MySQL Connector/Node.js Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL Connector/Node.js 8.0.

For additional Connector/Node.js documentation, see http://dev.mysql.com/.

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 legal information, see the Legal Notices.

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

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

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL Connector/Node.js 8.0.

Legal Notices

Documentation Accessibility

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

Access to Oracle Support for Accessibility

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

Changes in MySQL Connector/Node.js 8.0

Changes in MySQL Connector/Node.js 8.0.35 (2023-10-25, General Availability)

Bugs Fixed

- Some values stored in DECIMAL and NUMERIC columns lost precision when returned back to the client after executing a table query. The numeric conversion logic now returns all such values as raw strings rather than returning some as JavaScript numbers which lost precision. (Bug #35707417)
- Date and time operators are now allowed in X DevAPI expressions. (Bug #35690736)

- Rows can now be inserted into a table using CRUD with an X DevAPI expression. For example, a geometry object can now be inserted into a geometry column with table.insert(). (Bug #35666605)

Changes in MySQL Connector/Node.js 8.0.34 (Not released, General Availability)

There was not a MySQL Connector/Node.js 8.0.34 release.

Changes in MySQL Connector/Node.js 8.0.33 (2023-04-18, General Availability)

Bugs Fixed

- Updated the underlying TypeScript type declarations for method parameters to allow BigInt values. (Bug #35047820)
- Removed the getAutoIncrementValue() declaration from the RowResult TypeScript API definition. (Bug #34970587)
- Moved the getAffectedItems() method from the RowResult API, where it always returned 0, to the SqlResult API. (Bug #34970001)
- Made the "DocumentOrJSON" definition in the "CollectionDocuments" namespace less strict to allow any kind of JavaScript object (not just Plain Old JavaScript Objects) as: export type DocumentOrJSON = object | string. This improves TypeScript compatibility with --strict enabled. (Bug #34959626)
- The getAutoIncrementValue() method's return value would lose precision when a BIGINT AUTO_INCREMENT column contained a number greater than Number.MAX_SAFE_INTEGER in JavaScript. (Bug #34896695)
- The Table.select() method's TypeScript definition did not specify possible argument types; and this resulted in an error when the TypeScript compiler (tsc) compiled the application code to JavaScript. (Bug #109189, Bug #34947621)

Changes in MySQL Connector/Node.js 8.0.32 (2023-01-17, General Availability)

Functionality Added or Changed

- Optimized the X DevAPI expression parser for both structure and performance. The X DevAPI expressions are now encapsulated in a corresponding `Expr` module, and are parsed at the stage where they are used. The expression parser became stateless and is no longer re-created every time an expression is used, and all protobuf messages are now only constructed at the corresponding adapter.

  Also, `mysqlx.expr()` now returns an X DevAPI Expr instance instead of a protobuf message instance. (WL #12019)
- Updated dependencies to the latest versions compatible with the minimum required Node.js engine version (v14.0.0), namely google-protobuf to 3.21.2. Also updated these devDependencies entries: chai to v4.3.6, dns2 to v2.0.5, and mocha to v10.0.0. Updated the test suite to account for breaking changes introduced by the dns2 update. (WL #15434)
- On Linux and macOS, added a script that builds and runs a Docker container to execute the test suite. Only a running MySQL server is needed to run this test suite, see `CONTRIBUTING.md` for `./test/docker/run.sh` usage details. (WL #15353)

Bugs Fixed

- A document containing a numeric value not safely represented by a JavaScript number would lose precision and convert it to the nearest value capable of being represented as a JavaScript number. This new number was potentially unsafe for performing mathematical operations. Replaced the native JavaScript JSON parser with a bespoke version that is capable of safely representing unsafe numeric values as JavaScript strings. (Bug #34767204, Bug #34728259)

Changes in MySQL Connector/Node.js 8.0.31 (2022-10-11, General Availability)

Bugs Fixed

- Improved JSON_UNQUOTE + JSON_EXTRACT shorthand syntax behavior, which was inconsistent. Previously "->>" usage could differ between Session.sql() (SQL) and Table.select() (CRUD interface) usage. For example, '{ "key": '42' }' would return as '42' or 42, respectively, but now yields a utf8mb4 string '42' in both cases. (Bug #31017606)

- Relaxed the expression parser to allow leading and trailing whitespace in the expression strings. (Bug #29795595)

- Improved globstar ("**") error handling in the expression parser; expressions now explicitly fail in cases where the globstar is the last item in a document path. (Bug #27361584)

- Shift operator usage could return different results between CRUD and plain SQL contexts. (Bug #93834, Bug #29178528)

Changes in MySQL Connector/Node.js 8.0.30 (2022-07-26, General Availability)

Functionality Added or Changed

- Updated internal collation and character set mapping to include the new language-specific utf8mb4 collations added in MySQL Server 8.0.30. (WL #15227)

- The minimum required Node.js engine is now v14.0.0. This means that it's not guaranteed that Connector/Node.js will function with older Node.js versions. Previously the minimum version was v12.0.0 which has since reached EOL status (April 2022). (WL #15029)

- Added TypeScript support by introducing type definition files that include type declarations for the most relevant parts of the public API. This allows TypeScript applications to seamlessly use Connector/Node.js; and ensures proper support for TypeScript-enabled tooling available on IDEs (such as Visual Studio Code) to provide integrated type hints, contextual auto-completion, and development-time type errors. (WL #14631)

Bugs Fixed

- Now ensure that DECIMAL values are returned as JavaScript strings instead of numbers if there's a chance of losing precision. (Bug #34016587)

- Updated the client placeholder assignment logic to ensure that undefined placeholder values are ignored; and to externalize validation tasks to the server where the X Plugin yields errors. Those errors are then relayed to the application in a more consistent feedback loop.

For example, mixing conventions between single placeholder assignment and assignment maps with bind() resulted in the following unexpected error: "TypeError: Cannot read property 'valueOf' of undefined". (Bug #33940584)

Changes in MySQL Connector/Node.js 8.0.29 (2022-04-26, General Availability)

Functionality Added or Changed

- When a connection disabled SSL mode (such as `ssl-mode`=DISABLED), specifying other TLS/SSL connection options caused Connector/Node.js to throw an error. With this change, disabling SSL now means those additional TLS/SSL connection options are ignored. For example, defining `tls-versions` with `ssl-mode`=DISABLED raised an error in previous versions of Connector/Node.js.

  Now server certificates are only verified against a certificate authority if `ssl-mode` is either VERIFY_CA or VERIFY_IDENTITY. The exception is a warning message is logged if `ssl-ca` is set while `ssl-mode` is not equal to VERIFY_CA or VERIFY_IDENTITY. (Bug #26117220, WL #14837)

- Updated the google-protobuf dependency from v3.14.0 to v3.19.4. Issues fixed in the newer Protobuf versions are described at https://github.com/protocolbuffers/protobuf/releases. (WL #15079)

Bugs Fixed

- Removed the *timekeeper* package dependency, and replaced time travel functionality with manual time control using testdouble mocks. (Bug #33784970)

- Closing a connection pool now closes all open connection requests to the connection pool, when before the open requests remained open until the application process was terminated. (Bug #33761268)

- Creating a DATETIME index for a collection where the corresponding document field was provided as a native JavaScript Date instance yielded the following error: "Error: Incorrect datetime value". The Zulu Time indicator used for UTC date strings was replaced by an explicit "+00:00" suffix as the date indicator. (Bug #33740190)

Changes in MySQL Connector/Node.js 8.0.28 (2022-01-18, General Availability)

Deprecation and Removal Notes

- The TLSv1 and TLSv1.1 connection protocols were previously deprecated in Connector/Node.js 8.0.26 and support for them is removed starting with this release. Instead, use TLSv1.2 or TLSv1.3. (WL #14810)

Functionality Added or Changed

- Added support for multiple CA and CRL PEM files. (Bug #25589348, WL #14637)

Changes in MySQL Connector/Node.js 8.0.27 (2021-10-19, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Pluggable Authentication

- Added support for the following commercial authentication plugins: `authentication_kerberos_client`, `mysql_clear_password`, `authentication_ldap_sasl_client` (SCRAM-SHA1, SCRAM-SHA256 and GSSAPI), and `authentication_oci_client`

  Support is implemented using the required `node-mysql2` driver. (WL #14743, WL #14744, WL #14745, WL #14746, WL #14747)

Functionality Added or Changed

- Unified the error and warning handling infrastructure. (WL #14617)

Bugs Fixed

- An unexpected error occurred when closing a client instance without a pool. This fix makes `Client.close()` functionally equivalent to `Session.close()` for standalone connections. (Bug #33175234)
- Improved the connection pool mechanism, namely parallelism issues that stemmed from exceeding `mysqlx_max_connections` as the pool was not properly queuing the request to acquire further connections when they become available. (Bug #33175092, Bug #104374)

Changes in MySQL Connector/Node.js 8.0.26 (2021-07-20, General Availability)

- Deprecation and Removal Notes
- Functionality Added or Changed
- Bugs Fixed

Deprecation and Removal Notes

- The TLSv1.0 and TLSv1.1 connection protocols are now deprecated and support for them is subject to removal in a future Connector/Node.js version. Using these protocols now raises a warning. (WL #14565)

Functionality Added or Changed

- Management of connections to the MySQL server is now handled independently of the X DevAPI `Session` interface. (WL #12430)

Bugs Fixed

- Replaced all instances of the deprecated `Buffer()` API for the recommended alternatives (`Buffer.alloc()`, `Buffer.allocUnsafe()`, and `Buffer.from()`) to avoid unnecessary warning messages. (Bug #32820267, Bug #103419)
- For `addOrReplaceOne()` and `replaceOne()`, an error is now reported if the object provided as the second argument contains an `_id` property that does not match the value provided as the first argument.

  The previous behavior was unexpected. For example, when calling `addOrReplaceOne('1', { _id: '2', name: 'foo' })`, a document with `_id: '1'` was created if it did not exist, or if it did exist then any existing `name` property was updated with the value `'foo'`, or one was created. (Bug #32763298)

- SQL statement instances can now be executed with different placeholder values. (Bug #32750927)
- We now ensure that all floating point numbers are represented either by JavaScript numbers or by X DevAPI expression literals that are encoded as an X Protocol `V_DOUBLE` type; this helps avoid loss of precision. (Bug #32687374, Bug #103124)
- We now prevent column metadata callback functions used as push-based cursors in the `execute()` method from being called more than once for each result set generated by a statement. (Bug #32631911)
- Upon exceeding `connectTimeout` to a MySQL server, when a connection pool contained an idle connection that previously established a connection to same server and in the meantime the server became unresponsive but reachable, then an application became unresponsive after trying to acquire a connection from the pool. (Bug #32205365)
- If `connectTimeout` was exceeded as the client tried to connect to a server, the connection socket was not implicitly closed. (Bug #32200234)

Changes in MySQL Connector/Node.js 8.0.25 (2021-05-11, General Availability)

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

Changes in MySQL Connector/Node.js 8.0.24 (2021-04-20, General Availability)

Functionality Added or Changed

- **Important Change:** Starting with this release, Connector/Node.js ignores compatibility with Node.js versions that have reached their end of life. If you are planning to upgrade, make sure you are using Node.js 12.0.0 or higher. (WL #14258)
- Connector/Node.js no longer guarantees compatibility with Node.js versions that have reached their end of life. This means that, as of this release, Connector/Node.js officially supports Node.js versions 12.0.0 and later only. (WL #14257)
- Improved the inline documentation and added functionality for users to generate it locally using JSDoc. See `CONTRIBUTING.md` for instructions on how to generate the HTML documentation. (WL #13733)
- Server disconnection handling of X Protocol connections has been improved such that it now creates a log entry and returns an error message, as needed after Connector/Node.js receives notice of a closing connection from the server. Connector/Node.js detects three new types of warning notices.
  - *Connection idle*: This notice applies to a server connection that remains idle for longer than the relevant timeout setting. Connector/Node.js closes the connection when it receives the notice in an active session or while a new session is being created. An attempt to use the invalid session returns `Connection closed. Reason: connection idle too long`.
  - *Server shutdown*: If notice of a closing connection is received in a session as a result of a server shutdown, Connector/Node.js terminates the session with `Connection closed. Reason: server shutdown`. If connection pooling is in use, all other sessions that are connected to the same endpoint are removed from the pool.
  - *Connection killed*: If the connection being killed was made from another client session, Connector/Node.js closes the connection when it receives the notice in an active session or while a new session

is being created. An attempt to use the invalid session returns `Connection closed. Reason: connection killed by a different session` error message.

(WL #14201, WL #13492)

Bugs Fixed

- The `getSession()` method now releases connections from a pool when an exception is raised while connecting to the server. (Bug #32366743, Bug #101928)
- `Session.getDefaultSchema()` returned 'undefined' instead of a valid Schema instance if a default schema was not set. (Bug #32136490)

Changes in MySQL Connector/Node.js 8.0.23 (2021-01-18, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added custom debug functionality using the `NODE_DEBUG` environment variable to log and inspect low-level details for the application. Connector/Node.js now supports this feature and uses it, in particular, for logging information about the protocol messages (inbound and outbound) that are exchanged with the MySQL server. Messages sent by the client are available under the protocol:outbound scope, whereas messages sent by the server are available under the protocol:inbound scope.

  For example, the following writes a textual protobuf representation of every `Mysqlx.Crud.Find` and `Mysqlx.Resultset.Row` message to stderr:

  ```
  shell> NODE_DEBUG='protocol:outbound:Mysqlx.Crud.Find,protocol:inbound:Mysqlx.Resultset.Row' node app.js
  ```

  Node 10 and later supports wildcard pattern matching, such as `NODE_DEBUG='protocol:inbound:*'` to filter out inbound messages and `NODE_DEBUG='*'` to show all logs. (Bug #31729043, WL #13587)

- Added a deprecation warning to the Connector/Node.js installation process advising that future Connector/Node.js versions do not guarantee compatibility with Node.js versions past their end of life. If you are planning to upgrade, make sure you do so to Node.js 12.0.0 or later. (WL #14257)

Bugs Fixed

- Stored values in a `DOUBLE` column were truncated when encoded and sent to the MySQL server. For example, a value such as 1.000001 became 1. Such values are now encoded properly as double-precision floating numbers supporting the full range of 64-bit floating point precision in MySQL `DOUBLE` columns (with possible approximations performed by the database engine). (Bug #31734504)
- Improved consistency for method argument error handling when an argument is not set or uses JavaScript's "undefined". (Bug #31709879)
- All debugging utilities are now disabled when debug mode is off. (Bug #31584269)

Changes in MySQL Connector/Node.js 8.0.22 (2020-10-19, General Availability)

- Deprecation and Removal Notes
- Functionality Added or Changed

- Bugs Fixed

Deprecation and Removal Notes

- Deprecated the `dbPassword` and `dbUser` property names, which were aliases for, respectively, the `password` and `user` properties. Using the old names now raises deprecation errors. (Bug #31599660)

Functionality Added or Changed

- Test execution configuration has been aligned with that of other MySQL Connectors. This includes unifying environment variable names; for example, `NODE_TEST_MYSQL_HOST` has been changed to `MYSQLX_HOST`. See the Connector/Node.js documentation for more information. (WL #13854)

Bugs Fixed

- Values of types other than `BIGINT` which were stored in `BIGINT` columns were not decoded properly in result sets. (Bug #31686805, Bug #100324)
- Fetched results from a `SET` column contained only one value from the set. (Bug #31654667, Bug #100255)
- Added a `SERVER_GONE` error handler to avoid potential circular dependency warnings with Node.js 14.0.0 and later. (Bug #31586107, Bug #99869)
- The `offset()` method is now available only with `CollectionFind` and `TableSelect`, as described in the X DevAPI specification. Using `offset()` with other interfaces yielded the error `Error: The server has gone away`. The intended behavior is available by using a combination of either `sort()` or `orderBy()`, together with `limit()`. (Bug #31418813)
- The `nextResult()` method returned false when used with an empty result set; in such cases, it now returns true. To check whether a result set contains data, use the `hasData()` method. (Bug #31037211)
- The `column.getType()` method returned the numeric value for a type identifier; now it returns the type identifier's name. For example, the method now returns `DATETIME`, rather than `12` as before. (Bug #30922711)
- Improved memory management for work performed by third-party APIs. (Bug #30845472)
- Added support for “lazy” decoding of binary column metadata. (Bug #30845366)

Changes in MySQL Connector/Node.js 8.0.21 (2020-07-13, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Creating a collection now supports options to enable validation of a JSON schema that documents must adhere to before they are permitted to be inserted or updated. The `ModifyCollection()` method added in this release allows updating the schema of an existing collection. In addition, the `createCollection()` method's option for reusing an existing collection has been renamed from `ReuseExistingObject` to `reuseExisting`.

  Schema validation is performed by the server, which returns an error message if a document in a collection does not match the schema definition or if the server does not support validation.

If a given collection already exists in the database, `createCollection()` fails unless `reuseExisting` is enabled in an additional options object, as shown in the following example:

```
const mysqlx = require('@mysql/xdevapi');

mysqlx.getSession('mysqlx://localhost:33060')
    .then(sesion => {
        return session.getSchema('mySchema').createCollection('myCollection', { reuseExisting: true })
    });
```

You can also use the options object, for example, to create a serverside document validation schema. To do this, include a schema property matching a valid JSON schema definition within an outer validation object. You should also include the `level` property where `STRICT` enables validation and `OFF` disables it, as shown in the following example:

```
const mysqlx = require('@mysql/xdevapi');
const validation = { schema: { type: 'object', properties: { name: { type: 'string' } } }, level: mysqlx

mysqlx.getSession('mysqlx://localhost:33060')
    .then(sesion => {
        return session.getSchema('mySchema').createCollection('myCollection', { validation })
    });
```

The same `level` property logic applies to `modifyCollection()`. This example shows how to enable a JSON schema on an existing collection (or to update it, if it already exists) using the `STRICT` validation level:

```
const mysqlx = require('@mysql/xdevapi');
const validation = { schema: { type: 'object', properties: { name: { type: 'string' } } }, level: mysqlx

mysqlx.getSession('mysqlx://localhost:33060')
    .then(sesion => {
        return session.getSchema('mySchema').modifyCollection('myCollection', { validation })
    });
```

(WL #13333)

Bugs Fixed

- Row values for columns of type `BIGINT` were not correctly decoded by Connector/Node.js. We fix this problem by upgrading the included `google-protobuf` library to version 3.11.4. (Bug #27570685)

Changes in MySQL Connector/Node.js 8.0.20 (2020-04-27, General Availability)

Functionality Added or Changed

- Added two new connection options, evaluated during the TLS handshake, which restrict negotiated TLS protocols and ciphers, along with those set on the server that can further restrict the final choices. `tls-versions` determines permitted TLS protocol versions; `tls-ciphersuites` determines permitted cipher suites. These definitions are comma-separated, and accepted by `getSession()` and `getClient()`.

  `tls-versions`: Accepts one or more of the following: TLSv1, TLSv1.1, TLSv1.2, and TLSv1.3. Other values generate an error.

  `tls-ciphersuites`: Accepts IANA cipher suite names, as listed at TLS Cipher Suites. Unsupported or unknown values are ignored.

  The following examples demonstrate both plain JavaScript and JSON configuration object formats:

```
# tls versions:
mysqlx.getSession('mysqlx://root@localhost?tls-versions=[TLSv1,TLSv1.1,TLSv1.2,TLSv1.3]')

mysqlx.getSession({ user: 'root', tls: { versions: ['TLSv1', 'TLSv1.1', 'TLSv1.2', 'TLSv1.3'] } })

# tls cipher suites
mysqlx.getSession('mysqlx://root@localhost?tls-ciphersuites=[DHE-RSA-AES128-GCM-SHA256,DHE-RSA-AES256-SHA256

mysqlx.getSession({ user: 'root', tls: { ciphersuites: ['DHE-RSA-AES128-GCM-SHA256', 'DHE-RSA-AES256-SHA256'
```

(WL #12738)

- For X DevAPI applications, when creating a new connection, if the connection specification contains several target hosts with no assigned priority, the behavior of the failover logic now is the same as if all those target hosts have the same priority. That is, the next candidate for making a connection is chosen randomly from the remaining available hosts. If two hosts have the same priority then one is chosen at random. (WL #13546)

Changes in MySQL Connector/Node.js 8.0.19 (2020-01-13, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Added DNS SRV support.

  Using the `mysqlx+srv` scheme plus extension in a connection string (or enabling the `resolveSrv` option in a connection configuration object) allows automatic resolution of any SRV record available from a target DNS server or service discovery endpoint. (WL #13365)

Bugs Fixed

- Improved CRUD API error messages. (Bug #30423556)
- The `getAffectedItemsCount()` method did not work with result sets in Connector/Node.js 8.0.18. (Bug #30401962)
- The `Collection.existsInDatabase()` method did not function. (Bug #30401432)

Changes in MySQL Connector/Node.js 8.0.18 (2019-10-14, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Implemented the X DevAPI cursor model, including methods such as `fetchOne()`, `fetchAll()`, `getColumns()`, `hasData()`, and `nextResult()`. For more information, see Working with Result Sets, in the X DevAPI documentation.

  Previously, handling result set data or metadata required specific callback functions when calling `execute()`. With the addition of this interface, the Connector automatically switches to the pull-based cursor model if these callback functions are not provided. (WL #11840)

- Improved the performance of `Collection.getOne()` by parsing the underlying lookup expression only once, and using serverside prepared statements for subsequent calls to this method. (WL #13358)

- Added support for generating test coverage reports by running the `npm run coverage` or similar commands; see the bundled `CONTRIBUTING.md` file for requirements and other information. This was added to help users make contributions. (WL #13272)
- Added `linter` check support to help enforce coding style and convention rules for new contributions by running `npm run linter`; see the bundled `CONTRIBUTING.md` file for more information. (WL #13284)

Bugs Fixed

- Added support for assigning Node.js `Buffer` values to expressions and query placeholders. (Bug #30163003, Bug #96480)
- MySQL column values of binary types such as `BLOB`, `BINARY`, and `VARBINARY` can now be converted to instances of Node.js `Buffer`. (Bug #30162858, Bug #96478)
- Inserting a raw Node.js `Buffer` value into a MySQL `BLOB` column resulted in an error due to improper setting of the `content_type`; now the X Plugin handles this as a raw byte string. (Bug #30158425)
- The padding characters used for fixed-length columns now map to the collation code provided by the column metadata; previously it was based on the JavaScript native type of the values. (Bug #30030159)

Changes in MySQL Connector/Node.js 8.0.17 (2019-07-22, General Availability)

Functionality Added or Changed

- Document fields containing arrays can now be indexed by setting `array` to true in an index `fields` definition. (WL #12295)
- Added support for the `OVERLAPS` and `NOT OVERLAPS` operators; these are equivalent to the MySQL `JSON_OVERLAPS()` function.

  The syntax employed for these binary operators take the form *`expression`* `[NOT] OVERLAPS` *`expression`* where each *`expression`* returns a JSON array or object. For example, `[1, 2, 3]` overlaps `$.list`. (WL #12745)
- Added support for the `utf8mb4_0900_bin` collation added in MySQL Server 8.0.17. (WL #13138)
- The bundled `README.md` file was split and reformatted with some content moved into the new `README.txt` and `CONTRIBUTING.md` files. (WL #13097)

Bugs Fixed

- The MySQL `CAST` function did not work as a valid lookup expression. (Bug #29807792)
- Added backtick (`` ` ``) support for table column identifiers in valid expressions. (Bug #29789818)
- The `DIV` binary and `NOT` unary operators are now supported; these are case-insensitive. (Bug #29771833, Bug #29771027)
- *Collection.find()* now supports the JavaScript `Date` type. (Bug #29766014)
- The `collection.dropIndex()` method now silently fails if the index does not exist, as expected; previously, it raised a `Can't DROP` error. (Bug #29765589)

- In some cases, `Column.getCollationName()` returned the incorrect name. (Bug #29704185)

Changes in MySQL Connector/Node.js 8.0.16 (2019-04-25, General Availability)

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

X DevAPI Notes

- Connector/Node.js now supports connection attributes as key-value pairs that application programs can pass to the server. Connector/Node.js defines a default set of attributes, which can be disabled or enabled. In addition to these default attributes, applications can also provide their own set of custom attributes, as listed here:

  - The application can specify connection attributes as a `connection-attributes` parameter in a connection string, or by using the `connectionAttributes` property using either a plain JavaScript object or JSON notation to specify the connection configuration options.

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

    The `connectionAttributes` property allows passing user-defined attributes to the application using either a plain JavaScript object or JSON notation to specify the connection configuration options. In such cases, define each attribute in a nested object under `connectionAttributes`, where the property names match the attribute names and the property values match the attribute values. Unlike `connection-attributes`, and while using plain JavaScript objects or JSON notation, if the `connectionAttributes` object contains duplicate keys then no error is thrown and the last value specified for a duplicate object key is chosen as the effective attribute value.

    Examples:

    Not sending the default client-defined attributes:

    ```
    mysqlx.getSession('{ "user": "root", "connectionAttributes": false }')

    mysqlx.getSession('mysqlx://root@localhost?connection-attributes=false')

    mysqlx.getSession({ user: 'root', connectionAttributes: { foo: 'bar', baz: 'qux', quux: '' } })
    mysqlx.getSession('mysqlx://root@localhost?connection-attributes=[foo=bar,baz=qux,quux]')
    ```

  Attribute names defined by applications cannot begin with _; such names are reserved for internal attributes.

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

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

Functionality Added or Changed

- Optimized the reuse of existing connections through `client.getSession()` by reauthenticating only if required. (WL #12484)

- For X DevAPI, performance for statements that are executed repeatedly is improved by using serverside prepared statements for the second and any subsequent executions. This happens internally; applications need take no action and API behavior should be the same as previously. For statements that change, repreparation occurs as needed. Providing different data values, or different `offset()` or `limit()` values, does not count as a change for this purpose. Instead, the new values are passed to a new invocation of the previously prepared statement. (WL #12482)

Bugs Fixed

- Idle pooled connections to MySQL Server were not reused; new connections had to be recreated instead. (Bug #29436892)
- Executing `client.close()` did not close all associated connections in the connection pool. (Bug #29428477)
- `connectTimeout` instead of `maxIdleTime` determined whether idle connections in the connection pool were reused rather than creating new connections. (Bug #29427271)
- Released connections from the connection pool were not reset and reused; new connections were made instead. (Bug #29392088)
- Date values in documents were converted to empty objects when inserted into a collection. (Bug #29179767, Bug #93839)
- A `queueTimeout` value other than 0 (infinite) prevented the acquisition of old released connections from the connection pool. (Bug #29179372, Bug #93841)

Changes in MySQL Connector/Node.js 8.0.15 (2019-02-01, General Availability)

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

Changes in MySQL Connector/Node.js 8.0.14 (2019-01-21, General Availability)

Functionality Added or Changed

- Removed deprecation notices from the `count()` methods. (WL #12487)
- Setting the default schema using the connection now sets the default schema on the server; that is, subsequent queries executed using `session.sql()` no longer need to specify the schema. (WL #12627)

Bugs Fixed

- Setting the default schema with a connection URI using a schema name that contained special characters requiring URL encoding resulted in the encoded name being used instead of the original one (for example, `%25%26%5E*%5E_` instead of `%&^*^_`). (Bug #28990682)
- The client hung when the CA specified by `sslOption` differed from the certificate authority used to sign the server certificate, or the CA had been revoked. Now in such cases, we once again throw the appropriate error. (Bug #28977649)
- Attempting to use synonyms for `false` such as `0`, `false`, `null`, and `undefined` raised errors when updating or inserting documents in a collection or rows in a table. In addition, boolean values are now

converted to numeric values (true to 1, false to 0) while `null` and `undefined` are converted to MySQL `NULL`. (Bug #28970727, Bug #93315)

- `Collection.existsInDatabase()` always returned true if any other collection existed in the database. (Bug #28745240)
- Setting the default schema from the connection string created the schema if it did not exist. Now, an `Unknown database` error is thrown in such cases instead. (WL #12627)
- It was possible for an empty short-form notice to throw `Error: Invalid type for Mysqlx.Notice.Frame` on the client. Now these are ignored, as with long-form empty notices. (WL #12486)

Changes in MySQL Connector/Node.js 8.0.13 (2018-10-22, General Availability)

Functionality Added or Changed

- To complement the existing asynchronous `mysqlx.getSession(conn_str)` method, a new synchronous `mysqlx.getClient(conn_str, options)` method is added; the new method creates a connection pool handler that provides an asynchronous `getSession()` method to create and retrieve connections from the pool. Collection pooling options are listed here:
  - `enabled`: Enables or disables connection pooling; a boolean which defaults to true.
  - `maxSize`: Maximum number of connections available in the pool; a positive integer which defaults to 25.
  - `maxIdleTime`: Maximum number of milliseconds a connection can be idle in the queue before being closed; a nonnegative integer which defaults to 0 (infinite).
  - `queueTimeout`: Maximum number of milliseconds a request waits for a connection to become available; a nonnegative integer which defaults to 0 (infinite).

    This differs from the `connectTimeout` used when connection pooling is not employed. When a pooling is used, there may already be connections in the pool; `queueTimeout` controls how long to wait for a connection from the pool.

  Example:

```
var mysqlx = require('@mysql/xdevapi')
var client = mysqlx.getClient(
  { user: 'root', host: 'localhost', port: 33060 },
  { pooling: { enabled: true, maxIdleTime: 5000, maxSize: 25, queueTimeout: 20000 } }
);

client.getSession()
  .then(session => {
    console.log(session.inspect())
    return session.close() // the connection becomes idle in the client pool
  })
  .then(() => {
    return client.getSession()
  })
  .then(session => {
    console.log(session.inspect())
```

```
    return client.close() // closes all connections and destroys the pool
  })
```

Closing a session attached to the pool makes the connection available in the pool for subsequent `getSession()` calls, while closing (that is, destroying) the pool effectively closes all server connections. (WL #11831)

- Added a connection timeout query parameter to both the `mysqlx.getClient()` (pooling) and `mysqlx.getSession()` (no pooling) interfaces. This parameter determines the length of time in milliseconds that the client waits for a MySQL server to become available from among the given network addresses. The default value is 10000 (10 seconds). Setting it to 0 disables the timeout so that the client waits until the underlying socket times out. The socket timeout is platform-dependent

  As with other Connector/Node.js parameters, kebab case is used for URI definitions (`connection-timeout`) and camel case for JavaScript configuration objects (`connectionTimeout`).

  Example 1:

```
const mysqlx = require('@mysql/xdevapi');
var client = mysqlx.getClient('root@localhost?connect-timeout=5000')
client.getSession()
    .catch(err => {
        console.log(err.message) // "Connection attempt to the server was aborted. Timeout of 5000 ms wa
    })
```

  Example 2:

```
const mysqlx = require('@mysql/xdevapi');
var client = mysqlx.getClient('mysqlx://root:passwd@[localhost:33060, 127.0.0.1:33060]?connect-timeout=5
client.getSession()
    .catch(err => {
        // connection could not be established after 10 seconds (5 seconds for each server)
        console.log(err.message); // All server connection attempts were aborted. Timeout of 5000 ms was
    });
```

  When using multiple hosts, the `connect-timeout` value applies separately to each host. (WL #12197)

Bugs Fixed

- Improved the handling of X Protocol global notices by properly logging and then ignoring nonfatal errors, and by making the connection unusable for subsequent operations in the case of a fatal error. (Bug #28653781)
- Calling `getCollationName()` on columns of nontext types such as `INT` threw `TypeError: Cannot read property 'collation' of undefined`. (Bug #28608923)
- The `fields()` method did not function with valid expressions generated by the `expr()` method. (Bug #28409639)
- The returned `Session.inspect()` object now includes the `user` property in addition to the `dbUser` property; each property contains the same value. (Bug #28362115)

Changes in MySQL Connector/Node.js 8.0.12 (2018-07-27, General Availability)

- Deprecation and Removal Notes
- Bugs Fixed

Deprecation and Removal Notes

- The following API changes have been made in order to comply better with the X DevAPI:
  - `Collection`:

    Deprecated: `count()`.

    Changed: `getSchema()` now returns a `Schema` instance instead of the schema name.
  - `CollectionModify`:

    Deprecated: Second parameter of `limit(x, y)`; `arrayDelete()`.
  - `CollectionFind`:

    Deprecated: Second parameter of `limit(x, y)`.

    Added: `limit(x).offset(y)`.
  - `CollectionRemove`:

    Deprecated: Second parameter of `limit(x, y)`.
  - `Table`:

    Deprecated: `count()` and `insert(Document)` API.

    Updated: `getSchema()` now returns a `Schema` instance instead of the schema name.

    Removed: `as()`.
  - `TableSelect`:

    Deprecated: Second parameter of `limit(x, y)`.

    Added: `limit(x).offset(y)`.
  - `TableDelete`:

    Deprecated: Second parameter of `limit(x, y)`; `delete(x)` in favor of `where(x)`.
  - `TableUpdate`:

    Deprecated: Second parameter of `limit(x, y)`; `update(x)` in favor of `where(x)`.
  - `SqlExecute`:

    Deprecated: `sqlExecute()` in favor of `sql()`.

    Added: `bind()`.
  - `Column`:

    Added: `isNumberSigned()`, `getCollationName()`, `getCharacterSetName()`, and `isPadded()`.

    (WL #11950)

Bugs Fixed

- The promise returned by the `session.sql().execute()` method resolved to a plain JavaScript object rather than a proper `Result` instance. This meant that methods such as `getAffectedItemsCount()` and `getWarnings()` lacked access to the API. (Bug #28146988)
- Retrieving rows containing `NULL` columns raised an unexpected assertion. (Bug #27978594)
- The `session.close()` method is now asynchronous, returning a JavaScript `Promise`, when before it simply returned immediately. (Bug #27893001)
- The right-padding mechanism was improved. (Bug #27839295, Bug #28275595, Bug #91503)
- While calling `getSession()` without arguments raises an `Invalid parameter` error, passing in {} raised `Cannot read property 'length' of undefined`. Now `{}` is supported, and `getSession()` defaults to using an empty string as the user name. (Bug #27730748)
- Improved performance for expression parsing and protocol message encoding. (WL #11839, WL #11830)

Changes in MySQL Connector/Node.js 8.0.11 (2018-04-19, General Availability)

Functionality Added or Changed

- The `protobuf.js` library was replaced with the official `google-protobuf npm` package. (WL #11402)
- Added `NOWAIT` and `SKIP_LOCKED` support to the `lockShared()` and `lockExclusive()` methods. Example: `lockShared( mysqlx.LockContention.SKIP_LOCKED )`. (WL #11275)
- Added the X DevAPI SHA256_MEMORY authentication mechanism. (WL #11633)
- Connector/Node.js now supports auto-generated document ID values generated by MySQL 8.0.11 and later server. (WL #11441)

Bugs Fixed

- Running a select query against a table containing `BIGINT` values and using those values as filtering criteria could fail to function. This was because those values were converted to JavaScript numbers when encoding the protobuf message, and lost precision since the maximum safe integer in JavaScript is 2^53 - 1. (Bug #27570761)
- Row values from columns using the `FLOAT` type were not rounded according to the maximum number of displayable decimal digits defined by the schema. For example, a column with type `FLOAT(3,2)` containing a value of 1.23456789 would display it as 1.2300000190734863 instead of the expected 1.23. (Bug #27570541)
- Row values from columns using the `BIT` data type were decoded as their signed integer counterparts rather than as unsigned values. For example, `b'111'` was decoded as -4 instead of 7. (Bug #27570462)
- Row values for columns of any type of unsigned integer (`TINYINT`, `SMALLINT`, `MEDIUMINT`, `INT`, or `BIGINT`) were interpreted by the Connector as their signed integer value counterparts. (Bug #27570342)
- The `sort()` method was added to the following operations: `CollectionFind`, `CollectionRemove`, and `CollectionModify`. (Bug #27429922)

- While adding a document, the expression parser rejected valid escaped string literals that constituted properties of the document, and throwing unexpected errors. (Bug #27429852)
- Messages split into multiple fragments (either because they exceeded the MTU or the maximum size of V8 buffers) were improperly reconstructed and could not be decoded. This behavior would throw an error similar to `Uncaught SyntaxError: Unexpected token`. (Bug #27429429)
- Several methods that previously returned plain JavaScript objects now return iterable arrays. `Schema.getCollections()` now returns an array of `Collection` instances, `Schema.getTables()` now returns an array of `Table` instances, and `Session.getSchemas()` now returns an array of `Schema` instances. (Bug #27294362, Bug #27221114)
- The expression parser was executed every time a document was added but now requires an explicit call to `mysqlx.expr()`. For example, before the change `collection.add({ name: 'foo' })` parsed the `name` property; to accomplish the same task following the change, use `collection.add({ name: mysqlx.expr('"foo"') })`. (Bug #27177864)

Changes in MySQL Connector/Node.js 8.0.10 (Skipped version number)

There are no release notes for this skipped version number.

Changes in MySQL Connector/Node.js 8.0.9 (2018-01-30, Release Candidate)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- Improved the UUID generation algorithm to align with RFC 4122, reducing the chance of generating duplicated values in a relatively short time. (Bug #26120588)
- *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/Node.js:
  - API components that support session configurations, such as the `SessionConfig` and `SessionConfigManager` classes.
  - The `mysqlx.config` namespace and all methods of the namespace, including `save()`, `get()`, `list()`, and `delete()`.
  - The `Schema` class methods `createTable()`, `foreignKey()`, `dropTable()`, `createView()`, `dropView()`, and `alterView()`.

  (WL #11323, WL #11413)
- Added the `Session` methods listed here:
  - `Session.setSavePoint`: accepts a name or generates one of the form connector-nodejs-{uuid}, and returns a Promise.
  - `Session.releaseSavePoint`: releases a specific savepoint.
  - `Session.rollbackTo`: rollbacks to a specified savepoint.

  (WL #11223)

- The `createIndex()` method was added to the `Collection` API. (WL #11152)

Bugs Fixed

- The expression parser used by the CRUD API was replaced with a new implementation written in pure JavaScript, which fixes several grammar related issues. (Bug #26729768, Bug #26636956, Bug #25036336, Bug #23148246, WL #11156)
- The `CollectionFind.fields()` method now supports flexible parameters to align with standard X DevAPI conventions. (Bug #22084545)

Changes in MySQL Connector/Node.js 8.0.8 (2017-09-28, Development Milestone)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The following `Collection` methods were added: `replaceOne()`, `addOrReplaceOne()`, `getOne()`, and `removeOne()`. For details, see Tutorial: Working with Documents. (WL #10903)
- `lockExclusive()` and `lockShared()` methods have been added in this release to the `CollectionFind` and `TableSelect` classes to provide row locking support. For additional information, see Tutorial: Row Locking. (WL #10902)
- Connector/Node.js now provides extended authentication support, including SHA-256. For additional information, see Tutorial: Secure Sessions. (WL #10757)
- Added containment operator support for objects and arrays. This allows additional types of expressions such as `IN [x, y, z]` and `IN { "x": "foo", "y": "bar" }`, as well as expressions referencing field names that map to arrays and objects, such as `someArray IN $.field` and `someObject IN $.field`. (WL #10901)

Bugs Fixed

- Added support for parenthetical `IN` syntax, such as `IN (x, y, z, ...)`, as defined by the X DevAPI. (Bug #26666817)

Changes in MySQL Connector/Node.js 8.0.7 (2017-07-10, Development Milestone)

MySQL Connectors and other MySQL client tools and applications now synchronize the first digit of their version number with the (highest) MySQL server version they support. This change makes it easy and intuitive to decide which client version to use for which server version.

Connector/Node.js 8.0.7 is the first release to use the new numbering. It is the successor to Connector/Node.js 1.0.6.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- The methods `Collection.modify()` and `Collection.remove()` no longer accept an empty search condition, such as `NULL` or an empty string. (WL #10765)

- A number of changes have been implemented for methods which drop objects. These changes are listed here:

  - Drop methods are now made available at the same level as the corresponding create methods. For example, the `dropCollection()` and `dropTable()` methods have been removed from the `XSession` class (which has now been consolidated into the `Session` class) and moved under the `Schema` class; under the same principle, the `drop()` method has been removed from the `Collection` and `Table` classes.

  - Drop methods now succeed even if the objects to be dropped do not exist.

  - `dropView()` is now asynchronous and behaves exactly like `dropTable()` and `dropCollection()` by implicitly executing the drop operation and returning a promise that holds the result.

  (WL #10532)

- A configuration handler interface, `mysqlx.config`, has been added for managing persisted session configurations. See MySQL Connector/Node.js with X DevAPI for details. (WL #10586)

- The following changes have been made with regarding encrypted connections to MySQL servers:

  - Connections are now encrypted by default.

  - The connection option `ssl-enable` has been replaced by the `ssl-mode` option, which has `DISABLED`, `REQUIRED` (the default), and `VERIFY_CA` as its permitted values.

  - Using the `ssl-crl` option requires the use of the `ssl-ca` and that `ssl-mode=VERIFY_CA`; this is due to an internal requirement of the Node.js core platform.

  (WL #10415)

- `BaseSession`, `NodeSession`, and `XSession` have been consolidated into a single `Session` class. The following related changes were also made:

  - The `mysqlx.getNodeSession()` method is renamed to `getSession` and returns a `Session` object.

  - `DatabaseObject.getSession()` now returns a `Session` object.

  (WL #10413)

- A new clientside failover feature has been implemented such that, when creating a new connection, multiple hosts can now be specified in the connection string; Connector/Node.js tries each host until a successful connection is established or until all hosts have been tried. See Tutorial: Getting Started for details (WL #10346)

- Connector/Node.js now supports connecting to a local server using Unix sockets. See Tutorial: Getting Started for details. (WL #10345)

- The format of the document ID value generated when adding a document to a collection has changed. It is still a string of 32 hexadecimal digits based on a UUID, but the order of digits has been changed to match the requirement of a stable ID prefix. (WL #10683)

Bugs Fixed

- It was not possible to create a new session for a user with a SHA256 password using the PLAIN authentication mechanism. (Bug #26117627)

- The handling of large JSON arrays was problematic, and caused an exception to be thrown. (Bug #26084604)
- Attempting to use `bind` when removing a document from a collection threw an exception. (Bug #26029551, WL #10687)
- `Table.update()` did not require a `SearchConditionStr` parameter; not including this parameter could result in deleting all rows from a given table. A clientside exception is now thrown if `SearchConditionStr` is empty or undefined. (Bug #25993174, WL #10687)
- `Table.delete()` did not require a `SearchConditionStr` parameter; not including this parameter could result in deleting all rows from a given table. A clientside exception is now thrown if `SearchConditionStr` is empty or undefined. (Bug #25992969, WL #10687)