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++ 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 (http://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 legal information, see the Legal Notices.

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

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

Document generated on: 2018-03-16 (revision: 14318)

Table of Contents

Preface and Legal Notices

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

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Changes in MySQL Connector/C++ 8.0

Changes in MySQL Connector/C++ 8.0.11 (Not yet released)

Version 8.0.11 has no changelog entries, or they have not been published because the product version has not been released.

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 XAPI), 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 XAPI 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. For information about using the legacy API, refer to the documentation at http://dev.mysql.com/doc/connector-cpp/en/connector-cpp-getting-started-examples.html.

- Deprecation and Removal Notes
- Security Notes
- X DevAPI and XAPI 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 XAPI 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 XAPI enumerations:

```
mysqlx_view_algorithm_t
mysqlx_view_security_t
mysqlx_view_check_option_t
```

Removed XAPI macros:

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

Security Notes

- MySQL 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 applications that use X DevAPI or XAPI, 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 protocol, it is not 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 and XAPI 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");
```

  XAPI 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 Syntax.

  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();
```

  XAPI 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);
```

Functionality Added or Changed

- MySQL 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 http://dev.mysql.com/doc/dev/connector-cpp/8.0/building.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 and XAPI 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 XAPI, 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 authentiation 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 XAPI, a new `auth` setting can be specified in connection strings or URIs to indicate the authentiation 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();
  ```

  XAPI 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 XAPI 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" });
  ```

  XAPI 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
- Deprecation and Removal Notes
- X DevAPI and XAPI Notes
- Functionality Added or Changed
- Bugs Fixed

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 and XAPI Notes

- For X DevAPI or XAPI, 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" });
```

XAPI 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 and XAPI Notes
- Functionality Added or Changed
- Bugs Fixed

X DevAPI and XAPI 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 session settings, see http://dev.mysql.com/doc/dev/connector-cpp/8.0/classmysqlx_1_1_session_settings.html. For XAPI session settings, see http://dev.mysql.com/doc/dev/connector-cpp/8.0/group__xapi.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()`.

  XAPI 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 XAPI 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 XAPI 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 XAPI, 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 XAPI, 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 XAPI, refer to the documentation at http://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 http://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.

Version 2.0.2 has no changelog entries, or they have not been published because the product version has not been released.

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.

Version 2.0.1 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL Connector/C++ 1.1

Changes in MySQL Connector/C++ 1.1.11 (Not yet released, General Availability)

Version 1.1.11 has no changelog entries, or they have not been published because the product version has not been released.

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 MySQL 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

- 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
- Functionality Added or Changed
- Bugs Fixed

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 names. 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 connection protocols in MySQL, see Encrypted Connection 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) or Connector/C 6.1.8 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
- Functionality Added or Changed
- Bugs Fixed

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
- Bugs Fixed

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

- Connector/C++ is now compiled and linked with Connector/C 6.1.5 rather than with `libmysql`.
- 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

- 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

- **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
- Bugs Fixed

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)`.
- Added support for Connector/C.
- 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
- Bugs Fixed

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
- Functionality Added or Changed

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

A

B

T

V

W