MySQL Connector/C++ Release Notes

Abstract

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

For additional MySQL Connector/C++ documentation, see MySQL Connector/C++ 8.0 Developer Guide, and MySQL Connector/C++ 1.1 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: 2021-10-25 (revision: 23570)

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
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Changes in MySQL Connector/C++ 8.0

Changes in MySQL Connector/C++ 8.0.28 (Not yet released, General Availability)

Version 8.0.28 has no release notes, or they have not been published because the product version has not been released.

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

- Pluggable Authentication
- Bugs Fixed

Pluggable Authentication

- 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.

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
- Pluggable Authentication
- Functionality Added or Changed
- Bugs Fixed

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.

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.

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.

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 the version number with the MySQL Server 8.0.25 release.

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

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.

- 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.

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
- Functionality Added or Changed
- Bugs Fixed

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.

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.

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.

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();
```

- 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.

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.

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.

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.

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
- Packaging Notes

- Bugs Fixed

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.

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.

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.

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.

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)

- 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.

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.

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.
- Connector/C++ no longer supports using wolfSSL as an alternative to OpenSSL. All Connector/C++ builds now use OpenSSL.

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.

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.

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.

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.

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)" }] })"
);
```

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

- 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)

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>
```

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.
- Connector/C++ 8.0 configuration now requires a minimum `CMake` version of 3.0.

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)
- 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.
- 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.)

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.

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.

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.

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.

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 version number with 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.
- 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.
- Installers for Connector/C++ are now available as Debian packages. See Installing Connector/C++ from a Binary Distribution.

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`.

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

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.
- 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.

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
```

- 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), ...);
  ```

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.

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

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.

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.

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.

- 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.
- 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.

- 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.

- 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.

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 base name `mysqlcppconn` as opposed to `mysqlcppconn8` library 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
- Security Notes
- X DevAPI Notes
- Functionality Added or Changed

- Bugs Fixed

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
```

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.

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");
```

- 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.

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.

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
- Functionality Added or Changed
- Bugs Fixed

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.

- 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`.

- 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.

- 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
  );
  ```

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.

- 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.

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 (highest) 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 lower). 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.

Deprecation and Removal Notes

- The `NodeSession` class has been renamed to `Session`, and the `XSession` class has been removed.

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
);
```

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.
- 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`.
- Option names within connection strings are now treated as case insensitive. Option values are still case sensitive by default.

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)

Changes in MySQL Connector/C++ 2.0

Changes in MySQL Connector/C++ 2.0.4 (2017-03-21, Development Milestone)

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

X DevAPI Notes

- Support was added for encrypted sessions over TLS connections. An encrypted session can be requested either via the `ssl-enable` and `ssl-ca` options of a connection string, or using explicit session creation options. For X DevAPI or X DevAPI for C session settings, see the Connector/C++ X DevAPI Reference, available at https://dev.mysql.com/doc/index-connectors.html (check the documentation for `enum mysqlx_opt_type_t`).

- The X DevAPI `Schema` object now supports methods for view manipulation: `createView()`, `alterView()`, and `dropView()`.

  X DevAPI for C now contains functions that implement similar functionality: `mysqlx_view_create()`, `mysqlx_view_replace()`, `mysqlx_view_modify()`, and (implemented previously) `mysqlx_view_drop()`.

  As with other X DevAPI for C operations, there are functions that create a statement handle without executing it: `mysqlx_view_create_new()`, `mysqlx_view_replace_new()`, and `mysqlx_view_modify_new()`.

  These X DevAPI for C functions modify view DDL statements before execution: `mysqlx_set_view_algorithm()`, `mysqlx_set_view_security()`, `mysqlx_set_view_check_option()`, `mysqlx_set_view_definer()`, and `mysqlx_set_view_columns()`.

Functionality Added or Changed

- The format of document ID values generated when adding documents to a collection has changed. It is still a string of 32 hexadecimal digits based on UUID, but the order of digits was changed to match the requirement of a stable ID prefix.

- Connector/C++ now supports IPv6 target hosts in connection strings and when creating sessions using other methods.

Bugs Fixed

- When `rList` is an empty list, `table.insert().rows(rList)` caused a segmentation fault. (Bug #25515964)

Changes in MySQL Connector/C++ 2.0.3 (2016-10-19, Development Milestone)

MySQL Connector/C++ 2.0.3 is the next development milestone of the MySQL Connector/C++ 2.0 series, and the first public release. Apart from covering more X DevAPI features, it adds a new, plain C API, called X DevAPI for C, that offers functionality similar to X DevAPI to applications written in plain C. Thus, not only can MySQL Connector/C++ be used to write C++ applications, as before. Now, using the X DevAPI for C, MySQL Connector/C++ can be used to write plain C applications to access MySQL Database implementing a document store as well as execute traditional plain SQL statements. For more information about X DevAPI for C, refer to the documentation at https://dev.mysql.com/doc/dev/connector-cpp/xapi_ref.html.

Note

The X DevAPI requires at least MySQL Server version 5.7.12 or higher with the X Plugin enabled. For general documentation about how to get started using MySQL as a document store, see Using MySQL as a Document Store.

X DevAPI Notes

- New X DevAPI features added in this Connector/C++ release:
  - Methods for starting and controlling transactions
  - Using an X DevAPI URI or connection string to specify new session parameters
  - Capability of binding a session to the default shard and execute SQL statements there (using `XSession.bindToDefaultShard()`)
  - Methods for counting elements in a table or collection
  - Access to multiple result sets if present in a query result
  - Methods to count items in a result set and fetch a complete result set at once (using `fetchAll()`), instead of accessing items one by one (using `fetchOne()`)
  - Access to warnings reported when processing a statement (`getWarnings()`)
  - Access to information about affected rows, generated auto-increment values, and identifiers of documents added to a collection

Changes in MySQL Connector/C++ 2.0.2 (Not released, Development Milestone)

MySQL Connector/C++ 2.0.2 is the next development milestone of the MySQL Connector/C++ 2.0 series. This series implements the new X DevAPI. The X DevAPI enables application developers to write code that combines the strengths of the relational and document models using a modern, NoSQL-like syntax that does not assume previous experience writing traditional SQL. It also allows working in a traditional way, using SQL queries, and thus provides functionality similar to the API used in the previous versions of the connector.

To learn more about how to write applications using the X DevAPI, see X DevAPI User Guide. For more information about how to use Connector/C++ 2.0 and how the X DevAPI is implemented in it, see https://dev.mysql.com/doc/dev/connector-cpp/.

Note

The X DevAPI requires at least MySQL Server version 5.7.12 or higher with the X Plugin enabled. For general documentation about how to get started using MySQL as a document store, see Using MySQL as a Document Store.

Changes in MySQL Connector/C++ 2.0.1 (Not released, Development Milestone)

MySQL Connector/C++ 2.0.1 is the first development milestone of the MySQL Connector/C++ 2.0 series.

Changes in MySQL Connector/C++ 1.1

Changes in MySQL Connector/C++ 1.1.13 (2019-10-25, General Availability)

- Character Set Support
- Bugs Fixed

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.

Bugs Fixed

- 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++ 1.1.12 (2019-01-28, General Availability)

- Compilation Notes
- Security Notes
- Bugs Fixed

Compilation Notes

- Connector/C++ now compiles against MySQL 5.5 and 5.6. Thanks to Marco Busemann for the patch. (Bug #28280066, Bug #91529)

Security Notes

- Connector/C++ now supports the improvements to named-pipe access control implemented for MySQL Server and controlled by the `named_pipe_full_access_group` system variable.

Bugs Fixed

- Compiling Connector/C++ from source using dynamic linking resulted in link errors for the `mysql_sys` and `mysql_strings` libraries. (Bug #27961505, Bug #90727)

Changes in MySQL Connector/C++ 1.1.11 (2018-04-30, General Availability)

For MySQL Connector/C++ 1.1.11 and higher, Commercial and Community distributions require the Visual C++ Redistributable for Visual Studio 2015 to work on Windows platforms. This is a change from previous versions, which required Visual C++ Redistributable for Visual Studio 2013, and only for Community (but not Commercial) distributions. The Redistributable is available at the Microsoft Download Center; install it before installing Connector/C++.

- Packaging Notes

- Functionality Added or Changed

Packaging Notes

- Connector/C++ binary distributions now include a `BUILDINFO.txt` file that contains information about the build environment used to produce the distribution. (Bug #23556661)

Functionality Added or Changed

- Connector/C++ 1.1 now works with both MySQL 5.7 GA and MySQL 8.0 GA servers.
  - Applications can connect to MySQL 8.0 servers using accounts that authenticate using the `caching_sha2_password` authentication plugin.
  - Applications can connect to MySQL 8.0 servers using unencrypted connections by using the `OPT_GET_SERVER_PUBLIC_KEY` connection option with a value of true.
  - Connector/C++ 1.1 can be built from source against either MySQL 5.7 and MySQL 8.0 server installations.
  - A new `BUNDLE_DEPENDENCIES CMake` option is available. If enabled, the external libraries on which Connector/C++ depends at runtime (such as OpenSSL), are packaged together with the connector.
- Connector/C++ 1.1 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.

Changes in MySQL Connector/C++ 1.1.10 (2017-07-21, General Availability)

For this release of MySQL Connector/C++, only Commercial packages are available. No Community packages are available.

Security Notes

- The linked OpenSSL library for Connector/C++ 1.1 Commercial has been updated to version 1.0.2l. For a description of issues fixed in this version, see http://www.openssl.org/news/vulnerabilities.html. (Bug #26321027)

Changes in MySQL Connector/C++ 1.1.9 (2017-05-16, General Availability)

- Compilation Notes
- Security Notes
- Bugs Fixed

Compilation Notes

- The Windows version of Connector/C++ Community is now built using the dynamic C++ runtime library (that is, with the `/MD` compiler option), with the following implications for users:
  - Target hosts running Windows applications that use Connector/C++ Community now need the Visual C++ Redistributable for Visual Studio 2013 installed on them.
  - Client applications on Windows that use Connector/C++ Community should be compiled with the `/MD` compiler option.

Security Notes

- The linked OpenSSL library for Connector/C++ 1.1.9 Commercial has been updated to version 1.0.2k. For a description of issues fixed in this version, see http://www.openssl.org/news/vulnerabilities.html.

  This change does not affect the Oracle-produced MySQL Community build of Connector/C++, which uses the yaSSL library instead.

Bugs Fixed

- Values returned by `getDouble()` from `DOUBLE` table columns were truncated (decimal part missing) if the locale was set to `fr_CA`, which uses comma as the decimal separator. (Bug #17227390, Bug #69719)
- Connections to `localhost` failed if the local server was bound only to its IPv6 interface. (Bug #17050354, Bug #69663)

Changes in MySQL Connector/C++ 1.1.8 (2016-12-16, General Availability)

Security Notes

- OpenSSL is ending support for version 1.0.1 in December 2016; see https://www.openssl.org/policies/releasestrat.html. Consequently, Connector/C++ Commercial builds now use version 1.0.2 rather than version 1.0.1, and the linked OpenSSL library for the Connector/C++ Commercial has been updated from version 1.0.1 to version 1.0.2j. For a description of issues fixed in this version, see https://www.openssl.org/news/vulnerabilities.html.

  This change does not affect Oracle-produced MySQL Community builds of Connector/C++, which use the yaSSL library instead.

Functionality Added or Changed

- Connector/C++ now supports a `OPT_TLS_VERSION` connection option for specifying the protocols permitted for encrypted connections. The option value is string containing a comma-separated list of one or more protocol versions. Example:

  ```
  connection_properties["OPT_TLS_VERSION"] = sql::SQLString("TLSv1.1,TLSv1.2");
  ```

  The permitted values depend on the SSL library used to compile MySQL: `TLSv1`, `TLSv1.1`, `TLSv1.2` if OpenSSL was used; `TLSv1` and `TLSv1.1` if yaSSL was used. The default is to permit all available protocols.

  For more information about TLS protocols in MySQL, see Encrypted Connection TLS Protocols and Ciphers. (Bug #23496967)

- Connector/C++ now supports a `OPT_SSL_MODE` connection option for specifying the security state of the connection to the server. Permitted option values are `SSL_MODE_PREFERRED` (the default), `SSL_MODE_DISABLED`, `SSL_MODE_REQUIRED`, `SSL_MODE_VERIFY_CA`, and `SSL_MODE_VERIFY_IDENTITY`. These values correspond to the values of the `--ssl-mode` option supported by MySQL client programs; see Command Options for Encrypted Connections. For example, this setting specifies that the connection should be unencrypted:

  ```
  connection_properties["OPT_SSL_MODE"] = sql::SSL_MODE_DISABLED;
  ```

The `OPT_SSL_MODE` option comprises the capabilities of the `sslEnforce` and `sslVerify` connection options. Consequently, both of those options are now deprecated. (Bug #23496952)

- Connector/C++ now supports `OPT_MAX_ALLOWED_PACKET` and `OPT_NET_BUFFER_LENGTH` connection options. Each option takes a numeric value. They correspond to the `MYSQL_OPT_MAX_ALLOWED_PACKET` and `MYSQL_OPT_NET_BUFFER_LENGTH` options for the `mysql_options()` C API function.
- Issues compiling Connector/C++ under Visual Studio 2015 were corrected.

Bugs Fixed

- A segmentation fault could occur for attempts to insert a large string using a prepared statement. (Bug #23212333, Bug #81213)
- The certification verification checks that are enabled by the `verifySSL` connection option were not performed properly. (Bug #22931974)
- Connector/C++ failed to compile against a version of the MySQL C API older than 5.7. (Bug #22838573, Bug #80539, Bug #25201287)

Changes in MySQL Connector/C++ 1.1.7 (2016-01-20, General Availability)

Configuration Notes

- Binary distributions for this release of Connector/C++ were linked against `libmysqlclient` from MySQL 5.7.10, except for OS X 10.8/10.9, for which distributions were linked against MySQL 5.7.9. This enables Connector/C++ to take advantage of features present in recent client library versions. Some examples:
  - Support for the MySQL `JSON` data type is available. Current versions of MySQL Workbench require `JSON` support, so to build MySQL Workbench 6.3.5 or higher from source, it is necessary to use a version of Connector/C++ at least as recent as 1.1.7.
  - Applications attempt to connect using encryption by default if the server support encrypted connections, falling back to an unencrypted connection if an encrypted connection cannot be established. (This is as described at Configuring MySQL to Use Encrypted Connections.) To enforce an encrypted connection, such that an error occurs if encrypted connections are not available, applications can enable the `sslEnforce` connection option.

  To build Connector/C++ from source, you must use either a General Availability version of MySQL 5.7 (5.7.9 or higher). Set the `MYSQL_DIR` `CMake` option appropriately at configuration time as necessary. (Bug #22351273)

Security Notes

- The linked OpenSSL library for Connector/C++ Commercial has been updated to version 1.0.1q. Issues fixed in the new OpenSSL version are described at http://www.openssl.org/news/vulnerabilities.html.

  This change does not affect Oracle-produced MySQL Community builds of Connector/C++, which use the yaSSL library instead.

Spatial Data Support

- The required version of the Boost library for Connector/C++ builds has been raised to 1.56.0.

Bugs Fixed

- `MySQL_Prepared_ResultSet::relative()` failed to fetch the record due to a missing `proxy->fetch()` call. (Bug #21152054)
- During Connector/C++ builds, the MySQL Server `CXXFLAGS` and `CFLAGS` values were used rather than the system default values. To specify explicitly to use the server values, enable the new `USE_SERVER_CXXFLAGS` `CMake` option. (Bug #77655, Bug #21391025)

Changes in MySQL Connector/C++ 1.1.6 (2015-06-10, General Availability)

Security Notes

- Connector/C++ 1.1.6 Commercial upgrades the linked OpenSSL library to version 1.0.1m which has been publicly reported as not vulnerable to CVE-2015-0286.

Functionality Added or Changed

- The `std::auto_ptr` class template is deprecated in C++11, and its usage has been replaced with `boost::scoped_ptr/shared_ptr`.

  The `CMAKE_ENABLE_C++11` `CMake` option has been added to permit enabling C++11 support. (Bug #75251)
- Connector/C++ now provides macros to indicate the versions of libraries against which it was built: `MYCPPCONN_STATIC_MYSQL_VERSION` and `MYCPPCONN_STATIC_MYSQL_VERSION_ID` (MySQL client library version, string and numeric), and `MYCPPCONN_BOOST_VERSION` (Boost library version, numeric). (Bug #75250)

Bugs Fixed

- With `defaultStatementResultType=FORWARD_ONLY` and a row position after the last row, using getter methods such as `getInt()` or `getString()` resulted in a segmentation fault. (Bug #20085944)
- For prepared statements, calling `wasNull()` before fetching data resulted in an assertion failure. (Bug #19938873)
- Result sets from prepared statements were not freed. (Bug #18135088)
- Connector/C++ failed to build against Boost-devel-1.41.0-25 on OLE6. (Bug #75063, Bug #20125824)
- Configuration failed if the `MYSQL_CONFIG_EXECUTABLE` option was specified and the MySQL installation path contained the characters `-m`. Installation failed if the build directory was not in the top source directory. (Bug #73502, Bug #19447498)
- For prepared statements, `getString()` did not return the fractional seconds part from temporal columns that had a fractional sections part. (Bug #68523, Bug #17218692)
- For queries of the form `SELECT MAX(bit_col) FROM table_with_bit_col`, `getString()` returned an incorrect result. (Bug #66235, Bug #14520822)

- For Connector/C++ builds from source, `make install` failed if only the static library had been built without the dynamic library. (Bug #52281, Bug #11759926)

Changes in MySQL Connector/C++ 1.1.5 (2014-11-26, General Availability)

Functionality Added or Changed

- `MySQL_Prepared_Statement::getMoreResults()` functionality has been implemented, so multiple result sets now can be fetched using a prepared statement. (Bug #19147677)
- Connector/C++ now supports the `defaultAuth`, `OPT_CONNECT_ATTR_DELETE`, `OPT_CONNECT_ATTR_RESET`, `OPT_LOCAL_INFILE`, `pluginDir`, `readDefaultFile`, `readDefaultGroup`, and `charsetDir` connection options, which correspond to the `MYSQL_DEFAULT_AUTH`, `MYSQL_OPT_CONNECT_ATTR_DELETE`, `MYSQL_OPT_CONNECT_ATTR_RESET`, `MYSQL_OPT_LOCAL_INFILE`, `MYSQL_PLUGIN_DIR`, `MYSQL_READ_DEFAULT_FILE`, `MYSQL_READ_DEFAULT_GROUP`, and `MYSQL_SET_CHARSET_DIR` options for the `mysql_options()` C API function.

  It is also possible to get and set the statement execution-time limit using the `MySQL_Statement::getQueryTimeout()` and `MySQL_Statement::setQueryTimeout()` methods. (Bug #73665, Bug #19479950)
- These methods were added: `Connection::isValid()` checks whether the connection is alive, and `Connection::reconnect()` reconnects if the connection has gone down. (Bug #65640, Bug #14207722)
- The Boost dependency was removed from the Connector/C++ API headers. These headers were using the `boost::variant` type, making it impossible to use Connector/C++ binaries without having Boost installed.

Bugs Fixed

- For installation from MSI packages, `variant.h` and `version_info.h` were missing from the `include/cppconn` folder. (Bug #19973637)
- For several valid client options, `getClientOption()` did not return a value. (Bug #19940314)
- A memory leak occurred when adding the `OPT_CONNECT_ATTR_ADD` parameter to the options list. (Bug #19938970)
- `getClientOption()` raised an assertion if the specified option was not set at connect time. (Bug #19938922)
- Several metadata flaws were corrected:
  - `getTables()` did not return a correct result when `TableType=VIEW` and `metadataUseInfoSchema=false`.
  - `getColumns()` did not return column inforomation when `metadataUseInfoSchema=TRUE`.
  - `getColumnName()` returned the display name instead of the actual column name.
  - `getProcedures()` returned a syntax error when `metadataUseInfoSchema=false`.

  (Bug #19505348, Bug #19147897, Bug #19244736, Bug #19505421)
- The `LOCALHOST` global variable was referenced at two places in Connector/C++ code, which could result in a double-free corruption error. (Bug #74616, Bug #19910311)

- `driver/version_info.h` (containing version macros) was not included in the installed header files. (Bug #73795, Bug #19553971)
- Several `CMake` issues were corrected:
  - `CMake` could misconfigure the link flags.
  - `CMake` did not pick up the `libmysqlclient` path from the `MYSQL_LIB_DIR` option.
  - For test suite compilation, `CMake` did not pick up `libmysqlclient` from the user-specified path, even if `MYSQL_LIB_DIR` and `DYNLOAD_MYSQL_LIB` were given.

  (Bug #73427, Bug #19315635, Bug #19370844, Bug #19940663)
- Connector/C++ issued a ping command every time `isClosed()` was called in a `Connection`, rather than just checking whether `close()` had been called earlier or when a fatal error occurred in an earlier operation. (Bug #69785, Bug #17186530)
- With the result set type set to `TYPE_FORWARD_ONLY`, `Statement::executeQuery()` returns almost immediately, but `MySQL_ResultSet::next()` and `MySQL_Prepared_ResultSet::next()` returned false if the connection was lost rather than throwing an exception, making it impossible to distinguish loss of connection from normal end of the result set. `MySQL_ResultSet::next()` and `MySQL_Prepared_ResultSet::next()` now throw an exception when the connection is lost. (Bug #69031, Bug #18886278)
- `Connection` objects shared internal state with `Statement` objects they spawned, preventing a connection close unless the `Statement` objects were destroyed first. A connection to the server now is closed by calling `Connection::close()` and invoking the `Connection` object destructor, without explicitly destroying the statement object.

Changes in MySQL Connector/C++ 1.1.4 (2014-07-31, General Availability)

Compilation Notes

- Binary distributions of this Connector/C++ release were compiled using Boost 1.54.0. If you compile this Connector/C++ release from source, you must also use that Boost version.

Functionality Added or Changed

- Connector/C++ now supports the following connection options: `sslVerify` (boolean), `sslCRL` (string), and `sslCRLPath` (string). These correspond to the `MYSQL_OPT_SSL_VERIFY_SERVER_CERT`, `MYSQL_OPT_SSL_CRL`, and `MYSQL_OPT_SSL_CRLPATH` options for the `mysql_options()` C API function. (Bug #18461451)
- Connector/C++ has new methods to provide schema, table, and column character set and collation metadata for result sets:
  - `ResultSet * DatabaseMetaData::getSchemaCollation(const sql::SQLString& catalog, const sql::SQLString& schemaPattern)`
  - `ResultSet * DatabaseMetaData::getSchemaCharset(const sql::SQLString& catalog, const sql::SQLString& schemaPattern)`
  - `ResultSet * DatabaseMetaData::getTableCollation(const sql::SQLString& catalog, const sql::SQLString& schemaPattern, const sql::SQLString& tableNamePattern)`

- `ResultSet * DatabaseMetaData::getTableCharset(const sql::SQLString& catalog, const sql::SQLString& schemaPattern, const sql::SQLString& tableNamePattern)`
- `SQLString ResultSetMetaData::getColumnCollation(unsigned int columnIndex)`
- `SQLString ResultSetMetaData::getColumnCharset(unsigned int columnIndex)`

(Bug #72698, Bug #18803345)

- Connector/C++ now supports the `OPT_CONNECT_ATTR_ADD` option, which accepts an `std::map` argument. This option corresponds to the `MYSQL_OPT_CONNECT_ATTR_ADD` option for `mysql_options4()`. (Bug #72697, Bug #18803313)
- Connector/C++ now supports a `useLegacyAuth` connection option, which corresponds to the `MYSQL_SECURE_AUTH` option for the `mysql_options()` C API function, except that the sense is the logical negation. For example, to disable secure authentication, pass a `useLegacyAuth` value of true. (Bug #69492, Bug #16970753)

Bugs Fixed

- `MySQL_ResultSetMetaData::getColumnTypeName()` returned `UNKNOWN` for `LONG_BLOB` fields. (Bug #72700, Bug #18803414)
- Definitions for character sets and collations were added (`utf8mb4` in particular). (Bug #71606, Bug #18193771)
- Connector/C++ version-information methods have been revised to return the correct values. (Bug #66975, Bug #14680878)

Changes in MySQL Connector/C++ 1.1.3 (2013-03-08, General Availability)

Functionality Added or Changed

- Connector/C++ now supports an `OPT_ENABLE_CLEARTEXT_PLUGIN` connection option. If true, it enables the client-side cleartext authentication plugin. This client-side plugin is required, for example, for accounts that use the PAM server-side authentication plugin. (Bug #16520952)

Bugs Fixed

- `MySQL_ConnectionMetaData::getBestRowIdentifier()` considered only `PRIMARY KEY` columns usable for row identifiers. It now also considers `UNIQUE NOT NULL` columns suitable for the same purpose. (Bug #16277170)

Changes in MySQL Connector/C++ 1.1.2 (2013-02-05, General Availability)

Functionality Added or Changed

- Connector/C++ applications now can handle connecting to the server using an account for which the password had expired. Connector/C++ now supports three new connection options:
  - `OPT_CAN_HANDLE_EXPIRED_PASSWORDS`: If true, this indicates to the driver that the application can handle expired passwords.

If the application specifies `OPT_CAN_HANDLE_EXPIRED_PASSWORDS` but the underlying `libmysqlclient` library does not support it, the driver returns `sql::mysql:deCLIENT_DOESNT_SUPPORT_FEATURE(820)`.

- `preInit`: A string containing statements to run before driver initialization.
- `postInit`: A string containing statements to run after driver initialization.

A new file `driver/mysql_error.h` is being added to the MSI package. This file defines an `enum DRIVER_ERROR`, which contains the definition of `deCL_CANT_HANDLE_EXP_PWD`.

In addition to the preceding changes, these problems with `Statement::executeUpdate` were fixed:

- If `Statement::executeUpdate` executed multiple statements, the connection became unusable.
- There was no exception if one of statements returned a result set. Now `executeUpdate` returns and update count for the last executed query.

For example code showing how to use the new options, see the file `test/unit/bugs/bugs.cpp` in the Connector/C++ distribution. (Bug #67325, Bug #15936764)

Changes in MySQL Connector/C++ 1.1.1 (2012-08-07, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Connector/C++ is now built against `libmysqlclient` from MySQL 5.5.27, enabling support of authentification plugins and IPv6.
- URI format has been extended to better fit IPv6 addresses. You can now use `[]` to separate the host part of the URI.
- Added new method `ResultSetMetaData::isNumeric()` and implemented it in all classes that subclass from it.
- Added `MySQL_Connection::getLastStatementInfo()` which returns back the value of the `mysql_info()` function of the MySQL Client Library (`libmysqlclient`).

Bugs Fixed

- Compiling with Visual Studio 2010 could fail with compilation errors if the source contained a `#include <stdint.h>` line. The errors were typically of the form `cannot convert from 'type1' to 'type2'`. (Bug #14113387, Bug #60307)
- Fixed `stores(Lower|Mixed)Case(Quoted)Identifiers` methods.
- A statement that did not raise any warning could return warnings from a previously executed statement.
- `DatabaseMetaData::getSQLKeywords()` updated to match MySQL 5.5. Note that Connector/C++, just like Connector/J, returns the same list for every MySQL database version.

Changes in MySQL Connector/C++ 1.1.0 (2010-09-13, General Availability)

This fixes bugs since the first GA release 1.0.5 and introduces new features.

- Functionality Added or Changed

- Bugs Fixed

Functionality Added or Changed

- **Incompatible Change:** API incompatible change: `ConnectPropertyVal` is no longer a `struct` by a typedef that uses `boost::variant`. Code such as:

```
sql::ConnectPropertyVal tmp;
tmp.str.val=passwd.c_str();
tmp.str.len=passwd.length();
connection_properties["password"] = tmp;
```

  Should be changed to:

```
connection_properties["password"] = sql::ConnectPropertyVal(passwd);
```

- Instances of `std::auto_ptr` have been changed to `boost::scoped_ptr`. Scoped array instances now use boost::scoped_array. Further, `boost::shared_ptr` and `boost::weak_ptr` are now used for guarding access around result sets.
- `LDFLAGS`, `CXXFLAGS` and `CPPFLAGS` are now checked from the environment for every binary generated.
- The connection map property `OPT_RECONNECT` was changed to be of type `boolean` from `long long`.
- `get_driver_instance()` is now only available in dynamic library builds; static builds do not have this symbol. This was done to accommodate loading the DLL with `LoadLibrary` or `dlopen`. If you do not use `CMake` for building the source code you will need to define `mysqlcppconn_EXPORTS` if you are loading dynamically and want to use the `get_driver_instance()` entry point.
- `Connection::getClientOption(const sql::SQLString & optionName, void * optionValue)` now accepts the `optionName` values `metadataUseInfoSchema`, `defaultStatementResultType`, `defaultPreparedStatementResultType`, and `characterSetResults`. In the previous version only `metadataUseInfoSchema` was permitted. The same options are available for `Connection::setClientOption()`.

Bugs Fixed

- Certain header files were incorrectly present in the source distribution. The fix excludes dynamically generated and platform specific header files from source packages generated using CPack. (Bug #45846)
- `CMake` generated an error if configuring an out of source build, that is, when `CMake` was not called from the source root directory. (Bug #45843)
- Using Prepared Statements caused corruption of the heap. (Bug #45048)
- Missing includes when using GCC 4.4. Note that GCC 4.4 is not yet in use for any official Connector/C++ builds. (Bug #44931)
- A bug was fixed in Prepared Statements. The bug occurred when a stored procedure was prepared without any parameters. This led to an exception. (Bug #44931)
- Fixed a Prepared Statements performance issue. Reading large result sets was slow.
- Fixed bug in `ResultSetMetaData` for statements and prepared statements, `getScale` and `getPrecision` returned incorrect results.

Changes in MySQL Connector/C++ 1.0

Changes in MySQL Connector/C++ 1.0.5 (2009-04-21, General Availability)

This is the first General Availability (GA) release.

Functionality Added or Changed

- The interface of `sql::ConnectionMetaData`, `sql::ResultSetMetaData` and `sql::ParameterMetaData` was modified to have a protected destructor. As a result the client code has no need to destruct the metadata objects returned by the connector. Connector/C++ handles the required destruction. This enables statements such as:

  ```
  connection->getMetaData->getSchema();
  ```

  This avoids potential memory leaks that could occur as a result of losing the pointer returned by `getMetaData()`.

- Improved memory management. Handling of potential memory leak situations is more robust.

- Changed the interface of `sql::Driver` and `sql::Connection` so they accept the options map by alias instead of by value.

- Changed the return type of `sql::SQLException::getSQLState()` from `std::string` to `const char *` to be consistent with `std::exception::what()`.

- Implemented `getResultSetType()` and `setResultSetType()` for `Statement`. Uses `TYPE_FORWARD_ONLY`, which means unbuffered result set and `TYPE_SCROLL_INSENSITIVE`, which means buffered result set.

- Implemented `getResultSetType()` for `PreparedStatement`. The setter is not implemented because currently `PreparedStatement` cannot do refetching. Storing the result means the bind buffers will be correct.

- Added the option `defaultStatementResultType` to `MySQL_Connection::setClientOption()`. Also, the method now returns `sql::Connection *`.

- Added `Result::getType()`. Implemented for the three result set classes.

- Enabled tracing functionality when building with Microsoft Visual C++ 8 and later, which corresponds to Microsoft Visual Studio 2005 and later.

- Added better support for named pipes, on Windows. Use `pipe://` and add the path to the pipe. Shared memory connections are currently not supported.

Bugs Fixed

- A bug was fixed in `MySQL_Connection::setSessionVariable()`, which had been causing exceptions to be thrown.

Changes in MySQL Connector/C++ 1.0.4 (2009-03-31, Beta)

Functionality Added or Changed

- An installer was added for the Windows operating system.

- Minimum `CMake` version required was changed from 2.4.2 to 2.6.2. The latest version is required for building on Windows.

- `metadataUseInfoSchema` was added to the connection property map, which enables control of the `INFORMATION_SCHEMA` for metadata.
- Implemented `MySQL_ConnectionMetaData::supportsConvert(from, to)`.
- Introduced `ResultSetMetaData::isZerofill()`, which is not in the JDBC specification.

Bugs Fixed

- A bug was fixed in all implementations of `ResultSet::relative()` which was giving a wrong return value although positioning was working correctly.
- A leak was fixed in `MySQL_PreparedResultSet`, which occurred when the result contained a `BLOB` column.

Changes in MySQL Connector/C++ 1.0.3 (2009-03-02, Alpha)

Functionality Added or Changed

- Added new tests in `test/unit/classes`. Those tests are mostly about code coverage. Most of the actual functionality of the driver is tested by the tests found in `test/CJUnitPort`.
- New data types added to the list returned by `DatabaseMetaData::getTypeInfo()` are `FLOAT UNSIGNED`, `DECIMAL UNSIGNED`, `DOUBLE UNSIGNED`. Those tests may not be in the JDBC specification. However, due to the change you should be able to look up every type and type name returned by, for example, `ResultSetMetaData::getColumnTypeName()`.
- `MySQL_Driver::getPatchVersion` introduced.
- Major performance improvements due to new buffered `ResultSet` implementation.
- Addition of `test/unit/README` with instructions for writing bug and regression tests.
- Experimental support for STLPort. This feature may be removed again at any time later without prior warning! Type `cmake -L` for configuration instructions.
- Added properties enabled methods for connecting, which add many connect options. This uses a dictionary (map) of key value pairs. Methods added are `Driver::connect(map)`, and `Connection::Connection(map)`.
- New `BLOB` implementation. `sql::Blob` was removed in favor of `std::istream`. C++'s `IOStream` library is very powerful, similar to PHP's streams. It makes no sense to reinvent the wheel. For example, you can pass a `std::istringstream` object to `setBlob()` if the data is in memory, or just open a file `std::fstream` and let it stream to the DB, or write its own stream. This is also true for `getBlob()` where you can just copy data (if a buffered result set), or stream data (if implemented).
- Implemented `ResultSet::getBlob()` which returns `std::stream`.
- Fixed `MySQL_DatabaseMetaData::getTablePrivileges()`. Test cases were added in the first unit testing framework.
- Implemented `MySQL_Connection::setSessionVariable()` for setting system variables such as `sql_mode`.
- Implemented `MySQL_DatabaseMetaData::getColumnPrivileges()`.
- `cppconn/datatype.h` has changed and is now used again. Reimplemented the type subsystem to be more usable - more types for binary and nonbinary strings.

- Implementation for `MySQL_DatabaseMetaData::getImportedKeys()` for MySQL versions before 5.1.16 using `SHOW`, and above using `INFORMATION_SCHEMA`.
- Implemented `MySQL_ConnectionMetaData::getProcedureColumns()`.
- `make package_source` now packs with bzip2.
- Re-added `getTypeInfo()` with information about all types supported by MySQL and the `sql::DataType`.
- Changed the implementation of `MySQL_ConstructedResultSet` to use the more efficient O(1) access method. This should improve the speed with which the metadata result sets are used. Also, there is less copying during the construction of the result set, which means that all result sets returned from the metadata functions will be faster.
- Introduced, internally, `sql::mysql::MyVal` which has implicit constructors. Used in `mysql_metadata.cpp` to create result sets with native data instead of always string (varchar).
- Renamed `ResultSet::getLong()` to `ResultSet::getInt64()`. `resultset.h` includes typdefs for Windows to be able to use `int64_t`.
- Introduced `ResultSet::getUInt()` and `ResultSet::getUInt64()`.
- Improved the implementation for `ResultSetMetaData::isReadOnly()`. Values generated from views are read only. These generated values don't have `db` in `MYSQL_FIELD` set, while all normal columns do have.
- Implemented `MySQL_DatabaseMetaData::getExportedKeys()`.
- Implemented `MySQL_DatabaseMetaData::getCrossReference()`.

Bugs Fixed

- Bug fixed in `MySQL_PreparedResultSet::getString()`. Returned string that had real data but the length was random. Now, the string is initialized with the correct length and thus is binary safe.
- Corrected handling of unsigned server types to return correct values.
- Fixed handling of numeric columns in `ResultSetMetaData::isCaseSensitive` to return `false`.

Changes in MySQL Connector/C++ 1.0.2 (2008-12-19, Alpha)

Functionality Added or Changed

- Implemented `getScale()`, `getPrecision()` and `getColumnDisplaySize()` for `MySQL_ResultSetMetaData` and `MySQL_Prepared_ResultSetMetaData`.
- Changed `ResultSetMetaData` methods `getColumnDisplaySize()`, `getPrecision()`, `getScale()` to return `unsigned int` instead of `signed int`.
- `DATE`, `DATETIME` and `TIME` are now being handled when calling the `MySQL_PreparedResultSet` methods `getString()`, `getDouble()`, `getInt()`, `getLong()`, `getBoolean()`.
- Reverted implementation of `MySQL_DatabaseMetaData::getTypeInfo()`. Now unimplemented. In addition, removed `cppconn/datatype.h` for now, until a more robust implementation of the types can be developed.
- Implemented `MySQL_PreparedStatement::setNull()`.

- Implemented `MySQL_PreparedStatement::clearParameters()`.
- Added PHP script `examples/cpp_trace_analyzer.php` to filter the output of the debug trace. Please see the inline comments for documentation. This script is unsupported.
- Implemented `MySQL_ResultSetMetaData::getPrecision()` and `MySQL_Prepared_ResultSetMetaData::getPrecision()`, updating example.
- Added new unit test framework for JDBC compliance and regression testing.
- Added `test/unit` as a basis for general unit tests using the new test framework, see `test/unit/example` for basic usage examples.

Bugs Fixed

- Fixed `MySQL_PreparedStatementResultSet::getDouble()` to return the correct value when the underlying type is `MYSQL_TYPE_FLOAT`.
- Fixed bug in `MySQL_ConnectionMetaData::getIndexInfo()`. The method did not work because the schema name wasn't included in the query sent to the server.
- Fixed a bug in `MySQL_ConnectionMetaData::getColumns()` which was performing a cartesian product of the columns in the table times the columns matching `columnNamePattern`. The example `example/connection_meta_schemaobj.cpp` was extended to cover the function.
- Fixed bugs in `MySQL_DatabaseMetaData`. All `supportsCatalogXXXXX` methods were incorrectly returning `true` and all `supportsSchemaXXXX` methods were incorrectly returning `false`. Now `supportsCatalogXXXXX` returns `false` and `supportsSchemaXXXXX` returns `true`.
- Fixed bugs in the `MySQL_PreparedStatements` methods `setBigInt()` and `setDatetime()`. They decremented the internal column index before forwarding the request. This resulted in a double-decrement and therefore the wrong internal column index. The error message generated was:

```
setString() ... invalid "parameterIndex"
```

- Fixed a bug in `getString()`. `getString()` is now binary safe. A new example was also added.
- Fixed bug in `FLOAT` handling.
- Fixed `MySQL_PreparedStatement::setBlob()`. In the tests there is a simple example of a class implementing `sql::Blob`.

Changes in MySQL Connector/C++ 1.0.1 (2008-12-01, Alpha)

Deprecation and Removal Notes

- `sql::mysql::MySQL_SQLException` was removed. The distinction between server and client (connector) errors, based on the type of the exception, has been removed. However, the error code can still be checked to evaluate the error type.
- Driver Manager was removed.

Functionality Added or Changed

- Support for (n)make install was added. You can change the default installation path. Carefully read the messages displayed after executing cmake. The following are installed:

  - Static and the dynamic version of the library, `libmysqlcppconn`.
  - Generic interface, `cppconn`.
  - Two MySQL specific headers:

    `mysql_driver.h`, use this if you want to get your connections from the driver instead of instantiating a `MySQL_Connection` object. This makes your code portable when using the common interface.

    `mysql_connection.h`, use this if you intend to link directly to the `MySQL_Connection` class and use its specifics not found in `sql::Connection`.

    However, you can make your application fully abstract by using the generic interface rather than these two headers.

- Added `ConnectionMetaData::getSchemas()` and `Connection::setSchema()`.
- `ConnectionMetaData::getCatalogTerm()` returns not applicable, there is no counterpart to catalog in Connector/C++.
- Added experimental GCov support, `cmake -DMYSQLCPPCONN_GCOV_ENABLE:BOOL=1`
- All examples can be given optional connection parameters on the command line, for example:

  ```
  examples/connect tcp://host:port user pass database
  ```

  or

  ```
  examples/connect unix:///path/to/mysql.sock user pass database
  ```

- Renamed `ConnectionMetaData::getTables: TABLE_COMMENT` to `REMARKS`.
- Renamed `ConnectionMetaData::getProcedures: PROCEDURE_SCHEMA` to `PROCEDURE_SCHEM`.
- Renamed `ConnectionMetaData::getPrimaryKeys(): COLUMN` to `COLUMN_NAME`, `SEQUENCE` to `KEY_SEQ`, and `INDEX_NAME` to `PK_NAME`.
- Renamed `ConnectionMetaData::getImportedKeys(): PKTABLE_CATALOG` to `PKTABLE_CAT`, `PKTABLE_SCHEMA` to `PKTABLE_SCHEM`, `FKTABLE_CATALOG` to `FKTABLE_CAT`, `FKTABLE_SCHEMA` to `FKTABLE_SCHEM`.
- Changed metadata column name `TABLE_CATALOG` to `TABLE_CAT` and `TABLE_SCHEMA` to `TABLE_SCHEM` to ensure JDBC compliance.
- Introduced experimental CPack support, see make help.
- All tests changed to create TAP compliant output.
- Renamed `sql::DbcMethodNotImplemented` to `sql::MethodNotImplementedException`
- Renamed `sql::DbcInvalidArgument` to `sql::InvalidArgumentException`
- Changed `sql::DbcException` to implement the interface of JDBC's `SQLException`. Renamed to `sql::SQLException`.
- Converted Connector/J tests were added.
- MySQL Workbench 5.1 changed to use Connector/C++ for its database connectivity.
- New directory layout.

Index

P

Q

R

S

T

U

V

W

X

Y