



# SQream Connector Native C++

SQream Technologies

Version 1.2.0

**Copyright © 2010-2019. All rights reserved.**

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchant- ability or fitness for a particular purpose.

We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document.

This document may not be reproduced in any form, for any purpose, without our prior written permission.

# Table of Contents

<b>Table of Contents</b>	<b>3</b>
<b>SQream Connector Native C++</b>	<b>4</b>
The SQream Native C++ Connector - Overview	4
API Reference	4
Connection	4
Statement	5
Helper functions	5
High level protocol functions	7
Code Samples	9
Import and establish a connection	9
Code Samples	9
Import and establish a connection	9
Run a query - Create a table	9
Run a query - Insert values into table	9
Run a query - Get column values from table	10
Run a query - Use bulk insert to insert large amount of data in a programmatic way	10
Run a query - Starting and finishing	11

# SQream Connector Native C++

Version 1.2.0

## The SQream Native C++ Connector - Overview

- This guide describes the implementation of the SQream Native C++ connector and is designed for SQream DB administrators and developers.
- The SQream Native C++ connector gives structures to initialize a connection, run SQL queries through the connection (statements), enables network streaming (insert, select).
- SQream connector protocol version: 7

## API Reference

To use the functions include the connector main .h file ("SQream-cpp-connector.h"). This will give the application access to the interfaces. It is also needed later to possess the .so of the library in a way that the using program can compile and link to it.

The connector functions are situated in the name space "sqream::driver::" and for the rest of the documentation all the functions are assumed to be called using this name space.

## Connection

Table 1. Initializing and closing connections

Function	Description
<code>driver()</code>	The constructor creating the handle to the connection
<code>connect(const std::string &amp;ipv4,int port,bool ssl,const std::string &amp;username,const std::string &amp;password,const std::string &amp;database),std::string &amp;service)</code>	Connects the handle to the sqream, accessing its database ipv4 - IP address as a string. port - port number SQream is listening on. ssl - True / false. If true, connect to SQream using SSL port.

Function	Description
	username, password - connection credentials. default is 'sqream' for both. database - name of database to connect to. service - name of the service to connect to.
disconnect()	Close the connection handle and reset the driver

## Statement

Table 2. Statement execution

Function	Description
new_query(const std::string &sql_query)	Starts a new statement resetting any statement related datas and follows it by Prepare
execute_query()	Executes the statement of the current statement. Needs to be called after new_query () (This advance the SQL query from Prepare -> Execute).
next_query_row (const size_t min_put_size=CONSTS::MIN_PUT_SIZE)	On an insert query - start setting the next row for insertion. SQream does not support partial inserts. If min_put_size is set, the connector will flush and put the rows if the buffer size is bigger than this value. If the function is called without argument a default value of 67108864 bytes (64 MiB) is set for comparison with the buffer size. On a select query - move to next row index to start selecting items from various columns using get() functions, the min_put_size argument has no influence on select.
finish_query()	Closes the statement

## Helper functions

**Table 3. Additional functions in order to accomplish a broader amount of operations or give access to some internal numbers. Their namespace is "sqream::" only**

Function	Description
<code>new_query_execute</code> (sqream::driver *drv, std::string sql_query)	Operates the protocol until the statement is executed, this function is a shortcut and set_types or get_types functions can be executed right after
<code>void run_direct_query</code> (sqream::driver *drv, std::string sql_query)	Executes a query from start to end (closes the statement). This function is a shortcut for when you don't need to see any input/output like a DDL type of query ("create table" etc)
<code>std::vector&lt;sqream::column&gt;</code> <code>get_metadata(driver *drv)</code>	Returns the metadata of the current statement if available (after a Prepare) or else an empty vector
<code>uint32_t retrieve_statement_id</code> (sqream::driver *drv)	Returns the current statement id, will throw an error if no statement is executed
<code>sqream::CONSTS::statement_type</code> <code>retrieve_statement_type</code> (sqream::driver *drv)	Returns the enum representing the type of statement currently executing. A statement can be a select, an insert or a DML (direct)
<code>uint32_t date</code> (int32_t year, int32_t month, int32_t day)	Convert date to sqream date (Julian day number, stored in an uint)
<code>uint64_t datetime</code> (int32_t year, int32_t month, int32_t day, int32_t hour, int32_t minute, int32_t second, int32_t millisecond)	Convert datetime to sqream datetime (Julian day number, stored in an uint in the high 32 bits and Julian day fraction stored in an uint in the low 32 bits)
<code>sqream::date_t make_date</code> (uint32_t date)	Convert sqream date to sqream::date_t structure
<code>sqream::datetime_t make_datetime</code> (uint64_t datetime)	Convert sqream datetime to sqream::datetime_t structure

## High level protocol functions

*Table 4. Retrieve results from a select query by column index*

Function	Description
<code>is_null(size_t col_id)</code>	Check whether the value in column index <code>col_id</code> is a null
<code>get_bool(size_t col_id)</code>	Get bool value from column index <code>col_id</code> at the current row
<code>get_ubyte(size_t col_id)</code>	Get <code>uint8_t</code> value from column index <code>col_id</code> at the current row
<code>get_short(size_t col_id)</code>	Get <code>int16_t</code> value from column index <code>col_id</code> at the current row
<code>get_int(size_t col_id)</code>	Get <code>int32_t</code> value from column index <code>col_id</code> at the current row
<code>get_long(size_t col_id)</code>	Get <code>int64_t</code> value from column index <code>col_id</code> at the current row
<code>get_float(size_t col_id)</code>	Get float value from column index <code>col_id</code> at the current row
<code>get_double(size_t col_id)</code>	Get double value from column index <code>col_id</code> at the current row
<code>get_date(size_t col_id)</code>	Get <code>uint32_t</code> value from column index <code>col_id</code> at the current row
<code>get_datetime(size_t col_id)</code>	Get <code>uint64_t</code> value from column index <code>col_id</code> at the current row
<code>get_varchar(size_t col_id)</code>	Get string value from column index <code>col_id</code> at the current row
<code>get_nvarchar(size_t col_id)</code>	Get string value from column index <code>col_id</code> at the current row

*Table 5. Retrieve results from a select query by column name*

Function	Description
<code>is_null(string col_name)</code>	Check whether the value in column named <code>col_name</code> is a null
<code>get_bool(string col_name)</code>	Get Boolean value from column named <code>col_name</code> at the current row
<code>get_ubyte(String col_name)</code>	Get <code>UByte</code> value from column named <code>col_name</code> at the current row
<code>get_short(string col_name)</code>	Get <code>Short</code> value from column named <code>col_name</code> at the current row
<code>get_int(string col_name)</code>	Get <code>Int</code> value from column named <code>col_name</code> at the current row
<code>get_long(string col_name)</code>	Get <code>Long</code> value from column named <code>col_name</code> at the current row

Function	Description
<code>col_name)</code>	at the current row
<code>get_float(string col_name)</code>	Get Float value from column named <code>col_name</code> at the current row
<code>get_double(string col_name)</code>	Get Double value from column named <code>col_name</code> at the current row
<code>get_date(string col_name)</code>	Get Date value from column named <code>col_name</code> at the current row
<code>get_datetime(string col_name)</code>	Get Datetime value from column named <code>col_name</code> at the current row
<code>get_varchar(string col_name)</code>	Get Varchar value from column named <code>col_name</code> at the current row
<code>get_nvarchar(string col_name)</code>	Get Nvarchar value from column named <code>col_name</code> at the current row

**Table 6. Set data by index following a bulk insert query**

Function	Description
<code>set_null(size_t col)</code>	Set column at index <code>col</code> in the current row to null
<code>set_bool(size_t col, bool val)</code>	Set column at index <code>col</code> of type Boolean in the current row
<code>set_ubyte(size_t col, uint8_t val)</code>	Set column at index <code>col</code> of type UByte in the current row - unsigned bytes only
<code>set_short(size_t col, uint16_t val)</code>	Set column at index <code>col</code> of type Short in the current row
<code>set_int(size_t col, uint32_t val)</code>	Set column at index <code>col</code> of type Int in the current row
<code>set_long(size_t col, uint64_t val)</code>	Set column at index <code>col</code> of type Long in the current row
<code>set_float(size_t col, float val)</code>	Set column at index <code>col</code> of type Float in the current row
<code>set_double(size_t col, double val)</code>	Set column at index <code>col</code> of type Double in the current row
<code>set_date(size_t col, uint32_t val)</code>	Set column at index <code>col</code> of type Date in the current row
<code>set_datetime(size_t col, uint64_t val)</code>	Set column at index <code>col</code> of type Datetime in the current row
<code>set_varchar(size_t col, string val)</code>	Set column at index <code>col</code> of type Varchar in the current row
<code>set_nvarchar(size_t col, string val)</code>	Set column at index <code>col</code> of type Nvarchar in the current row



## Code Samples

### Import and establish a connection

#### Example

```
#include "SQream-cpp-connector.h"

// Connection parameters: IP, Port, Database, Username, Password
sqream::driver sqc;
sqc.connect("127.0.0.1", 5000, false, "sqream", "sqream",
"master");
```

## Code Samples

### Import and establish a connection

#### Example

```
#include "SQream-cpp-connector.h"

// Connection parameters: IP, Port, Database, Username, Password
sqream::driver sqc;
sqc.connect("127.0.0.1", 5000, false, "sqream", "sqream",
"master");
```

### Run a query - Create a table

#### Example

```
string statement = "create or replace table table_name (int_
column int)";
sqc.new_query(statement);
sqc.execute_query();
sqc.finish_query();

OR

run_direct_query(&sqc, "create or replace table table_name (int_
column int)");
```

### Run a query - Insert values into table

#### Example

```
string statement = "insert into table_name(int_column) values
(5), (6), (7), (8)";
sqc.new_query(statement);
sqc.execute_query();
```

```

sqc.finish_query();

OR

run_direct_query(&sqc, "insert into table_name(int_column) values
(5), (6), (7), (8)");

```

## Run a query - Get column values from table

### Example

```

// Retrieve data
string statement = "select int_column from table_name";
sqc.new_query(statement);
sqc.execute_query();

// Pull out the actual data row by row
while (sqc.next_query_row())
    cout << "Received: " << sqc.get_int(0) << endl;
sqc.finish_query();

OR

new_query_execute(&sqc, "select int_column from table_name");
while (sqc.next_query_row())
    cout << "Received: " << sqc.get_int(0) << endl;
sqc.finish_query();

```

## Run a query - Use bulk insert to insert large amount of data in a programmatic way

### Example

```

// Example of classic Set data loop, using network streaming (also
called Network Insert)
int len = 3;
int row1[len] = {1,2,3};
string row2[len] = {"s1","s2","s3"};

string statement = "create or replace table table_name (int_
column int, varchar_column varchar(10))";
run_direct_query(&sqc, statement);

// each interrogation symbol represents a column to which the
network insertion can push
statement = "insert into table_name(int_column, varchar_column)
values(?, ?)";
sqc.new_query(statement);
sqc.execute_query();

```

```
for (int idx = 0; idx < len; idx++) {
    sqc.set_int(0, row1[idx]);           // put a value at column 0 of
the table
    sqc.set_varchar(1, row2[idx]);      // put a value at column 1 of
the table

    sqc.next_query_row();
}

sqc.finish_query();
```

## Run a query - Starting and finishing

### Example

```
//Initialization - Termination Example
#include "SQream-cpp-connector.h"

void Query() {
    // Connection parameters: IP, Port, Database, Username, Password
    sqream::driver sqc;
    sqc.connect("127.0.0.1", 5000, false, "sqream", "sqream",
"master");
    string statement = SQL_STATEMENT;
    new_query_execute(&sqc, statement);

    //.
    //. Do something
    //.

    // closes the statement (to do after execute + necessary
fetch/put to close the
    // statement and be able to open another one through prepare())
    sqc.finish_query();

    // closes the connection completely, destroying the socket, a call
to "connect(..)"
    // needs to be done do continue using this "driver sqc" object
    sqc.disconnect();
}
```