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 (http://dev.mysql.com/downloads/), the version has not yet been released.

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

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

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

Document generated on: 2016-08-17 (revision: 9605)

Table of Contents

Preface and Legal Notices

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

Legal Notices

Changes in MySQL Connector/Python 2.2

Changes in MySQL Connector/Python 2.2.1 (Not yet released)

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)
- Session creating using a URL-format connection string failed. (Bug #23550057)

Changes in MySQL Connector/Python 2.2.0 (2016-07-12, Milestone 1)

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 http://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 2.1.3 (2015-09-24)

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 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. For more information, see Requesting a Fabric Connection.

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 backend 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 backend 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)

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 backend 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 yet been published because the product version has not yet been released.

Changes in MySQL Connector/Python 2.0

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

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 backend 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)

Bugs Fixed

- The Django backend 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 backend, 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)

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 backend 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 backend. (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 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 backend 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 yet 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)

- Functionality Added or Changed
- Bugs Fixed

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 backend 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. For more information, see Requesting a Fabric Connection.

- 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, ...])
    ```

  For more information, see Requesting a Fabric Connection.

- 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. See Connector/Python Fabric Support. 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)

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)

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)

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)

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)

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)

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 backend for MySQL. This backend supports new features found in MySQL 5.6 such as fractional seconds support for temporal data types. For more information, see Connector/Python Django Backend.
- 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)

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)

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)

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)

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)

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 (29 September 2012, 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 (30 August 2012)

Second beta release.

- Functionality Added or Changed
- Bugs Fixed

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 (17 July 2012, 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 (07 July 2012)

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 (08 June 2012)

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 (19 May 2012)

Internal alpha release.

- Functionality Added or Changed
- Bugs Fixed

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 (26 April 2012)

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 (22 April 2012, Alpha)

Internal alpha release.

- Functionality Added or Changed

- Bugs Fixed

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)