Jetter automation

How-to Manual

SQL4automation Library

We automate your success.

This document has been compiled by Jetter AG with due diligence, and based on the known state of the art. Revisions and further development of our products are not automatically mentioned in a reviewed document. Jetter AG shall not be liable for errors in form or content, or for missing updates, as well as for damages or disadvantages resulting from such failure.

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg
Germany

Phone

Switchboard	+49 7141 2550-0
Sales	+49 7141 2550-531
Technical Hotline	+49 7141 2550-444

E-mail

Technical Hotline	hotline@jetter.de
Sales	sales@jetter.de

Translation of the original User Manual

Revision	1.00
Date of issue	3/19/2021

Table of Contents

1 Introduction

1.1 Typographical conventions

This manual uses different typographical effects to support you in finding and classifying information. Below, there is an example of a step-by-step instruction:

✓ This symbol indicates requirements which have to be met before executing the following action.

► This sign or a numbering at the beginning of a paragraph marks an action instruction that must be executed by the user. Execute the instructions one after the other.

⇨ The target after a list of instructions indicates reactions to, or results of these actions.

ⓘ INFO

Further information and practical tips

In the info box you will find helpful information and practical tips about your product.

1.2 Product description

This document describes how to integrate and use the SQL4automation library in a JetSym STX project and how to use and adapt the attached sample project for own applications.

For more information on SQL4automation, visit the homepage:
https://www.sql4automation.com

2 Including the library in an STX project

The SQL4automation library must first be added to JetSym via the library manager and then included in a project. The procedure is described below.

2.1 Adding the library to JetSym

1. Copy the .libpackage file to any location on your PC.
2. Open JetSym and create an STX project.
3. Open the Library Manager by clicking on "Tools" and then on "Library Manager".
4. Click "Add" and select the file SQL4Automation_Library_1.0.00.00_Beta_00.libdesc on your PC.
5. Click on "Open".

⇨ The library is added to your project.

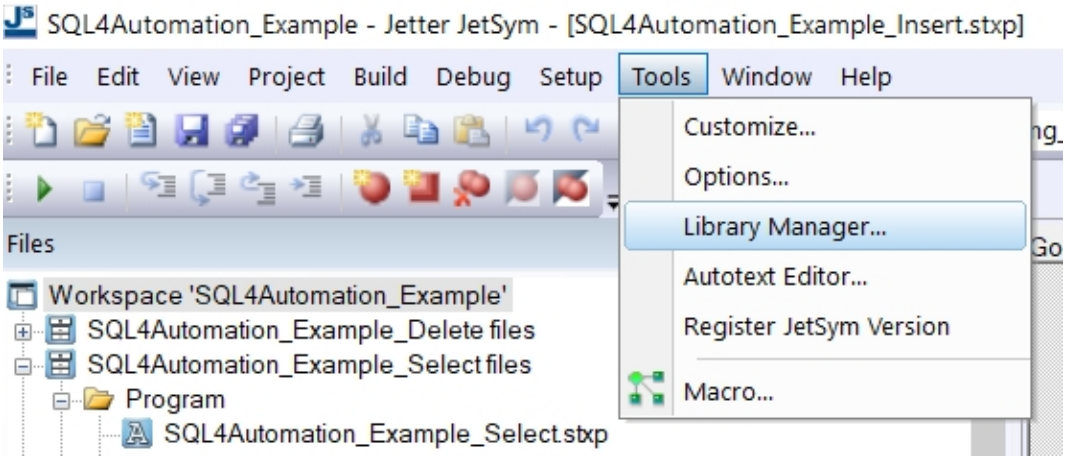

Fig. 1: Library Manager

2.2 Including the library in and STX project

1. Open or create an STX project (e.g. the sample project [▶ 8] "SQL4Automation_example.wsw").
2. Select the "Files" tab.
3. Right-click on the entry "Library" and select "Add libraries…".
   ⇨ The Library Manager opens.
4. Select the latest version of the installed library.

⇨ The library is included.

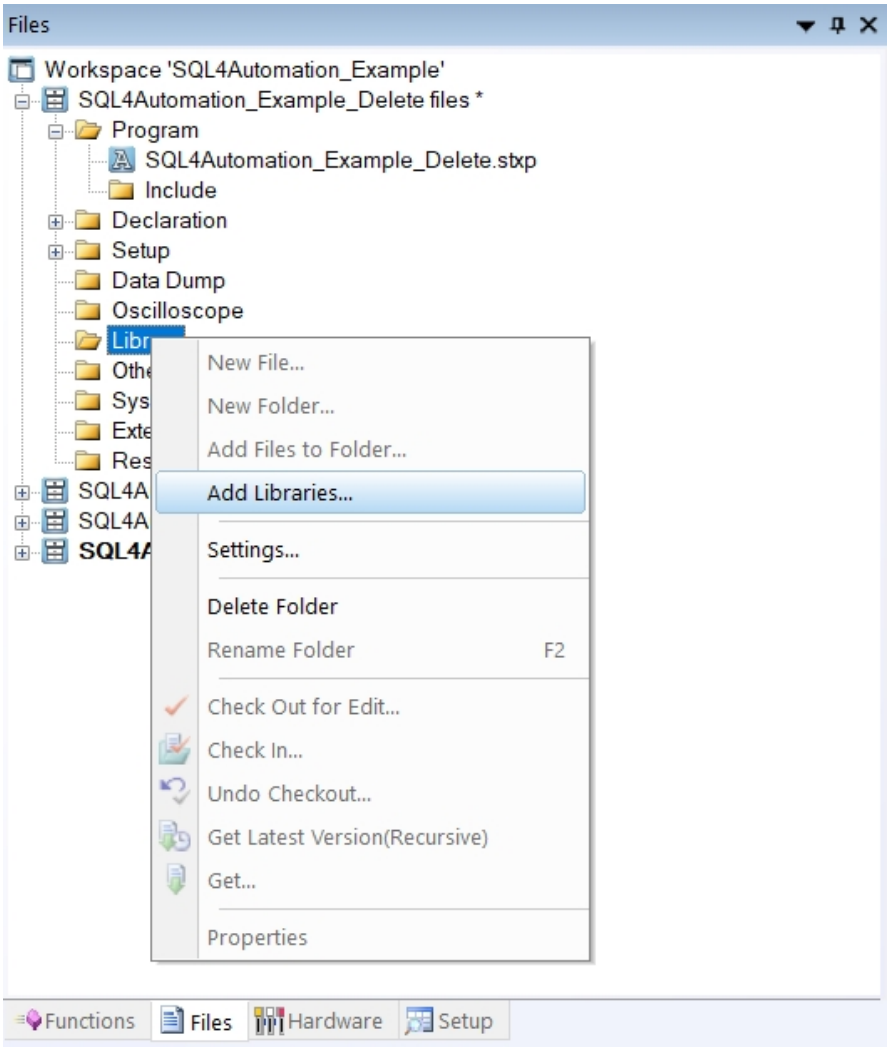

Fig. 2: Add library

2.3 Calling up the library in STX

In the STX program, the library is called up using the following command:

```
SQLTest : CfbSQL4JetSym(
    enumTCPIPServerClient.Client,
    enumTCPIPProtocoll.TCP,
    '192.168.10.208',            //IP address
    1101                         //Port number
    );
```

Parameters of Cfb-SQL4JetSym

Parameter	Type	Description
tTimeout	Int8	Timeout
dwResponseMaxReturn-Params	DWORD	Maximum number of return parameters
dwResponseMaxRows	DWORD	Maximum number of rows in the database query response
dwResponseMaxColumns	DWORD	Maximum number of columns in the database query response

Parameter	Type	Description
dwResponse-MaxStringLen	DWORD	Maximum string length in the database query response
dwResponseCutStringLen	DWORD	Strings in the response are truncated to the appropriate length
stResult.diResultState	eSQL4JetSym_SQLError-Code	Status of the database query/error number
stResult.dwResultRows	DWORD	Number of rows received from the database query
stResult.dwResult-Columns	DWORD	Number of columns received from the database query
stResult.sResultColumn-Name	Array of string	Column name received from the database query

Tab. 1: Parameters CfbSQL4JetSym

3 Working with the Sample Project

The library comes with an example project which can be used as a basis for own adaptations to match your application. Note that this program should run without errors before making any adjustments. This makes it easier to find and fix any errors that may occur.

ⓘ INFO

Note that the connector must be started and the connection to the sample database S4A_Test_DB.mdb must be configured in order to use the SQL functions. More information on this topic can be found in the user manual in the download area of SQL4automation.
(https://www.sql4automation.com/de/lizenzmodelle/download.php)

The sample project includes four sample programs that represent the basic commands of SQL:

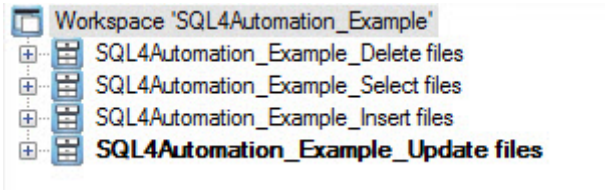

Fig. 3: Sample programs

3.1 Structure of the Sample Project

The structure of the sample project consists of four steps:

1. Calling the `Initialize()` function.
   This initializes the TCP/IP interface and establishes the connection to the connector. If the DefaultInputParameter variable has the value True, additional default parameters are set. Otherwise, the following parameters can be defined manually:
   - `InputFbSQL4.tTimeout := 30;`
   - `InputFbSQL4.dwResponseMaxReturnParams := 10;`
   - `InputFbSQL4.dwResponseMaxRows := 100;`
   - `InputFbSQL4.dwResponseMaxColums := 15;`
   - `InputFbSQL4.dwResponseCutStringLen := 0;`
   - `InputFbSQL4.dwResponseMaxStringLen := 255;`
2. Calling the `Reset()` function.
   In this step, all variables and data buffers are deleted.
3. Creating the SQL query according to the customer application and sending the telegram to the database by the SQL4automation framework. Then, the system waits for a response from the connector. If data is returned, it can be evaluated if it is successfully executed. In the event of an error or if no response is received from the connector, a corresponding error message is output.
4. Calling the `Delete()` function.
   This function closes the connection to the connector.

3.2 Communication Principle

The communication principle is the same for all sample projects:

The controller sends a request to the database and receives a response. The data flow is secondary here and data can be both read (Select) and written (Insert, Update) from the database. The exact syntax of the SQL command is determined on the one hand by the database model, and on the other hand by the task to be executed.

3.2.1 Submitting a Request

A query (request) to the database can be made using both a SELECT and an INSERT command.

Example: Select command

```
SQLTest.DefaultInputParameter := true;

//Initializing and starting the request
SQLTest.Initialize();
SQLTest.Reset();
SQLTest.AddRequest(
    'Select id,
    iParam,
    fParam2,
    sText1 FROM tTable WHERE id < 100 ORDER BY id ASC;'
);
```

Example: Insert command

```
SQLTest.DefaultInputParameter := true;

//Initializing and starting the request
SQLTest.Initialize();
SQLTest.Reset();
SQLTest.AddRequest(
    StrFormat(
    'INSERT INTO tTable1(
        iParam1, fParam2, sText1)
    VALUES(
        %d, %g, ''This_is_a_string''); ',
    iWert,
    fWert)
);
```

3.2.2 Parsing the Response

The response telegram from the database is checked for correctness on receipt. Important information is stored internally and is available for reading out the data:

- `OutputFbSQL4.stResult.diResultState`
  - 0 = No error
  - >0 = Error code
- `OutputFbSQL4.stResult.dwResultRows`
  Number of data records in the response telegram
- `OutputFbSQL4.stResult.dwResultColumns`
  Number of data columns in the response telegram

Reading out the data

There are two methods for reading out the data:

1. Once the query has successfully passed, the `aResponseReturnParams[]` array contains the received records. The row index and the column index are zero-based.
2. `OutputFbSQL4.stResult.sResultColumnName[]` returns the column name based on the corresponding column index. The column index is zero-based.

Example: Readout via array (approach 1)

```
FOR dwRow := 0 TO SQLTest.OutputFbSQL4.stResult.dwResultRows DO
    astSQLExampleData[dwRow].diID := StrToInt(
        SQLTest.aResponseReturnParams[dwRow, 0]);
    astSQLExampleData[dwRow].iParam1 := StrToInt(
        SQLTest.aResponseReturnParams[dwRow, 1])
    astSQLExampleData[dwRow].rParam2 := StrToFloat(
        SQLTest.aResponseReturnParams[dwRow, 2])
    astSQLExampleData[dwRow].sText :=
        SQLTest.aResponseReturnParams[dwRow, 3]
END_FOR;

if SQLTest.OutputFbSQL4.stResult.diResultState <> 0 then
    diErrorCount += 1;
else
    diDoneCount += 1;
END_IF;
```

4 Error Codes (Return Values)

Errors that occur are described in the `diResultState` variable:

Error number	Description
-1	Error during connection set-up
-2	Internal error when connecting
-3	Invalid parameter when connecting
-4	Error during sending
-5	Invalid handle on send/receive
-6	Error during reception
-8	Timeout
-9	Error when clearing the connection
0	No errors
1	Unknown SQL command
2	Query returns more records than are defined with dwMaxRows. Adjust query or increase dwMaxRows.
3	Query returns more columns than are defined with dwMaxColumn. Adjust query or increase dwMaxColumn.
4	Query returns more data than the defined buffer size. Adjust query or buffer size. Increase the size in the connector and in the controller.
5	Query returns at least one value greater than the value defined with MaxStringLength. Adjust query or increase MaxStringLength.
10	Internal connector error
11	Internal connector error. Database cannot be opened.
21	No IP address defined
22	No port number defined
23	Request string is empty
24	Number of maxRows not defined / diMaxRows=0
25	Number of MaxColumns not defined / diMax-Columns=0
29	Size of MaxStringLength not defined / diS-tringLength=0
51	Request string is larger than the send buffer.
>100	Error numbers of the ODBC database connection
40002	General error in SQL query. Request string is in-valid.

Tab. 2: Error Codes

Jetter
automation

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg
www.jetter.de

E-mail info@jetter.de
Phone +49 7141 2550-0

We automate your success.