MySQL Connector/Python Release Notes

Abstract

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

For additional Connector/Python documentation, see MySQL Connector/Python 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 either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

Document generated on: 2018-12-10 (revision: 16662)

Table of Contents

Preface and Legal Notices

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

Legal Notices

Documentation Accessibility

Access to Oracle Support

Changes in MySQL Connector/Python 8.0

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

Functionality Added or Changed

- Added Python 3.7 support. (Bug #27081809, Bug #87818)
- To go with the existing `mysqlx.get_session(conn_str)` method, a new `mysqlx.get_client(conn_str, options)` method was added that creates a connection pool handler that provides a `get_session()` method to create and retrieve connections from the pool. The collection pooling options are:
  - `enabled`: enables or disables connection pooling; boolean and defaults to true.
  - `max_size`: maximum number of connections available in the pool; positive integer and defaults to 25.
  - `max_idle_time`: maximum number of milliseconds a connection can be idle in the queue before being closed; integer >= 0 and defaults to 0 (infinite).
  - `queue_timeout`: maximum number of milliseconds a request will wait for a connection to become available; integer >= 0 and defaults to 0 (infinite).

    This is different than connect_timeout that's used for non-pooling. In a pooling scenario there are already connections in the pool, so queue_timeout controls how long to wait for a connection in the pool.

  Example usage:

```
client = mysqlx.get_client(
    {
    'host': 'localhost',
    'port': 33060,
    'user': 'mike',
    'password': 'password'
    },
    { pooling: {
        enabled: true,
        max_idle_time: 5000,
        max_size: 25,
        queue_timeout: 20000
        }
    }
)
```

  Closing a session attached to the pool makes the connection available in the pool for subsequent get +session() calls, while closing (destroying) the pool effectively closes all server connections.
- Added a `connection-timeout` connection timeout query parameter. This defines the length of time (milliseconds) the client waits for a MySQL server to become available in the given network addresses. It was added to both the mysqlx.get_session() (non-pooling sessions) and mysqlx.get_client() (pooling sessions) interfaces. This option defaults to 10000 (10 seconds). The value 0 disables the timeout so the client will wait until the underlying socket (platform dependent) times out.

  Example usages:

```
mysqlx.get_session("root@localhost?connect-timeout=0");
mysqlx.get_session("root@[localhost:33060, 127.0.0.1:33060]?connect-timeout=5000");
```

  In a multi-host scenario, the `connect-timeout` value applies to each individual host.

Bugs Fixed

- On Windows, the 32-bit MSI failed to install. The registry key path was updated to allow the CEXT prerequisite check to execute and pass. (Bug #28395599, Bug #28464866)

- Subsequent collection.add() method calls would leak memory if the C extension was enabled. (Bug #28278352)
- Missing bind() parameters could cause an unclear error message or unexpectedly halt. (Bug #28037275)
- The username and password fields are now quoted to allow special characters when making X DevAPI connections. (Bug #27528819, Bug #89614)

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

Installation Notes

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

X DevAPI Notes

- To increase compliance with the X DevAPI, these Connector/Python changes were made:
  - `DatabaseObject`: Deprecated: `am_i_real()` and `who_am_i()`. Added: `get_session()` and the `session` property.
  - `Collection.modify()`: Deprecated: `limit(x, y)`'s second parameter, `where(condition)`, and `change(CollectionField, ExprOrLiteral)`. Changed: the `modify(condition)` condition is now mandatory.
  - `Collection.find()`: Deprecated: `limit(x, y)`'s second parameter. Added: `limit(x).offset(y)`.
  - `Collection.remove()`: Deprecated: `limit(x, y)`'s second parameter and `where(condition)`. Changed: the `modify(condition)` condition is now mandatory.
  - `Table.select()`: Deprecated: `limit(x, y)`'s second parameter and `sort()`. Added: `limit(x).offset(y)`.
  - `Table.delete()`: Deprecated: `limit(x, y)`'s second parameter and `sort()`. Removed: `delete(x)`'s parameter in favor of using `where()` instead. Added: `order_by()`.
  - `Table.update()`: Deprecated: `limit(x, y)`'s second parameter, and the `sort()` method. Added: `order_by()`.
  - `Session`: Added: `get_schemas()`.
  - `Result`: Deprecated: `get_document_id()` and `get_generated_insert_id()`. Moved: `get_affected_items_count()` to the `BaseResult` class.

- `RowResult`: Added: `get_columns()`.
- `SqlResult`: Added: `has_data()`.
- `Column`: Renamed: `ColumnMetaData` to `Column`. Added properties: `schema_name`, `table_name`, `table_label`, `column_name`, `column_label`, `type`, `length`, `fractional_digits`, `collation_name`, `character_set_name`.

Functionality Added or Changed

- Removed MySQL Fabric support.
- An RPM package for installing ARM 64-bit (aarch64) binaries of Connector/Python 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.

Bugs Fixed

- The default character set changed from 'utf8' (an alias to the deprecated 'utf8mb3' character set) to 'utf8mb4'. (Bug #28188883)
- Fixed datetime conversion compatibility between Django 2.0 and MySQL 8.0.

  A workaround was to use Connector/Python's pure Python implementation instead the C extension by setting "use_pure=True" in Django's database options. (Bug #27962293, Bug #90541)
- The `get_row()` and `get_rows()` behavior differed with the C (connections with CMySQLConnection) and pure Python (connections with MySQLConnection) implementations of the connector. The resolved differences are:
  - With the pure Python implementation, all data was returned as bytearrays; while the C implementation returned all data as Python types with CMySQLConnection (cext). Both now return Python types.
  - With the pure Python implementation, they returned a tuple with (row(s), eof), but with the C Extension they only returned the row(s). Now both implementations return the tuple form; (row(s), eof).
  - For queries returning a result, with pure Python the warning count was part of the returned eof. With the C extension, warning count was only available using the warning_count property of the connection object. Related, the warning_count property was not available in the pure Python implementation. Now, result includes the warning count for both implementations.
  - Fetching rows using pure Python would automatically set the unread_rows property to False. With the C extension, explicitly calling the free_result() method of the connection object was required. Related, free_result() was only available with the C extension. Now, both implementations set unread_rows to False.

  (Bug #27411275, Bug #27991948, Bug #27802700, Bug #28133321, Bug #27650437, Bug #89305, Bug #90799, Bug #90292, Bug #91107)
- Connecting with a collation unknown to Connector/Python would yield an unknown character set error. It now properly references the unknown collation. (Bug #27277937)
- Deprecated the `Row.get_string()` method in favor of `__getitem__`. (Bug #26834200, Bug #87777)

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

Functionality Added or Changed

- *X DevAPI:* Previously, when documents without an `_id` attribute were added to a collection, Connector/Python automatically generated _id for them. Now a MySQL 8 server generates the `_id` attribute unless a document already contains one. The generated IDs resulting from a document-add operation can be obtained using the new `get_generated_ids()` method.

  This capability requires a MySQL 8.0 server. Because MySQL 5.7 does not support document ID generation, the document-add operation returns an error if you do not define the _id's.

  Incompatibility: The `get_generated_ids()` method replaces the now removed `get_document_ids()`. (Bug #27627366)

- Added *NOWAIT* and *SKIP_LOCKED* support to the ReadStatement.lock_shared() and ReadStatement.lock_exclusive() methods. Example usage: `lock_exclusive(mysqlx.LockContention.SKIP_LOCKED)`.

- The C extension (cext) is now enabled by default, as the `use_pure` option changed from *True* to *False* by default.

  If the C extension is not available on the system then the Python implementation is used instead, and `use_pure` is set to *True*.

- Added the X DevAPI SHA256_MEMORY authentication mechanism.

  Example mysqlx.get_session() usages: *?auth=SHA256_MEMORY* via a connection string, *"auth": mysqlx.Auth.SHA256_MEMORY* via a dictionary, or *auth=mysqlx.Auth.SHA256_MEMORY* via method parameters.

Bugs Fixed

- Warnings are now stored as a list of dictionaries instead of a list of tuples. In other words, *get_warnings()* returns the likes of *[{"level": _level_, "code": _code_, "msg": _msg_}]* instead of *[(_level_, _code_, _msg_)]*. (Bug #27639119)

- The mapped MySQL Server error codes were synced with MySQL Server 8.0.11. (Bug #27634885)

- Removed *upsert* functionality from *InsertStatement* as it can only be used by collections, so *upsert* remains available to *AddStatement*. (Bug #27589450)

- *MySQLConverter.escape()* functionality was added to *create_schema()*'s count mechanism. (Bug #27528842)

- When using prepared statements, string columns were returned as bytearrays instead of strings. The returned value is now a string decoded using the connection's `charset` (defaults to 'utf8'), or as a bytearray if this conversion fails. (Bug #27364914)

- The result from JSON_TYPE() was returned as a bytearray instead of a string. The returned value is now a string decoded using the connection's `charset` (defaults to 'utf8'), or as a bytearray if this conversion fails. (Bug #24948205, Bug #83516)

- JSON integer values were cast to bytes in Python instead of integers. (Bug #24948186, Bug #83513)

Changes in MySQL Connector/Python 8.0.7 - 8.0.10 (Skipped version numbers)

There are no release notes for these skipped version numbers.

Changes in MySQL Connector/Python 8.0.6 (2018-02-01, Release Candidate)

Functionality Added or Changed

- A new `bdist_wheel` distutils command was added to build a Connector/Python wheel package.

  A new `--static` option was added that enables static linking for the C extension variant.

- *X DevAPI:* In the process of refining the definition of the X DevAPI to cover the most relevant usage scenarios, the following API components have been removed from the X DevAPI implementation for Connector/Python:

  - **API components that support session configurations**.

    The `mysqlx.config` namespace and all members of the namespace.

  - The `create_table`, `drop_table`, `create_view`, `drop_view`, and `alter_view` methods from the `Schema` class.

- A *Pylint* test was added for the *mysqlx* module.

- A new `Modify.patch()` method was added to the X DevAPI as a way to change several document attributes in one operation; otherwise known as a JSON Merge Patch via RFC 7386.

- The `create_index()` method was added to the Collection API.

- The transaction API was extended to allow setting savepoints. The following methods have been added to the *Session* object:

  - `set_savepoint([name])`: executes the SAVEPOINT name SQL statement to generate a savepoint. If a name is not provided (or None), one is generated.

    The SAVEPOINT statement sets a named transaction savepoint with a name of identifier. If the current transaction has a savepoint with the same name, the old savepoint is deleted and a new one is set.

  - `release_savepoint(name)`: executes the RELEASE name SQL statement to release a savepoint.

    The RELEASE SAVEPOINT statement removes the named savepoint from the set of savepoints of the current transaction. No commit or rollback occurs. It returns an error if the savepoint does not exist.

  - `rollback_to(name)`: executes the ROLLBACK TO name SQL statement to rollback a savepoint.

    The ROLLBACK TO identifier command reverts the state of the transaction back to what was when executed the command SAVEPOINT identifier.

  Names passed to these functions are checked to make sure that the name is not null or an empty string. Names such as '', "", ``, and so on, are not allowed even though they are allowed by the server. For more information, see SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT Syntax.

Bugs Fixed

- On Enterprise Linux 7, SSL connections could fail due to the Python 2.7.9 or higher requirement. Since EL7 backported the SSL module from Python 3 (PEP466) into its default Python 2.7.5, SSL connections are now enabled on EL7. (Bug #27368032)
- MySQL Server 8.0 utf8mb4 collations were missing from Connector/Python. (Bug #27277964)
- The LICENSE and README files were missing from the C extension (“cext”) builds. (Bug #26912787)
- On Linux, commercial packages included source (.py) files in the package instead of only .pyc/.pyo files. (Bug #26821756)
- Python 3.6 is now officially supported and tested.

Changes in MySQL Connector/Python 8.0.5 (2017-09-28, Development Milestone)

Packaging Notes

- MySQL Connector/Python packages are now available in two formats: Pure Python packages that contain only Python files, and packages that contain the Python files plus the C Extension and C Protobuf extension. Exception platforms are Solaris, macOS, and Windows, for which packages containing the Python files and C extensions are available but not pure Python packages. (Bug #26648417)

Functionality Added or Changed

- MySQL Connector/Python now supports connections to MySQL accounts that use the `caching_sha2_password` authentication plugin (see Caching SHA-2 Pluggable Authentication). This requires MySQL server version 8.0.3 or higher. It also requires use of a secure connection because Connector/Python does not support RSA encryption for password exchange.
- MySQL Connector/Python now supports an `auth` connection option to specify the authentication methanism. Permitted values are `plain`, `mysql41`, and `external`. The option name and value are not case sensitive.

  If the authentication mechanism is not specified, it defaults to `plain` for secure (TLS) or Unix socket connections, or `mysql41` for insecure connections.
- MySQL Connector/Python now supports a pure Python implementation of Protobuf. Consequently, the Protobuf C extension has become optional. Connector/Python will use the Python implementation if the C extension is not available. The Protobuf Python package is required if it is desired not to use the C extension.

  The version requirements are Protobuf C++ 2.6.0 or higher, Protobuf Python 3.0.0 or higher.
- A `mysqlx.sessions` variable is now exposed to scripts that can be used for session-related tasks such as saving or loading session configuration information.
- These methods have been added for `Collection`: `add_or_replace_one()`, `get_one()`, `replace_one()`, and `remove_one()`.
- These methods have been added for `FindStatement` and `SelectStatement`, to enable shared and exclusive locks to be acquired: `lock_shared()` and `lock_exclusive()`.

- There is support for new forms of comparisons that use the `IN` operator:

```
item IN list
item IN document path
dict IN dict
```

  The left-hand-side value must be castable to the `JSON` type.

Changes in MySQL Connector/Python 8.0.4 (2017-07-17, 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/Python 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/Python 8.0.4 is the first release to use the new numbering. It is the successor to Connector/Python 2.2.3.

Character Set Support

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

Functionality Added or Changed

- 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. To intentionally apply an operation to an entire collection, pass a condition that always evaluates to true, such as `True`. A similar change was made to `Table.update()` and `Table.delete()`. (Bug #25991574)
- For MSI and Solaris packages, the pure Python Protobuf support implementation was replaced by a C++ extension. This enables Connector/Python to support Python 2 and 3 as well Protobuf 2 and 3.
- The `NodeSession` class has been renamed to `Session`, and the `XSession` class has been removed.
- 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`.
- 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.

Bugs Fixed

- The C Extension was not installed by some Connector/Python installers, such as Solaris `.pkg` and macOS `.dmg` installer packages. (Bug #24422244)
- `Collection.drop_index("name")` incorrectly returned an instance of `DropCollectionIndexStatement`.

Changes in MySQL Connector/Python 2.2

Changes in MySQL Connector/Python 2.2.3 (2017-03-22, 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/Python 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/Python 2.2.3 was the final release to use the old numbering. It is the predecessor to Connector/Python 8.0.4.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The pure Python implemention of Protobuf has been replaced by a C++ extension. This enables Connector/Python to support Python 2 and 3 as well as Protobuf 2 and 3. (Bug #25209469)
- Connector/Python now supports IPv6 target hosts in X DevAPI connection strings.

Bugs Fixed

- The `defined_as(statement)` method used to create views did not permit a `SelectStatement` object argument (generated by `Table.select()`). (Bug #25614860)
- The `SelectStatement` object returned by `Table.select()` failed to provide the `order_by()` method. (Bug #25519251)
- `import mysqlx` caused an error with Python 2.6 on Solaris and EL6 platforms. (Bug #24578507)
- The error message for `get_session()` failure was incorrect. (Bug #23636962)
- The `Collection.find()` method failed to work with the `LIKE` operator or aggregate functions.

  The `Collection.find()` method failed to work with several operators. Support was added for these operators:

  - Nullary Operators:

    ```
    *
    ```

  - Unary Operators:

    ```
    !, NOT, +, -, ~
    ```

  - Binary Operators:

    ```
    AND, &&, OR, ||, XOR, <>, ^
    IS NOT, NOT REGEXP, NOT LIKE, CAST, NOT IN
    ```

  - Ternary Operators:

    ```
    NOT BETWEEN
    ```

In addition, arrow notation to access `JSON` columns is now functional (for example, `schema.table.column->'$.document field'`). (Bug #23567724, Bug #23568207, Bug #25436568, Bug #84585)

Changes in MySQL Connector/Python 2.2.2 (2016-12-01, Development Milestone)

Functionality Added or Changed

- If the MySQL server is configured to support encrypted connections, Connector/Python now attempts to establish an encrypted connection by default, falling back to an unencrypted connection otherwise. This is behavior similar to the `--ssl-mode=PREFERRED` option supported by MySQL client programs.

  The following TLS/SSL options have been implemented for the `mysqlx` URI type string connection schema. All require Python 2.7.9 or higher.

  - `ssl-enable`: This option enforces SSL connections. If given, a connection attempt must be able to establish an encrypted connection or the attempt fails.
  - `ssl-ca`: This option is used to verify the server certificate.
  - `ssl-cert`, `ssl-key`: These options are used to pass the client certificate and key, but the server currently does not validate the client using these.

  The `ssl-enable` parameter can be specified in a parameter dictionary or URL, like this:

```
mysqlx.get_session({"user": "root", "host": "localhost", "port": 33060,
                    "password": "pass", "ssl-enable": True})
```

  Or:

```
mysqlx.get_session("mysqlx://root:pass@localhost?ssl-enable")
```

  The other parameters are used similarly. In a URI type string, path name values should be given within parentheses; for example, `ssl-cert=(path_name)`. See Connecting Using a URI or Key-Value Pairs. (Bug #24954646)

- There is now a standard API to create a table: `Schema` objects have a `create_table` function. It throws an error if the table exists.
- For any method that takes a value list of parameters for its argument, there is now more flexibility with how the parameters can be specified: Either as a value list or a list of individual parameters. For example, these method calls are the same:

```
Collection.add([{"a": 27}, {"a": 28}])
Collection.add({"a": 27}, {"a": 28})
```

- For `Schema` objects, `get_view`, `create_view`, `alter_view`, and `drop_view` functions were added to support retrieval, create, alter, and drop operations on `View` objects.
- On Unix and Unix-like systems, Unix domain socket files are now supported as a connection transport. The socket file can be specified in a parameter dictionary or URL, like this:

```
mysqlx.get_session({"user": "root", "password": "pass",
                    "socket": "/path/to/socket"})
```

Or:

```
mysqlx.get_session("mysqlx://user:pass@(/path/to/sock)/schema")
mysqlx.get_session("mysqlx://user:pass@/path%2Fto%2Fsock/schema")
mysqlx.get_session("mysqlx://user:pass@.%2Fpath%2Fto%2Fsock/schema")
mysqlx.get_session("mysqlx://user:pass@..%2Fpath%2Fto%2Fsock/schema")
```

Bugs Fixed

- For a user created with `REQUIRE SSL`, establishing an SSL connection by specifying `--ssl-key` but not `--ssl-ca` or `--ssl-cert` fails for standard MySQL client programs. The same connection configuration was (improperly) permitted in Connector/Python. (Bug #24953032)
- Connection failures due to an improper SSL CA resulted in an uninformative error message. (Bug #24948054)
- Using a schema object to alter a view failed if the view selected from a non-`INFORMATION_SCHEMA` table and it was altered to select from an `INFORMATION_SCHEMA` table. (Bug #24947078)
- `schema.create_collection()` with an empty collection name threw an improper error. (Bug #24520850)

Changes in MySQL Connector/Python 2.2.1 (2016-09-13, Development Milestone)

MySQL Connector/Python 2.2.1 has Protobuf 3 as a prerequisite, and is available in fewer distribution formats than usual (RPM and `tar.gz` packages only).

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

X DevAPI Notes

- Views can now be accessed like tables:
  - `Schema.get_tables()` now also returns a `Table` object for each existing `View`.
  - `Schema.get_table(name)` now also returns a `Table` object if the given name is a `View`.
  - All the operations available through a `Table` object are also available if the underlying object is a `View`. (The client will not do any validation in this regard.)
  - A new `Table.is_view()` method determines whether the underlying object is a `View`.
- The `Collection.add([]).execute()` construct now is treated as a no-operation and does not return an error. This is true even if there is no connection to the server or the collection does not exist on the server. The reasoning is that a request to add nothing to something that does not exist is trivially fulfilled.
- Connector/Python now implements support for these X DevAPI features:
  - Client failover
  - URI type string connections. See Connecting Using a URI or Key-Value Pairs.

  References: See also: Bug #23550057.

Functionality Added or Changed

- A new `get_default_schema()` method retrives a `Schema` object from the current session, given the schema name configured in the connection settings. For example, if the connection string is `"mysqlx://user:@127.0.0.1:33060/my_schema"`, `session.get_default_schema()` returns a `Schema` object for `my_schema`.
- Protobuf support was upgraded from Protobuf 2 to Protobuf 3 (which means that Protobuf3 is now a prerequsite for Connector/Python).

Bugs Fixed

- Attempts to fetch a value having the `BIT` data type produced an error. (Bug #23729357)
- The `fetchone()` result set method and `close()` session method were missing. They are now included. (Bug #23568257, Bug #23550743)

Changes in MySQL Connector/Python 2.2.0 (2016-07-12, Development Milestone)

MySQL Connector/Python 2.2.0 M1 is the first development release of the MySQL Connector/Python 2.2 series. This series adds support for 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.

To learn more about how to write applications using the X DevAPI, see X DevAPI User Guide. For more information about how the X DevAPI is implemented in MySQL Connector/Python, and its usage, see https://dev.mysql.com/doc/dev/connector-python/

Please note that 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.

Bugs Fixed

- When using the C Extension with `raise_on_warnings=True`, errors were not thrown as exceptions when an executed statement produced an error, and it was not possible to reuse the cursor if the statement produced a result set. (Bug #21536507)
- When using the C Extension, character decoding of identifiers (database, table, column names) in result sets could fail. (Bug #21535573)
- When using the C Extension with the `auth_plugin` option, `connect()` calls failed. (Bug #21529781)
- In connections for which `consume_results=True`, `callproc()` could hang. (Bug #21492815)
- Connections failed if the password began or ended with spaces because they were being stripped before the connection attempt. (Bug #21492428)
- Installation after configuring with the `--with-mysql-capi` option could fail if the download package had been renamed. (Bug #21490865)

Changes in MySQL Connector/Python 2.1

Changes in MySQL Connector/Python 2.1.8 (2018-08-31, General Availability)

Known limitation: MySQLConnection.cmd_change_user() with the C extension enabled does not establish a connection with the changed user. Instantiate a new MySQLConnection() connection as a workaround.

Functionality Added or Changed

- MySQL Connector/Python now supports connections to MySQL accounts that use the `caching_sha2_password` authentication plugin (see Caching SHA-2 Pluggable Authentication). This requires MySQL server version 8.0.3 or higher. It also requires use of a secure connection because Connector/Python does not support RSA encryption for password exchange.

Bugs Fixed

- On Windows, the 32-bit MSI failed to install. The registry key path was updated to allow the CEXT prerequisite check to execute and pass. (Bug #28395599, Bug #28464866)
- In pure Python mode, connecting to a MySQL server with *default_authentication_plugin* = *caching_sha2_password* would fail, even for accounts not using *caching_sha2_password*, because the authentication plugin was unknown to the Python connector. (Bug #27371245)
- When using prepared statements, string columns were returned as bytearrays instead of strings. The returned value is now a string decoded using the connection's `charset` (defaults to 'utf8'), or as a bytearray if this conversion fails. (Bug #27364914)
- Connecting to a MySQL server configured to use TLS versions other than TLSv1, such as as TLSv1.1 and TLSv1.2), was not possible and failed with a "[SSL: WRONG_VERSION_NUMBER] wrong version number" error. This is because the connector restricted the connection to use TLSv1.

  In addition, tsl_version support was added. (Bug #26484601, Bug #87121)
- The result from JSON_TYPE() was returned as a bytearray instead of a string. The returned value is now a string decoded using the connection's `charset` (defaults to 'utf8'), or as a bytearray if this conversion fails. (Bug #24948205, Bug #83516)
- JSON integer values were cast to bytes in Python instead of integers. (Bug #24948186, Bug #83513)

Changes in MySQL Connector/Python 2.1.7 (2017-08-18, General Availability)

Functionality Added or Changed

- Connector/Python is now compatible with Django 1.11. Thanks to Brian Tyndall for the patch for some of the work. (Bug #25965922, Bug #86105, Bug #26257499, Bug #86652)
- Connector/Python now attempts to connect using TLS by default if the server supports encrypted connections. (Bug #21947091)

Bugs Fixed

- Prepared statements did not work with a MySQL 8.0 server. (Bug #26376334)
- With a connection character set of `utf8mb4`, multiple-row insert operations failed with an error of `LookupError: unknown encoding: utf8mb4`. (Bug #24659561, Bug #82948)
- Creating a `Connection` instance with `use_pure=True` could lead to the underlying socket not being closed if the user name or password were invalid. Thanks to Vilnis Termanis for the patch. (Bug #24342757, Bug #82324)

- For cursors created with `named_tuple=True`, `MySQLCursorNamedTuple` objects could leak memory. Thanks to Piotr Jurkiewicz for the patch on which this fix was based. (Bug #22880163, Bug #80621)
- The C Extension leaked memory if used to execute `INSERT` statements that inserted Unicode strings. (Bug #22825962, Bug #79887)
- The `escape_string()` method leaked memory. (Bug #22810810, Bug #79903)
- With Python 2.x, for a call to `encode('utf8')` on a bytestring that was serialized from unicode, Python attempted to decode the string using the 'ascii' codec and encode it back using 'utf8'. The result was encoding failure for bytestrings that contain non-ASCII characters. (Bug #22564149, Bug #79993)

Changes in MySQL Connector/Python 2.1.6 (2017-04-18, General Availability)

Functionality Added or Changed

- Compatibility issues with Django 1.9 were corrected. (Bug #25726671)
- Methods for filtering time and datetime fields were changed in Django 1.9 from `value_to_db_datetime` to `adapt_datetimefield_value` and from `value_to_db_time` to `adapt_timefield_value`. Proxy methods with the previous names were added to Connector/Python ensure compatibility. Thanks to Brian Tyndall for the patch. (Bug #25349918, Bug #84410)
- Connector/Python added support for a database backend API change introduced in Django 1.9 for the `bulk_insert_sql` method. Thanks to Brian Tyndall for the patch. (Bug #25349897, Bug #84408)
- An `ssl-cipher` option is now supported for specifying the encryption cipher for encrypted connections. (Bug #22545879, Bug #78186)

Bugs Fixed

- The fix for Bug #22529828 caused Python 2.7 to be unable to insert binary data. (Bug #25589496, Bug #85100)

  References: This issue is a regression of: Bug #22529828.
- Some SQL statements that worked using pure Python failed with the Connector/Python C Extension enabled. (Bug #25558885)
- Connector/Python produced no error or warning if the server certificate was expired. (Bug #25397650)
- If an exception reset the underlying session, connections in a connection pooled could become unavailable to the pool. (Bug #25383644, Bug #84476)
- Extra encapsulation was removed from the `get_constraints` method for the `foreign_key` parameter. Thanks to Brian Tyndall for the patch. (Bug #25349912, Bug #84409)
- Loading the `world` sample database worked using pure Python but failed with the Connector/Python C Extension enabled. (Bug #22476689, Bug #79780)
- If the output from the `mysql_config --include` command included more than one directory, the C Extension failed to compile. (Bug #20736339, Bug #76350)

Changes in MySQL Connector/Python 2.1.5 (2016-12-15, General Availability)

Bugs Fixed

- Incorrect logic for handling EOF packets caused MySQL Utilities replication to fail. (Bug #25111218)
- Using the C Extension, connection failure occurred when the user name or password contained Unicode characters. (Bug #21656282)
- For connections established using the `use_pure=True` parameter, queries that returned more than 4096 columns produced an `InterfaceError`. (Bug #21530841)
- Using Python 3.3, connection failure occurred if the `option_groups` parameter named multiple groups. (Bug #21530100)
- Using `executemany()` to execute an `INSERT INTO ... SELECT` statement produced an `InterfaceError`. (Bug #21477493)
- If a call to `set_charset_collation()` failed, the character set became invalid rather than being unchanged. (Bug #21476495)

Changes in MySQL Connector/Python 2.1.4 (2016-10-06, General Availability)

Security Notes

- The linked OpenSSL library for Connector/Python 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/Python, which use the yaSSL library instead. The change also does not affect connections made using any pure-Python implementation of Connector/Python, for which the version of OpenSSL used is whatever is installed on the system.

Bugs Fixed

- Connector/Python failed to establish connections using the cleartext client-side authentication plugin for accounts using the PAM server-side authentication plugin. Thanks to Daniël van Eeden for the patch. (Bug #22873551, Bug #80609)
- A potential SQL injection vector was eliminated. (Bug #22529828, Bug #24816150, Bug #19487642, Bug #73611)
- Connections made using the C Extension failed when the `ssl_ca` parameter was given without `ssl_cert` and `ssl_key`. (Bug #21879914, Bug #79835, Bug #22494320)
- For connections made with `consume_results=True`, `consume_results` was reset to `False` after `callproc()` execution failure. (Bug #21879859)

  References: This issue is a regression of: Bug #21492815.
- In connections for which `compress=True`, `LOAD DATA LOCAL INFILE` statements produced “Packets out of error” errors. (Bug #21449996)

Changes in MySQL Connector/Python 2.1.3 (2015-09-24, General Availability)

Functionality Added or Changed

- Connector/Python is now compatible with Django 1.8. (Bug #76752, Bug #20987205)

Bugs Fixed

- When using the C Extension with `raise_on_warnings=True`, errors were not thrown as exceptions when an executed statement produced an error, and it was not possible to reuse the cursor if the statement produced a result set. (Bug #21536507)
- When using the C Extension, character decoding of identifiers (database, table, column names) in result sets could fail. (Bug #21535573)
- When using the C Extension with the `auth_plugin` option, `connect()` calls failed. (Bug #21529781)
- In connections for which `consume_results=True`, `callproc()` could hang. (Bug #21492815)
- Connections failed if the password began or ended with spaces because they were being stripped before the connection attempt. (Bug #21492428)
- Connection failure occurred for accounts authenticated with the `sha256_password` authentication plugin that had a blank password. (Bug #21420906)
- RPM packages of Connector/Python were missing some required `__init_py__` files. (Bug #77819, Bug #21505096)
- The Connector/Python C Extension could exit when fetching a result set containing many `NULL` values. (Bug #77678, Bug #21420633)
- Connector/Python failed to complete the connection handshake with MySQL Server 5.5.8. (Bug #77040, Bug #21090014)
- Connector/Python hung until timeout if the query it was running was killed. (Bug #76156, Bug #20653441)
- Writing to a table with a `BinaryField` from Django resulted in a `UnicodeDecodeError` exception. (Bug #75175, Bug #21054559)
- Stripping `NoneType` objects from Django resulted in an `AttributeError` exception. (Bug #74675, Bug #21054556)

Changes in MySQL Connector/Python 2.1.2 (2015-04-30, Beta)

Functionality Added or Changed

- Previously, connections from Connector/Python to Fabric were always made using the XML-RPC protocol. It is now possible to connect using the MySQL client/server protocol. To specify the protocol, use the `protocol` value in the `fabric` dictionary argument for the `connect()` method.

Permitted `protocol` values are `xmlrpc` (the default) and `mysql`. With `mysql`, the default port becomes 32275, although that can be changed with an explicit `port` value.

Bugs Fixed

- Connector/Python could raise an `AttributeError` exception for Fabric connections with MySQL Utilities 1.5.4 or 1.6.1. (Bug #20834643)
- The `setup.py install` command did not retain the value provided by the `--install-lib` option. (Bug #20217174)
- Encoding failure could occur for prepared cursors with UTF-8 statement parameters. (Bug #75542, Bug #20407036)
- The Connector/Python version checker for MySQL did not handle nonnumeric suffixes. During the build process, if the `--with-mysql-api` option was given, the check failed for installed versions of MySQL such as 5.7.6-m16. (Bug #75520, Bug #20365619)
- Values of the `SET` data type were not translated correctly if empty. (Bug #75402, Bug #20301989)
- `HASH` sharding for Fabric failed. (Bug #75287, Bug #20324089)
- Queries that produced a large result could result in an `IndexError: bytearray index out of range` exception. (Bug #74933, Bug #20462427)
- The Django back end was creating excessive connections (immediately when each `DatabaseWrapper` object was created rather than waiting until the object actually needed the connection.) (Bug #74696, Bug #19972427)
- Error messages containing non-ASCII characters caused an exception to be raised. (Bug #74345, Bug #19803702)
- The Django back end sometimes failed to properly convert `SafeText` objects, which then appeared in queries. (Bug #74336, Bug #20106629)
- When using the `callproc()` cursor method, warnings generated by statements executed within the procedure or generated by the procedure itself were not available to the client. (Bug #74252, Bug #19777815)
- Connection pooling did not work when using MySQL Fabric. (Bug #73445, Bug #19331658)

Changes in MySQL Connector/Python 2.1.1 (2015-02-23, Alpha)

C Extension Notes

- MySQL Connector/Python distributions now are available that include a C Extension that interfaces with the MySQL C client library. For queries that return large result sets, using the C Extension can improve performance compared to a “pure Python” implementation of the MySQL client/server protocol.

  For binary Connector/Python distributions, some packaging types have a single distribution file that includes the pure-Python Connector/Python code together with the C Extension. (Windows MSI and OS X Disk Image packages fall into this category.) Other packaging types have two related distribution files: One that includes the pure-Python Connector/Python code, and one that includes only the C Extension. For packaging types that have separate distribution files, install both distributions if you want to use the C Extension. The two files have related names, the difference being that the one that contains the C Extension has “cext” in the distribution file name.

Source distributions include both the pure-Python code and the C Extension, and distribution names do not contain “cext”. Instead, availability of the C Extension is determined by whether you compile the distribution with the `--with-mysql-capi` option.

Packages for Connector/Python with the C Extension are available at the Connector/Python download site. For installation instructions, see Connector/Python Installation. For information about using the C Extension, see The Connector/Python C Extension.

For Connector/Python installations that support the C Extension, the `use_pure` connection argument to `connect()` controls whether to use the extension. If `use_pure` is `True` (the default), the connection uses pure Python. If `use_pure` is `False`, the connection uses the C Extension.

It is also possible to use the C Extension directly, by importing the `_mysql_connector` module rather than the `mysql.connector` module. See The _mysql_connector C Extension Module.

Functionality Added or Changed

- A new `connect()` option, `consume_results`, if enabled, causes result sets generated by queries to be automatically consumed and discarded. The `can_consume_results` connection object property indicates the value of `consume_results`.

Bugs Fixed

- With `mysql.connector.django` as the engine, the `python manage.py inspectdb` command failed with an “Unread result found” error. (Bug #20022533)
- If the `pool_size` and `pool_name` connection arguments were specified using the option file (as opposed to being passed explicitly to the connect call), the pooled connection was successfully created, but an exception was raised when closing it. (Bug #19549363)
- The Django back end raised an exception when converting `"0000-00-00 00:00:00"` to `None`. (Bug #73940, Bug #19667984)
- The `type_code` in `cursor.description` did not compare equal to any of the type objects defined in `mysql.connector.dbapi`. (Bug #73798, Bug #19584051)
- Data corruption occurred when inserting sufficiently large data in a table with a `TEXT` type column using prepared statements, due to incorrect encoding of the data length while sending the prepared statement packet. (Bug #73690, Bug #19522948)
- When the character set was `binary`, character set conversion could occur. Conversion is no longer done and `binary` data is returned as is. (Bug #71909, Bug #19500097)

Changes in MySQL Connector/Python 2.1.0 (Not released, Alpha)

MySQL Connector/Python 2.1.0 is a labs-only release.

Version 2.1.0 has no changelog entries, or they have not been published because the product version has not been released.

Changes in MySQL Connector/Python 2.0

Changes in MySQL Connector/Python 2.0.5 (2016-10-26, General Availability)

Bugs Fixed

- A potential SQL injection vector was eliminated. (Bug #22529828, Bug #24816150, Bug #19487642, Bug #73611)

Changes in MySQL Connector/Python 2.0.4 (2015-04-21, General Availability)

Bugs Fixed

- The `Changes.txt` file was missing from `.msi` and `.dmg` packages. (Bug #20323087)
- Encoding failure could occur for prepared cursors with UTF-8 statement parameters. (Bug #75542, Bug #20407036)
- Values of the `SET` data type were not translated correctly if empty. (Bug #75402, Bug #20301989)
- `HASH` sharding for Fabric failed. (Bug #75287, Bug #20324089)
- Queries that produced a large result could result in an `IndexError: bytearray index out of range` exception. (Bug #74933, Bug #20462427)
- The Django back end sometimes failed to properly convert `SafeText` objects, which then appeared in queries. (Bug #74336, Bug #20106629)

Changes in MySQL Connector/Python 2.0.3 (2015-01-28, General Availability)

Bugs Fixed

- The Django back end was creating excessive connections (immediately when each `DatabaseWrapper` object was created rather than waiting until the object actually needed the connection.) (Bug #74696, Bug #19972427)
- Using the Django back end, it was not possible to connect to a `connection_created` signal. (Bug #74679, Bug #19954882)
- `recv_plain()` could fail to read a packet header correctly, resulting in a lost connection. (Bug #74483, Bug #19930054)
- Error messages containing non-ASCII characters caused an exception to be raised. (Bug #74345, Bug #19803702)
- When using the `callproc()` cursor method, warnings generated by statements executed within the procedure or generated by the procedure itself were not available to the client. (Bug #74252, Bug #19777815)
- Connection pooling did not work when using MySQL Fabric. (Bug #73445, Bug #19331658)

Changes in MySQL Connector/Python 2.0.2 (2014-11-03, General Availability)

Bugs Fixed

- If the `pool_size` and `pool_name` connection arguments were specified using the option file (as opposed to being passed explicitly to the connect call), the pooled connection was successfully created, but an exception was raised when closing it. (Bug #19549363)
- The Django back end raised an exception when converting `"0000-00-00 00:00:00"` to `None`. (Bug #73940, Bug #19667984)
- Using a `connection_created` signal defined in `django.db.backends.signals` caused a “maximum recursion depth reached” runtime error. (Bug #73847, Bug #19584116)
- The `type_code` in `cursor.description` did not compare equal to any of the type objects defined in `mysql.connector.dbapi`. (Bug #73798, Bug #19584051)

- Data corruption occurred when inserting sufficiently large data in a table with a `TEXT` type column using prepared statements, due to incorrect encoding of the data length while sending the prepared statement packet. (Bug #73690, Bug #19522948)
- When the character set was `binary`, character set conversion could occur. Conversion is no longer done and `binary` data is returned as is. (Bug #71909, Bug #19500097)

Changes in MySQL Connector/Python 2.0.1 (2014-09-24, General Availability)

Functionality Added or Changed

- Connector/Python is now compatible with Django 1.7. (Bug #72746, Bug #19163169)
- `RANGE_DATETIME` is now supported as a sharding type. This is similar to the regular `RANGE` sharding type, but instead of an integer key, requires a datetime or date object. For example, to get the shard which holds employees hired after the year 2000, you could do the following, with lower bounds set as "group1/1980-01-01, group2/2000-01-01":

  ```
  cnx.set_property(tables=["employees.employees"],
                   key=datetime.date(2000, 1, 1), mode=fabric.MODE_READONLY)
  ```

  If the lower bounds included a time, it would have been like this:

  ```
  cnx.set_property(tables=["employees.employees"],
                   key=datetime.datetime(2000, 1, 1, 12, 0, 0),
                   mode=fabric.MODE_READONLY)
  ```

  Only `datetime.datetime` and `datetime.date` values are supported. Any other type given when using a shard defined using `RANGE_DATETIME` causes a `ValueError` to be raised.
- `RANGE_STRING` is now supported as a sharding type. This is similar to the regular `RANGE` sharding type, but instead of an integer key, requires a UTF-8 encoded string. For example:

  ```
  cnx.set_property(tables=["employees.employees"],
                   key=u'employee_name', mode=fabric.MODE_READONLY)
  ```

  Only Unicode strings are supported. Any other type given when using a shard defined using `RANGE_STRING` causes a `ValueError` to be raised.

Bugs Fixed

- Connector/Python failed to catch an exception when SSL capability was found to be unavailable. (Bug #19440592)
- Date and time query formatting was fixed for the Django back end. (Bug #19179711)
- Multiple `[connector_python]` option groups sometimes caused an error. (Bug #19170287)
- An error failed to occur if an option file was named multiple times. (Bug #19169143)
- Some valid Connector/Python connection options were not recognized when specified in the `[connector_python]` option group. (Bug #19168737)
- `!include` and `!includedir` directives in option files were not handled properly. (Bug #73660, Bug #19481761)

- Binding `None` (`NULL`) to a parameter marker in a prepared statement did not work. (Bug #73370, Bug #19282158)
- With Python 2, Connector/Python could truncate digits of floating-point values. (Bug #73266, Bug #19225481)
- An exception was raised when a cursor tried to convert `LINESTRING` data as UTF-8 data. Now such values are returned without decoding. (Bug #73187, Bug #19164627)
- Connector/Python now supports a `shutdown()` method that, unlike `disconnect()`, closes the client connection without attempting to send a `QUIT` command to the server first. Thus, it will not block if the connection is disrupted for some reason such as network failure. (Bug #72691, Bug #18798953)

Changes in MySQL Connector/Python 2.0.0 (2014-07-24, Alpha)

Functionality Added or Changed

- **Incompatible Change:** Previous series of Connector/Python had separate Python 2 and Python 3 code bases. For Connector/Python 2.0, the source tree has been reorganized to have a single code base, for easier maintenance, testing, and distribution.

  This reorganization results in an incompatible change in behavior: With the use of “raw” cursors, the returned values is of the `bytearray` type. This is necessary for having both Python 2 and 3 return the same data. Consider the following example:

  ```
  import mysql.connector

  cnx = mysql.connector.connect(raw=True)
  cursor = cnx.cursor()
  cursor.execute('SELECT 1')
  print(cursor.fetchall())
  ```

  In Connector/Python 1.x, the output is:

  - Using Python 2: `[('1',)]`
  - Using Python 3: `[(b'1',)]`

  In Connector/Python 2.0, for both Python versions, the output is: `[(bytearray(b'1'),)]`

  To get the same value as in Connector/Python 1.x, do this:

  - Using Python 2: `str(bytearray(b'1'))`
  - Using Python 3: `bytes((bytearray(b'1'))`
- **Important Change:** Previously, to enable use of `LOAD DATA LOCAL INFILE`, clients had to explicitly set the `ClientFlag.LOCAL_FILES` flag. This flag is now enabled by default. To disable it, the `allow_local_infile` option for `connect()`can be set to `False`.
- For a stored procedure that produces multiple result sets, it is now possible to execute the procedure and process its results by executing a `CALL` statement. Execute the statement using `execute()` with a `multi=True` argument, and use the returned iterator to process each result in turn. (Bug #73291, Bug #19207922)
- Connector/Python now supports option files using two new options for `connect()`:

- `option_files`: Which option files to read. The value can be a file path name (a string) or a sequence of path name strings. By default, Connector/Python reads no option files, so this argument must be given explicitly to cause option files to be read. Files are read in the order specified.
- `option_groups`: Which groups to read from option files, if option files are read. The value can be an option group name (a string) or a sequence of group name strings. If this argument is not given, the default value is `['client, 'connector_python']` to read the `[client]` and `[connector_python]` groups.

For more information, see Connector/Python Option-File Support.

- The `mysql.connector.cursor` module supports four new cursor classes:
  - The `MySQLCursorDict` cursor class returns each row as a dictionary. The keys for each dictionary object are the column names of the MySQL result.

    ```
    cursor = cnx.cursor(dictionary=True)
    ```

  - The `MySQLCursorBufferedDict` cursor class is like `MySQLCursorDict`, but fetches the entire result set after executing the query and buffers the rows.

    ```
    cursor = cnx.cursor(dictionary=True, buffered=True)
    ```

  - The `MySQLCursorNamedTuple` cursor class returns each row as a named tuple. Each column is accessible through an attribute of the tuple-like object.

    ```
    cursor = cnx.cursor(named_tuple=True)
    ```

  - The `MySQLCursorBufferedNamedTuple` cursor class is like `MySQLCursorNamedTuple`, but fetches the entire result set after executing the query and buffers the rows.

    ```
    cursor = cnx.cursor(named_tuple=True, buffered=True)
    ```

  For more information, see Subclasses cursor.MySQLCursor.

- The packaging modules and supporting files have been removed from the main repository and from the source packages for Connector/Python. They are still available in the Connector/Python 1.x series.

Bugs Fixed

- Django `TimeField` values of `00:00:00` were incorrectly converted to `NULL` because Python considered that value equal to `False`. (Bug #72732, Bug #18956789)
- Fetching results from a prepared statement that returned many columns could produce an error. (Bug #72602, Bug #18742429)
- Previously, a `RuntimeError` exception was raised when a Django application was inactive for a while. Now, the Django back end verifies that the database connection is still valid each time a database request is made. (Bug #72545, Bug #18843153)

Changes in MySQL Connector/Python 1.2

Changes in MySQL Connector/Python 1.2.4 (Not released)

Bugs Fixed

- Using a `connection_created` signal defined in `django.db.backends.signals` caused a "maximum recursion depth reached" runtime error. (Bug #73847, Bug #19584116)

Changes in MySQL Connector/Python 1.2.3 (2014-08-22, General Availability)

Functionality Added or Changed

- Connector/Python is now compatible with Django 1.7. (Bug #72746, Bug #19163169)

Bugs Fixed

- The specification and control files were updated to reflect that Connector/Python 1.2 cannot be used with MySQL Utilities 1.5.1 (which requires Connector/Python 2.0 or greater). (Bug #19444705)
- Connector/Python failed to catch an exception when SSL capability was found to be unavailable. (Bug #19440592)
- With Python 2, Connector/Python could truncate digits of floating-point values. (Bug #73266, Bug #19225481)
- An exception was raised when a cursor tried to convert `LINESTRING` data as UTF-8 data. Now such values are returned without decoding. (Bug #73187, Bug #19164627)
- Django `TimeField` values of `00:00:00` were incorrectly converted to `NULL` because Python considered that value equal to `False`. (Bug #72732, Bug #18956789)
- Fetching results from a prepared statement that returned many columns could produce an error. (Bug #72602, Bug #18742429)
- Previously, a `RuntimeError` exception was raised when a Django application was inactive for a while. Now, the Django back end verifies that the database connection is still valid each time a database request is made. (Bug #72545, Bug #18843153)
- Negative timedelta values were incorrectly converted to and from Python. Thanks to Vitali Graf for the patch. (Bug #72493, Bug #18694096)

Changes in MySQL Connector/Python 1.2.2 (2014-05-27, General Availability)

Bugs Fixed

- The Fabric connection configuration permitted `username` but not `user` as a parameter name, which is inconsistent with the connection arguments permitted by Connector/Python itself. Now either can be used. (Using both raises a `ValueError`.) (Bug #18463182)
- In the `MySQLProtocol._auth_response` method of the `mysql.connector.protocol` module, the `auth_response` variable was changed without being defined first. (Bug #18463182)
- Commercial Debian Connector/Python packages included a copyright file containing a GPL license. (Bug #18422727)

- For Fabric connections, the Weighted Round Robin (WRR) load balancing algorithm stopped working properly due to cache problems. (Bug #17995416)
- Building an RPM package using `python setup.py bdist_rpm` did not work. (Bug #72261, Bug #18550039)
- The community MSI Connector/Python packages contained empty documentation PDF and HTML files. These have been removed and replaced with the `README_DOCS.txt` file which contains a URL to the online manual. (Bug #72245, Bug #18527132)
- For Python 3, when parameters were passed as a dictionary to the `MySQLCursor` methods `execute()` and `executemany()`, only first occurrence of each element in the query was replaced by the parameter value. (Bug #71975, Bug #18389196)
- Connector/Python raised all deprecation warnings as errors when Django was run in debug mode. Now only database warnings are raised as errors in debug mode. (Bug #71806, Bug #18380134)
- when `MySQLCursor.execute()` was passed values of a data type which cannot be converted, the exception raised was not easy to understand. Now a nicer error message is displayed when unconvertible Python types are given. (Bug #71729, Bug #18258807)

Changes in MySQL Connector/Python 1.2.1 (2014-03-31, Release Candidate)

Functionality Added or Changed

- Connector/Python now permits the type for stored procedure parameters to be specified. To do this, specify a parameter as a two-item tuple consisting of the parameter value and type. For more information, see MySQLCursor.callproc() Method. (Bug #71124, Bug #17965619)
- It was not possible to initiate an SSL session without explicitly giving a key and certificate. Now it is possible to connect to a MySQL server using only the `ssl_ca` connection argument pointing to a file of CA certificates. This means the `ssl_key` and `ssl_cert` connection arguments are optional. However, when either is given, both must be given or an `AttributeError` is raised. (Bug #69418, Bug #17054848)
- Connector/Python now supports authentication plugins found in MySQL 5.6. This includes `mysql_clear_password` and `sha256_password`, both of which require an SSL connection. The `sha256_password` plugin does not work over a non-SSL connection because Connector/Python does not support RSA encryption.

  The `connect()` method now supports an `auth_plugin` parameter that can be used to force use of a particular plugin. For example, if the server is configured to use `sha256_password` by default and you want to connect to an account that authenticates using `mysql_native_password`, either connect using SSL or specify `auth_plugin='mysql_native_password'`. (Bug #68054, Bug #16217765)
- The `connect()` method now accepts a `failover` argument that provides information to use for server failover in the event of connection failures. The argument value is a tuple or list of dictionaries (tuple is preferred because it is nonmutable). Each dictionary contains connection arguments for a given server in the failover sequence. Permitted dictionary values are: `user`, `password`, `host`, `port`, `unix_socket`, `database`, `pool_name`, `pool_size`.
- Connector/Python now enables applications to specify additional information to be used when connecting to Fabric: User name and credentials, and information to use for establishing an SSL connection. The `fabric` dictionary argument to the `connect()` method accepts these additional values: `username`, `password`, `ssl_ca`, `ssl_cert`, `ssl_key`. Only the `ssl_ca` value is required to establish an SSL connection. If `ssl_cert` or `ssl_key` are given, both must be specified.

- Connector/Python now can report errors to Fabric that occur while accessing a MySQL instance. The information can be used to update the backing store and trigger a failover operation, provided that the instance is a primary server and Fabric has received a sufficient number of problem reports from different connectors.

  - The `fabric` dictionary argument to the `connect()` method now accepts a `report_errors` value. Its default value is `False`; pass a value of `True` to enable error reporting to Fabric.

  - To define which errors to report, use the `extra_failure_report()` function:

    ```
    from mysql.connector.fabric import extra_failure_report
    extra_failure_report([error_code_0, error_code_1, ...])
    ```

- A new `MySQLConnection` class `reset_connection()` method enables applications to send a `COM_RESET_CONNECTION` to the server. This method is analogous to the `mysql_reset_connection()` C API function added in MySQL 5.7.3.

  A new `MySQLConnection` class `reset_session()` method is similar to `reset_connection()` but falls back to use reauthentication for older servers that do not support `COM_RESET_CONNECTION`. For more information, see MySQLConnection.cmd_reset_connection() Method, and MySQLConnection.reset_session() Method.

Bugs Fixed

- The `MySQLConnection.autocommit` attribute failed to set the value of the `self._autocommit` attribute. (Bug #18172769)

- Uninstalling Connector/Python using an RPM package failed to remove the `fabric` folder. (Bug #18143073)

- The global `MYSQL_FABRIC_PORT` variable was changed from 8080 to 32274 to match the port change made in Fabric. (Bug #18075339)

  References: See also: Bug #70954.

- For Fabric connections, any `connect_attempts` and `connect_delay` values specified by the user were ignored. (Bug #18055719)

- For Fabric sharding operations, Connector/Python raised an incorrect error when a table was given with the `tables` connection property for which no sharding information was available. This now results in a `DatabaseError` (with `errorcode.ER_BAD_TABLE_ERROR`) mentioning that the table is unknown. (Bug #18047794)

- For Fabric operations, an incorrect exception was raised by `set_property()` when a connection property value had the wrong type (for example, when the `tables` property was not a tuple or a list). `set_property()` now correctly raises a `ValueError`. (Bug #18047758)

- For Fabric operations, the default mode was supposed to be read/write but was set to read-only. (Bug #18047591)

- The delay between attempts when trying to connect to a MySQL Fabric-managed server was not honored. (Bug #71905, Bug #18335432)

- Fabric has renamed the dump functionality to a new command called `dump`. Consequently, Connector/Python now uses the new functions `dump.sharding_information`, `dump.fabric_nodes`, and `dump.servers`. (Bug #71124, Bug #17965619)

- `MySQLCursor.executemany()` caused a `UnicodeDecodeError` when non-ASCII characters existed in the `seq_params` parameter and the operation was a Unicode instance with Python 2. This is now corrected by encoding the operation per the current connection character set. (Bug #69067, Bug #18220593)

Changes in MySQL Connector/Python 1.2.0 (2013-12-23, Alpha)

Functionality Added or Changed

- Connector/Python now supports Fabric. Supported capabilities include:
  - High-Availability group lookup using read-only or read-write mode
  - Range and hash sharding support
  - Failure reporting to Fabric
  - Failover support
  - Load balancing based on MySQL server weight

Changes in MySQL Connector/Python 1.1

Changes in MySQL Connector/Python 1.1.7 (2014-05-13, General Availability)

Bugs Fixed

- Commercial Debian Connector/Python packages included a copyright file containing a GPL license. (Bug #18422727)
- For Django, introspecting to get the primary key of MySQL tables could fail in Python 3. (Bug #72001, Bug #18380100)
- In prepared statements, Unicode arguments in Python 2 and bytes arguments in Python 3 were causing errors, as were the symbols of character sets other than `utf8` or `ascii`. (Bug #71482, Bug #18144971)

Changes in MySQL Connector/Python 1.1.6 (2014-02-19, General Availability)

Bugs Fixed

- Connector/Python produced errors using time functions with Django 1.6 due to not using the autocommit value from Django. Now the value is set to that specified in the Django configuration file. (Bug #71438, Bug #18187561)

Changes in MySQL Connector/Python 1.1.5 (2014-01-31, General Availability)

Functionality Added or Changed

- Connector/Python is now compatible with Django 1.6. (Bug #17857712)
- `utf8mb4` is now recognized as a valid character set. (Bug #70596, Bug #17780576)
- The `start_transaction()` method now supports a `readonly` argument. This argument can be `True` to start the transaction in `READ ONLY` mode or `False` to start it in `READ WRITE` mode. If `readonly` is omitted, the server's default access mode is used. For details about transaction access mode, see the description for the `START TRANSACTION` statement at START TRANSACTION, COMMIT, and ROLLBACK Syntax. If the server is older than MySQL 5.6.5, it does

not support setting the access mode and Connector/Python raises a `ValueError`. (Bug #70545, Bug #17573172)

Bugs Fixed

- When using connection pooling, a connection returned to the pool was not reset, so session variables retained their values. Now these variables are reset by re-authenticating the user when the connection is returned to the pool. To disable this behavior, pass a `pool_reset_session` argument to `connect()` when requesting a pooled connection:

  ```
  cnx = mysql.connector.connect(pool_reset_session=False,...)
  ```

  (Bug #18040042)

- An incorrectly handled error in `MySQLProtocol.parse_column_count()` method could lead to a misreported error message. (Bug #17958420)

- `executemany()` failed with `INSERT INTO ... SELECT` statements. (Bug #70529, Bug #17826833)

Changes in MySQL Connector/Python 1.1.4 (2013-12-17, General Availability)

MySQL Connector/Python 1.1.4 is a new version of the pure Python database driver for MySQL. This is the first GA (General Availability) version of Connector/Python 1.1.

MySQL Connector/Python version 1.1 is compatible with MySQL Server versions 5.5 and greater, but should work with earlier versions greater than 4.1. Python 2.6 and greater as well as Python 3.1 and greater are supported. Python 2.4 and 2.5 are not supported.

Bugs Fixed

- Python method call overhead was reduced for certain update and select operations. (Bug #17890173)

Changes in MySQL Connector/Python 1.1.3 (2013-11-15, Alpha)

Functionality Added or Changed

- The Connector/Python source code has been made compliant with PEP-8 to the extent possible.

Bugs Fixed

- Connection pooling did not correctly handle unavailable servers; for a connection that could not be established, it failed to return the connection to the pool. Now reconnection is attempted and if that fails, the connection is returned to the pool. (Bug #17578937)

- There was a problem saving data containing the backslash character or 0x5c using multibyte character sets such as `sjis`, `big5`, or `gbk`. To handle this, there is a new `HexLiteral` type. When a backslash is found in such as `sjis`, `big5`, or `gbk` data, the string is sent as a hexadecimal literal to MySQL. (Bug #69710, Bug #17079344)

- Connection attempts failed on older versions of FreeBSD. (Bug #69088, Bug #17372107)

Changes in MySQL Connector/Python 1.1.2 (2013-10-23, Alpha)

- Bugs Fixed

Functionality Added or Changed

- The error message raised when a connection pool has no more connections available now indicates “pool exhausted” rather than “queue is empty”. (Bug #17406263)
- Previously, instantiating a cursor for prepared statements was done using `MySQLConnection.cursor(cursor_class=MySQLCursorPrepared)`. Now this can be done using `MySQLConnection.cursor(prepared=True)`. (Bug #17215197)
- Previously, setting a custom converter class was possible after instantiating a new connection object. The `connect()` method now accepts a `converter_class` connection argument that takes a class and sets it when configuring the connection. An `AttributeError` is raised if the custom converter class is not a subclass of `conversion.MySQLConverterBase`. (Bug #13551483)
- The `connect()` method now accepts a boolean `compress={False|True}` argument indicating whether to use the compressed client/server protocol (default `False`). This provides an easier alternative to setting the `ClientFlag.COMPRESS` flag. (Bug #13369592)

Bugs Fixed

- In some cases, when a Connector/Python application exited, a `RuntimeError` was raised when using Python 3. (Bug #17424009)
- `cmd_shutdown()` did not work correctly when a server for MySQL 5.6 or higher raised a `DatabaseError` (1835: Malformed communication packet). (Bug #17422299)
- Attempts to change the size of an existing connection pool were not rejected. (Bug #17372107)
- The `DatabaseOperations.last_executed_query()` method in the Django base module was unnecessarily decoding the string, resulting in an error when using Python 3. (Bug #70324, Bug #17473273)

Changes in MySQL Connector/Python 1.1.1 (2013-09-10, Alpha)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Incompatible Change:** The original message passed to `errors.Error()` was not saved in such a way that it could be retrieved. Instead, the `Error.msg` attribute was formatted with the error number and SQLSTATE value. Now only the original message is saved in the `Error.msg` attribute. The formatted value together with the error number and SQLSTATE value can be obtained by printing or getting the string representation of the error object. Example:

```
try:
  conn = mysql.connector.connect(database = "baddb")
except mysql.connector.Error as e:
  print "Error code:", e.errno        # error number
  print "SQLSTATE value:", e.sqlstate # SQLSTATE value
  print "Error message:", e.msg       # error message
  print "Error:", e                   # errno, sqlstate, msg values
  s = str(e)
  print "Error:", s                   # errno, sqlstate, msg values
```

  (Bug #16933795)

- Output for individual unit tests did not show timings, making it more difficult to debug problems that involve a change in test execution time. `unittest.py` now has a new `--stats` option that runs tests and shows elapsed time for each.

It is also possible to save the data to a MySQL server. When the `--stats-host` option is given with other options such as `--stats-user`, results are saved to a table called `myconnpy_X_Y_Z`. The table contains the name of the test case and columns that combine Python and MySQL versions; for example, py27my55 or py33my56.

For example, to see the difference between MySQL 5.1 and 5.6, using Python 2.7, after running the test cases for both using Connector/Python 1.1.0, use this statement:

```
SELECT test_case, py27my51, py27my56, (py27my56-py27my51) AS diff51
FROM myconnpy_1_1_0 WHERE (py27my56-py27my51) > 0.5;
```

(Bug #17028999)

- Connector/Python now includes a `mysql.connector.django` module that provides a Django back end for MySQL. This back end supports new features found in MySQL 5.6 such as fractional seconds support for temporal data types. For more information, see Connector/Python Django Back End.
- MySQL Connector/Python now supports simple connection pooling that has these characteristics:
  - A pool opens a number of connections and handles thread safety when providing connections to requesters.
  - The size of a connection pool is configurable at pool creation time. It cannot be resized thereafter.
  - A connection pool can be named at pool creation time. If no name is given, one is generated using the connection parameters.
  - The connection pool name can be retrieved from the connection pool or connections obtained from it.
  - It is possible to have multiple connection pools. This enables applications to support pools of connections to different MySQL servers, for example.
  - For each connection request, the pool provides the next available connection. No round-robin or other scheduling algorithm is used.
  - It is possible to reconfigure the connection parameters used by a pool. These apply to connections obtained from the pool thereafter. Reconfiguring individual connections obtained from the pool by calling the connection `config()` method is not supported.

  Applications that can benefit from connection-pooling capability include:

  - Middleware that maintains multiple connections to multiple MySQL servers and requires connections to be readily available.
  - Web sites that can have more “permanent” connections open to the MySQL server.

  The connection pooling implementation involves these interface elements:

  - A new module, `mysql.connector.pooling`, provides two classes: `MySQLConnectionPool` instantiates and manages connection pools, and `PooledMySQLConnection` is similar to `MySQLConnection` but is used for connections that are part of a connection pool.
  - A new exception, `PoolError`, occurs for pool-related exceptions. `PoolError` is a subclass of `Error`.

  For more information, see Connector/Python Connection Pooling.

Bugs Fixed

- Following `fetchone()` or `fetchmany()`, the result returned by `fetchall()` was missing one row. (Bug #17041412)
- Previously, executing a statement after the connection was closed raised an `OperationalError` with an unclear error. Connector/Python now returns the client error 2006, `MySQL Server has gone away`, with an extra message.

  The `Error()` class has been extended to accept a new argument, `extra_msg`. When given, it is appended between brackets. For example: `[2000] Unknown MySQL Error (Some extra message)` (Bug #17022399)
- `LOAD DATA LOCAL INFILE` failed for files approximately 14MB or larger. (Bug #17002411)
- Invoking `executemany()` without any data produced a `ProgrammingError` rather than doing nothing. (Bug #16660356)
- An `InternalError` was raised during transaction rollback if there were unread results. The `MySQLConnection.rollback()` method now consumes unread results instead of raising an error. (Bug #16656621)
- Python 2.6 and 2.7 raised a `UnicodeDecodeError` when `unicode_literals` was used and a database name contained nonlatin Unicode characters. (Bug #16655208)
- The `MySQLCursor.executemany()` method raised an exception when an SQL function was used as a column value when executing an `INSERT` statement. (Bug #69675, Bug #17065366)
- An unclear `OperationalError` was raised if a cursor object was closed while there were unread results. Connector/Python now raises an `InternalError` indicating that there are still unread results. This provides information that to avoid the error it is necessary to consume the result by reading all rows. (Bug #67649, Bug #17041240)

Changes in MySQL Connector/Python 1.1.0 (2013-07-02, Alpha)

Functionality Added or Changed

- **Incompatible Change:** Python 2 code was changed to use new features introduced in Python 2.6 and 2.7. Some examples:
  - `print()` is used as a function, not a statement.
  - Exceptions are handled using the `as` keyword.
  - The `in` keyword is used instead of the `has_key()` dictionary method.

  This change means that MySQL Connector/Python 1.1 does not work with versions of Python older than 2.6.
- Connector/Python was updated with error information from MySQL 5.7.1. (Bug #16896702)
- `mysql.connector.__version__` and `mysql.connector.__version_info__` now are available to provide MySQL Connector/Python version information in a more standard, Pythonic manner.
- `MySQLConnection` objects now support an `in_transaction` property that returns `True` or `False` to indicate whether a transaction is active for the connection.

- `MySQLConnection` objects now support a `start_transaction()` method to begin a transaction. This method accepts arguments indicating whether to use a consistent snapshot and which transaction isolation level to use:

```
cnx.start_transaction(consistent_snapshot=bool,
                      isolation_level=level)
```

  The default `consistent_snapshot` value is `False`. The default `isolation_level` value is `None`, and permitted values are `'READ UNCOMMITTED'`, `'READ COMMITTED'`, `'REPEATABLE READ'`, and `'SERIALIZABLE'`.

- Connector/Python supports a new `MySQLCursorPrepared` class that enables execution of prepared SQL statements using the binary client/server protocol. For details, see cursor.MySQLCursorPrepared Class.

Bugs Fixed

- Relative imports were removed from Python 3 code. PEP-8 indicates that relative imports are discouraged. (Bug #16234372)

Changes in MySQL Connector/Python 1.0

Changes in MySQL Connector/Python 1.0.12 (2013-07-24, General Availability)

Bugs Fixed

- Following `fetchone()` or `fetchmany()`, the result returned by `fetchall()` was missing one row. (Bug #17041412)
- `LOAD DATA LOCAL INFILE` failed for files approximately 14MB or larger. (Bug #17002411)
- The `fetchall()` methods for buffered cursors were returning all rows after `fetchone()` or `fetchmany()` were used. `fetchall()` now correctly returns all or remaining, just like the nonbuffered cursors. (Bug #16662920)
- Python 2.6 and 2.7 raised a `UnicodeDecodeError` when `unicode_literals` was used and a database name contained nonlatin Unicode characters. (Bug #16655208)
- The `MySQLCursor.executemany()` method raised an exception when an SQL function was used as a column value when executing an `INSERT` statement. (Bug #69675, Bug #17065366)
- An unclear `OperationalError` was raised if a cursor object was closed while there were unread results. Connector/Python now raises an `InternalError` indicating that there are still unread results. This provides information that to avoid the error it is necessary to consume the result by reading all rows. (Bug #67649, Bug #17041240)

Changes in MySQL Connector/Python 1.0.11 (2013-07-01, General Availability)

Functionality Added or Changed

- Connector/Python was updated with error information from MySQL 5.7.1. (Bug #16896702)
- Debian (`.deb`) packages for Connector/Python are now available.

Changes in MySQL Connector/Python 1.0.10 (2013-05-07, General Availability)

Functionality Added or Changed

- A new connection option `ssl_verify_cert` checks the SSL certificate for the server against the certificate found in the file specified by the `ssl_ca` option. This option is disabled by default. Any certificate mismatch of invalid combination of SSL options will raise a `ValueError` exception. (Bug #16400735)
- Connector/Python now supports the `LOCAL` keyword for `LOAD DATA LOCAL`. (Bug #16369511, Bug #16736916)
- The `MySQLConnection.cmd_shutdown()` method now accepts an optional shutdown type. A new `ShutdownType` constants class was added. (Bug #16234441)
- The GPL Connector/Python packages contained non-GPL documentation. This could be an issue when Linux distributions would like to repackage. PDF and other documentation formats now are removed from the GPL packages, which point in the `README_DOCS.txt` file to online availability of the manual. (Bug #68509, Bug #16430013)

Changes in MySQL Connector/Python 1.0.9 (2013-02-26, General Availability)

Functionality Added or Changed

- Previously, when setting up an SSL connection, the developer had to set the `ClientFlag.SSL` explicitly in the `client_flags` argument of the `mysql.connector.connect()` function call. Now, whenever SSL arguments are specified, the client flag is set automatically. This change makes the SSL behavior of Connector/Python more consistent with other MySQL connectors. (Bug #16217667, Bug #68172)

Bugs Fixed

- The `DistUtils` command was not copying `version.py` into the `build` directory, so that the `build/lib` directory could not be used for development without manually copying `version.py`. (Bug #16236136)
- Passing string parameters to a stored procedure resulted in extra quotes being included in the value. This was caused by the conversion from Python to MySQL data types being applied two times. We now only convert once, and pass the values correctly.

  `MySQLCursor.callproc()` now also raises a `ValueError` when the type of an argument is incorrect. (Bug #16217743, Bug #68066)
- Fixed IPv6 for older Microsoft Windows versions. Also improved the associated code for all operating systems: we now use `socket.getaddrinfo()` instead of `inet_pton()` to check whether we are connecting using IPv4 or IPv6.

  A new connection option `force_ipv6` has been introduced. When set to `True`, IPv6 will be used when an address resolves to both IPv4 and IPv6. Otherwise, IPv4 is favored. (Bug #16209119)

Changes in MySQL Connector/Python 1.0.8 (2012-12-21, General Availability)

Fixes bugs since the initial 1.0.7 GA release.

Bugs Fixed

- When a stored procedure was called with arguments, and produced multiple result sets, the result sets were not returned properly. (Bug #15916486, Bug #67710)
- The `ping()` method was always reconnecting to the database, ignoring the `reconnect` argument. This means that there would be a rollback when pinging the MySQL server during a transaction.

  Now `ping()` will honor the `reconnect` option and only reestablish the connection when needed. (Bug #15915243, Bug #67650)
- Connector/Python could not connect to MySQL servers using IPv6 addresses. An `InterfaceError` or `ConnectionRefusedError`was raised:

  ```
  mysql.connector.errors.InterfaceError: 2003: Can't connect to MySQL server on
  'IPv6-style address' (Address family for hostname not supported)

  ConnectionRefusedError: [Errno 111] Connection refused
  ```

  (Bug #15876886, Bug #15927825)
- When connecting to a MySQL server from a host whose IP address was not allowed, Connector/Python reported a handshake problem and raised an `InterfaceError` exception. (Bug #15836979)
- When a username or password was passed in as Unicode to Connector/Python, connection attempts failed with `UnicodeDecodeError` exceptions due to string concatenation of mixed-charset types. This issue affected programs running under Python 2, and did not affect Python 3. (Bug #14843456, Bug #67306)
- Intermittent errors could occur on Windows systems: `InterfaceError(errno=2013)`. The cause was incorrect handling of `sock.recv()` library calls that returned less data than was requested. (Bug #14829471, Bug #67303)
- A socket error would produce a `NameError` exception instead of the expected `InterfaceError`, due to a misnamed variable:

  ```
  NameError: global name 'e' is not defined
  ```

  (Bug #14802017)
- The `executemany()` function now supports the pyformat parameter style. In the pyformat style, all the substitution variables are passed in using a single dictionary parameter, and the `%` format specifier is encoded like `%(dict_key)s` for a string. `MySQLCursor.executemany()` can now use both ANSI C printf and Python extended format codes. (Bug #14754894, Bug #67146)
- The error message was clarified when a non-integer value was used for the TCP/IP port connection argument. (Bug #13808727, Bug #64543)

Changes in MySQL Connector/Python 1.0.7 (2012-09-29, General Availability)

GA release. Connector/Python is now production-ready.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Client and server error messages have been regenerated using the MySQL 5.6.6 development release.

Bugs Fixed

- Fixed formatting of client errors changing numeric to string placeholders. (Bug #14548043)

Changes in MySQL Connector/Python 1.0.6 (2012-08-30, Beta)

Second beta release.

Functionality Added or Changed

- Changed how MySQL server errors are mapped to Python exceptions. We now use the `SQLState` (when available) to raise a better error.
  - Incompatibility: some server errors now are raised with a different exception.
  - It is possible to override how errors are raised using the `mysql.connector.custom_error_exception()` function, defined in the `mysql.connector.errors` module. This can be useful for certain frameworks to align with other database drivers.
- Changed name and version of distributions to align with other MySQL projects:
  - The version now includes the suffix 'b' for beta and 'a' for alpha followed by a number. This version is used in the source and built distributions. GA versions will have no suffix.
  - The RPM spec files have been updated to create packages whose names are aligned with RPMs from other MySQL projects.

Bugs Fixed

- Fixed version-specific code so Connector/Python works with Python 3.3. (Bug #14524942)
- Fixed `MySQLCursorRaw.fetchall()` so it does not raise an exception when results are available. (Bug #14517262, Bug #66465)
- Timeout for unit tests has been set to 10 seconds. Test cases can individually adjust it to be higher or lower. (Bug #14487502)
- Fixed installation of `version.py` on OS X:
  - `version.py` is now correctly installed on OS X in the `mysql.connector` package. Previously, it was installed through `data_files`, and `version.py` ended up in the system-wide package location of Python, from which it could not be imported.
  - `data_files` is not used any longer in `setup.py` and is removed. Extra files like `version.py` now are copied in the custom `Distutils` commands.

  (Bug #14483142)
- Fixed test cases in `test_mysql_database.py` that failed when using `YEAR(2)` with MySQL 5.6.6 and greater. (Bug #14460680)
- Fixed SSL unit testing for source distributions:
  - The SSL keys and certificates were missing and now are added to the source distribution. Now SSL testing works properly.
  - Additionally for the Windows platform, forward slashes were added to the option file creation so the MySQL server can pick up the needed SSL files.

(Bug #14402737)

Changes in MySQL Connector/Python 1.0.5 (2012-07-17, Beta)

First beta release.

Functionality Added or Changed

- Added descriptive error codes for both client and server errors in the module `errorcode`. A new sub-package `locales` has been added, which currently only supports English client error messages.

  For example, `errorcode.CR_CONNECTION_ERROR` is 2002.

- Added SQLMode class in the constants module to make it easier to set modes. For example:

```
cnx.sql_mode = [SQLMode.REAL_AS_FLOAT, SQLMode.NO_ZERO_DATE]
```

Changes in MySQL Connector/Python 1.0.4 (2012-07-07, Alpha)

Internal alpha release.

Bugs Fixed

- **Incompatible Change:** The method `MySQLConnection.set_charset()` has been removed and replaced by `MySQLConnection.set_charset_collation()` to simplify setting and retrieving character set and collation information. The `MySQLConnection` properties `collation` and `charset` are now read-only. (Bug #14260052)

- **Incompatible Change:** The `MySQLConnection` methods `unset_client_flag()` and `set_client_flag()` have been removed. Use the`set_client_flags()` method instead using a sequence. (Bug #14259996)

- **Incompatible Change:** Fixed `MySQLConnection.cmd_query()` to raise an error when the operation has multiple statements. We introduced a new method `MySQLConnection.cmd_query_iter()` which needs to be used when multiple statements send to the MySQL server. It returns a generator object to iterate through results.

  When executing single statements, `MySQLCursor.execute()` will always return `None`. You can use the `MySQLCursor` property `with_rows` to check whether a result could have rows or not.

  `MySQLCursor.execute()` returns a generator object with which you can iterate over results when executing multiple statements.

  The `MySQLCursor.next_resultset()` became obsolete and was removed and the `MySQLCursor.next_proc_result()` method has been renamed to `MySQLCursor.proc_results()`, which returns a generator object. The `MySQLCursor.with_rows` property can be used to check if a result could return rows. The `multiple_resultset.py` example script shows how to go through results produced by sending multiple statements. (Bug #14208326)

- Fixed `MySQLCursor.executemany()` when `INSERT` statements use the `ON DUPLICATE KEY` clause with a function such as `VALUES()`. (Bug #14259954)

- Fixed unit testing on the Microsoft Windows platform. (Bug #14236592)

- Fixed converting a `datetime.time` to a MySQL type using Python 2.4 and 2.5. The `strftime()` function has no support for the `%f` mark in those Python versions. (Bug #14231941)

- Fixed `cursor.CursorBase` attributes `description`, `lastrowid` and `rowcount` to be read-only properties. (Bug #14231160)
- Fixed `MySQLConnection.cmd_query()` and other methods so they check first whether there are unread results. (Bug #14184643)

Changes in MySQL Connector/Python 1.0.3 (2012-06-08, Alpha)

Internal alpha release.

Functionality Added or Changed

- Adding support for time values with a fractional part, for MySQL 5.6.4 and greater. A new example script `microseconds.py` was added to show this functionality.
- Adding new `Distutils` commands to create Windows Installers using WiX and RPM packages.

Changes in MySQL Connector/Python 1.0.2 (2012-05-19, Alpha)

Internal alpha release.

Functionality Added or Changed

- Added more unit tests for modules like `connection` and `network` as well as testing the SSL functionality.

Bugs Fixed

- Fixed bootstrapping MySQL 5.6 running unit tests.

  Messages send by the bootstrapped MySQL server to `stdout` and `stderr` now are discarded. (Bug #14048685)
- Fixing and refactoring the `mysql.connector.errors` module. (Bug #14039339)

Changes in MySQL Connector/Python 1.0.1 (2012-04-26, Alpha)

Internal alpha release.

Functionality Added or Changed

- Change the version so it only contain integers. The 'a' or 'alpha' suffix will not be present in packages, but it will be mentioned in the `_version.py` module since `metasetupinfo.py` uses this information to set, for example, the Trove classifiers dynamically.

Changes in MySQL Connector/Python 1.0.0 (2012-04-22, Alpha)

Internal alpha release.

Functionality Added or Changed

- **Incompatible Change:** `MySQLConnection.reconnect()` can be used to reconnect to the MySQL server. It accepts number of retries and an optional delay between attempts.

`MySQLConnectiong.ping()` is now a method and works the way the MySQL C API `mysql_ping()` function works: it raises an error. It can also optionally reconnect.

`MySQLConnection.is_connected()` now returns `True` when connection is available, `False` otherwise.

`ping()` and `is_connected()` are backward incompatible. (Bug #13392739)

- Refactored the modules connection and protocol and created a new module `network`. The `MySQLProtocol` does not keep a reference to the connection object any more and deals only with creating and parsing MySQL packets. Network interaction is now done by the `MySQLConnection` objects (with the exception of `MySQLProtocol.read_text_result()`).

Bugs Fixed

- Fixed `metasetupinfo.py` to use the Connector/Python which is being installed instead of the version already installed. (Bug #13962765)
- Fixed `MySQLCursor.description` so it stores column names as Unicode. (Bug #13792575)
- Fixed `dbapi.Binary` to be a bytes types for Python 3.x. (Bug #13780676)
- Fixed automatic garbage collection which caused memory usage to grow over time. Note that `MySQLConnection` does not keep track of its cursors any longer. (Bug #13435186)
- Fixed setting time zone for current MySQL session. (Bug #13395083)
- Fixed setting and retrieving character set and collation. (Bug #13375632)
- Fixed handling of errors after authentication for Python 3. (Bug #13364285)

Index

L

M

P

T

U