



# MariaDB

## **MariaDB ColumnStore Java API Usage Documentation**

*Release 1.2.2-d9bf620-dirty*

**MariaDB Corporation**

Nov 25, 2018

# CONTENTS

<b>1</b>	<b>Licensing</b>	<b>1</b>
1.1	Documentation Content . . . . .	1
1.2	MariaDB ColumnStore Java API . . . . .	1
<b>2</b>	<b>Version History</b>	<b>2</b>
<b>3</b>	<b>Using javamcsapi</b>	<b>4</b>
3.1	Usage Introduction . . . . .	4
3.2	Basic Bulk Insert . . . . .	4
3.3	Advanced Bulk Insert . . . . .	5
<b>4</b>	<b>Compiling with javamcsapi</b>	<b>8</b>
4.1	Pre-requisites . . . . .	8
4.2	Compiling . . . . .	8
<b>5</b>	<b>javamcsapi API Reference</b>	<b>10</b>
5.1	ColumnStoreBulkInsert . . . . .	10
5.2	ColumnStoreDateTime . . . . .	14
5.3	ColumnStoreDecimal . . . . .	15
5.4	ColumnStoreDriver . . . . .	17
5.5	ColumnStoreException . . . . .	19
5.6	ColumnStoreSummary . . . . .	19
5.7	ColumnStoreSystemCatalog . . . . .	20
5.8	ColumnStoreSystemCatalogColumn . . . . .	21
5.9	ColumnStoreSystemCatalogTable . . . . .	23
5.10	ColumnStoreTime . . . . .	25
5.11	TableLockInfo . . . . .	26
5.12	TableLockInfoVector . . . . .	28
5.13	columnstore_data_convert_status_t . . . . .	30
5.14	columnstore_data_types_t . . . . .	30
5.15	columnstore_lock_types_t . . . . .	33
	<b>Index</b>	<b>34</b>

## LICENSING

### 1.1 Documentation Content



The javamcsapi documentation is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

### 1.2 MariaDB ColumnStore Java API

The MariaDB ColumnStore Java API (jvamcsapi) is licensed under the [GNU Lesser General Public License, version 2.1](https://www.gnu.org/licenses/old-licenses/lgpl-2.1.html).

## VERSION HISTORY

This is a version history of Java API interface changes. It does not include internal fixes and changes.

Version	Changes
1.2.2	<ul style="list-style-type: none"> <li>• Added <code>ColumnStoreDriver.clearTableLock</code></li> <li>• Added <code>ColumnStoreDriver.isTableLocked</code></li> <li>• Added <code>ColumnStoreDriver.listTableLocks</code></li> </ul>
1.1.6	<ul style="list-style-type: none"> <li>• Java documentation added</li> <li>• Added <code>ColumnStoreDriver.getJavaMcsapiVersion</code> to return version information about the java wrapper library <code>javamcsapi.jar</code></li> <li>• Windows support added (Alpha)</li> </ul>
1.1.5	<ul style="list-style-type: none"> <li>• Added overloaded functions of <code>ColumnStoreBulkInsert.setColumn</code> and <code>ColumnStoreBulkInsert.setNull</code> to return additional status information</li> </ul>
1.1.4	<ul style="list-style-type: none"> <li>• Make <code>ColumnStoreSystemCatalog.getTable</code> and <code>ColumnStoreSystemCatalogTable.getColumn</code> case insensitive</li> <li>• Add <code>ColumnStoreDriver.setDebug</code> to enable debugging output to <code>stderr</code></li> </ul>
1.1.1	<ul style="list-style-type: none"> <li>• Add <code>ColumnStoreBulkInsert.isActive</code></li> <li>• Make <code>ColumnStoreBulkInsert.rollback</code> fail without error</li> <li>• Add <code>ColumnStoreBulkInsert.resetRow</code></li> <li>• <code>ColumnStoreDateTime</code> now uses <code>uint32_t</code> for every parameter</li> <li>• <code>ColumnStoreSystemCatalog</code> now uses <code>const</code> for the sub-class strings</li> </ul>
1.1.0 $\beta$	<ul style="list-style-type: none"> <li>• First beta release</li> </ul>

## USING JAVAMCSAPI

### 3.1 Usage Introduction

The Java bulk insert API (`javamcsapi`) is a wrapper around the C++ bulk insert API (`mcsapi`) generated by [SWIG](#). As a result Java programmers can utilize the same functions to insert data into `ColumnStore` tables as C++ developers can do. `javamcsapi` is available for Java 8 and upwards.

### 3.2 Basic Bulk Insert

In this example we will insert 1000 rows of two integer values into table `test.t1`. The full code for this can be found in the `example/Basic_bulk_insert.java` file in the `mcsapi` codebase.

You will need the following table in the test database to execute this:

Listing 1: `example/basic_bulk_insert.sql`

```
1 CREATE TABLE `t1` (  
2   `a` int(11) DEFAULT NULL,  
3   `b` int(11) DEFAULT NULL  
4 ) ENGINE=Columnstore;
```

Listing 2: `example/Basic_bulk_insert.java`

```
23 import com.mariadb.columnstore.api.*;
```

We need to import all classes from the package `com.mariadb.columnstore.api` to use `mcsapi`.

Listing 3: `example/Basic_bulk_insert.java`

```
25 public class Basic_bulk_insert {  
26  
27     public static void main(String[] args) {  
28         try {  
29             ColumnStoreDriver d = new ColumnStoreDriver();
```

A new instance of `ColumnStoreDriver` is created which will attempt to find the `Columnstore.xml` configuration file by first searching for the environment variable `COLUMNSTORE_INSTALL_DIR` and then the default path of `/usr/local/mariadb/columnstore/etc/Columnstore.xml`. Alternatively we could provide a path as a parameter to `ColumnStoreDriver`.

Listing 4: example/Basic\_bulk\_insert.java

```
30 ColumnStoreBulkInsert b = d.createBulkInsert("test", "t1", (short)0, 0);
```

Once we have the ColumnStore installation's configuration in the driver we use this to initiate a bulk insert using `ColumnStoreDriver.createBulkInsert`. We are using the `test` database and the `t1` table. The remaining two parameters are unused for now and set to 0.

Listing 5: example/Basic\_bulk\_insert.java

```
31     for (int i=0; i<1000; i++){
32         b.setColumn(0, i);
33         b.setColumn(1, 1000-i);
34         b.writeRow();
35     }
```

A “for” loop is used to loop over 1000 arbitrary inserts in this example. We use `ColumnStoreBulkInsert.setColumn` to specify that column 0 (column a) should be set to the integer from the “for” loop and column 1 (column b) is set to 1000 minus the integer from the “for” loop.

When we have added something to every column `ColumnStoreBulkInsert.writeRow` is used to indicate we are finished with the row. The library won't necessarily write the row at this stage, it buffers up to 100,000 rows by default.

Listing 6: example/Basic\_bulk\_insert.java

```
36         b.commit();
37     }
```

At the end of the loop we execute `ColumnStoreBulkInsert.commit` which will send any final rows and initiate the commit of the data. If we do not do this the transaction will be implicitly rolled back instead.

Listing 7: example/Basic\_bulk\_insert.java

```
38     catch (ColumnStoreException e) {
39         System.err.println("Error caught: " + e.getMessage());
40     }
41 }
```

If anything fails then we should catch `ColumnStoreException` to handle it.

### 3.3 Advanced Bulk Insert

In this example we will insert 2 rows in a more complex table. This will demonstrate using different kinds of data types, chained methods and getting the summary information at the end of a transaction.

You will need the following table in the test database to execute this:

Listing 8: example/advanced\_bulk\_insert.sql

```
1 CREATE TABLE `t2` (
2   `id` int(11) DEFAULT NULL,
3   `name` varchar(40) DEFAULT NULL,
4   `dob` date DEFAULT NULL,
5   `added` datetime DEFAULT NULL,
```

(continues on next page)

(continued from previous page)

```

6   `salary` decimal(9,2) DEFAULT NULL
7 ) ENGINE=Columnstore;

```

Listing 9: example/Advanced\_bulk\_insert.java

```

23 import com.mariadb.columnstore.api.*;
24
25 public class Advanced_bulk_insert {
26
27     public static void main(String[] args) {
28         try {
29             ColumnStoreDriver d = new ColumnStoreDriver();
30             ColumnStoreBulkInsert b = d.createBulkInsert("test", "t2", (short)0, 0);

```

As with the basic example we create an instance of the driver and use it to create a bulk insert instance.

Listing 10: example/Advanced\_bulk\_insert.java

```

31         b.setColumn(0, 1);
32         b.setColumn(1, "Andrew");
33         b.setColumn(2, "1936-12-24");
34         b.setColumn(3, "2017-07-07 15:14:12");
35         b.setColumn(4, "15239.45");
36         b.writeRow();

```

This demonstrates setting several different data types using strings of data. The second column (column 1) is a VARCHAR(40) and is set to “Andrew”. The third column is a DATE column and the API will automatically convert this into a binary date format before transmitting it to ColumnStore. The fourth is a DATETIME and the fifth a DECIMAL column which again the API will convert from the strings into the binary format.

Listing 11: example/Advanced\_bulk\_insert.java

```

37         b.setColumn(0, 2).setColumn(1, "David").setColumn(2, new
↳ ColumnStoreDateTime("1972-05-23", "%Y-%m-%d")).setColumn(3, new ColumnStoreDateTime(
↳ "2017-07-07 15:20:18", "%Y-%m-%d %H:%M:%S")).setColumn(4, new
↳ ColumnStoreDecimal(2347623, (short)2)).writeRow();

```

Many of the *ColumnStoreBulkInsert* methods return a pointer to the class and a return status which means multiple calls can be chained together in a similar way to ORM APIs. Here we use additional datatypes *ColumnStoreDateTime* and *ColumnStoreDecimal*.

*ColumnStoreDateTime* is used to create an entry for a DATE or DATETIME column. It can be used to define custom formats for dates and times using the *strptime* format.

A decimal is created using the *ColumnStoreDecimal* class. It can be set using a string, double or a pair of integers. The first integer is the precision and the second integer is the scale. So this number becomes 23476.23.

Listing 12: example/Advanced\_bulk\_insert.java

```

38         b.commit();
39         ColumnStoreSummary summary = b.getSummary();
40         System.out.println("Execution time: " + summary.getExecutionTime());
41         System.out.println("Rows inserted: " + summary.getRowsInsertedCount());
42         System.out.println("Truncation count: " + summary.getTruncationCount());
43         System.out.println("Saturated count: " + summary.getSaturatedCount());
44         System.out.println("Invalid count: " + summary.getInvalidCount());
45     }

```



After a commit or rollback we can obtain summary information from the bulk insert class. This is done using the `ColumnStoreBulkInsert.getSummary` method which will return a reference `ColumnStoreSummary` class. In this example we get the number of rows inserted (or would be inserted if there was a rollback) and the execution time from the moment the bulk insert class is created until the commit or rollback is complete.

Listing 13: example/Advanced\_bulk\_insert.java

```
46         catch (ColumnStoreException e) {
47             System.err.println("Error caught: " + e.getMessage());
48         }
49     }
50 }
```

At the end we clean up in the same way as the basic bulk insert example.

## COMPILING WITH JAVAMCSAPI

### 4.1 Pre-requisites

To link javamcsapi to your application and to execute it you first need install the following mcsapi pre-requisites:

#### 4.1.1 Ubuntu

```
sudo apt-get install libuv1
```

#### 4.1.2 CentOS 7

```
sudo yum install epel-release  
sudo yum install libuv
```

### 4.2 Compiling

To compile a Java program from command line you have to explicitly state where to find javamcsapi.jar. Its path can differ depending on your operating system.

#### 4.2.1 Ubuntu

```
javac -classpath ".:usr/lib/javamcsapi.jar" Basic_bulk_insert.java  
java -classpath ".:usr/lib/javamcsapi.jar" Basic_bulk_insert
```

#### 4.2.2 CentOS 7

```
javac -classpath ".:usr/lib64/javamcsapi.jar" Basic_bulk_insert.java  
java -classpath ".:usr/lib64/javamcsapi.jar" Basic_bulk_insert
```

### 4.2.3 Windows 10 (x64)

```
javac -classpath ".;%mcsapiInstallDir%\lib\java\javamcsapi-1.1.7.jar" Basic_bulk_
↳insert.java
java -classpath ".;%mcsapiInstallDir%\lib\java\javamcsapi-1.1.7.jar" -Djava.library.
↳path="%mcsapiInstallDir%\lib" Basic_bulk_insert
```

The variable `%mcsapiInstallDir%` represents the base installation directory of the Bulk Write SDK. (e.g. `C:\Program Files\MariaDB\ColumnStore Bulk Write SDK`)

If you don't want to change the `java.library.path` you can copy `javamcsapi`'s DLLs `libiconv.dll`, `libuv.dll`, `libxml2.dll`, `mcsapi.dll` and `javamcsapi.dll` from `%mcsapiInstallDir%\lib` to the directory of the Java class to execute. Another option is to just add `%mcsapiInstallDir%\lib` to your `PATH` environment variable, which is the default setting when you install the Bulk Write SDK.

## JAVAMCSAPI API REFERENCE

### 5.1 ColumnStoreBulkInsert

public class **ColumnStoreBulkInsert**

#### 5.1.1 Fields

**swigCMemOwn**

protected transient boolean **swigCMemOwn**

#### 5.1.2 Constructors

**ColumnStoreBulkInsert**

protected **ColumnStoreBulkInsert** (long *cPtr*, boolean *cMemoryOwn*)

#### 5.1.3 Methods

**addReference**

protected void **addReference** (*ColumnStoreDriver columnStoreDriver*)

**commit**

public void **commit** ()

**delete**

public synchronized void **delete** ()

**finalize**

protected void **finalize** ()

### **getCPtr**

protected static long **getCPtr** (*ColumnStoreBulkInsert obj*)

### **getColumnCount**

public int **getColumnCount** ()

### **getSummary**

public *ColumnStoreSummary* **getSummary** ()

### **isActive**

public boolean **isActive** ()

### **resetRow**

public *ColumnStoreBulkInsert* **resetRow** ()

### **rollback**

public void **rollback** ()

### **setBatchSize**

public void **setBatchSize** (long *batchSize*)

### **setColumn**

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *String value*, int[] *status*)

### **setColumn**

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *String value*)

### **setColumn**

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *String value*, long *length*, int[] *status*)

### **setColumn**

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *String value*, long *length*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, java.math.BigInteger *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, java.math.BigInteger *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, long *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, long *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, int *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, int *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, short *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, short *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, byte *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, byte *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, boolean *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, boolean *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, double *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, double *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *ColumnStoreDateTime* *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *ColumnStoreDateTime* *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *ColumnStoreTime* *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *ColumnStoreTime* *value*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *ColumnStoreDecimal* *value*, int[] *status*)

### setColumn

public *ColumnStoreBulkInsert* **setColumn** (int *columnNumber*, *ColumnStoreDecimal* *value*)

### setNull

public *ColumnStoreBulkInsert* **setNull** (int *columnNumber*, int[] *status*)

### setNull

public *ColumnStoreBulkInsert* **setNull** (int *columnNumber*)

### **setTruncateIsError**

public void **setTruncateIsError** (boolean *set*)

### **writeRow**

public *ColumnStoreBulkInsert* **writeRow** ()

## **5.2 ColumnStoreDateTime**

public class **ColumnStoreDateTime**

### **5.2.1 Fields**

#### **swigCMemOwn**

protected transient boolean **swigCMemOwn**

### **5.2.2 Constructors**

#### **ColumnStoreDateTime**

protected **ColumnStoreDateTime** (long *cPtr*, boolean *cMemoryOwn*)

#### **ColumnStoreDateTime**

public **ColumnStoreDateTime** ()

#### **ColumnStoreDateTime**

public **ColumnStoreDateTime** (SWIGTYPE\_p\_tm *time*)

#### **ColumnStoreDateTime**

public **ColumnStoreDateTime** (long *year*, long *month*, long *day*, long *hour*, long *minute*, long *second*, long *microsecond*)

#### **ColumnStoreDateTime**

public **ColumnStoreDateTime** (long *year*, long *month*, long *day*, long *hour*, long *minute*, long *second*)

#### **ColumnStoreDateTime**

public **ColumnStoreDateTime** (long *year*, long *month*, long *day*, long *hour*, long *minute*)



### ColumnStoreDateTime

public **ColumnStoreDateTime** (long *year*, long *month*, long *day*, long *hour*)

### ColumnStoreDateTime

public **ColumnStoreDateTime** (long *year*, long *month*, long *day*)

### ColumnStoreDateTime

public **ColumnStoreDateTime** (*String dateTime*, *String format*)

## 5.2.3 Methods

### delete

public synchronized void **delete** ()

### finalize

protected void **finalize** ()

### getCPtr

protected static long **getCPtr** (*ColumnStoreDateTime obj*)

### set

public boolean **set** (SWIGTYPE\_p\_tm *time*)

### set

public boolean **set** (*String dateTime*, *String format*)

## 5.3 ColumnStoreDecimal

public class **ColumnStoreDecimal**

### 5.3.1 Fields

#### swigCMemOwn

protected transient boolean **swigCMemOwn**

## 5.3.2 Constructors

### ColumnStoreDecimal

protected **ColumnStoreDecimal** (long *cPtr*, boolean *cMemoryOwn*)

### ColumnStoreDecimal

public **ColumnStoreDecimal** ()

### ColumnStoreDecimal

public **ColumnStoreDecimal** (long *value*)

### ColumnStoreDecimal

public **ColumnStoreDecimal** (String *value*)

### ColumnStoreDecimal

public **ColumnStoreDecimal** (double *value*)

### ColumnStoreDecimal

public **ColumnStoreDecimal** (long *number*, short *scale*)

## 5.3.3 Methods

### delete

public synchronized void **delete** ()

### finalize

protected void **finalize** ()

### getCPtr

protected static long **getCPtr** (*ColumnStoreDecimal obj*)

### set

public boolean **set** (long *value*)

### set

public boolean **set** (String *value*)

**set**

public boolean **set** (double *value*)

**set**

public boolean **set** (long *number*, short *scale*)

## 5.4 ColumnStoreDriver

public class **ColumnStoreDriver**

### 5.4.1 Fields

**swigCMemOwn**

protected transient boolean **swigCMemOwn**

### 5.4.2 Constructors

**ColumnStoreDriver**

protected **ColumnStoreDriver** (long *cPtr*, boolean *cMemoryOwn*)

**ColumnStoreDriver**

public **ColumnStoreDriver** (String *path*)

**ColumnStoreDriver**

public **ColumnStoreDriver** ()

### 5.4.3 Methods

**clearTableLock**

public void **clearTableLock** (java.math.BigInteger *lockId*)

**clearTableLock**

public void **clearTableLock** (TableLockInfo *tbi*)

**clearTableLock**

public void **clearTableLock** (String *db*, String *table*)

### **createBulkInsert**

public *ColumnStoreBulkInsert* **createBulkInsert** (String *db*, String *table*, short *mode*, int *pm*)

### **delete**

public synchronized void **delete** ()

### **finalize**

protected void **finalize** ()

### **getCPtr**

protected static long **getCPtr** (*ColumnStoreDriver* *obj*)

### **getJavaMcsapiVersion**

public String **getJavaMcsapiVersion** ()

### **getSystemCatalog**

public *ColumnStoreSystemCatalog* **getSystemCatalog** ()

### **getVersion**

public String **getVersion** ()

### **isTableLocked**

public boolean **isTableLocked** (String *db*, String *table*, *TableLockInfo* *rtm*)

### **isTableLocked**

public boolean **isTableLocked** (String *db*, String *table*)

### **listTableLocks**

public *TableLockInfoVector* **listTableLocks** ()

### **setDebug**

public void **setDebug** (short *level*)

## 5.5 ColumnStoreException

public class **ColumnStoreException** extends java.lang.RuntimeException

### 5.5.1 Constructors

#### ColumnStoreException

public **ColumnStoreException** ()

#### ColumnStoreException

public **ColumnStoreException** (*String message*)

#### ColumnStoreException

public **ColumnStoreException** (*String message, Throwable cause*)

#### ColumnStoreException

public **ColumnStoreException** (*Throwable cause*)

## 5.6 ColumnStoreSummary

public class **ColumnStoreSummary**

### 5.6.1 Fields

#### swigCMemOwn

protected transient boolean **swigCMemOwn**

### 5.6.2 Constructors

#### ColumnStoreSummary

protected **ColumnStoreSummary** (*long cPtr, boolean cMemoryOwn*)

#### ColumnStoreSummary

public **ColumnStoreSummary** ()

#### ColumnStoreSummary

public **ColumnStoreSummary** (*ColumnStoreSummary summary*)

### 5.6.3 Methods

#### **delete**

public synchronized void **delete** ()

#### **finalize**

protected void **finalize** ()

#### **getCPtr**

protected static long **getCPtr** (*ColumnStoreSummary obj*)

#### **getExecutionTime**

public double **getExecutionTime** ()

#### **getInvalidCount**

public java.math.BigInteger **getInvalidCount** ()

#### **getRowsInsertedCount**

public java.math.BigInteger **getRowsInsertedCount** ()

#### **getSaturatedCount**

public java.math.BigInteger **getSaturatedCount** ()

#### **getTruncationCount**

public java.math.BigInteger **getTruncationCount** ()

## 5.7 ColumnStoreSystemCatalog

public class **ColumnStoreSystemCatalog**

### 5.7.1 Fields

#### **swigCMemOwn**

protected transient boolean **swigCMemOwn**

## 5.7.2 Constructors

### ColumnStoreSystemCatalog

protected **ColumnStoreSystemCatalog** (long *cPtr*, boolean *cMemoryOwn*)

### ColumnStoreSystemCatalog

public **ColumnStoreSystemCatalog** ()

### ColumnStoreSystemCatalog

public **ColumnStoreSystemCatalog** (*ColumnStoreSystemCatalog obj*)

## 5.7.3 Methods

### addReference

protected void **addReference** (*ColumnStoreDriver columnStoreDriver*)

### delete

public synchronized void **delete** ()

### finalize

protected void **finalize** ()

### getCPtr

protected static long **getCPtr** (*ColumnStoreSystemCatalog obj*)

### getTable

public *ColumnStoreSystemCatalogTable* **getTable** (String *schemaName*, String *tableName*)

## 5.8 ColumnStoreSystemCatalogColumn

public class **ColumnStoreSystemCatalogColumn**

### 5.8.1 Fields

#### swigCMemOwn

protected transient boolean **swigCMemOwn**

## 5.8.2 Constructors

### ColumnStoreSystemCatalogColumn

protected `ColumnStoreSystemCatalogColumn` (long *cPtr*, boolean *cMemoryOwn*)

### ColumnStoreSystemCatalogColumn

public `ColumnStoreSystemCatalogColumn` ()

### ColumnStoreSystemCatalogColumn

public `ColumnStoreSystemCatalogColumn` (*ColumnStoreSystemCatalogColumn obj*)

## 5.8.3 Methods

### compressionType

public short `compressionType` ()

### delete

public synchronized void `delete` ()

### finalize

protected void `finalize` ()

### getCPtr

protected static long `getCPtr` (*ColumnStoreSystemCatalogColumn obj*)

### getColumnName

public `String` `getColumnName` ()

### getDefaultValue

public `String` `getDefaultValue` ()

### getDictionaryOID

public long `getDictionaryOID` ()

### getOID

public long `getOID` ()



### **getPosition**

public long **getPosition** ()

### **getPrecision**

public long **getPrecision** ()

### **getScale**

public long **getScale** ()

### **getType**

public *columnstore\_data\_types\_t* **getType** ()

### **getWidth**

public long **getWidth** ()

### **isAutoincrement**

public boolean **isAutoincrement** ()

### **isNullable**

public boolean **isNullable** ()

## **5.9 ColumnStoreSystemCatalogTable**

public class **ColumnStoreSystemCatalogTable**

### **5.9.1 Fields**

#### **swigCMemOwn**

protected transient boolean **swigCMemOwn**

### **5.9.2 Constructors**

#### **ColumnStoreSystemCatalogTable**

protected **ColumnStoreSystemCatalogTable** (long *cPtr*, boolean *cMemoryOwn*)

## ColumnStoreSystemCatalogTable

```
public ColumnStoreSystemCatalogTable ()
```

## ColumnStoreSystemCatalogTable

```
public ColumnStoreSystemCatalogTable (ColumnStoreSystemCatalogTable obj)
```

### 5.9.3 Methods

#### delete

```
public synchronized void delete ()
```

#### finalize

```
protected void finalize ()
```

#### getCPtr

```
protected static long getCPtr (ColumnStoreSystemCatalogTable obj)
```

#### getColumn

```
public ColumnStoreSystemCatalogColumn getColumn (String columnName)
```

#### getColumn

```
public ColumnStoreSystemCatalogColumn getColumn (int columnNumber)
```

#### getColumnCount

```
public int getColumnCount ()
```

#### getOID

```
public long getOID ()
```

#### getSchemaName

```
public String getSchemaName ()
```

#### getTableName

```
public String getTableName ()
```

## 5.10 ColumnStoreTime

public class **ColumnStoreTime**

### 5.10.1 Fields

**swigCMemOwn**

protected transient boolean **swigCMemOwn**

### 5.10.2 Constructors

**ColumnStoreTime**

protected **ColumnStoreTime** (long *cPtr*, boolean *cMemoryOwn*)

**ColumnStoreTime**

public **ColumnStoreTime** ()

**ColumnStoreTime**

public **ColumnStoreTime** (SWIGTYPE\_p\_tm *time*)

**ColumnStoreTime**

public **ColumnStoreTime** (int *hour*, long *minute*, long *second*, long *microsecond*, boolean *is\_negative*)

**ColumnStoreTime**

public **ColumnStoreTime** (int *hour*, long *minute*, long *second*, long *microsecond*)

**ColumnStoreTime**

public **ColumnStoreTime** (int *hour*, long *minute*, long *second*)

**ColumnStoreTime**

public **ColumnStoreTime** (String *time*, String *format*)

### 5.10.3 Methods

**delete**

public synchronized void **delete** ()

## finalize

protected void **finalize** ()

## getCPtr

protected static long **getCPtr** (*ColumnStoreTime obj*)

## set

public boolean **set** (SWIGTYPE\_p\_tm *time*)

## set

public boolean **set** (*String time, String format*)

## 5.11 TableLockInfo

public class **TableLockInfo**

### 5.11.1 Fields

#### swigCMemOwn

protected transient boolean **swigCMemOwn**

### 5.11.2 Constructors

#### TableLockInfo

protected **TableLockInfo** (long *cPtr*, boolean *cMemoryOwn*)

#### TableLockInfo

public **TableLockInfo** ()

### 5.11.3 Methods

#### delete

public synchronized void **delete** ()

#### finalize

protected void **finalize** ()

### **getCPtr**

protected static long **getCPtr** (*TableLockInfo obj*)

### **getCreationTime**

public SWIGTYPE\_p\_time\_t **getCreationTime** ()

### **getDbrootList**

public SWIGTYPE\_p\_std\_\_vectorT\_unsigned\_int\_t **getDbrootList** ()

### **getId**

public java.math.BigInteger **getId** ()

### **getOwnerName**

public String **getOwnerName** ()

### **getOwnerPID**

public long **getOwnerPID** ()

### **getOwnerSessionID**

public long **getOwnerSessionID** ()

### **getOwnerTxnID**

public long **getOwnerTxnID** ()

### **getState**

public *columnstore\_lock\_types\_t* **getState** ()

### **getTableOID**

public long **getTableOID** ()

### **setCreationTime**

public void **setCreationTime** (SWIGTYPE\_p\_time\_t *value*)

### **setDbrootList**

public void **setDbrootList** (SWIGTYPE\_p\_std\_\_vectorT\_unsigned\_int\_t *value*)

### **setId**

public void **setId** (java.math.BigInteger *value*)

### **setOwnerName**

public void **setOwnerName** (String *value*)

### **setOwnerPID**

public void **setOwnerPID** (long *value*)

### **setOwnerSessionID**

public void **setOwnerSessionID** (long *value*)

### **setOwnerTxnID**

public void **setOwnerTxnID** (long *value*)

### **setState**

public void **setState** (*columnstore\_lock\_types\_t value*)

### **setTableOID**

public void **setTableOID** (long *value*)

## **5.12 TableLockInfoVector**

public class **TableLockInfoVector**

### **5.12.1 Fields**

#### **swigCMemOwn**

protected transient boolean **swigCMemOwn**

## 5.12.2 Constructors

### TableLockInfoVector

protected **TableLockInfoVector** (long *cPtr*, boolean *cMemoryOwn*)

### TableLockInfoVector

public **TableLockInfoVector** ()

### TableLockInfoVector

public **TableLockInfoVector** (long *n*)

## 5.12.3 Methods

### add

public void **add** (*TableLockInfo x*)

### capacity

public long **capacity** ()

### clear

public void **clear** ()

### delete

public synchronized void **delete** ()

### finalize

protected void **finalize** ()

### get

public *TableLockInfo* **get** (int *i*)

### getCPtr

protected static long **getCPtr** (*TableLockInfoVector obj*)

### isEmpty

public boolean **isEmpty** ()

## **reserve**

public void **reserve** (long *n*)

## **set**

public void **set** (int *i*, *TableLockInfo val*)

## **size**

public long **size** ()

## **5.13 columnstore\_data\_convert\_status\_t**

public enum **columnstore\_data\_convert\_status\_t**

### **5.13.1 Enum Constants**

#### **CONVERT\_STATUS\_INVALID**

public static final *columnstore\_data\_convert\_status\_t* **CONVERT\_STATUS\_INVALID**

#### **CONVERT\_STATUS\_NONE**

public static final *columnstore\_data\_convert\_status\_t* **CONVERT\_STATUS\_NONE**

#### **CONVERT\_STATUS\_SATURATED**

public static final *columnstore\_data\_convert\_status\_t* **CONVERT\_STATUS\_SATURATED**

#### **CONVERT\_STATUS\_TRUNCATED**

public static final *columnstore\_data\_convert\_status\_t* **CONVERT\_STATUS\_TRUNCATED**

## **5.14 columnstore\_data\_types\_t**

public enum **columnstore\_data\_types\_t**

### **5.14.1 Enum Constants**

#### **DATA\_TYPE\_BIGINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_BIGINT**



#### **DATA\_TYPE\_BIT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_BIT**

#### **DATA\_TYPE\_BLOB**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_BLOB**

#### **DATA\_TYPE\_CHAR**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_CHAR**

#### **DATA\_TYPE\_CLOB**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_CLOB**

#### **DATA\_TYPE\_DATE**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_DATE**

#### **DATA\_TYPE\_DATETIME**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_DATETIME**

#### **DATA\_TYPE\_DECIMAL**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_DECIMAL**

#### **DATA\_TYPE\_DOUBLE**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_DOUBLE**

#### **DATA\_TYPE\_FLOAT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_FLOAT**

#### **DATA\_TYPE\_INT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_INT**

#### **DATA\_TYPE\_MEDINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_MEDINT**

#### **DATA\_TYPE\_SMALLINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_SMALLINT**

#### **DATA\_TYPE\_TEXT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_TEXT**

#### **DATA\_TYPE\_TIME**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_TIME**

#### **DATA\_TYPE\_TINYINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_TINYINT**

#### **DATA\_TYPE\_UBIGINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UBIGINT**

#### **DATA\_TYPE\_UDECIMAL**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UDECIMAL**

#### **DATA\_TYPE\_UDOUBLE**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UDOUBLE**

#### **DATA\_TYPE\_UFLOAT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UFLOAT**

#### **DATA\_TYPE\_UINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UINT**

#### **DATA\_TYPE\_UMEDINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UMEDINT**

#### **DATA\_TYPE\_USMALLINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_USMALLINT**

## **DATA\_TYPE\_UTINYINT**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_UTINYINT**

## **DATA\_TYPE\_VARBINARY**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_VARBINARY**

## **DATA\_TYPE\_VARCHAR**

public static final *columnstore\_data\_types\_t* **DATA\_TYPE\_VARCHAR**

## **5.15 columnstore\_lock\_types\_t**

public enum **columnstore\_lock\_types\_t**

### **5.15.1 Enum Constants**

#### **LOCK\_TYPE\_CLEANUP**

public static final *columnstore\_lock\_types\_t* **LOCK\_TYPE\_CLEANUP**

#### **LOCK\_TYPE\_LOADING**

public static final *columnstore\_lock\_types\_t* **LOCK\_TYPE\_LOADING**

## A

add(TableLockInfo) (*Java method*), 29  
 addReference(ColumnStoreDriver) (*Java method*), 10, 21

## C

capacity() (*Java method*), 29  
 clear() (*Java method*), 29  
 clearTableLock(java.math.BigInteger) (*Java method*), 17  
 clearTableLock(String, String) (*Java method*), 17  
 clearTableLock(TableLockInfo) (*Java method*), 17  
 columnstore\_data\_convert\_status\_t (*Java enum*), 30  
 columnstore\_data\_types\_t (*Java enum*), 30  
 COLUMNSTORE\_INSTALL\_DIR, 4  
 columnstore\_lock\_types\_t (*Java enum*), 33  
 ColumnStoreBulkInsert (*Java class*), 10  
 ColumnStoreBulkInsert(long, boolean) (*Java constructor*), 10  
 ColumnStoreDateTime (*Java class*), 14  
 ColumnStoreDateTime() (*Java constructor*), 14  
 ColumnStoreDateTime(long, boolean) (*Java constructor*), 14  
 ColumnStoreDateTime(long, long, long) (*Java constructor*), 15  
 ColumnStoreDateTime(long, long, long, long) (*Java constructor*), 15  
 ColumnStoreDateTime(long, long, long, long, long) (*Java constructor*), 14  
 ColumnStoreDateTime(long, long, long, long, long, long) (*Java constructor*), 14  
 ColumnStoreDateTime(long, long, long, long, long, long, long) (*Java constructor*), 14  
 ColumnStoreDateTime(String, String) (*Java constructor*), 15  
 ColumnStoreDateTime(SWIGTYPE\_p\_tm) (*Java constructor*), 14

ColumnStoreDecimal (*Java class*), 15  
 ColumnStoreDecimal() (*Java constructor*), 16  
 ColumnStoreDecimal(double) (*Java constructor*), 16  
 ColumnStoreDecimal(long) (*Java constructor*), 16  
 ColumnStoreDecimal(long, boolean) (*Java constructor*), 16  
 ColumnStoreDecimal(long, short) (*Java constructor*), 16  
 ColumnStoreDecimal(String) (*Java constructor*), 16  
 ColumnStoreDriver (*Java class*), 17  
 ColumnStoreDriver() (*Java constructor*), 17  
 ColumnStoreDriver(long, boolean) (*Java constructor*), 17  
 ColumnStoreDriver(String) (*Java constructor*), 17  
 ColumnStoreException (*Java class*), 19  
 ColumnStoreException() (*Java constructor*), 19  
 ColumnStoreException(String) (*Java constructor*), 19  
 ColumnStoreException(String, Throwable) (*Java constructor*), 19  
 ColumnStoreException(Throwable) (*Java constructor*), 19  
 ColumnStoreSummary (*Java class*), 19  
 ColumnStoreSummary() (*Java constructor*), 19  
 ColumnStoreSummary(ColumnStoreSummary) (*Java constructor*), 19  
 ColumnStoreSummary(long, boolean) (*Java constructor*), 19  
 ColumnStoreSystemCatalog (*Java class*), 20  
 ColumnStoreSystemCatalog() (*Java constructor*), 21  
 ColumnStoreSystemCatalog(ColumnStoreSystemCatalog) (*Java constructor*), 21  
 ColumnStoreSystemCatalog(long, boolean) (*Java constructor*), 21  
 ColumnStoreSystemCatalogColumn (*Java class*), 21  
 ColumnStoreSystemCatalogColumn() (*Java*

- constructor*), 22
  - ColumnStoreSystemCatalogColumn (ColumnStoreSystemCatalogColumn (Java constructor), 22
  - ColumnStoreSystemCatalogColumn (long, boolean) (Java constructor), 22
  - ColumnStoreSystemCatalogTable (Java class), 23
  - ColumnStoreSystemCatalogTable () (Java constructor), 24
  - ColumnStoreSystemCatalogTable (ColumnStoreSystemCatalogColumn (Java constructor), 24
  - ColumnStoreSystemCatalogTable (long, boolean) (Java constructor), 23
  - ColumnStoreTime (Java class), 25
  - ColumnStoreTime () (Java constructor), 25
  - ColumnStoreTime (int, long, long) (Java constructor), 25
  - ColumnStoreTime (int, long, long, long) (Java constructor), 25
  - ColumnStoreTime (int, long, long, long, boolean) (Java constructor), 25
  - ColumnStoreTime (long, boolean) (Java constructor), 25
  - ColumnStoreTime (String, String) (Java constructor), 25
  - ColumnStoreTime (SWIGTYPE\_p\_tm) (Java constructor), 25
  - com.mariadb.columnstore.api (package), 10
  - commit () (Java method), 10
  - compressionType () (Java method), 22
  - CONVERT\_STATUS\_INVALID (Java field), 30
  - CONVERT\_STATUS\_NONE (Java field), 30
  - CONVERT\_STATUS\_SATURATED (Java field), 30
  - CONVERT\_STATUS\_TRUNCATED (Java field), 30
  - createBulkInsert (String, String, short, int) (Java method), 18
- D**
- DATA\_TYPE\_BIGINT (Java field), 30
  - DATA\_TYPE\_BIT (Java field), 31
  - DATA\_TYPE\_BLOB (Java field), 31
  - DATA\_TYPE\_CHAR (Java field), 31
  - DATA\_TYPE\_CLOB (Java field), 31
  - DATA\_TYPE\_DATE (Java field), 31
  - DATA\_TYPE\_DATETIME (Java field), 31
  - DATA\_TYPE\_DECIMAL (Java field), 31
  - DATA\_TYPE\_DOUBLE (Java field), 31
  - DATA\_TYPE\_FLOAT (Java field), 31
  - DATA\_TYPE\_INT (Java field), 31
  - DATA\_TYPE\_MEDINT (Java field), 31
  - DATA\_TYPE\_SMALLINT (Java field), 32
  - DATA\_TYPE\_TEXT (Java field), 32
  - DATA\_TYPE\_TIME (Java field), 32
  - DATA\_TYPE\_TINYINT (Java field), 32
  - DATA\_TYPE\_UBIGINT (Java field), 32
  - DATA\_TYPE\_UDECIMAL (Java field), 32
  - DATA\_TYPE\_UDOUBLE (Java field), 32
  - DATA\_TYPE\_UFLOAT (Java field), 32
  - DATA\_TYPE\_UINT (Java field), 32
  - DATA\_TYPE\_UMEDINT (Java field), 32
  - DATA\_TYPE\_USMALLINT (Java field), 32
  - DATA\_TYPE\_UTINYINT (Java field), 33
  - DATA\_TYPE\_VARBINARY (Java field), 33
  - DATA\_TYPE\_VARCHAR (Java field), 33
  - delete () (Java method), 10, 15, 16, 18, 20–22, 24–26, 29
- E**
- environment variable
    - COLUMNSTORE\_INSTALL\_DIR, 4
- F**
- finalize () (Java method), 10, 15, 16, 18, 20–22, 24, 26, 29
- G**
- get (int) (Java method), 29
  - getColumn (int) (Java method), 24
  - getColumn (String) (Java method), 24
  - getColumnCount () (Java method), 11, 24
  - getColumnName () (Java method), 22
  - getCPtr (ColumnStoreBulkInsert) (Java method), 11
  - getCPtr (ColumnStoreDateTime) (Java method), 15
  - getCPtr (ColumnStoreDecimal) (Java method), 16
  - getCPtr (ColumnStoreDriver) (Java method), 18
  - getCPtr (ColumnStoreSummary) (Java method), 20
  - getCPtr (ColumnStoreSystemCatalog) (Java method), 21
  - getCPtr (ColumnStoreSystemCatalogColumn) (Java method), 22
  - getCPtr (ColumnStoreSystemCatalogTable) (Java method), 24
  - getCPtr (ColumnStoreTime) (Java method), 26
  - getCPtr (TableLockInfo) (Java method), 27
  - getCPtr (TableLockInfoVector) (Java method), 29
  - getCreationTime () (Java method), 27
  - getDbrootList () (Java method), 27
  - getDefaultValue () (Java method), 22
  - getDictionaryOID () (Java method), 22
  - getExecutionTime () (Java method), 20
  - getId () (Java method), 27
  - getInvalidCount () (Java method), 20
  - getJavaMcsapiVersion () (Java method), 18

getOID() (*Java method*), 22, 24  
 getOwnerName() (*Java method*), 27  
 getOwnerPID() (*Java method*), 27  
 getOwnerSessionID() (*Java method*), 27  
 getOwnerTxnID() (*Java method*), 27  
 getPosition() (*Java method*), 23  
 getPrecision() (*Java method*), 23  
 getRowsInsertedCount() (*Java method*), 20  
 getSaturatedCount() (*Java method*), 20  
 getScale() (*Java method*), 23  
 getSchemaName() (*Java method*), 24  
 getState() (*Java method*), 27  
 getSummary() (*Java method*), 11  
 getSystemCatalog() (*Java method*), 18  
 getTable(String, String) (*Java method*), 21  
 getTableName() (*Java method*), 24  
 getTableOID() (*Java method*), 27  
 getTruncationCount() (*Java method*), 20  
 getType() (*Java method*), 23  
 getVersion() (*Java method*), 18  
 getWidth() (*Java method*), 23

## I

isActive() (*Java method*), 11  
 isAutoincrement() (*Java method*), 23  
 isEmpty() (*Java method*), 29  
 isNullable() (*Java method*), 23  
 isTableLocked(String, String) (*Java method*), 18  
 isTableLocked(String, String, TableLockInfo) (*Java method*), 18

## L

listTableLocks() (*Java method*), 18  
 LOCK\_TYPE\_CLEANUP (*Java field*), 33  
 LOCK\_TYPE\_LOADING (*Java field*), 33

## R

reserve(long) (*Java method*), 30  
 resetRow() (*Java method*), 11  
 rollback() (*Java method*), 11

## S

set(double) (*Java method*), 17  
 set(int, TableLockInfo) (*Java method*), 30  
 set(long) (*Java method*), 16  
 set(long, short) (*Java method*), 17  
 set(String) (*Java method*), 16  
 set(String, String) (*Java method*), 15, 26  
 set(SWIGTYPE\_p\_tm) (*Java method*), 15, 26  
 setBatchSize(long) (*Java method*), 11  
 setColumn(int, boolean) (*Java method*), 13  
 setColumn(int, boolean, int[]) (*Java method*), 12

setColumn(int, byte) (*Java method*), 12  
 setColumn(int, byte, int[]) (*Java method*), 12  
 setColumn(int, ColumnStoreDateTime) (*Java method*), 13  
 setColumn(int, ColumnStoreDateTime, int[]) (*Java method*), 13  
 setColumn(int, ColumnStoreDecimal) (*Java method*), 13  
 setColumn(int, ColumnStoreDecimal, int[]) (*Java method*), 13  
 setColumn(int, ColumnStoreTime) (*Java method*), 13  
 setColumn(int, ColumnStoreTime, int[]) (*Java method*), 13  
 setColumn(int, double) (*Java method*), 13  
 setColumn(int, double, int[]) (*Java method*), 13  
 setColumn(int, int) (*Java method*), 12  
 setColumn(int, int, int[]) (*Java method*), 12  
 setColumn(int, java.math.BigInteger) (*Java method*), 12  
 setColumn(int, java.math.BigInteger, int[]) (*Java method*), 12  
 setColumn(int, long) (*Java method*), 12  
 setColumn(int, long, int[]) (*Java method*), 12  
 setColumn(int, short) (*Java method*), 12  
 setColumn(int, short, int[]) (*Java method*), 12  
 setColumn(int, String) (*Java method*), 11  
 setColumn(int, String, int[]) (*Java method*), 11  
 setColumn(int, String, long) (*Java method*), 11  
 setColumn(int, String, long, int[]) (*Java method*), 11  
 setCreationTime(SWIGTYPE\_p\_time\_t) (*Java method*), 27  
 setDbrootList(SWIGTYPE\_p\_std\_vectorT\_unsigned\_int) (*Java method*), 28  
 setDebug(short) (*Java method*), 18  
 setId(java.math.BigInteger) (*Java method*), 28  
 setNull(int) (*Java method*), 13  
 setNull(int, int[]) (*Java method*), 13  
 setOwnerName(String) (*Java method*), 28  
 setOwnerPID(long) (*Java method*), 28  
 setOwnerSessionID(long) (*Java method*), 28  
 setOwnerTxnID(long) (*Java method*), 28  
 setState(columnstore\_lock\_types\_t) (*Java method*), 28  
 setTableOID(long) (*Java method*), 28  
 setTruncateIsError(boolean) (*Java method*),

14

size() (*Java method*), 30

swigCMemOwn (*Java field*), 10, 14, 15, 17, 19–21, 23,  
25, 26, 28

## T

TableLockInfo (*Java class*), 26

TableLockInfo() (*Java constructor*), 26

TableLockInfo(long, boolean) (*Java constructor*), 26

TableLockInfoVector (*Java class*), 28

TableLockInfoVector() (*Java constructor*), 29

TableLockInfoVector(long) (*Java constructor*),  
29

TableLockInfoVector(long, boolean) (*Java constructor*), 29

## W

writeRow() (*Java method*), 14