tried until a successful connection is established. A list of hosts can be given in a connection string or in the session creation options, with or without priorities.

  X DevAPI examples:

```
Session sess(
  "mysqlx://user:password@["
    "server.example.com,"
    "192.0.2.11:33060,"
    "[2001:db8:85a3:8d3:1319:8a2e:370:7348]:1"
  "]/database"
);

Session sess({ SessionSettings::USER, "user",
               SessionSettings::PWD, "password,
               SessionSettings::HOST, "server.example.com",
               SessionSettings::HOST, "192.0.2.11",
               SessionSettings::PORT, 33060,
               SessionSettings::HOST, "[2001:db8:85a3:8d3:1319:8a2e:370:7348]",
               SessionSettings::PORT, 1,
               SessionSettings::DB, "database" });
```

  X DevAPI for C examples:

```
sess = mysqlx_get_session_from_url(
        "mysqlx://user:password@["
           "(address=127.0.0.1,priority=2),"
           "(address=example.com:1300,priority=100)"
        "]/database",
        err_msg, &err_code);

mysqlx_opt_type_t *sess_opt = mysqlx_session_option_new();
mysqlx_session_option_set(sess_opt,
  MYSQLX_OPT_USER, "user",
  MYSQLX_OPT_PWD, "password",
  MYSQLX_OPT_HOST, "127.0.0.1",
  MYSQLX_OPT_PRIORITY, 2,
  MYSQLX_OPT_HOST, "example.com",
  MYSQLX_OPT_PORT, 1300,
  MYSQLX_OPT_PRIORITY, 100,
  MYSQLX_OPT_DB, "database");

mysqlx_session_t *sess = mysqlx_get_session_from_options(
  sess_opt, err_buf, &err_code
);
```

(WL #9978)

Functionality Added or Changed

- The `SqlResult` class now implements the `getAffectedRowsCount()` and `getAutoIncrementValue()` X DevAPI methods. (Bug #25643081)
- To avoid unintentional changes to all items in a collection, the `Collection::modify()` and `Collection::remove()` methods now require a nonempty selection expression as argument. (WL #10786)
- Connections created using `Session` objects now are encrypted by default. Also, the `ssl-enabled` connection option has been replaced by `ssl-mode`. Permitted `ssl-mode` values are `disabled`, `required` (the default), `verify_ca` and `verify_identity`. (WL #10442)
- Option names within connection strings are now treated as case insensitive. Option values are still case sensitive by default. (WL #10717)

Bugs Fixed

- It is now possible to call `stmt.execute()` multiple times. Calling methods that modify statement parameters should modify the statement sent with `execute()`. This is also true for binding new values to named parameters. (Bug #25858159)
- Compiler errors occurred when creating a `SessionSettings` object due to ambiguity in constructor resolution. (Bug #25603191)
- `collection.add()` failed to compile if called with two STL container arguments. (Bug #25510080)
- These expression syntaxes are now supported:

```
CHARSET(CHAR(X'65'))
'abc' NOT LIKE 'ABC1'
'a' RLIKE '^[a-d]'
'a' REGEXP '^[a-d]'
POSITION('bar' IN 'foobarbar')
```

  These expression syntaxes are not supported but a better error message is provided when they are used:

```
CHARSET(CHAR(X'65' USING utf8))
TRIM(BOTH 'x' FROM 'xxxbarxxx')
TRIM(LEADING 'x' FROM 'xxxbarxxx')
TRIM(TRAILING 'xyz' FROM 'barxxyz')
```

```
'Heoko' SOUNDS LIKE 'h1aso'
```

(Bug #25505482)

Index

Symbols

A

B

C

D

E

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

X

Y

Z