MySQL Connector/J 8.0 Release Notes

Abstract

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

For additional Connector/J documentation, see MySQL Connector/J 8.0 Developer Guide.

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: 2022-05-06 (revision: 24655)

Table of Contents

Preface and Legal Notices .......................................................... 1
Changes in MySQL Connector/J 8.0.29 (2022-04-26, General Availability) ................................................................. 3
Changes in MySQL Connector/J 8.0.28 (2022-01-18, General Availability) ................................................................. 4
Changes in MySQL Connector/J 8.0.27 (2021-10-19, General Availability) ................................................................. 6
Changes in MySQL Connector/J 8.0.26 (2021-07-20, General Availability) ................................................................. 7
Changes in MySQL Connector/J 8.0.25 (2021-05-11, General Availability) ................................................................. 9
Changes in MySQL Connector/J 8.0.24 (2021-04-20, General Availability) ................................................................. 9
Changes in MySQL Connector/J 8.0.23 (2021-01-18, General Availability) ................................................................. 11
Changes in MySQL Connector/J 8.0.22 (2020-10-19, General Availability) ................................................................. 13
Changes in MySQL Connector/J 8.0.21 (2020-07-13, General Availability) ................................................................. 15
Changes in MySQL Connector/J 8.0.20 (2020-04-27, General Availability) ................................................................. 16
Changes in MySQL Connector/J 8.0.19 (2020-01-13, General Availability) ................................................................. 17
Changes in MySQL Connector/J 8.0.18 (2019-10-14, General Availability) ................................................................. 19
Changes in MySQL Connector/J 8.0.17 (2019-07-22, General Availability) ................................................................. 19
Changes in MySQL Connector/J 8.0.16 (2019-04-25, General Availability) ................................................................. 21
Changes in MySQL Connector/J 8.0.15 (2019-02-01, General Availability) ................................................................. 23
Changes in MySQL Connector/J 8.0.14 (2019-01-21, General Availability) ................................................................. 23
Changes in MySQL Connector/J 8.0.13 (2018-10-22, General Availability) ................................................................. 25
Changes in MySQL Connector/J 8.0.12 (2018-07-27, General Availability) ................................................................. 27
Changes in MySQL Connector/J 8.0.11 (2018-04-19, General Availability) ................................................................. 29
Changes in MySQL Connector/J 8.0.10 (Skipped version number) ................................................................. 30
Changes in MySQL Connector/J 8.0.9 (2018-01-30, Release Candidate) ................................................................. 30
Changes in MySQL Connector/J 8.0.8 (2017-09-28, Development Milestone) ................................................................. 32
Changes in MySQL Connector/J 8.0.7 (2017-07-10, Development Milestone) ................................................................. 34
Index ................................................................. 36

Preface and Legal Notices

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

Legal Notices

Copyright © 1997, 2022, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted
in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed
MySQL Connector/J 8.0 Release Notes

copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

For information about Oracle’s commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

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

Changes in MySQL Connector/J 8.0.29 (2022-04-26, General Availability)

Version 8.0.29 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0 and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• Historically, MySQL has used utf8 as an alias for utf8mb3. Since release 8.0.29, utf8mb3 has become a recognized (though deprecated) character set on its own for MySQL Server. Therefore, Connector/J has added utf8mb3 to its character set mapping, and users are encouraged to update to Connector/J 8.0.29 to avoid potential issues when working with MySQL Server 8.0.29 or later. (Bug #33850155)

• A new connection property socksProxyRemoteDns has been added, which, when set to true, makes the SocksProxySocketFactory execute its own connect() implementation that passes the unresolved InetSocketAddress of a MySQL Server host to the created proxy socket, instead of having the address resolved locally. (Bug #77924, Bug #25710160)

• The code for prepared statements has been refactored to make the code simpler and the logic for binding more consistent between ServerPreparedStatement and ClientPreparedStatement.


Bugs Fixed

• X DevAPI: If the connection property xdevapi.ssl-mode was set to DISABLED (or xdevapi.ssl-mode was not set, but the value was picked up from the sslMode setting), specifying some of the security properties caused Connector/J to throw an error. With this fix, even when encryption is turned off and irrelevant security properties are set, Connector/J does not throw an error.

• DatabaseMetaData.getDefaultTransactionIsolation() returned a wrong value. It now returns the correct value of Connection.TRANSACTION_REPEATABLE_READ. (Bug #33723611)
• Statement executions failed for replication connections when `useCursorFetch` was true and `defaultFetchSize` was greater than 0. (Bug #25701740)

• Prepared statements were parsed incorrectly sometimes when they contained comments marked by `/*` and `*/`. (Bug #21978230)

• A connection did not maintain the correct autocommit state when it was used in a pool with `useLocalSessionState=true`. (Bug #106435, Bug #33850099)

  References: This issue is a regression of: Bug #33054827.

• A spelling error in the error message for the buffer length being less than the expected payload length has been corrected. Thanks to Jianjian Song for contributing the fix. (Bug #106397, Bug #33893591)

• When using client-side prepared statements, if the VALUES clause came after the ON DUPLICATE KEY UPDATE clause or it came at the end of the statement, a StringIndexOutOfBoundsException was thrown. This patch refactors the query parser to fix the problem behind the issue, and also to improve the parser's performance. (Bug #106240, Bug #33781440)

• An unnecessary boxing has been removed from `findColumn()` in the ResultSetImpl class. Thanks to Pei Pei Ning for contributing this improvement. (Bug #106171, Bug #33757217)

• When decoding decimals, the constructor used for creating the BigDecimal object has been changed from `BigDecimal(String)` to `BigDecimal(char[])` in order to save memory. Thanks to Chen Yi for contributing to this improvement. (Bug #106065, Bug #33726184)

• When inserting BigDecimal values into a database using rewritable server-side prepared statements with cursor-based fetching, the values suffered precision loss. (Bug #105915, Bug #33678490)

• When the Connector/J logger level was at TRACE, a null bind value for a PreparedStatement resulted in a NullPointerException when the logger tried to read the value. This patch added a null check to avoid the exception to be thrown under the situation. (Bug #104349, Bug #33563548)

• When the connection property `rewriteBatchedStatements` was set to true, inserting a BLOB using a prepared statement and `executeBatch()` resulted in a NullPointerException. (Bug #85317, Bug #25672958)

• `ResultSetMetaData` and `DatabaseMetaData` returned `Types.DATE` for a YEAR table column even when `yearIsDateType=false`. With this fix, `Types.SMALLINT` was returned correctly in the situation. (Bug #82084, Bug #23743938)

• A PreparedStatement could not be rewritten for batch insert if any table column involved contained "select" as a substring in the column name. (Bug #81468, Bug #23312764)

• When using server-side prepared statements and the connection property `profileSQL` was set to true, setting a parameter of type LONGTEXT using a StringReader() resulted in a javax.io.NotSerializableException. (Bug #62006, Bug #16714956)

• Data truncation occurred for INOUT type parameters of data type BIT(1) for stored procedures. (Bug #38954, Bug #11749415)

Changes in MySQL Connector/J 8.0.28 (2022-01-18, General Availability)

Version 8.0.28 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0 and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.
Functionality Added or Changed

• **X DevAPI:** A document \_id check has been added to the `Collection.addOrReplaceOne(id, doc)` and `Collection.replaceOne(id, doc)` methods: If the id supplied as an argument for the functions does not match the \_id in doc (the supplied JSON document), the operation fails. (Bug #32770013)

• Connector/J now supports multifactor authentication for connections to MySQL Servers. Three new connection properties, `password1`, `password2`, and `password3`, have been added for that purpose. See Connecting Using Multifactor Authentication for details.

• The TLSv1 and TLSv1.1 connection protocols were deprecated in Connector/J 8.0.26 and are now removed in this release. The removed values are considered invalid for use with connection options and session settings. Connections can be made using the more-secure TLSv1.2 and TLSv1.3 protocols. Using TLSv1.3 requires that the server be compiled with OpenSSL 1.1.1 or higher and Connector/J be run with a JVM that supports TLSv1.3 (for example, Oracle Java 8u261 and above).

Also, the following connection properties have been renamed:

• `enabledTLSProtocols` renamed to `tlsVersions`; the original name remains as an alias.

• `enabledSSLCipherSuites` renamed to `tlsCiphersuites`; the original name remains as an alias.

Bugs Fixed

• The `README` file in the Connector/J distribution contained broken links in the Additional Resources section. (Bug #33507321)

• Storing a `java.time.LocalDate` object onto the server as a `DATE` value using a batched `PreparedStatement` failed with the complaint that `java.time.LocalDate` could not be cast to `java.sql.Date`. With this fix, the object is encoded correctly into a `DATE` value. (Bug #33468860, Bug #105211)

• `ResultSet` navigation methods like `absolute()`, `relative()`, `first()`, `last()`, and so on returned a `NullPointerException` when they were applied to a non-navigable `ResultSet`. This was because the methods did not check if the `ResultSet` really had any rows before trying to navigate it. This patch makes all navigation methods perform the check and throw a proper `SQLException` when the `ResultSet` has no rows. (Bug #33461744)

• Running a `PreparedStatement` in the form of `INSERT INTO ... VALUE ... ON DUPLICATE KEY UPDATE ... VALUES(...)` resulted in a `StringIndexOutOfBoundsException`. It was because Connector/J did not recognize `VALUE` as an alias for `VALUES` in an `INSERT` statement, and mistook the `VALUES()` function as a `VALUES` clause for the statement. With this fix, statements like this are parsed properly. (Bug #33425867, Bug #84365)

• When the connection session's character set was SJIS, `BLOB` values could not be handled properly by client-side prepared statements. (Bug #33350185)

• Some Java frameworks prevent changes of the `autocommit` value on the MySQL Server. When an application tried to change `autocommit`, the value was changed for the connection `Session` on the client side, but it failed to change on the server, so the settings became out of sync. With this fix, Connector/J resets `autocommit` to its original value for the `Session` under the situation. Thanks to Tingyu Wei for contributing to the fix. (Bug #33054827, Bug #104067)
• `getWarnings()` always sent a `SHOW WARNINGS` query to the server. With this fix, the query is sent only if there really are warnings to be shown, in order to save resources. (Bug #32089018, Bug #101389)

• After calling `Statement.setQueryTimeout()`, when a query timeout was reached, a connection to the server was established to terminate the query, but the connection remained open afterward. With this fix, the new connection is closed after the query termination. (Bug #31189960, Bug #99260)

• A new session created for executing `Statement.cancel()` remained open after the `Statement` had been cancelled. With this fix, the session is closed after the `Statement` cancellation. (Bug #30355150, Bug #96900)

• Using the `setString()` method on an `SQLXML` object resulted in a `NullPointerException`. (Bug #25656020, Bug #85223)

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

Version 8.0.27 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• The `ResultSet.getBoolean()` method now returns `true`/`false` for `VARCHAR` values of “T”/“F”. (Bug #33185116)

• In line with good XML practices, the `getSource()` method of `MysqlSQLXML` no longer supports external DTD, external general entities, and external general parameters in XML sources. (Bug #33094420)

• To allow compatibility with other implementations of Java, the `TLS_` suffix has been removed from the names of cipher suites in the `TlsSettings.properties` file, and both the `TLS_` and `SSL_` prefixes are added to the cipher suites names in the `ALLOWED_CIPHERS` list of the `ExportControlled` object. (Bug #31117686)

• Synchronization in the `MultiHostConnectionProxy#invoke()` method forced connection pools to wait for statements to finish executing before new or free connections could be provided, creating performance issues in some cases. It was because invocation by multiple threads of the `equals()` method was blocked due to the synchronization. This patch restructures the synchronization to avoid locking on ubiquitous methods like `equals()` and `toString()`. (Bug #28725534)

• Connector/J now supports authentication of MySQL users created using the `authenticationoci` plugin. There is a new connection property, `ociConfigFile`, for specifying the location of the required OCI SDK and Client configuration file when it is not found at the default location of `~/.oci/config` for Unix-like systems and `%HOMEDRIVE%%HOMEPATH%\.oci\config` for Windows systems (where `~` and `%HOMEPATH%` represent the home directory of the user who runs the application).

• The `Connector/J test suite` has been reworked to run against a single MySQL Server compiled with either yaSSL or OpenSSL; the Ant build properties `com.mysql.cj.testsuite.url.openssl` and `com.mysql.cj.testsuite.mysqlx.url.openssl` are now ignored.
Bugs Fixed

- The method `DatabaseMetaData.getImportedKeys()` sometimes returned multiple rows for the same foreign key. (Bug #33237255, Bug #104641)

- When `cacheServerConfiguration` was enabled and subsequent connections were configured with different character set options, inserting strings into a table resulted in an `Incorrect string value` error from the server. It was because the cached character set information reused by Connector/J was mutable, and this patch fixed the issue. (Bug #29807572)

- When `ResultSet.getObject(columnIndex, java.util.Date.class)` was expected to return null, it caused Connector/J to throw a `NullPointerException` instead. (Bug #104559, Bug #33232419)

- After a `Calendar` was passed as a parameter to a `ClientPreparedStatement` set method, the `SimpleDateFormat` object used to render the parameter value actually modified the `Calendar`, so that using the same `Calendar` object with several set methods resulted sometimes in the wrong date or time being set. With this fix, a clone of the `Calendar` is passed to the `SimpleDateFormat` object when it is created, to avoid changing the original `Calendar`. Thanks to Björn Michael for his contribution to the fix. (Bug #104170, Bug #33064455)

- When using cursor-based fetching (`useCursorFetch=true`), `SHOW` and `EXPLAIN` statements failed with an `SQLException`. (Bug #103878, Bug #32954449)

- `setQueryTimeout()` failed to set a timeout for queries when a cursor was used for fetching rows (`useCursorFetch=true`). Thanks to Hong Wang for contributing the fix. (Bug #103796, Bug #32922715)

- When the connection property `createDatabaseIfNotExist` was set to true, a non-existing database could not be created when its name included a hyphen. Thanks to Lukasz Sanek for contributing the fix. (Bug #95564, Bug #29894324)

- When `Statement.executeQuery()` was called, Connector/J’s check for whether a statement would return results was inadequate, so that sometimes appropriate statements were rejected (for examples, `SELECT` statements starting with a `WITH` clause, statements preceded by consecutive comments, and so on) and, at other times, inappropriate statements were executed (for example, `DO` statements), resulting in various kinds of errors. With this fix, Connector/J performs more accurate checks by looking at the statement keywords and the context, as well as handling properly different corner cases. In this new mechanism, Connector/J takes a permissive approach: statements that might return results are allowed to run. (Bug #71929, Bug #18346501, Bug #103612, Bug #32902019, Bug #23204652)

Changes in MySQL Connector/J 8.0.26 (2021-07-20, General Availability)

Version 8.0.26 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

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

- The TLSv1 and TLSv1.1 connection protocols are now deprecated. While connections to the server using those TLS versions can still be made with the same negotiation process as before, for any connections established using those TLS versions, Connector/J writes to its logger the message "This connection is using TLSv1[.1] which is now deprecated and will be removed in a future release of Connector/J."

For background to this deprecation, refer to the IETF memo Deprecating TLSv1.0 and TLSv1.1. It is recommended that connections be made using the more-secure TLSv1.2 and TLSv1.3 protocols.

Functionality Added or Changed

- **Important Change:** The mechanism for deciding on the character set and collation for communication between the server and Connector/J has been changed: instead of following the server’s settings by default, they are now determined by the Connector/J connection properties. See Using Character Sets and Unicode for details. (Bug #25554464)

- Connector/J can now receive client session state changes tracked by the server if the tracking has been enabled on the server. A new connection property, trackSessionState, has been created to enable Connector/J to receive the information with query results when the property is set to true. Thanks to William Lee for contributing to this feature. (Bug #32435618, Bug #102404)

- Connector/J can now establish connections for accounts that use the authentication_kerberos server-side authentication plugin, which is supported by commercial editions of the MySQL Server since release 8.0.26. See Connecting Using Kerberos for details.

- Connector/J now supports Query Attributes when they have been enabled on the server. See Using Query Attributes for details.

Bugs Fixed

- **X DevAPI:** The streaming of DocResult failed if rows were preceded by any Notice messages in the stream. This was because the addProtocolEntity() method of StreamingDocResultBuilder did not return false for attempts to add Notice messages with the method, and that has been fixed by this patch. (Bug #30438500, Bug #97269)

- When using cursor-based fetching to retrieve rows from the server, Connector/J hung after receiving an error packet, as it kept waiting for an actual data packet to arrive. With this fix, Connector/J throws an error in the situation. (Bug #32954396)

- Inserting a BLOB into a table using a server-side PreparedStatement resulted in a ClassCastException. (Bug #32766143, Bug #103303)

- Connector/J of release 8.0.23 and later could not be used in OSGi. It was because the manifest file was missing a few imports needed for OSGi, and this patch fixed the issue. (Bug #32459408, Bug #102372)

- Connector/J issued unnecessary SET NAMES statements to set collation just to the server’s own setting, which caused performance degradation. To fix this and other problems, a new mechanism for negotiating the character set and collation for communication between the server and Connector/J has been introduced; see Using Character Sets and Unicode for details. Thanks to Marc Fletcher for contributing to the fix. (Bug #31818423, Bug #100606)

- Rows were duplicated in the ResultSet returned by the DatabaseMetaData.getImportedKeys() method. It was due to a faulty query in the method, which has been corrected by this patch. Thanks to Miron Balcerzak for contributing to the fix. (Bug #29757140, Bug #95280)
• Connection failed if the ID for the server’s default connection collation is bigger than 255 or if Connector/J specified a custom connection collation. This patch corrects the issue by implementing a new mechanism and a new connection property customCharsetMapping for the negotiation of a collation between the server and Connector/J. See Using Character Sets and Unicode for details. (Bug #25554464)

Changes in MySQL Connector/J 8.0.25 (2021-05-11, General Availability)

Version 8.0.25 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

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/J 8.0.24 (2021-04-20, General Availability)

Version 8.0.24 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

• Functionality Added or Changed
  • A new connection property, scrollTolerantForwardOnly, has been introduced, which preserved the legacy behavior of Connector/J 8.0.17 and earlier by tolerating backward and absolute cursor movements on result sets of type ResultSet.TYPE_FORWARD_ONLY. This is for maintaining compatibility with legacy code that took advantage of the old behavior. See the description for scrollTolerantForwardOnly for details. (Bug #31747910)

  References: See also: Bug #30474158.

• Connector/J now supports "userless" authentication for JDBC connections: When the user for the connection is unspecified, Connector/J uses the name of the OS user who runs the application.
for authentication with the MySQL server. See Connector/J: Obtaining a connection from the
DriverManager for more details.

• Starting from this release, whenever an authentication plugin is explicitly set for the connection property
defaultAuthenticationPlugin, the specified plugin takes precedence over the server’s default when
Connector/J negotiates a plugin with the server. There is no behavioral change for Connector/J if
no value is explicitly set for the property, in which case the server’s choice of default plugin takes
precedence over the implicit default of mysql_native_password for Connector/J. See the description
of defaultAuthenticationPlugin for details.

• In the past, for JDBC connections, when the server closed a session because a client was idling beyond
the period specified by the server’s wait_timeout system variable, Connector/J returned a generic IO
error. Connector/J now relays a clearer error message from the server.

Bugs Fixed

• X DevAPI: Concurrently getting and closing multiple sessions from the same X DevAPI Client object
might result in a ConcurrentModificationException thrown by Connector/J at the closing of a
session. (Bug #31699993)

• X DevAPI: Under some specific conditions, when using Deflate as the algorithm for compression of X
Protocol connections, Connector/J threw an AssertionFailedException (ASSERTION FAILED:
Unknown message type: 57). It was because when a compressed packet was just a few bytes
longer than the size of some internal buffer used by a Java InflaterInputStream, the leftover bytes
from the inflate procedure were being discarded by Connector/J, causing inflation of subsequent packets
to fail. With this fix, no data bytes are discarded, and the inflation works as expected. (Bug #31510398,
Bug #99708)

• When a SecurityManager was in place, connections to a MySQL Server could not be established
unless the client had been properly configured to use SASL-based LDAP authentication. It was because the
AuthenticationLdapSaslClientPlugin in Connector/J requires a special permission to load
the provider MySQLScramShaSasl when a SecurityManager is in place; but since the provider was
loaded by a static initializer during initialization for the plugin, the lack of the permission was causing an
error and then failures for all connections, even if the plugin was never used or enabled. This fix changes
how the provider is loaded: the loading now happens only at the plugin instance’s initialization and the
initialization was deferred to the time when the plugin is actually needed, so that connections that do
not use SASL-based LDAP authentication are unaffected by security settings regarding the plugin. (Bug
#32526663, Bug #102188)

• When using Connector/J 8.0.23, ResultSetMetaData.getColumnClassName() did not return the
correct class name corresponding to DATETIME columns. (Bug #32405590, Bug #102321)

• Creation of an UpdatableResultSet failed with a NullPointerException when it was generated
by querying a view with a derived value. (Bug #32338451, Bug #102131)

• Using getLong() on the CHAR_OCTET_LENGTH column of the ResultSet for
DatabaseMetaData.getProcedureColumns() (or getFunctionColumns()) resulted in a
NumberFormatException exception when the column’s value exceeded $2^{32} - 1$. With this patch, the value of
$2^{32} - 1$ is returned in the situation. (Bug #32329915, Bug #102076)

• Connections to a server could not be established when the user supplied an implementation of the
ConnectionPropertiesTransform interface using the connection property propertiesTransform.
This was because Connector/J called PropertyKey.PORT.getKeyName() instead of
PropertyKey.HOST.getKeyName() for getting the host name, and that has been corrected by this
fix. (Bug #32151143, Bug #101596)
MySQL Connector/J 8.0 Release Notes

- A **NullPointerException** was returned when a statement that could not be used as a **ServerPreparedStatement** was executed and the connection property **useUsageAdvisor** was set to **true**. With this fix, a **SQLException** is returned instead. (Bug #32141210, Bug #101558)

- Using the **setSessionMaxRows()** method on a closed connection resulted in a **NullPointerException**. With this fix, a **SQLNonTransientConnectionException** is thrown instead, with the error message "No operations allowed after connection closed." (Bug #22508715)

- Using the **setObject()** method for a target type of **Types.TIME** resulted in a **SQLException** when the value to be set had a fractional part, or when the value did not fit any pattern described in **Date and Time Literals**. This patch introduced a new logic that can handle fractional parts; also, it performs the conversion according to the patterns of the literals and, when needed, the target data type. (Bug #20391832)

Changes in MySQL Connector/J 8.0.23 (2021-01-18, General Availability)

Version 8.0.23 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0 and 5.7. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

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

Deprecation and Removal Notes

- As an implementation of the **MySQL Terminology Updates**, connection properties and public method names have been adjusted in the following manners:

  - Changing "master" to "source": For example, the connection property **queriesBeforeRetryMaster** becomes **queriesBeforeRetrySource**, and the method **isMasterConnection()** becomes **isSourceConnection()**

  - Changing "slave" to "replica": For example, the connection property **allowSlavesDownConnections** becomes **allowReplicaDownConnections**, and the method **getSlaveHosts()** becomes **getReplicaHosts()**

  - Changing "blacklist" to "blocklist": For example, the connection property **loadBalanceBlacklistTimeout** becomes **loadBalanceBlocklistTimeout**.

Old names have been deprecated—though they are still usable for now, they are to be removed eventually in future releases; users are therefore encouraged to switch to the new names.

See the **MySQL Connector/J 8.0 Developer Guide**, the Connector/J API documentation (generated by Javadoc), and the **MySQL Connector/J X DevAPI Reference** available at Connectors and APIs for information on any new property and method names.

Functionality Added or Changed

- **Important Change**: A new mechanism has been introduced for users to configure how time zone conversions should occur when time instants are saved to or retrieved from a server by
Connector/J. Three new connection properties, `preserveInstants`, `connectionTimeZone`, and `forceConnectionTimeZoneToSession`, control the new mechanism—see Preserving Time Instants for details.

**Important**

To preserve the default behavior of Connector/J 8.0.22 and earlier to query the session time zone from the server and then convert a timestamp between that and the JVM time zone, set the new connection property `connectionTimeZone` to `SERVER`, and leave the other two new properties at their default values (i.e., `preserveInstants=true` and `forceConnectionTimeZoneToSession=false`). Users who had `serverTimeZone=user-defined-time-zone` and keep it the same, without configuring the new connection properties, can expect the same behavior as before, but testing is recommended.

Also, with the implementation of the new mechanism, a `getObject(columnIndex)` call on a `DATETIME` column returns a `LocalDateTime` object now instead of a `String`. To receive a `String` like before, use `getObject(columnIndex, String.class)` instead.

- While a `java.sql.TIME` instance, according to the JDBC specification, is not supposed to contain fractional seconds by design, because `java.sql.TIME` is a wrapper around `java.util.Date`, it is possible to store fractional seconds in a `java.sql.TIME` instance. However, when Connector/J inserted a `java.sql.TIME` into the server as a MySQL `TIME` value, the fractional seconds were always truncated. To allow the fractional seconds to be sent to the server, a new connection property, `sendFractionalSecondsForTime`, has been introduced: when the property is `true` (which is the default value), the fractional seconds for `java.sql.TIME` are sent to the server; otherwise, the fractional seconds are truncated.

  Also, the connection property `sendFractionalSeconds` has been changed into a global control for the sending of fractional seconds for ALL date-time types. As a result, if `sendFractionalSeconds=false`, fractional seconds are not sent irrespective of the value of `sendFractionalSecondsForTime`. (Bug #20959249, Bug #76775)

- Connector/J now supports the following authentication methods for LDAP Pluggable Authentication with the MySQL Enterprise Server:

  - **The GSSAPI/Kerberos Authentication Method**: A new connection property, `ldapServerHostname`, has been introduced for specifying the LDAP service host principal as configured in the Kerberos key distribution centre (KDC). See the description for `ldapServerHostname` in the MySQL Connector/J 8.0 Developer Guide for details.

  - **The SCRAM-SHA-256 method.**

**Bugs Fixed**

- Storing a `java.time.LocalDateTime` object onto the server as a `TIMESTAMP` value using a batched `PreparedStatement` failed with the complaint that `java.time.LocalDateTime` could not be cast to `java.sql.Timestamp`. With this fix, the casting works again. (Bug #32099505, Bug #101413)

- Using the `setObject()` method to set a `ByteArrayInputStream` instance for a `PreparedStatement` resulted in a `SQLException`. (Bug #32046007, Bug #101242)

- The returned value for a `TIMESTAMP` was incorrect when a `temporal interval expression` was used in the SQL statement for the query. (Bug #31074051, Bug #99013)
After upgrading from Connector/J 5.1 to 8.0, the results of saving and then retrieving DATETIME and TIMESTAMP values became different sometimes. It was because while Connector/J 5.1 does not preserve a time instant by default, Connector/J 8.0.22 and earlier tried to do so by converting a timestamp to the server’s session time zone before sending its value to the server. In this release, new mechanisms for controlling timezone conversion has been introduced—see Preserving Time Instants for details. Under this new mechanism, the default behavior of Connector/J 5.1 in this respect is preserved by setting the connection property preserveInstants=false. (Bug #30962953, Bug #98695, Bug #30573281, Bug #95644)

Conversion of a MySQL DATETIME or TIMESTAMP value to a Java OffsetDateTime using the getObject(i, OffsetDateTime.class) method failed with a "Conversion not supported for type ..." error. It was because the OffsetDateTime.parse() method on DATETIME and TIMESTAMP values yielded an unexpected string format. With this patch, conversions between OffsetDateTime and the DATE, TIME, DATETIME, TIMESTAMP, and YEAR data types are now possible, and an instant point on the timeline is preserved as such during a conversion, when possible—see Preserving Time Instants for details. (Bug #29402209, Bug #94457)

When the server’s session time zone setting was not understandable by Connector/J (for example, it was set to CEST), a connection could not be established with the server unless Connector/J specified the correct IANA time zone name in the serverTimezone connection property. This happened even if there was actually no need to use any date-time functionality in Connector/J. The issue was fixed by the new connection properties for Connector/J that control date-time handling—see Preserving Time Instants for details. The following now happens with respect to the above-mentioned situation:

- If the new connection property connectionTimeZone is set to LOCAL or a specified time zone, the time_zone variable on the server is no longer checked
- If connectionTimeZone=SERVER, the check for the time_zone variable is delayed until date-time driver functionality is first invoked, so that an unrecognizable server time zone does not prevent connection to be established. However, when date-time functionality is invoked and the value of time_zone cannot be recognized by Connector/J, an exception is thrown.

(Bug #21789378)

Changes in MySQL Connector/J 8.0.22 (2020-10-19, General Availability)

Version 8.0.22 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Security Enhancement**: Previously, the LOCAL data loading capability for the LOAD DATA statement can be controlled on the client side only by enabling it for all files accessible to the client or by disabling it altogether, using the connection property allowLoadLocalInfile. A new connection property, allowLoadLocalInfileInPath, has been introduced to allow LOCAL data loading only from a specific filepath; see the description for the option in Configuration Properties for details.

- **X DevAPI**: A new connection property, xdevapi.compression-algorithms, has been introduced for specifying the compression algorithms to use and the priority in which they will be negotiated for.
Also, the older connection property `xdevapi.compression-algorithm` (without an “s” at the end of its name) has been renamed to `xdevapi.compression-extensions`; its function remains the same as before (providing implementations for compression algorithms), but the syntax for its value has been changed: each element in a triplet for an algorithm is now separated by a colon (:). See Connection Compression Using X DevAPI for details.

- **X DevAPI**: The asynchronous variant of the X Protocol is no longer supported by Connector/J; the connection properties `useAsyncProtocol` and `asyncResponseTimeout` are now deprecated and have no effect when used.

- **X DevAPI**: Connector/J now supports Java keystore for SSL client certificates for X DevAPI sessions. The following new connection properties have been introduced for the purpose:
  - `xdevapi.ssl-keystore`
  - `xdevapi.ssl-keystore-type`
  - `xdevapi.ssl-keystore-password`

  See Connecting Securely Using SSL for details.

- **X DevAPI**: Similar control for fallbacks for X DevAPI connections are provided by the new connection properties, `xdevapi.fallback-to-system-keystore` and `xdevapi.fallback-to-system-truststore`.

When trying to open multiple connections to a MySQL server using Connector/J with a named pipe on Windows systems, the attempt sometimes failed with the “All pipe instances are busy” error. With this fix, if a timeout has been set using the connection property `connectTimeout` or the method `DriverManager.setLoginTimeout()`, Connector/J will retry opening the named pipe repeatedly until the timeout is reached or the named pipe is opened successfully. As a side-effect of this new behavior, there will be a delay, equal to the length of the timeout, for throwing an error for a failed named-pipe connection even when it is caused by an error other than “All pipe instances are busy.” (Bug #31711961, Bug #98667)

When the connection option `sslMode` is set to `VERIFY_IDENTITY`, Connector/J now validates the host name in the connection string against the host names or IP addresses provided under the Subject Alternative Name (SAN) extension in the server’s X.509 certificate. Also, verification against the Common Name (CN) is now performed when a SAN is not provided in the certificate or if it does not contain any DNS name or IP address entries. Host names listed in the certificate, under either the SAN or the CN, can contain a wildcard character as specified in the RFC 6125 standard. Thanks to Daniël van Eeden for contributing to the patch. (Bug #31443178, Bug #99767, Bug #28834903, Bug #92903)

When using Connector/J, the `AbandonedConnectionCleanupThread` thread can now be disabled completely by setting the new system property `com.mysql.cj.disableAbandonedConnectionCleanup` to `true` when configuring the JVM. The feature is for well-behaving applications that always close all connections they create. Thanks to Andrey Turbanov for contributing to the new feature. (Bug #30304764, Bug #96870)

Connector/J now supports client authentication with MySQL Enterprise Server using simple or SASL (SCRAM-SHA-1) LDAP authentication on Windows and Linux platforms.

Connector/J can now be prevented from falling back to the system-wide truststore and keystore for server and client identity authentication, respectively. For JDBC connections, two new connection properties, `fallbackToSystemKeyStore` and `fallbackToSystemTrustStore`, have been introduced for the control. While those properties are `true` by default (fallbacks enabled), setting them to `false` disable fallbacks. See Connecting Securely Using SSL for details.
• The integration classes for JBoss have been removed from Connector/J.

**Bugs Fixed**

• In a load balancing setup, if the connection parameter `loadBalanceBlacklistTimeout` was set, a server that was once unavailable remained in the blocklist even after a connection to it has been reestablished, and this affected the system's performance. With this fix, the server is removed from the blocklist as soon as it becomes available again. (Bug #31699357, Bug #96309)

• Using a `PreparedStatement` to store a `Date` into a database sometimes resulted in a `NullPointerException`. It was because some assignments were missing in `ServerPreparedQueryBindValue.clone()`, and this patch corrects the issue. (Bug #31418928, Bug #99713)

• When a client attempted to establish a JDBC connection using the server’s X Protocol port, Connector/J threw an `ArrayIndexOutOfBoundsException`. With this fix, Connector/J throws the proper exception for using the wrong protocol with the port and returns a proper error message. (Bug #31083755, Bug #99076)

• `LocalDate`, `LocalDateTime`, and `LocalTime` values set through Connector/J were altered when there was a timezone difference between the server and the client. This fix corrects the issue by handling the `LocalDate`, `LocalTime`, and `LocalDateTime` with no time zone conversion and no intermediate conversions to other date-time classes. Thanks to Iwao Abe for his contribution to the fix. (Bug #29015453, Bug #93444)

**Changes in MySQL Connector/J 8.0.21 (2020-07-13, General Availability)**

Version 8.0.21 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

In the documentation for MySQL 8.0.21, we have started changing the term “master” to “source”, the term “slave” to “replica”, the term “whitelist” to “allowlist”, and the term “blacklist” to “blocklist”. There are currently no changes to the product’s syntax, so these terms are still present in the documentation where the current code requires their use. See the blog post MySQL Terminology Updates for more information.

• **Functionality Added or Changed**

• **Bugs Fixed**

**Functionality Added or Changed**

• **X DevAPI**: The [JSON Schema Validation Functions](https://dev.mysql.com/doc/connector-j/8.0/en/json-schema-validation.html) on MySQL servers are now supported by Connector/J; see [Schema Validation](https://dev.mysql.com/doc/connector-j/8.0/en/schema-validation.html) for details.

• The required versions of the 3rd-party libraries needed for running or compiling Connector/J have been changed; new requirements for additional libraries have also been added. See [Installing Connector/J from a Binary Distribution](https://dev.mysql.com/doc/connector-j/8.0/en/install-binary-macos.html) and [Installing from Source](https://dev.mysql.com/doc/connector-j/8.0/en/install-source-macos.html) for details.

**Bugs Fixed**

• When trying to set a parameter for a `PreparedStatement` using the method `PreparedStatement.setObject(parameterIndex, "false", Types.BOOLEAN)`, the value was set to `true` instead of `false`. (Bug #30911870, Bug #98237)
Changes in MySQL Connector/J 8.0.20 (2020-04-27, General Availability)

Version 8.0.20 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **X DevAPI**: Connector/J now supports data compression for X Protocol connections. See Connection Compression Using X DevAPI for details.

- A new method, `getElapsedTime()`, has been added to the implementation of the `Statement` interface in Connector/J, to expose the elapsed time for a query. Thanks to Matti Sillanpää for contributing the code. (Bug #30570249, Bug #97714)

Bugs Fixed

- When a custom `Calendar` was used in the `setDate` method for a `PreparedStatement`, it was being used by subsequent calls of the same method that did not use the same calendar, resulting in the wrong date being set. It was because the `SimpleDateFormat` object created internally with the custom calendar was cached and reused. With this fix, the object is no longer cached. (Bug #30877755)

- Setting the connection property `clientInfoProvider` without using the fully qualified class name for `ClientInfoProviderSP` caused a `NullPointerException`. This was due to some wrong exception handling, which has been corrected by this fix. (Bug #30832513)

- Authentication failed when a client tried to connect to a server that used Windows Authentication Plugin and the Kerberos protocol. It was because the implementation of the `NativeAuthenticationProvider` class by Connector/J did not interact correctly with a custom-made Kerberos authentication plugin, and this patch fixes the issue. (Bug #30805426)

- Methods from the `ResultSetUtil` class are no longer used in Connector/J 8.0.20; the class has therefore been removed. (Bug #30636056)

- A `NullPointerException` was returned when the connection had `cacheResultSetMetadata=true` and a query containing any `SET` statements was executed. This fix corrects the issue by adding the missing variable assignment, and also a null check. (Bug #30584907, Bug #97757)

- A `DataConversionException` was thrown when an application tried to store a string starting with "d." [d was any digit] into a `VARCHAR` column. It was due to a parsing error in the `AbstractNumericValueFactory`, which has been fixed by this patch. Thanks to Nick Pollett for contributing the code. (Bug #30570721, Bug #97724)

- When creating a `Statement`, the specification for the `resultSetType` parameter was not honored, so that the `ResultSet` type was always set to `ResultSet.TYPE_FORWARD_ONLY`. With this fix, the `resultSetType` parameter is now honored. Also, type validation has been added so that calling the methods `beforeFirst`, `afterLast`, `first`, `last`, `absolute`, `relative`, or `previous` results in an exception if the `ResultSet` type is `ResultSet.TYPE_FORWARD_ONLY`. (Bug #30474158)
• When a Calendar was not used, a `java.sql.Date` value could not always be stored into and then retrieved from a MySQL server consistently. It was because Connector/J always converted a `Date` value to the server's time zone when storing it on the server as a MySQL `DATE`; but since a MySQL `DATE` does not have any time value, the hour, minute, and second parts of the original date was effectively lost. If the converted value is one day ahead of or behind the original value, when the value was retrieved through Connector/J and converted back to the local time zone, there was no time value for adjusting the date back to its original value, resulting in a one-day error. With this fix, any `Date` value is converted to MySQL `DATE` value using the JVM's time zone, so that the value is always consistent when being stored and then read back.

Also, the `cacheDefaultTimezone` connection property, previously removed from Connector/J 8.0, has now been restored so that when it is set to `false`, Connector/J becomes aware of the time zone changes of the JVM during runtime and converts dates with the updated time zone. (Bug #28125069, Bug #91112)

### Changes in MySQL Connector/J 8.0.19 (2020-01-13, General Availability)

Version 8.0.19 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- **Functionality Added or Changed**

- **Bugs Fixed**

### Functionality Added or Changed

- **X DevAPI:** The server failover support for connections using X DevAPI has been enhanced with the following features:
  - When the `priority` property is NOT set for each host in the connection URL, failover connections are attempted on the hosts in a random order, until a connection is successfully established (Connector/J used to attempt connections to the hosts in the sequence they appear in the connection URL).
  - Once all hosts have been tried and no connections can be made, Connector/J throws a `com.mysql.cj.exceptions.CJCommunicationsException` and returns the message **Unable to connect to any of the target hosts**.
  - When using connection pooling, Connector/J keeps track of any host it failed to connect to and, for a short waiting period after the failure, avoids connecting to it during the creation or retrieval of a `Session`. However, if all other hosts have already been tried, those excluded hosts will be retried without waiting. Once all hosts have been tried and no connections can be established, Connector/J throws a `com.mysql.cj.exceptions.CJCommunicationsException` and returns the message **Unable to connect to any of the target hosts**.

- **X DevAPI:** The allowed TLS versions and cipher suites for X DevAPI connections can now be restricted by two new connection properties:
  - `xdevapi.tls-versions` restricts the allowable TLS protocol versions to be used for X DevAPI connections.
  - `xdevapi.tls-ciphersuites` restricts the allowable cipher suites to be used for X DevAPI connections.
MySQL Connector/J 8.0 Release Notes

See the descriptions for them in Configuration Properties and also Connecting Securely Using SSL for details.

• MySQL Server 8.0.17 deprecated the display width for the TINYINT, SMALLINT, MEDIUMINT, INT, and BIGINT data types when the ZEROFILL modifier is not used, and MySQL Server 8.0.19 has removed the display width for those data types from results of SHOW CREATE TABLE, SHOW CREATE FUNCTION, and queries on INFORMATION_SCHEMA.COLUMNS, INFORMATION_SCHEMA.ROUTINES, and INFORMATION_SCHEMA.PARAMETERS (except for the display width for signed TINYINT(1)). This patch adjusts Connector/J to those recent changes of MySQL Server and, as a result, DatabaseMetaData, ParameterMetaData, and ResultSetMetaData now report identical results for all the above-mentioned integer types and also for the FLOAT and DOUBLE data types. (Bug #30477722)

• The cipher suites usable by Connector/J are now pre-restricted by a properties file that can be found at src/main/resources/com/mysql/cj/TlsSettings.properties inside the src folder on the source tree or in the platform-independent distribution archive (in .tar.gz or .zip format) for Connector/J. See Connecting Securely Using SSL for details.

• Connector/J now supports the use of DNS SRV records for connections. Here is a brief summary for Connector/J's support for DNS SRV records:

  • These new schemas in the connection URLs enable DNS SRV support:

    • jdbc:mysql+srv: For ordinary and basic failover JDBC connections that make use of DNS SRV records.

    • jdbc:mysql+srv:loadbalance: For load-balancing JDBC connections that make use of DNS SRV records.

    • jdbc:mysql+srv:replication: For replication JDBC connections that make use of DNS SRV records.

    • mysqlx+srv: For X DevAPI connections that make use of DNS SRV records.

  • Besides using the new schemas in the connection URLs, DNS SRV record support can also be enabled or disabled using the two new connection properties, dnsSrv and xdevapi.dns-srv, for JDBC and X DevAPI connections respectively.


• The allowable versions of TLS protocol used for connecting to the server, when no restrictions have been set using the connection properties enabledTLSProtocols, have been changed to:

  • TLSv1, TLSv1.1, TLSv1.2, and TLSv1.3 for MySQL Community Servers 8.0, 5.7.28 and later, and 5.6.46 and later, and for all commercial versions of MySQL Servers.

  • TLSv1 and TLSv1.1 for all other versions of MySQL Servers.

**Bugs Fixed**

• The RPM package for Connection/J provided by the MySQL Yum repository did not have its epoch set; as a result, the package could not be installed on an Enterprise Linux or Fedora system even if the MySQL Yum repository has been enabled, because the Connector/J package in the native repository had the epoch set to 1. This fix sets the epoch also to 1 for the RPM package provided by the MySQL Yum repository, in order to prevent the installation problem. (Bug #30566384, Bug #97700)
MySQL Connector/J 8.0 Release Notes

• For some prepared statements, calling `getMetaData()` on them resulted in an Incorrect DATE error, even when no DATE values were involved. This was due to some recent changes on the MySQL Server, to which this patch adjusts Connector/J. (Bug #30151808, Bug #96442)

References: See also: Bug #29025656, Bug #28940878.

• When retrieving TIME values using `ResultSet.getTimestamp()`, the fractional seconds are truncated when `useCursorFetch=true`. This patch corrects the problem by fixing the TIME value decoding in the `MysqlBinaryValueDecoder`. It also corrects some inconsistencies in formatting the fractional seconds when returning the TIME values as strings. (Bug #30119545, Bug #96383)

• Streaming of multiple result sets failed with an error. It was due to an error in switching the streamer source from one result set to another, and this fix corrects the issue. (Bug #29999318, Bug #96059)

Changes in MySQL Connector/J 8.0.18 (2019-10-14, General Availability)

Version 8.0.18 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

Bugs Fixed

• A minor code improvement has been put into `DatabaseMetaDataUsingInfoSchema.getColumns()`. (Bug #29898567, Bug #95741)

• For a replication setup, when the connection property `loadBalanceAutoCommitStatementThreshold` was set to any values other than 0, load-balancing server switching failed. It was because in this case, `LoadBalancedAutoCommitInterceptor` did not have the required access to the parent connection proxy that had established the connection, and this fix enables such access. (Bug #25223123, Bug #84098)

• An attempt to retrieve multiple result sets returned by asynchronous executions of stored procedures resulted in an ExecutionException. With this fix, Connector/J now works properly when asynchronous executions return multiple result sets. (Bug #23721537)

Changes in MySQL Connector/J 8.0.17 (2019-07-22, General Availability)

Version 8.0.17 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• X DevAPI: The following methods have been deprecated:
  • `Collection.find().where()`
  • `Collection.modify().where()`
  • `Collection.remove().where()`
MySQL Connector/J 8.0 Release Notes

- **X DevAPI:** Two new operators for JSON objects and arrays, `OVERLAPS` and `NOT OVERLAPS`, are now supported.

- **X DevAPI:** Indexing for array fields is now supported. See Indexing Array Fields in the X DevAPI User Guide for details.

- The `README` and `LICENSE` files are now included inside the Connector/J JAR archive delivered in the platform-independent tarballs and zip files. (Bug #29591275)

- A number of private parameters of `ProfilerEvents` (for example, `hostname`) had no getters for accessing them from outside of the class instance. Getter methods have now been added for all the parameters of the class. (Bug #20010454, Bug #74690)

- A new connection property, `databaseTerm`, sets which of the two terms is used in an application to refer to a database. The property takes one of the two values `CATALOG` or `SCHEMA` and uses it to determine which `Connection` methods can be used to set/get the current database, which arguments can be used within the various `DatabaseMetaData` methods to filter results, and which fields in the `ResultSet` returned by `DatabaseMetaData` methods contain the database identification information. See the entry for `databaseTerm` in Configuration Properties for details.

  Also, the connection property `nullCatalogMeansCurrent` has been renamed to `nullDatabaseMeansCurrent`. The old name remains an alias for the connection property.

  Thanks to Harald Aamot for contributing to the patch. (Bug #11891000, Bug #27356869, Bug #89133)

- A new CONTRIBUTING file has been added to the Connector/J repository on GitHub, which provides guidelines for code contribution and bug reporting.

- The MySQL Connector/J X DevAPI Reference can now be generated from the Connector/J source code as an Ant target, `xdevapi-docs`.

- Added support for host names that are longer than 60 characters (up to 255 characters), as they are now supported by MySQL Server 8.0.17.

- Added support for the `utf8mb4_0900_bin` collation, which is now supported by MySQL Server 8.0.17.

- A cached server-side prepared statement can no longer be effectively closed by calling `Statement.close()` twice. To close and de-cache the statement, do one of the following:
  - Close the connection (assuming the connection is tracking all open resources).
  - Use the implementation-specific method `JdbcPreparedStatement.realClose()`.
  - Set the statement as non-poolable by calling the method `Statement.setPoolable(false)` before or after closing it.

**Bugs Fixed**

- **X DevAPI:** The `IN` operator in X DevAPI expressions, when followed by a square bracket (`[]`), got mapped onto the wrong operation in X Protocol. (Bug #29821029)

- When using a replication connection, retrieving data from `BlobFromLocator` resulted in a `ClassCastException`. It was due to some wrong and unnecessary casting, which has been removed by this fix. (Bug #29807741, Bug #95210)

- `ResultSetMetaData.getTableName()` returned null when no applicable results could be returned for a column. However, the JDBC documentation specified an empty string to be returned in that
case. This fix makes the method behave as documented. The same correction has been made for
getCatalogName() and getSchemaName(). (Bug #29452669, Bug #94585)

• ResultSetImpl.getObject(), when autoboxing a value of a primitive type retrieved from a column,
  returned a non-null object when the retrieved value was null. (Bug #29446100, Bug #94533)

• ResultSetImpl.getDouble() was very inefficient because it called
  FloatingPointBoundsEnforcer.createFromBigDecimal, which needlessly recreated
  BigDecimal objects for the fixed minimum and maximum bounds. With this fix, the objects
  BigDecimal.valueOf(min) and BigDecimal.valueOf(max) are cached after they are first
  created, thus avoiding their recreations. (Bug #29446059, Bug #94442)

• Enabling logSlowQueries resulted in many unnecessary calls
  of LogUtils.findCallingClassAndMethod(). With this fix,
  LogUtils.findCallingClassAndMethod() is called only when profileSQL is true and even
  in that case, the number of calls are reduced to a minimal to avoid the excessive stack trace data
  the function used to generate. Thanks to Florian Agsteiner for contributing to the fix. (Bug #29277648, Bug
  #94101, Bug #17640628, Bug #70677)

• Characters returned in a ResultSet were garbled when a server-side PreparedStatement was
  used, and the query involved concatenation of a number and a string with multi-byte characters. That
  was due to an issue with the number-to-string conversion involved, which has been corrected by this fix.
  (Bug #27453692)

• Calling ProfilerEvent.pack() resulted in an ArrayIndexOutOfBoundsException. It was due to
  a mishandling of data types, which has been corrected by this fix. (Bug #11750577, Bug #41172)

Changes in MySQL Connector/J 8.0.16 (2019-04-25, General Availability)

Version 8.0.16 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is
suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity
(JDBC) 4.2 API, and implements the X DevAPI.

• Functionality Added or Changed

• Bugs Fixed

Functionality Added or Changed

• X DevAPI: Added BigInteger, BigDecimal, and Character as supported classes whose instances
  can be passed as parameters to a X DevAPI Table statement. Also made the error message clearer
  when applications try to pass instances of unsupported classes. (Bug #25650912)

• X DevAPI: Connector/J now supports the ability to send connection attributes (key-value pairs that
  application programs can pass to the server at connect time) for X Protocol connections. Connector/J
  defines a default set of attributes, which can be disabled or enabled. In addition, applications can specify
  attributes to be passed in addition to the default attributes. The default behavior is to send the default
  attribute set. See the description for the new configuration property xdevapi.connect-attributes
  for details.

  Note
  The aggregate size of connection attribute data sent by a client is limited by the value of the
  performance_schema_session_connect_attrs_size server
variable. The total size of the data package should be less than the value of the server variable, or the attribute data will be truncated.

- **X DevAPI:** When using X DevAPI, performance for statements that are executed repeatedly (two or more times) is improved by using server-side prepared statements for the second and subsequent executions. See Working with Prepared Statements in the X DevAPI User Guide for details.

- The version number has been removed from the name of the Connector/J JAR archive within the RPM packages for Connector/J. That makes upgrading Connector/J with RPM packages easier. (Bug #29384853)

- The collation `utf8mb4_zh_0900_as_cs` has been added to the CharsetMapping class. (Bug #29244101)

- The following third-party libraries have been removed from the distribution bundles for Connector/J:
  - Google protobuf for Java (required for using X DevAPI and for building Connector/J from source)
  - C3P0 (required for building Connector/J from source)
  - JBoss common JDBC wrapper (required for building Connector/J from source)
  - Simple Logging Facade API (required for using the logging capabilities provided by the default implementation of org.slf4j.Logger.Slf4JLogger by Connector/J, and for building Connector/J from source)

Users who need those libraries have to obtain them on their own. See Installing Connector/J from a Binary Distribution and Installing from Source for details.

### Bugs Fixed

- **X DevAPI:** The method `unquoteWorkaround()` has been removed from the ExprParser class, as the workaround is no longer needed, and it actually produced wrong results in some cases. (Bug #29257922)

- **X DevAPI:** Connector/J threw an error when a JSON document contained only a field with an empty array as its value. With this fix, Connector/J now takes that as a valid JSON document. (Bug #28834959, Bug #92819)

- **X DevAPI:** `getByte()` calls failed on table columns of the BINARY data type. This was due to issues with string conversion, which has been corrected with this fix. (Bug #25650385)

- **X DevAPI:** Any statements sent after a failed procedure call caused Connector/J to hang. This was because after the failed call, Connector/J was not aware that the result streamer had already been closed by the server. With this fix, an error is thrown when the procedure call fails, and the result streamer is nullified. (Bug #22038729)

- **X DevAPI:** Unary negative and positive operators inside expressions were parsed wrongly as binary minus and plus operators. (Bug #21921956)

- Because the SHOW PROCESSLIST statement might cause the server to fail sometimes, Connector/J now avoids using the statement, but queries the performance scheme instead for the information it needs. (Bug #29329326)

- Some unnecessary information has been removed from the Connector/J log. (Bug #29318273)

- In the DatabaseMetaDataUsingInfoSchema interface, the getProcedureColumns() and getFunctionColumns() methods returned wrong results for the PRECISION column, and the
getColumns() and getVersionColumns() methods returned wrong results for the COLUMN_SIZE column. The errors were due to the wrong handling of the temporal type precision by Connector/J, which has now been fixed. (Bug #29186870)

- For an SSL connection, after a client disconnected from a server by calling Connection.close(), the TCP connection remained in the TIME_WAIT state on the server side. With this fix, the connection remains in the TIME_WAIT state on the client side instead, in most cases. (Bug #29054329, Bug #93590)

- The function LoadBalancedConnectionProxy.getGlobalBlacklist() always returned an empty map, thus there was never a blacklist for load-balanced connections. (Bug #28860051, Bug #93007)

- The redundant file, changelog.gz, has been removed from the Debian 9 package for Connector/J. The file repeated the contents of the CHANGES.gz file. (Bug #27786499)

- Using getBytes() to retrieve TEXT data resulted in a NumberFormatException. With this fix, the proper exception (SQLDataException), is now thrown. (Bug #27784363)

- A changeUser() call failed with a java.io.IOException when the configuration property enablePacketDebug was set to true for a connection. (Bug #25642021)

- bindings.getBoolean() always returned false. It was due to a mishandling of data types, which has been corrected with this fix. (Bug #22931700)

### Changes in MySQL Connector/J 8.0.15 (2019-02-01, General Availability)

Version 8.0.15 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, and 5.6. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

#### Functionality Added or Changed

- Default value of the connection property allowLoadLocalInfile has been changed to false. Applications that use the LOAD DATA LOCAL INFILE statement on MySQL Servers need to set this property to true explicitly. (Bug #29261254, Bug #30866178)

### Changes in MySQL Connector/J 8.0.14 (2019-01-21, General Availability)

Version 8.0.14 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

#### Functionality Added or Changed

#### Bugs Fixed

- Important Change: For MySQL Server 8.0.14 and later, 5.7.25 and later, 5.6.43 and later, and 5.5.63 and later, minimal permissions on named pipes are granted to clients that use them to connect to the server. Connector/J, however, can only use named pipes when granted full access on them. As a workaround, the MySQL Server that Connector/J wants to connect to must be started with the system variable named_pipe_full_access_group; see the description for the system variable for more details. (Bug #28971500)
• **X DevAPI:** `getDefaultSchema()` now returns `null` when no default schema has been set for the `Session`.

• Connector/J now has a new property for building from source, `com.mysql.cj.build.verbose`, which controls the verbosity of the build process' output. Its default value is `false`, which makes the output considerably shorter comparing with earlier versions of Connector/J. (Bug #28970166)

• The method `ResultSet.getBoolean()` now returns `FALSE` when the designated column is of data type `CHAR` or `VARCHAR` and contains an "N" or "n". This makes Connector/J 8.0 behaves like Connector/J 5.1 when it comes to converting strings to booleans. (Bug #28706219, Bug #92574)

• Connector/J is now capable of reading and, if needed, ignoring any initial notice packets sent by X Plugin before an X Protocol connection is established.

### Bugs Fixed

• **X DevAPI:** Connector/J returned a `NullPointerException` when an application tried to establish an XProtocol connection using a Windows named pipe, which is not supported. With this fix, an `XProtocolException` is returned instead.

  This fix also makes sure that instead of a `NullPointerException`, a proper exception is thrown when an application tries to establish a Classic MySQL Protocol connection with a named pipe, but the named pipe is not specified at connection or it cannot be found on the specified path. (Bug #28606708)

• **X DevAPI:** Adding an empty document with `executeAsync()` resulted in an `ERROR 5013 (Missing row data for Insert)`. With this fix, no error or warning is returned in the case. (Bug #23045642)

• `Collection.count()` returned a wrong error message when the collection did not exist. (Bug #28924137)

• The source code of Connector/J contains non-ASCII characters, which might cause encoding issues during compilation if the system did not also use a UTF-8 locale. With this fix, the build script now handles non-ASCII characters well regardless of the system locale. (Bug #28894344)

• A memory leak occurred if Connector/J was loaded via the bootstrap class path instead of the main application classpath. It was because `AbandonedConnectionCleanupThread` failed to initialize its internal thread in that case, so that references for closed connections were not cleaned up, and their number kept growing. This fix repairs the clean up process for closed connections and also makes the process thread safe. (Bug #28747636, Bug #92508)

• `clearInputStream()` returned a `NullPointerException` when the `mysqlSocket`, `mysqlInput`, or `mysqlOutput` object it tried to retrieve was null. With this fix, an `IOException` is thrown instead in the situation. Thanks to Henning Schmiedehausen for contributing to the fix. (Bug #28731795, Bug #92625)

• Updating a result set returned by a server-side prepared statement with `SELECT ... FOR UPDATE` resulted in a `SQLException`. (Bug #28692243, Bug #92536)

• When the connection property `zeroDateTimeBehavior` was set to `CONVERT_TO_NULL`, Connector/J converted a `TIME` type value of `00:00:00` to `null`. With this fix, it returns a `java.sql.Time` instance of zero hours, minutes, and seconds, as expected. (Bug #28101003, Bug #91065)

• When using server-side prepared statements and working with a table with multicolumn primary key, an `updateRow()` call failed with a `NullPointerException` or a `SQLException`. (Bug #25650514)

• When using server-side prepared statements, a `refreshRow()` call after an `updateRow()` call failed with a `SQLException`. (Bug #25650482)
• `changeUser()` failed to change or reauthenticate a user when all of the following were true: (a) connection to the server was by SSL; (b) the `caching_sha2` or `sha256_password` authentication plugin was used for the user; and (c) the user password contained Unicode characters. (Bug #25642226)

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

Version 8.0.13 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** Connector/J now requires Protocol Buffers 3.6.1 as an external library for using X DevAPI and for building Connector/J from source.

  See Connector/J Installation on installation requirements for Connector/J. (Bug #28499094)

- **X DevAPI:** X DevAPI now provides a connection pooling feature, which can reduce overhead for applications by allowing idle connections to be reused. Connection pools are managed by the new `Client` objects, from which sessions can be obtained. See Connecting to a Single MySQL Server Using Connection Pooling in the X DevAPI User Guide for details.

- **X DevAPI:** A new connection property, `xdevapi.connect-timeout`, now defines the timeout (in milliseconds) for establishing an X-Protocol connection to the server. Default value is 10000 (10s), and a value of 0 disables timeout, which makes Connector/J wait for the underlying socket to time out instead. See Configuration Properties for details.

  Note that if `xdevapi.connect-timeout` is not set explicitly and `connectTimeout` is, `xdevapi.connect-timeout` takes up the value of `connectTimeout`.

- The connection property `useOldUTF8Behavior` is no longer supported. The connection property never had any meaning for the MySQL Server versions supported by Connector/J 8.0, but actually corrupted the data when it was used with them. (Bug #28444461)

- Connector/J now translates the legacy value of `convertToNull` for the connection property `zeroDateTimeBehavior` to `CONVERT_TO_NULL`. This allows applications or frameworks that use the legacy value (for example, NetBeans) to work with Connector/J 8.0. (Bug #28246270, Bug #91421)

- A new connection property, `sslMode`, has been introduced to replace the connection properties `useSSL`, `requireSSL`, and `verifyServerCertificate`, which are now deprecated. Also, when not explicitly set, the connection properties `xdevapi.ssl-mode`, `xdevapi.ssl-truststore`, `xdevapi.ssl-truststore-password`, and `xdevapi.ssl-truststore-type` now take up the values of `sslMode`, `trustCertificateKeyStoreUrl`, `trustCertificateKeyStorePassword`, and `trustCertificateKeyStoreType`, respectively. See Connecting Securely Using SSL and Configuration Properties for details.

  Note that for ALL server versions, the default setting of `sslMode` is `PREFERRED`, and it is equivalent to the legacy settings of `useSSL=true, requireSSL=false, and verifyServerCertificate=false`, which are different from their default settings for Connector/J.
8.0.12 and earlier in some situations. Applications that continue to use the deprecated properties and rely on their old default settings should be reviewed. (Bug #27102307)

- The value UTF-8 for the connection property characterEncoding now maps to the utf8mb4 character set on the server and, for MySQL Server 5.5.2 and later, characterEncoding=UTF-8 can now be used to set the connection character set to utf8mb4 even if character_set_server has been set to something else on the server. (Before this change, the server must have character_set_server=utf8mb4 for Connector/J to use that character set.)

Also, if the connection property connectionCollation is also set and is incompatible with the value of characterEncoding, characterEncoding will be overridden with the encoding corresponding to connectionCollation.

See Using Character Sets and Unicode for details, including how to use the utf8mb3 character set now for connection. (Bug #23227334, Bug #81196)

**Bugs Fixed**

- **X DevAPI:** Connector/J threw a WrongArgumentException when it encountered a JSON number with more than ten digits. This was due to an error in the JSON parser, which has now been fixed. (Bug #28594434, Bug #92264)

- **X DevAPI:** Session.getUri() returned a NullPointerException when the default value is null for any of the connection properties contained in the connection URL; and when Session.getUri() returned a URL, the URL contained a comma (",") before its first connection property. (Bug #23045604)

- **X DevAPI:** When handling an invalid JSON document, Connector/J threw a NullPointerException. With this fix, a WrongArgumentException is thrown instead in the situation. (Bug #21914769)

- Setting the connection property characterEncoding to an encoding that maps to the MySQL character set latin1 or utf8mb4 did not result in the corresponding default connection collation (latin1_swedish_ci or utf8mb4_0900_ai_ci, respectively) to be used on the server. With this fix, the server default is used in the situation. (Bug #28207422)

- Calling UpdatableResultSet.updateClob() resulted in a SQLFeatureNotSupportedException. It was because the implementation of the method was missing from Connector/J, and it has been added with this fix. (Bug #28207088)

- When a connection property's value contained an equal sign ("=") in itself, an exception ("WrongArgumentException: Malformed database URL") was thrown. This was due to an error in the parser for the connection URL, which has been corrected by this fix. (Bug #28150662, Bug #92485)

- Connector/J threw a SQLSyntaxErrorException when the parameter tableName for DatabaseMetaDataUsingInfoSchema.getTables() had a null argument. (Bug #28034570, Bug #90887)

- Setting rewriteBatchedStatements=true and useLocalTransactionState=true caused transactions to be uncommitted for batched UPDATE and DELETE statements. It was due to the intermediate queries for enabling multiquery support on the server resetting the local transaction state as a side effect. With this fix, the local transaction state is preserved when the intermediate queries are executed. (Bug #27658489, Bug #89948)

- Rewriting prepared INSERT statements in a multiquery batch failed with a BatchUpdateException when the statements did not contain place holders. This was due a faulty mechanism for query rewriting, which has been corrected by this fix. (Bug #25501750, Bug #84813)
When using batched prepared statements with multiple queries per statement, queries rewriting was incorrect, resulting in the wrong queries being sent to the server. (Bug #23098159, Bug #81063)

Record updates failed for a scrollable and updatable PreparedStatement when the WHERE clause for the updater or refresher contained fractional timestamp values and the connection property sendFractionalSeconds was set to false. It was because in the situation, Connector/J did not perform the proper adjustments of the fractional parts in the WHERE clause values according to the length of the field’s fractional part as defined in the database. This fix makes Connector/J perform the proper adjustment to the fractional part, so that the WHERE clause value can be properly compared to the value fetched from the database. (Bug #22305979)

Some tests in the testsuite failed as they could not recognize system time zone values like CEST or WEST, even with the connection property serverTimezone set. This was because the value of serverTimezone in the testsuite URLs, after being processed by the testsuite, was not actually propagated as a connection property to Connector/J. This fix makes sure the property is in the actual URLs passed to Connector/J. (Bug #21774249)

When a Java Date value was bound to a PreparedStatement parameter, attempts to format the value by a proleptic GregorianCalendar failed to make the dates proleptic, so that dates before the Julian-Gregorian cutover (October 15, 1582) were stored wrongly. With this fix, a proleptic calendar is properly used if supplied to the setDate() method.

Note that when trying to set or retrieve dates before the Julian-Gregorian cutover with PreparedStatement methods, a proleptic GregorianCalendar should always be explicitly supplied to the setDate() and getDate() method. For details, see Known Issues and Limitations. (Bug #18749544, Bug #72609)

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

Version 8.0.12 is the latest General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **X DevAPI:** The following changes have been made to the API:
  - Removed ModifyStatement.arrayDelete() and ModifyStatement.merge().
  - Renamed Collection.find().limit().skip() to Collection.find().limit().offset().
  - To simplify the class hierarchy and to have the class names reflect better the classes’ functions, the following changes have been made:
    - The FindParams class has been renamed to FilterParams
    - The AbstractFindParams class has been renamed to AbstractFilterParams
    - The DocFindParams class has been renamed to DocFilterParams
    - The TableFindParams class has been renamed to TableFilterParams
Notice that the methods in the original `FilterParams` class have been moved under the new `AbstractFilterParams` class.

(Bug #28027459)

- **X DevAPI**: Connector/J now uses synchronous client sockets (`java.net.Socket`) by default to communicate with MySQL servers for X Protocol connections. While asynchronous sockets can still be used by setting the connection property `xdevapi.useAsyncProtocol=true`, this is not recommended, as it might result in performance degradation for Connector/J. (Bug #27522054)

- **X DevAPI**: Connector/J now gives provision for the use of a custom socket factory for X Protocol connections to MySQL Servers using Unix domain sockets. See Section 6.8, "Connecting Using Unix Domain Sockets" for details.

- Connector/J now retrieves the MySQL keyword list from the `INFORMATION_SCHEMA.KEYWORDS` table on the MySQL server when a connection session is established. The list can then be accessed by calling `DatabaseMetaData.getSQLKeywords()`.

- To simplify the code, the `ReadableProperty` and `ModifiableProperty` classes have been consolidated into the `RuntimeProperty` class.

**Bugs Fixed**

- **X DevAPI**: When creating an X DevAPI session using a `Properties` map instead of a connection string, referring to property keys like `host`, `port`, and `protocol` in lowercase caused a `NullPointerException`. With the fix, both upper and lower cases can now be used. (Bug #27652379)

- **X DevAPI**: When creating an X DevAPI session with an SSL connection using a `Properties` map instead of a connection string, a `NullPointerException` was returned when no connection password was provided. (Bug #27629553)

- **X DevAPI**: When using the `getConnection()` method with the `mysqlx:` scheme in the connection URL, Connector/J returned an ordinary JDBC connection instead of an X-Protocol connection. (Bug #26089880)

- If `wait_timeout` was set on the server and the Connector/J had the connection property `interactiveClient=false`, or if `interactive_timeout` was set on the server and Connector/J had the connection property `interactiveClient=true`, a connection is invalidated when it has idled for a longer time than the set timeout. When such a timeout occurred, Connector/J threw a `CJCommunicationsException`, without indicating it was a timeout. With this fix, the error message returned explains the issue and suggests how to avoid it. (Bug #27977617, Bug #90753)

- When an application tried to connect to a non-MySQL database through some JDBC driver and Connector/J happened to be on the class path also, Connector/J threw a `SQLNonTransientConnectionException`, which prevented the application from connecting to its database. With this fix, Connector/J returns null whenever a connection string does not start with `jdbc:mysql:` or `mysqlx:`, so connections to non-MySQL databases are not blocked. (Bug #26724154, Bug #87600)

- A `wasNull()` call on a `ResultSet` did not return the proper value unless `AbstractResultSetRow.getNull()` or `AbstractResultSetRow.getValueFromByte()` was called before. This caused data loss when Connector/J was used with frameworks like Hibernate, which rely on `wasNull()` calls to properly retrieve data. With this fix, `wasNull()` returns a correct value as long as some getter method has been called before on the `ResultSet`. (Bug #25924324, Bug #85941)
Changes in MySQL Connector/J 8.0.11 (2018-04-19, General Availability)

Version 8.0.11 is the first General Availability release of the 8.0 series of MySQL Connector/J. It is suitable for use with MySQL Server versions 8.0, 5.7, 5.6, and 5.5. It supports the Java Database Connectivity (JDBC) 4.2 API, and implements the X DevAPI.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **X DevAPI:** The locking options `lockShared()` and `lockExclusive()`, available when retrieving data from `collection.find()` and `table.select()`, now also accept an optional locking contention value, which is exposed through the enumeration `Statement.LockContention`. The combinations of `lockShared([lockCont])` or `lockExclusive([lockCont])` with `Statement.LockContention.NOWAIT` or `Statement.LockContention.SKIP_LOCKED` map directly to the SQL statement `SELECT ... FOR SHARE` or `SELECT ... FOR UPDATE` with the SQL option `NOWAIT` or `SKIP LOCKED`, for the different InnoDB locking read modes.

- **X DevAPI:** Connector/J now supports the new server-side document ID generation feature. Client-side document ID generation is no longer supported. As a result, the methods `getDocumentId()` and `getDocumentIds()` have been removed and the method `getGeneratedIds()` has been added to the `AddResult` and `AddResultImpl` classes.

- **X DevAPI:** The `SHA256_MEMORY` authentication mechanism is now supported by Connector/J for connections using the X Protocol. See the entry for the connection property `xdevapi.auth` in Configuration Properties for details.

- Connector/J now recognizes the data type `GEOMCOLLECTION`, which has been introduced in MySQL 8.0.11 as an alias and preferred name to the previously known `GEOMETRYCOLLECTION` data type. (Bug #27678308)

- The lower bound for the connection property `packetDebugBufferSize` has been changed to 1, to avoid the connection errors that occur when the value is set to 0. (Bug #26819691)

- Connector/J now supports the use of a custom `SSLSocketFactory` for returning a custom-constructed SSL socket at the time of connection establishment. (Bug #26092824, Bug #86278)

- The source directory and Java package layouts of Connector/J have been revised to make it easier to use custom protocols, APIs, value decoders, and value factories with Connector/J. See the Connector/J source code and the MySQL Connector/J X DevAPI Reference for more details.

Bugs Fixed

- When an integer value in a JSON document is modified, it becomes a a `DOUBLE` value to the MySQL server, which is returned with a decimal when fetched from the JSON document. Therefore, calling `getInteger()` upon the changed value with Connector/J resulted in an `NumberFormatException`. With this fix, `getInteger()` parses such a value correctly and returns an integer. (Bug #27226293)

- In the Ant build file `build.xml`, `com.mysql.cj.api.conf` was missing in the list of OSGi exported packages, causing missing dependencies in OSGi bundles that use Connector/J. (Bug #25765250, Bug #85566)
Changes in MySQL Connector/J 8.0.10 (Skipped version number)

There are no release notes for this skipped version number.

Changes in MySQL Connector/J 8.0.9 (2018-01-30, Release Candidate)

Version 8.0.9 Release Candidate is the first release candidate of the 8.0 branch of MySQL Connector/J, providing an insight into upcoming features. It is suitable for use with MySQL Server versions 5.5, 5.6, 5.7, and 8.0. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **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/J:
  - Components that support DDLs for views, including the `createView()`, `dropView()`, and `modifyView()` methods.
  - Components that support DDLs for tables, including the `createTable()`, `dropTable()`, and `modifyTable()` methods.
  - Components that support session configurations, including the `SessionConfig` object, the `PersistenceHandler` interface, the `PasswordHandler` interface, and the `SessionConfigManager` class.

- **X DevAPI:** Added the `setSavepoint()`, `rollbackTo()`, and `releaseSavepoint()` methods to the `Session` interface to support the `SAVEPOINT`, `ROLLBACK TO SAVEPOINT`, and `RELEASE SAVEPOINT` statements. See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** A new `patch()` function has been added to the `ModifyStatement` interface. The function accepts an JSON-like object describing document changes and applies them to documents matched by the `modify()` filter. See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** The `createIndex()` method for the `Collection` interface now has a new syntax. See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** Added the following methods for single-document operations in the X DevAPI:
  - `replaceOne()`
  - `addOrReplaceOne()`
  - `getOne()`
  - `removeOne()`
See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI:** Setters and getters methods have been added for the configuration properties with the MysqlDataSource, MysqlXADataSource, and MysqlConnectionPoolDataSource classes.

- **X DevAPI:** The connection property enabledTLSProtocols can now be used to select the allowed TLS versions for an X Protocol connection to the server.

- Connector/J now supports the new caching_sha2_password authentication plugin, which is the default authentication plugin for MySQL 8.0.4 and later (see Caching SHA-2 Pluggable Authentication for details).

  **Note**

  To authenticate accounts with the caching_sha2_password plugin, either a secure connection to the server using SSL or an unencrypted connection that supports password exchange using an RSA key pair (enabled by setting one or both of the connecting properties allowPublicKeyRetrieval and serverRSAPublicKeyFile) must be used.

Because earlier versions of Connector/J 8.0 do not support the caching_sha2_password authentication plugin and therefore will not be able to connect to accounts that authenticate with the new plugin (which might include the root account created by default during a new installation of a MySQL 8.0 Server), it is highly recommended that you upgrade now to Connector/J 8.0.9, to help ensure that your applications continue to work smoothly with the latest MySQL 8.0 Server.

- Connector/J now takes advantage of the MySQL Server 8.0 data dictionary by making the connection property useInformationSchema true by default; this makes Connector/J, by default, access the data dictionary more efficiently by querying tables in the INFORMATION_SCHEMA. See INFORMATION_SCHEMA and Data Dictionary Integration for details. Users can still set useInformationSchema to false; but for MySQL 8.0.3 and later, some data dictionary queries might then fail, due to deprecations of older data dictionary features.

- In the past, query texts were always passed as strings to QueryInterceptor methods, even if the texts were not actually used by them. Now, only suppliers for the texts are passed, and the texts are only extracted by get() calls on the suppliers.

### Bugs Fixed

- The connection property nullNamePatternMatchesAll, when set to false (which was the default value), caused some DatabaseMetaData methods to throw an error when a null search string was used with them. The behavior was not compliant with the JDBC specification, which requires that a search criterion be ignored when a null search string is used for it. The connection property has now been removed from Connector/J 8.0. (Bug #26846249, Bug #87826)

- Trying to print the query in a PreparedStatement using the toString() method after it has been closed resulted in an exception (No operations allowed after statement closed) being thrown. (Bug #26748909)

- When working with MySQL Server 8.0, an update or delete statement for a CONCUR_UPDATABLE ResultSet failed when the ResultSet's primary keys included a boolean column and the character set used was not latin1. (Bug #26266731)

- Connector/J failed to recognize a server greeting error it received during a handshake with the server and parsed the error message as a normal greeting packet, causing an ArrayIndexOutOfBoundsException to be thrown. (Bug #24924097)
Changes in MySQL Connector/J 8.0.8 (2017-09-28, Development Milestone)

Version 8.0.8 Development Milestone is the latest development release of the 8.0 branch of MySQL Connector/J, providing an insight into upcoming features. It is suitable for use with MySQL Server versions 5.5, 5.6, 5.7, and 8.0. It supports the Java Database Connectivity (JDBC) 4.2 API.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Packaging**: RPM and Debian packages for installing Connector/J are now available from the Connector/J Download page.

- **X DevAPI**: Connector/J has implemented a new interface of the X Dev API that allows the retrieving, adding, removing, and updating of persistent session continuation data. The implementation includes the following:
  - A `SessionConfig` object that holds the information for a session configuration data set.
  - A `PersistenceHandler` interface that allows custom implementations of persistence handlers.
  - A `PasswordHandler` interface that allows custom implementations of password handling code.
  - A `SessionConfigManager` class for editing and fetching `Sessionconfig` objects, and defining instances of the `PersistenceHandler` and `PasswordHandler`.

  See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI**: A new connection property, `xdevapi.auth`, has been added for specifying the authentication mechanism for connections using the X Protocol. Allowed values are `MYSQL41`, `PLAIN`, and `EXTERNAL`. See the entry for the new property in Configuration Properties for details.

- **X DevAPI**: To support row locks for the `find()` method of the X DevAPI, the `FindStatement` and the `SelectStatement` interfaces have been extended with the following methods:
  - `lockExclusive()`, which works like `SELECT ... FOR UPDATE` for relational tables.
  - `lockShared()`, which works like the `SELECT ... LOCK IN SHARED MODE` (for MySQL 5.7) or `SELECT ... FOR SHARE` (for MySQL 8.0) for relational tables.

  See MySQL Connector/J X DevAPI Reference for more details.

- **X DevAPI**: Connector/J now supports the expanded syntax for the `IN` and `NOT IN` operator, which can check if a sub-expression is contained inside another one; for example:

```java
// For documents
coll.find("$.b IN [100,101,102]").execute();
coll.find("some text with 5432' in $.a").execute();
coll.find("1 in [1, 2, 4]").execute();
coll.find("'a': 3 not in {'a': 1, 'b': 2}").execute();
// For relational tables
tbl.select().where("3 not in [1, 2, 4]").execute();
tbl.select().where("qqq' not in $.a").execute();
tbl.select().where("'a': 1 in {'a': 1, 'b': 2}").execute();
```
• **X DevAPI**: A number of changes have been implemented for the “drop” methods for the X DevAPI:
  
  • Removed `dropCollection(schemaName, collectionName)` and `dropTable(schemaName, tableName)` from `Session`.
  
  • Added `dropCollection(collectionName)` and `dropTable(tableName)` to `Schema`.
  
  • `Schema.dropView()` now executes immediately and returns `void`; also, the `ViewDrop` interface has been removed.
  
  • `Collection.dropIndex()` now executes immediately and returns `void`; also the `DropCollectionIndexStatement` interface has been removed.
  
  • The “drop” methods now succeed even if the objects to be dropped do not exist.

  • Conversion from the MySQL `TIME` data to `java.sql.Date` is now supported. In the past, a `getDateTime()` retrieving data from a `TIME` column would throw an `SQLException`. Now, such a retrieval returns a `java.sql.Date` object containing the time value expressed in number of milliseconds from the Java epoch; also returned is the warning: “Date part does not exist in SQL TIME field, thus it is set to January 1, 1970 GMT while converting to java.sql.Date.” (Bug #26750807)

  • A new connection property, `enabledTLSProtocols`, can now be used to override the default restrictions on the TLS versions to be used for connections, which are determined by the version of the MySQL Server that is being connected to. By providing a comma-separated list of values to this option (for example, “TLSv1,TLSv1.1,TLSv1.2”) users can, for example, prevent connections from using older TLS version, or allow connections to use TLS versions only supported by a user-compiled MySQL Server. See the entry for the new property in Configuration Properties for details. Thanks to Todd Farmer for contributing the code. (Bug #26646676)

  • Updated the timezone mappings using the latest IANA and CLDR time zone databases. (Bug #25946965)

  • A new option for the `loadBalancingStrategy` connection property called `serverAffinity` has been added. The servers listed in the new connection property `serverAffinityOrder` (which should be a subset of the servers in the host list of the connection URL) are contacted in the order they are listed until a server is available or until the list of servers is exhausted, at which point a random load-balancing strategy is used with the hosts not listed by `serverAffinityOrder`. See descriptions for `loadBalancingStrategy` and `serverAffinityOrder` in Configuration Properties for details. (Bug #20182108)

**Bugs Fixed**

• **Important Change**: Following the changes in MySQL Server 8.0.3, the system variables `tx_isolation` and `tx_read_only` have been replaced with `transaction_isolation` and `transaction_read_only` in the code of Connector/J. Users should update Connector/J to this latest release in order to connect to MySQL 8.0.3. They should also make the same adjustments to their own applications if they use the old variables in their codes. (Bug #26440544)

• **X DevAPI**: Calling `schema.dropView()` with a null argument resulted in a `NullPointerException`. (Bug #26750807)

• **X DevAPI**: When `dropCollection()` was applied on a null collection, a `NullPointerException` occurred. (Bug #26393132)

• When using cached server-side prepared statements, a memory leak occurred as references to opened statements were being kept while the statements were being decached; it happened when either the
**close()** method has been called twice on a statement, or when there were conflicting cache entries for a statement and the older entry had not been closed and removed from the opened statement list. This fix makes sure the statements are properly closed in both cases. Thanks to Eduard Gurskiy for contributing to the fix. (Bug #26633984, Bug #87429)

- The regression test for Bug#63800 failed because the default value of the system variable `explicit_defaults_for_timestamp` of MySQL Server has been changed since release 8.0.2. The test has been adjusted to take the change into consideration. (Bug #26501245)

- Running callable statements against MySQL Server 8.0 resulted in the `SQLException: ResultSet is from UPDATE. No Data.` (Bug #26259384)

- Secure JDBC connections did not fall back to the default truststore when a custom one was not provided. (Bug #26243128)

- In `com/mysql/jdbc/ServerPreparedStatement.java`, the arguments `resultSetType` and `resultSetConcurrency` for a call of `Connection.preparedStatement()` were swapped. (Bug #25874048, Bug #85885)

- Some JDBC proxied objects were missing the proper handlings of the `equals()` methods, thus even comparison of one of these proxied objects to its own self with `equals()` yielded false. This patch introduces proper handlings for the `equals()` method in all the relevant proxies. (Bug #21931572, Bug #78313)

- A server-side prepared statement was not closed when the same statement was being prepared again while the original statement was being cached. This was caused by the silent replacement of the cache entry of the old statement by the new. When this happened repeatedly, it caused eventually the complaint that `max_prepared_stmt_count` was exceeded. This fix makes sure that when a cache entry for a statement replaces an older one, the older statement is immediately closed. (Bug #20066806, Bug #74932)

**Changes in MySQL Connector/J 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/J 8.0.7 is the first release to use the new numbering. It is the successor to Connector/J 6.0.6.

- **Functionality Added or Changed**

- **Bugs Fixed**

**Functionality Added or Changed**

- **X DevAPI:** There are changes to some methods related to the `Result` interface:
  - `getLastDocumentId()` and `getLastDocumentIds()` have been replaced with `getDocumentId()` and `getDocumentIds()`, which are put under a new `AddResult` interface that extends `Result`.
  - A new `getAutoIncrementValue()` method is added to the new `InsertResult` interface that extends `Result`.

See [MySQL Connector/J X DevAPI Reference](#) for more details. (Bug #25207784)
• **X DevAPI:** It is no longer permitted to pass an empty search condition, such as the NULL value or an empty string, to the `Collection.Modify()` and `Collection.Remove()` methods.

• **X DevAPI:** Connections using the X Protocol are now secure by default. Also, the `xdevapi.ssl-enable` connection option has been replaced by the `xdevapi.ssl-mode` option, which has `DISABLED`, `REQUIRED` (default), `VERIFY_CA`, and `VERIFY_IDENTITY` as its permitted values; see the description for the new option in Configuration Properties for details.

• **X DevAPI:** Consolidated the `BaseSession`, `NodeSession`, and `XSession` interfaces into a single `com.mysql.cj.api.xdevapi.Session` interface. The following related changes were also made:
  - Renamed `XSessionFactory` to `SessionFactory`.
  - Consolidated the `AbstractSession`, `NodeSessionImpl`, and `XSessionImpl` classes into the `com.mysql.cj.xdevapi.SessionImpl` class.
  - Removed the `Session.bindToDefaultShard()` method and the `VirtualNodeSession` interface.
  - The `mysqlx.getNodeSession()` method has been renamed to `mysqlx.getSession()` and it now returns a `Session` object.
  - The `DatabaseObject.getSession()` method now returns a `Session` object (instead of the old `Session` interface).

See MySQL Connector/J X DevAPI Reference for more details.

• To avoid using JDBC statements inside core Connector/J classes, the following changes have been implemented:
  - Created a new `com.mysql.cj.api.Query` interface, which is implemented by `StatementImpl`.
  - Replaced the `com.mysql.cj.api.jdbc.interceptors.StatementInterceptor` interface with the `com.mysql.cj.api.interceptors.QueryInterceptor` interface.
  - Added a new method, `PacketPayload preProcess(PacketPayload queryPacket)`, to `QueryInterceptor`.
  - Renamed the connection property `statementInterceptors` to `queryInterceptors`. See Configuration Properties for details.

• Added Japanese collation for the `utf8mb4` character set.

**Bugs Fixed**

• **X DevAPI:** `createView()` failed with a `NullPointerException` when there were null inputs to it. This fix adds checks for nulls, and makes Connector/J throw the proper errors for them. (Bug #25575156)

• **X DevAPI:** `createaTable()` failed with a `NullPointerException` when there were null inputs to it. This fix adds checks for nulls, and makes Connector/J throw the proper errors for them. (Bug #25575103)

• **X DevAPI:** The connection properties `enabledSSLCipherSuites`, `clientCertificateKeyStoreUrl`, `clientCertificateKeyStoreType`, and `clientCertificateKeyStorePassword` were ignored for connections using the X Protocol. (Bug #25494338)
MySQL Connector/J 8.0 Release Notes

• **X DevAPI:** Calling `getNodeSession()` with an URL string containing SSL parameters caused a `CJCommunicationsException`. This has been fixed by creating a byte buffer to handle SSL handshake data. (Notice that `getNodeSession()` has since been consolidated into `getSession()`.) (Bug #23597281)

• **X DevAPI:** Concurrent asynchronous operations resulted in hangs, null pointer exceptions, or other unexpected exceptions. This has been fixed by correcting a number of problems with the `SerializingBufferWriter` and by limiting the number of buffers sent with a gathering write. (Bug #23510958)

• **X DevAPI:** When a thread failed to make a connection to the server using the X Protocol, the client application hung. A new connection property, `xdevapi.asyncResponseTimeout` (default value is 300s), now provides a duration beyond which the attempt to connect timeouts, and a proper error is then thrown. See description for the new option in Configuration Properties for details. (Bug #22972057)

• Connector/J failed a number of regression tests in the test suite related to geographic information system (GIS) functions because of changes to GIS support by the MySQL server. The fix corrects the tests. (Bug #26239946, Bug #26140577)

• Attempts to connect to a server started with collation `utf8mb4_de_pb_0900_ai_ci` resulted in null pointer exceptions. (Bug #26090721)

• Configuration templates named by the connection property `useConfigs` were not recognized by Connector/J. (Bug #25757019, Bug #85555)

• A `NullPointerException` was returned when `getDate()`, `getTime()`, or `getTimestamp()` was called with a null `Calendar`. This fix makes Connector/J throw an `SQLException` in the case. (Bug #25650305)

• An `ArrayIndexOutOfBoundsException` was thrown when a server-side prepared statement was used and there was a NULL in a BLOB, TEXT, or JSON type column in the `ResultSet`. (Bug #25215008, Bug #84084)

Index

Symbols
3rd-party libraries, 15

A
AbandonedConnectionCleanupThread, 13
allowLoadLocalInfile, 23
asynchronous execution, 19
asynchronous X Protocol, 13
authentication, 29, 32
authentication plugin, 30
authentication plugins, 9
authentication_oci_client, 6
autocommit, 3

B
BigDecimal, 3, 21
BigDecimal(), 3
BigInteger, 21
bings, 21
BIT(1), 3
BLOB, 4, 7
boxed types, 19
build from source, 23
ByteArrayInputStream, 11

C
cached prepared statements, 19
Calendar, 6, 16
callable statements, 32
changeUser(), 21
Character, 21
calendar set, 7
calendar sets, 6
characterEncoding, 25
cipher suites, 17
client authentication, 13
clientInfoProvider=ClientInfoProviderSP, 16
ClientPreparedStatement, 3, 6
collation, 34
collation, 7, 21
Collection.addOrReplaceOne(), 4
Collection.replaceOne(), 4
com.mysql.cj.testsuite.mysqlx.url.openssl, 6
com.mysql.cj.testsuite.url.openssl, 6
comments, 3
Communications link failure, 27
compression, 9
ConcurrentModificationException, 9
connection attributes, 21
connection compression, 13, 16
connection pooling, 25
connection property, 19
connectionCollation, 25
ConnectionPropertiesTransform, 9
connectoin closed notifications, 9
connectTimeout, 25
convertToNull, 25
count(), 23
createDatabaseIfNotExist, 6
cursor, 6
custom collation, 7
custom load balancing, 32
custom SSLSocketFactory, 29

data dictionary, 30
DatabaseMetaData, 7
DatabaseMetaDataUsingInfoSchema, 21
databaseTerm, 19
Date, 13, 16
dates, 16
DATETIME, 11
Debian package, 32
defaultAuthenticationPlugin, 9
Deflate, 9
deprecations, 19
display width, 17
DNS SRV records, 17
document ID geneartion, 29
dropCollection(), 32
dropX(), 32

E
empty array, 21
enabledTLSProtocols, 17, 30, 32
enablePacketDebug, 21
epoch, 17
equals(), 32
error message, 3
executeAsync(), 23
EXPLAIN, 6
ExprParser, 21

F
FIDO authentication, 3
foreign keys, 6

G
GEOMCOLLECTION, 29
getBoolean(), 21, 23
getByte(), 21
getBytes(), 21
getConnection(), 27
date(), 34
defaultSchema(), 23
defaultTransactionIsolation(), 3
gElapsedTime(), 16
gGetGlobalBlacklist, 21
gGetImportedKeys(), 7
gGetLong(), 9
gGetSession(), 27
gGetTables, 25
gTime(), 34
gGetTimestamp(), 17, 34
gGetUri(), 25
gGetWarnings(), 4
GIS, 34

H
Hibernate, 27
host identity verification, 13
Important Change, 7, 11, 23, 25, 32
IN operator, 19
index for array field, 19
INSERT, 25

Japanese collation, 34
Java package layout, 29
java.sql.Date, 4
java.time.LocalDate, 4
JBoss, 13
JSON, 25
JSON number, 25
JsonNumber.getInteger(), 29

Kerberos, 16
keystore, 13
keyword list, 27

LDAP authentication, 11, 13
load balancing, 13
Load Balancing, 19
LOAD LOCAL DATA INFILE, 13
loadBalanceAutoCommitStatementThreshold, 19
LoadBalancedConnectionProxy, 21
LocalDate, 13
LocalDateTime, 13
LocalTime, 13
lockExclusive(), 32
lockShared(), 32
logger, 3

max_prepared_stmt_count, 32
memory leak, 23
ModifiableProperty, 27
modify(), 34
MultiHostConnectionProxy, 6
multiple result sets, 19
multiqueries, 25
MysqlSQLXML, 6
mysql, 27

named pipe, 23
named pipes, 13
named_pipe_full_access_group, 23
non-MySQL databases, 27
NotSerializableException, 3
not_overlaps, 19
NOWAIT, 29
nullNamePatternMatchesAll, 30

O
OSGi, 29
overlaps, 19

P
Packaging, 32
password, 27
passwords, 4
pathc(), 30
prepared statement, 17, 21, 30, 32
PreparedStatement, 3, 7, 13, 15, 19, 23
preparedStatement(), 32
procedures, 21
profileSQL, 3
proleptic GregorianCalendar, 25
propertiesTransform, 9
Protocol Buffers, 25
proxied objects, 32

Q
QueryInterceptor, 30

R
read-only connections, 6
ReadableProperty, 27
README, 4
refreshRow(), 23
regression test, 32
regression tests, 34
remove(), 34
replication, 3, 19
resultSetType, 16
ResultSet, 4, 9, 17
ResultSet.getBoolean(), 6
ResultSetImpl.getObject(), 19
ResultSetUtil, 16
rewriteBatchedStatements, 25
row locking, 32
RPM, 17
RPM package, 32
RPM packages, 21
RuntimeProperty, 27

S
SAVEPOINT, 30
schema validation, 15
schema.dropView(), 32
scrollTolerantForwardOnly, 9
Security Enhancement, 13
server authentication, 13
server failover, 17
server greeting error, 30
server-side prepared statement, 32
serverAffinity, 32
ServerPreparedStatement, 3
serverTimezone, 25
setautocommit(), 4
setObject(), 9, 11, 15
setQueryTimeout, 6
setSessionMaxRows(), 9
setters and getters, 30
SHA256_MEMORY, 29
SHOW, 6
SHOW PROCESSLIST, 21
SimpleDateFormat, 6
SKIP LOCKED, 29
socksProxyRemoteDns, 3
SocksProxySocketFactory, 3
source directory layout, 29
SQLXML, 4
SSL, 25
SSL connection, 13
SSL connections, 13
sslMode, 3, 13, 25
Statement.cancel(), 4
Statement.setQueryTimeout(), 4
stored procedure, 3
streaming, 17
StreamingDocResultBuilder, 7
synchronization, 6
synchronous sockets, 27

tableNamePattern, 25
terminology updates, 11
testsuite, 6
tf8mb4_de_pb_0900_ai_ci, 34
third-party libraries, 21
TIME, 11, 32
time zone, 11
time zone mappings, 32
timeout, 27
TIMESTAMP, 11
timestamp, 11
timezone conversion, 11
TIME_WAIT, 21
TLS, 7, 32
TLS cipher suites, 6
TLSv1, 4
TLSv1.1, 4
truststore, 13, 32
U
unary negative, 21
unary positive, 21
Unix domain socket, 27
unquoteWorkaround(), 21
UpdatableResultSet, 9
updateClob, 25
updateRow(), 23
useAsyncProtocol, 27
useConfigs, 34
useCursorFetch, 3, 6
useLocalTransactionState, 25
useOldUTF8Behavior, 25
userless authentication, 9
useSSL, 25
useUsageAdvisor, 9
utf8mb3, 3
utf8mb4, 34

V
VALUES, 4
VARCHAR, 16
VAULES(), 4
verbose, 23

W
wasNull(), 27
Windows Authentication Plugin, 16
Windows named pipe, 23

X
X DevAPI, 3, 4, 7, 9, 13, 15, 16, 17, 19, 21, 23, 25, 27, 29, 30, 32, 34
xdevapi.auth, 29, 32
xdevapi.connect-timeout, 25
xdevapi.ssl-mode, 3
xdevapi.tls-ciphersuites, 17
xdevapi.tls-versions, 17
XSession, 34

Z
zeroDateTimeBehavior, 25