The Cloud Database Professional plugin
PRINTED MANUAL

Cloud Database Professional plugin

Printed: 11/2/2024

Publisher

AGG Software

Production

© 2020-2024 AGG Software
http://www.aggsoft.com

Table of Contents

1 Introduction

The "Cloud Database Professional" data exporting module for our data loggers (for example, Advanced Serial Data Logger) is used for exporting parsed data into SQL-compatible cloud databases Microsoft Azure, MySQL, PostgreSQL, MariaDB, MongoDB, Amazon Aurora, Amazon Redshift.

The plugin uses the direct driver access methods provided by the developer of the corresponding database. It reduces system requirements, lowers the traffic between database clients and servers, and allows you to use features unique for every database (for example, stored procedures in Microsoft SQLServer or MySQL).

The plugin exports data to a database in real-time through a secure connection and can create a local backup temporarily until a remote database is offline.

The plugin allows executing one or more custom SQL statements in a queue.

2 System requirements

The following requirements must be met for "Cloud Database Professional" to be installed:

Operating system: Windows 2000 SP4 and above, including both x86 and x64 workstations and servers. The latest service pack for the corresponding OS is required.

Free disk space: Not less than 5 MB of free disk space is recommended.

Special access requirements: You should log on as a user with Administrator rights in order to install this module.

The main application (core) must be installed, for example, Advanced Serial Data Logger.

3 Installing Cloud Database Professional

1. Close the main application (for example, Advanced Serial Data Logger) if it is running;
2. Copy the program to your hard drive;
3. Run the module installation file with a double click on the file name in Windows Explorer;
4. Follow the instructions of the installation software. Usually, it is enough just to click the "Next" button several times;
5. Start the main application. The name of the module will appear on the "Modules" tab of the "Settings" window if it is successfully installed.

If the module is compatible with the program, its name and version will be displayed in the module list. You can see examples of installed modules on fig.1-2. Some types of modules require additional

configuration. To do it, just select a module from the list and click the "Setup" button next to the list. The configuration of the module is described below.

You can see some types of modules on the "Log file" tab. To configure such a module, you should select it from the "File type" list and click the "Advanced" button.

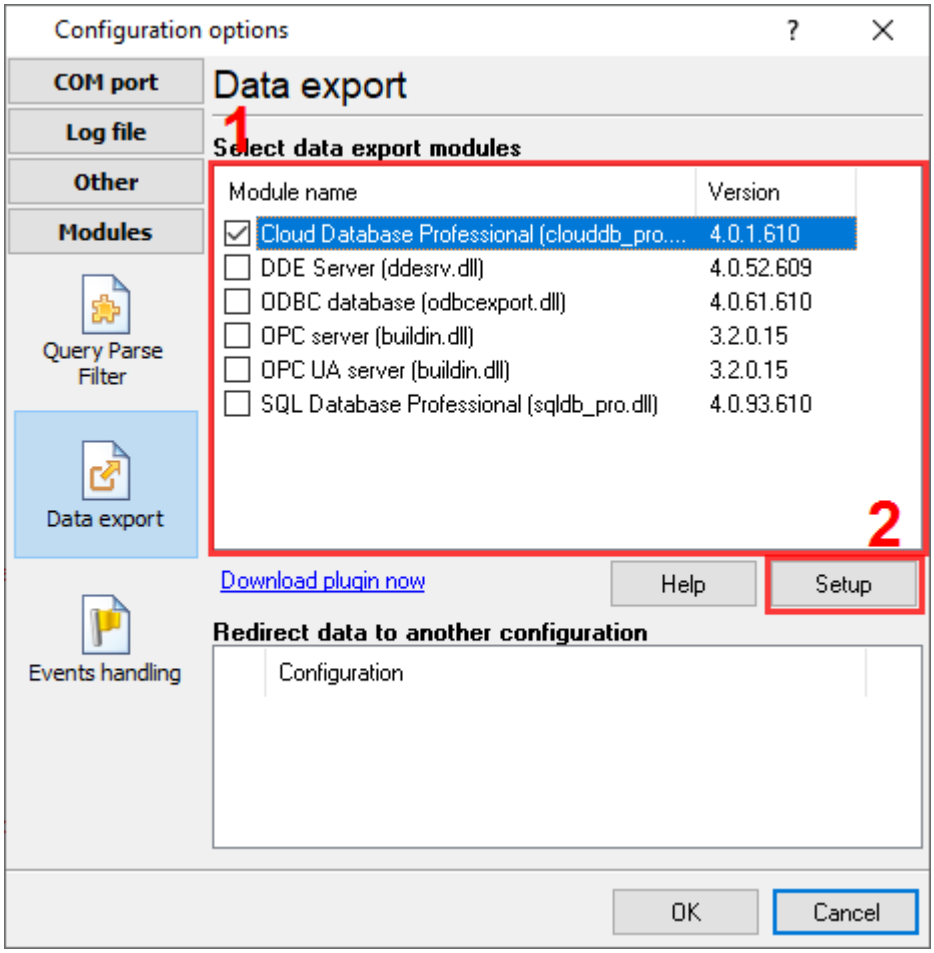

Fig. 1. Example of installed module

4 Glossary

Main program - it is the main executable of the application, for example, Advanced Serial Data Logger and asdlog.exe. It allows you to create several configurations with different settings and use different plugins.

Plugin - it is the additional plugin module for the main program. The plugin module extends the functionality of the main program.

Parser - it is the plugin module that processes the data flow, singling out data packets from it, and then variables from data packets. These variables are used in data export modules after that.

Core - see "Main program."

5 Setting up the connection

The module is configured in a special dialog box. To open the module settings dialog box, you should do the following:

1. Start the program if it is not running yet.
2. Select Options - Manage configurations - Change... in the main menu or click the  button on the toolbar.
3. Open the Modules - Data Export tab in the settings.
4. Select the "Cloud Database Professional" module from the list of data export modules on this tab. If there is no such module, go to the "Install" chapter and make sure you have done everything correctly to install the module.
5. Click the Setup button to configure the module settings.

6 Connection options

With our module, you can flexibly set the connection properties. The module can either maintain a permanent connection to the database or connect to it when necessary. You can set these parameters in the "Connection mode" group (fig. 2)

For the module to be activated, the "**Temporarily disable**" checkbox must be unchecked. You can select it to pause all operations with the database temporarily. It may be useful when you are configuring the module or administering the database.

Using the "**Stay connected**", "**Disconnect after each transaction**", and "**Disconnect when inactive**" options, you can specify the connection type. The "Stay connected" option makes the module connect to the database once needed and maintains the connection until the program is closed. The "Disconnect after each transaction" option makes the module connect to the database each time the module is called and terminates the connection after each piece of data is published (after all the data that should be published is published). The "Disconnect when inactive" option makes the module connect to the database once needed and disconnect if no data is published for the number of seconds specified in the "**Disconnect after**" field. It is recommended to use this option if data is received irregularly and at long time intervals. It allows you to lower the network traffic by reducing the number of empty requests.

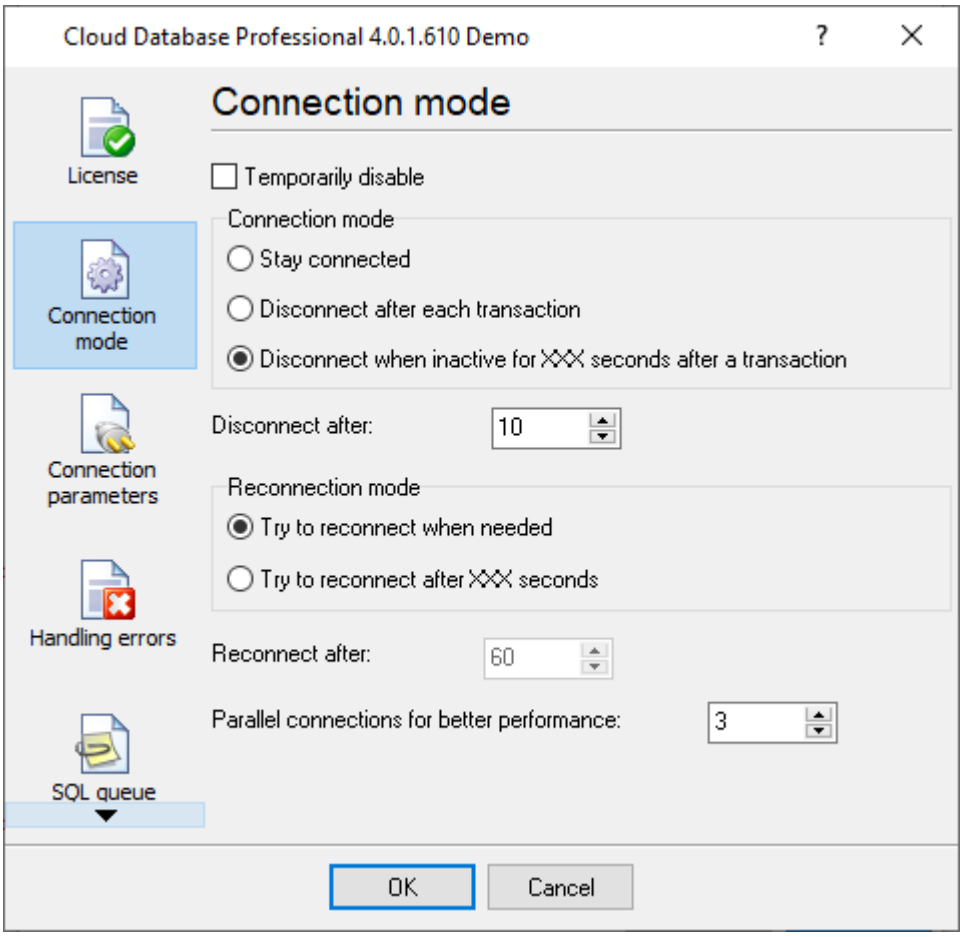

Fig. 2. Connection options

If a connection lasts for a long period, it can be lost due to a connection failure or a database crash. The "Reconnection mode" options allow you to set the mode in which the program will try to reconnect to the database.

- **Try to reconnect when needed** – the module will attempt to reconnect each time it is called;
- **Try to reconnect after XXX seconds** - the module will attempt to reconnect each time it is called but only after the number of seconds specified in the "**Reconnect after**" field.

7 Connection parameters

The options described in the previous section specify the properties for the physical connection, while the connection parameters described below configure the connection on the software level. You can set these parameters on the "**Connection parameters**" tab (fig. 3).

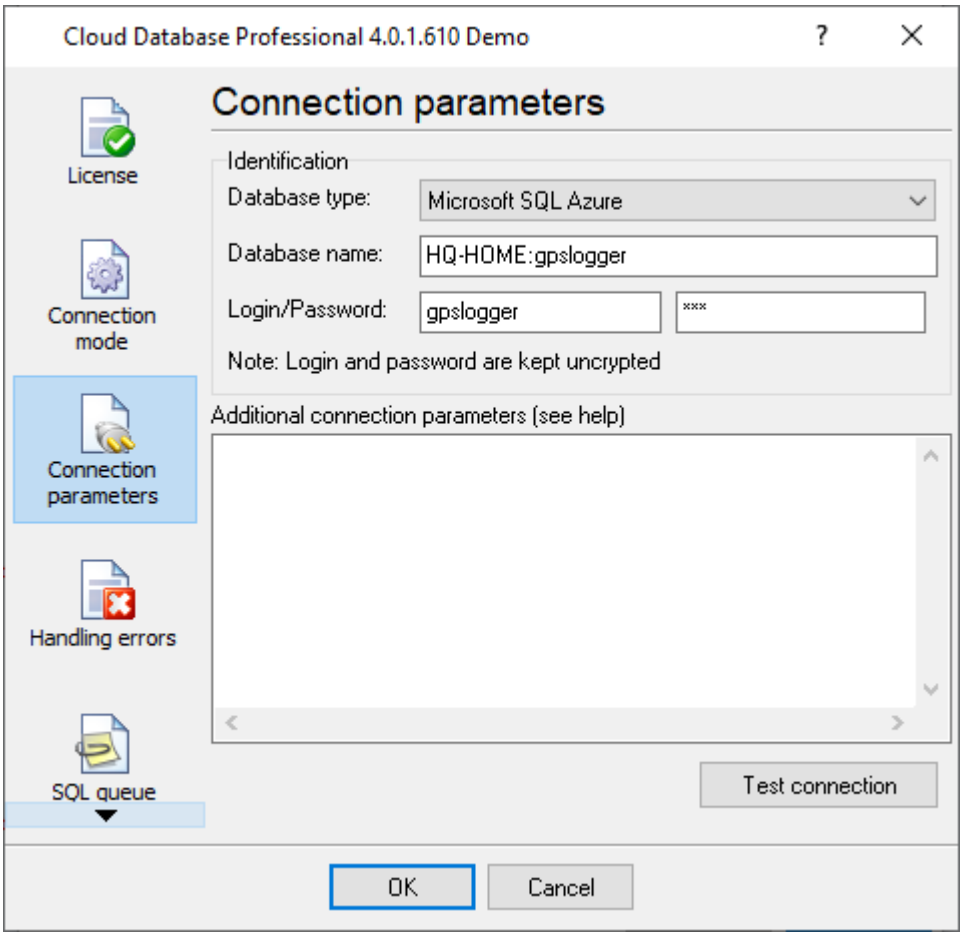

Fig. 3. Connection parameters

The database type is specified in the "**Identification**" group (you select it from the "**Database type**" list).

You should specify the hostname or the IP address and the database name in the "**Database name**" field using the following format:

hostname:database_name

Examples:

Aurora:
database-1.cluster-copvutiaj8an.us-east-1.rds.amazonaws.com:mydb

Azure:
aggsoft-test.database.windows.net:test

MongoDB:
cluster0-shard-00-02.j4arl.mongodb.net:test

Redshift endpoint address:
redshift-cluster-1.cm01xwy5h8ea.us-east-1.redshift.amazonaws.com:dev

The "Additional parameters" field may contain the following parameters:

Value	Description	Default	Note	Example
SERVER PORT	If the server uses a non-standard port to connect to a database	Database standard port	All databases	SERVER PORT=8897
SSL KEY	Path to the SSL client key file (*.pem) for your server	Not used	MySQL, MariaDB, PostgreSQL	SSL KEY=c:\MySQL8\data\client-key.pem
SSL CERT	Path to the SSL client certificate file (*.pem)	Not used	MySQL, MariaDB, PostgreSQL	SSL CERT=c:\MySQL8\data\client-cert.pem
SSL CA	Path to the SSL CA certificate (*.pem)	Not used	MySQL, MariaDB, PostgreSQL	SSL CA=c:\MySQL8\data\ca.pem
SSL CIPHER	(Optional) The desired SSL chiper type	Not used	MySQL, MariaDB, PostgreSQL	SSL CIPHER=`TLS_AES_128_CCM_SHA256`
COMPRESSED PROTOCOL	Enables the compressed protocol	Not used	MySQL, MariaDB	COMPRESSED PROTOCOL=TRUE
LOGIN TIMEOUT	Define the custom connection timeout in seconds	120	MySQL, MariaDB, PostgreSQL	LOGIN TIMEOUT=10
LOCAL CHARSET	Define connection charset	Database charset	MySQL, MariaDB	LOCAL CHARSET=utf-8
ConnectionOptions	Specify this parameter if a secured connection is required	Empty	MongoDB	ConnectionOptions=ssl=true

8 Amazon Aurora and Amazon Redshift

You should create inbound rules for incoming connections and redirect it to your data in your Virtual Private Cloud.

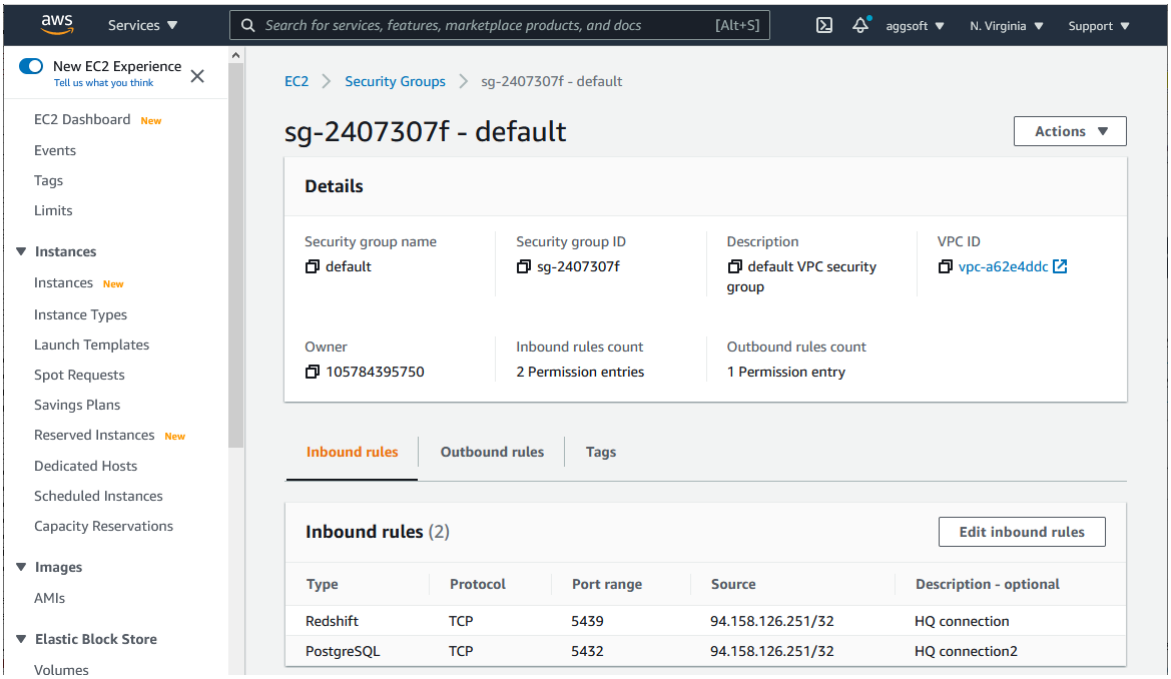

Fig. 4. Amazon VPC security group

Amazon Redshift

1. Start by navigating to the Redshift console --> Clusters screen and clicking on the Properties tab.

2. Scroll down to the Network and Security section. Make sure that the cluster is set with the value for Publicly Accessible to Yes. Then, click on the VPC Security Group to verify and/or modify the security rules.

3. In the Security Group screen, select the Inbound tab.

4. There should be rules for the IP addresses of your computers. If those rules need to be altered or don't exist, click Edit.

5. Edit any existing rules or click Add Rule to add a new rule. For each rule, select the type of database and enter the Redshift port. Then click Save.

6. To configure your cluster to only accept SSL encrypted connections. First, access the parameter group and edit it to set require_ssl to true. Then, Navigate to Config --> Workload management. If you created the cluster with a default parameter group, create a new parameter group and modify the cluster to associate to that parameter group. Click Edit on the cluster homepage, then go to Database Configurations to associate the parameter group with the current cluster.

9 MongoDB

The example below is based on MongoDB Atlas.

1. You should create inbound rules for incoming connections and redirect incoming connections to your database in your Virtual Private Cloud.

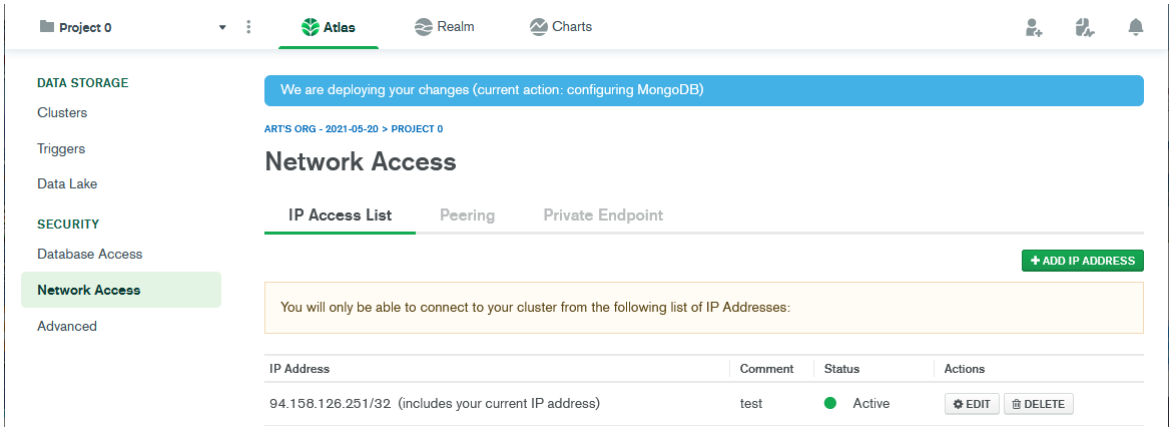

Fig. 5. IP access list

2. You should use the primary endpoint address as a hostname in the connection parameters.

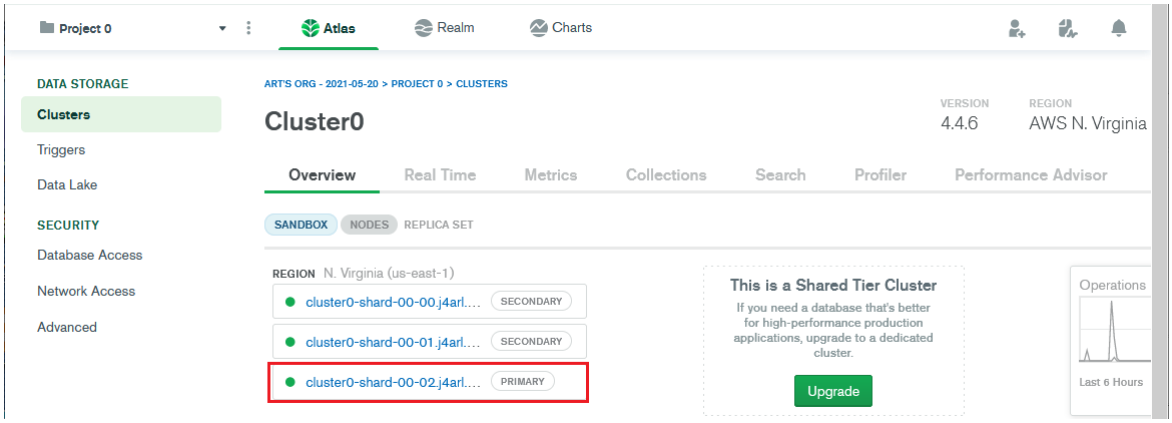

Fig. 6. Primary endpoint address

3. Create a new database and collection in it.

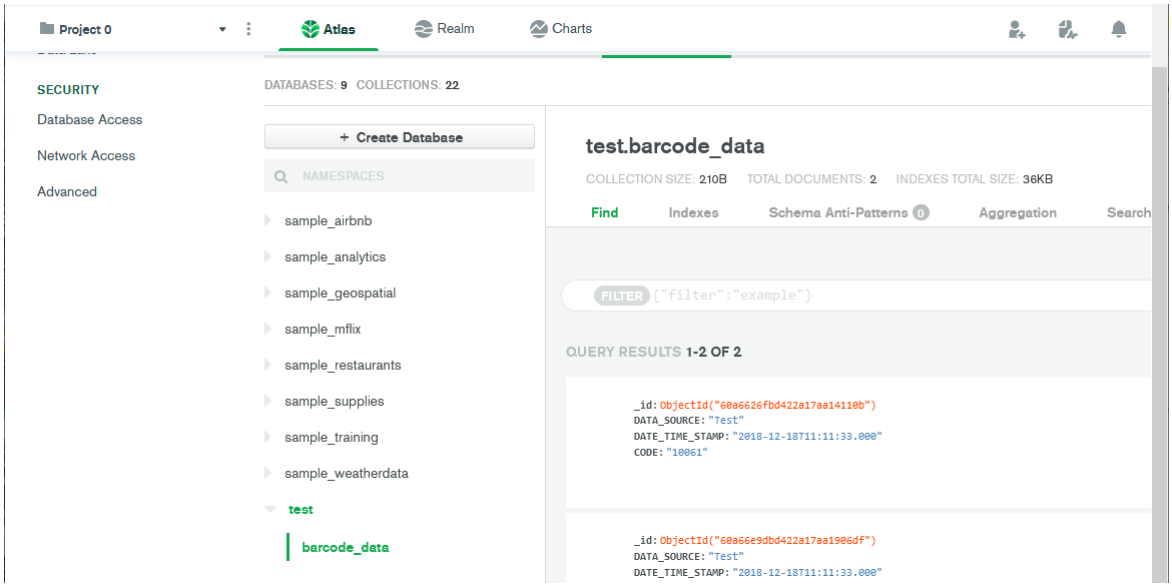

Fig. 7. Databases and collections

4. Secure connection settings

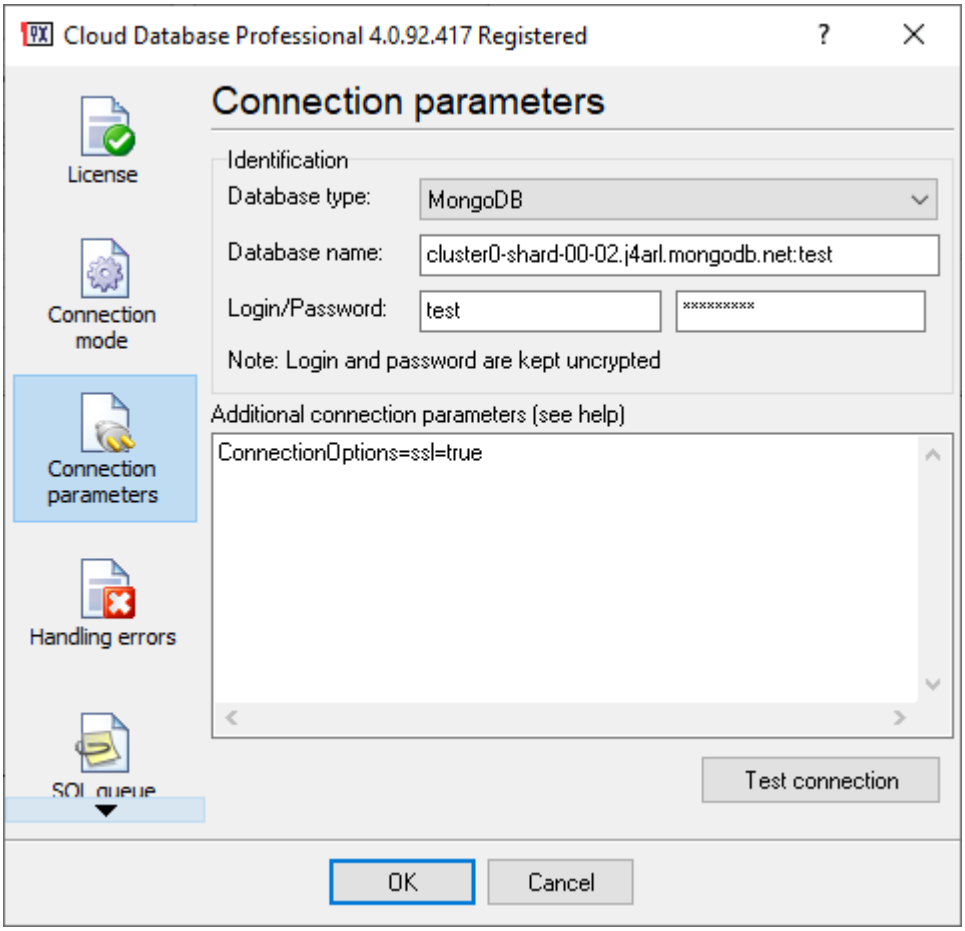

Fig. 8. Connection parameters

5. The "INSERT" statement for MongoDB.

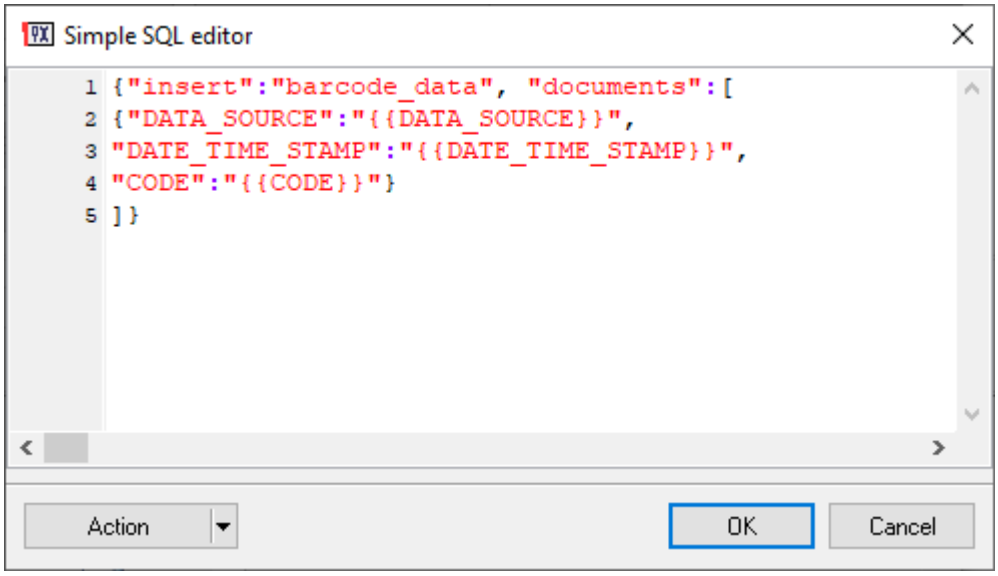

Fig. 9. Insert statement

10 Handling errors

When the module is performing its tasks, some errors may occur in the interaction with the database. It can be error messages about violated constraints (PRIMARY KEY), data integrity limitations (FOREIGN KEY), losing the database's connection, etc. You can set the behavior of the module when such errors occur. The parameters of this group are set on the "Errors handling" tab (fig. 10)

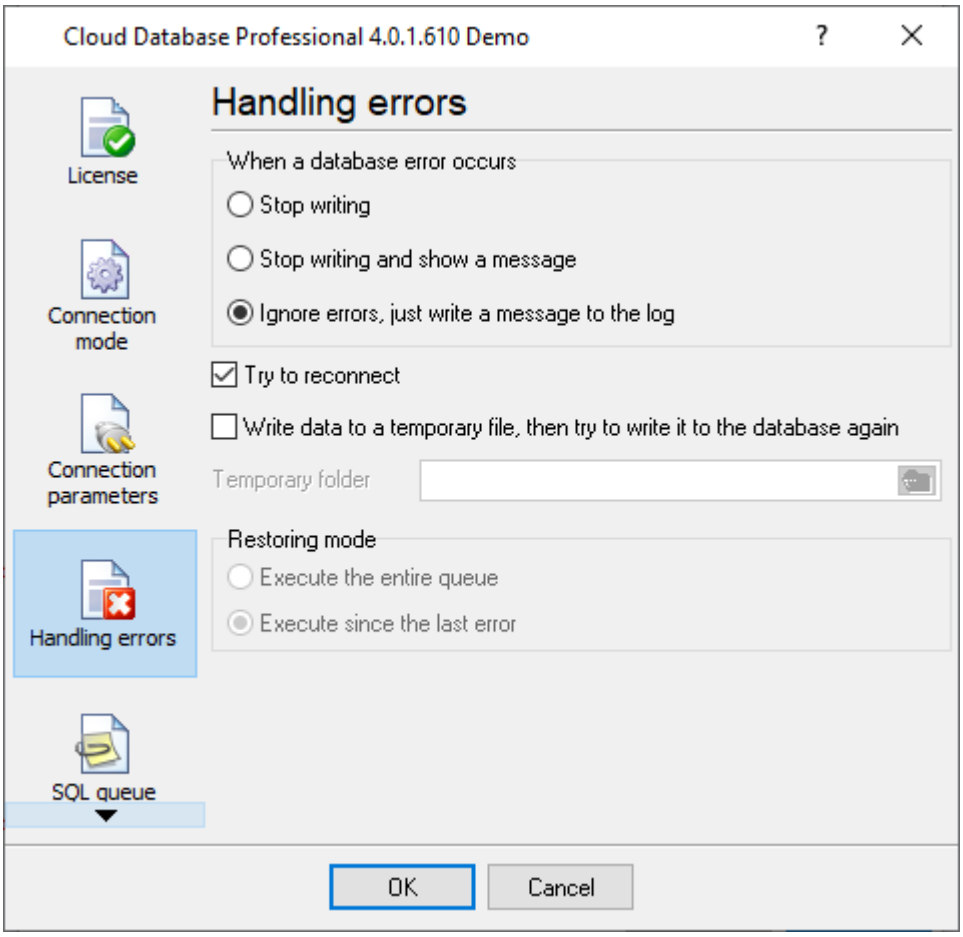

Fig. 10. Handling errors

There are four ways the module can react when an error occurs:

1. **Stop writing** – if an error occurs, the program generates a message and enables the internal "Temporary disabled" option; it stops publishing data until the module is reconfigured. The program will ignore all data after that moment until you restarted the program.
2. **Stop writing and show a message** – if an error occurs, the program generates a message, enables the internal "Temporary disabled" option, and displays a dialog box on the screen. The program will ignore all data after that moment until you clicked the "Yes" button in the dialog box.
3. **Ignore errors, just write a message to the log** – if an error occurs, the program generates a message and continues its work according to its configuration.
4. **Try to reconnect** – this option is similar to the previous one, except that the module will disconnect from the database and try to reconnect to it when the module is called next time. This option is useful if the database's connection is not stable.

The last two options allow you to save the data to a temporary file created in the "**Temporary**" folder to avoid losing data while publishing. The data will be placed into the temporary file only if the "**Write data to temporary file**" option is enabled. If any error occurs when the plugin publishes data, it saves data to that file and tries to restore data after the next successful write operation. Please, note that if your SQL statement contains syntax errors, the plugin will indefinitely try to backup and restore data.

If you added several queries to a SQL queue, an error might occur for any SQL query in the queue. If the "**Execute the entire queue**" option is selected, the plugin executes all queries in the queue when it restores data; otherwise, if the "**Execute since the last error**" option is active, the plugin executes only the remaining queries.

For example, the first option is necessary when an error occurs for the first SELECT query, and data from that query is used in a subsequent INSERT query.

The second option can be used if your queue contains several INSERT queries and you should skip successfully processed SQL queries to avoid duplicates.

Module messages

The module may output errors, warnings, and information messages when it operates. These messages are displayed at the log box in the main program window. You can disable or enable some message types in the "Program settings – Protocol – Data export" category.

11 SQL queue

Data is often written into a database with the help of the INSERT statement or a stored procedure. Suppose your INSERT query requires data from another table. In that case, you may combine several SQL statements to a queue of SQL queries (from now on called "queue") that will be executed one by one, starting from the upper one. The data you get using SELECT can be available for the following SQL queries. This feature enhances our module's use because you do not have to create complex and nested SQL queries or special stored procedures. You can configure the queue on the "**SQL queue**" tab (fig. 11).

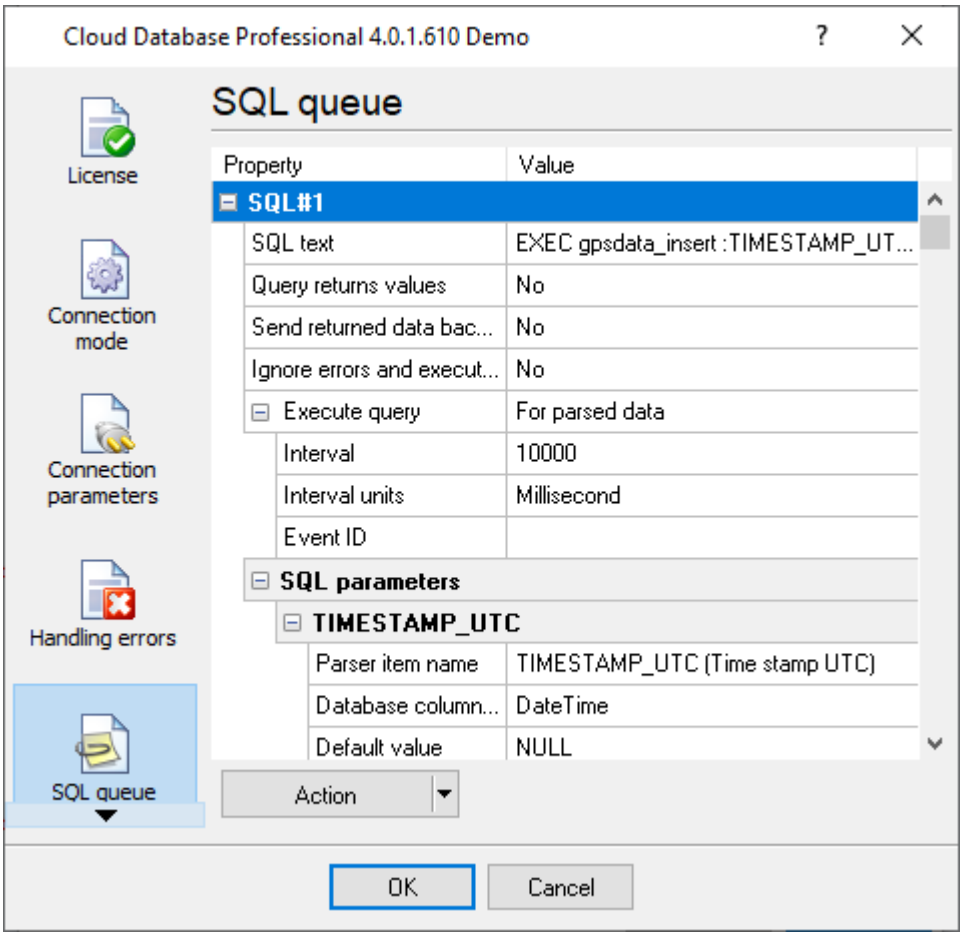

Fig. 11. SQL queue

You may execute all actions for the selected SQL query using either a popup menu or the "**Action**" button. To choose an SQL query, click either its title (the blue line in fig. 11) or any of its parameters.

Add SQL to the queue – add a new SQL query to the queue's end and select it.

Delete SQL from the queue – delete the selected SQL query from the queue.

Move SQL up, **Move SQL down** – move the selected SQL query up or down in the queue.

Load an SQL queue from a file – load a new queue from a file. This option can be useful when you move the configuration from one computer to another.

Save the entire SQL queue to a file – save the entire queue to a file. This option can be useful when you move the configuration from one computer to another. You can load the created file with the help of the previous option.

12 Creating a new SQL query

To create a new SQL query in the queue, click the "**Action - Add**" button. You will see a new SQL query and its parameters in the queue (fig. 12). The query will automatically acquire a name with its number.

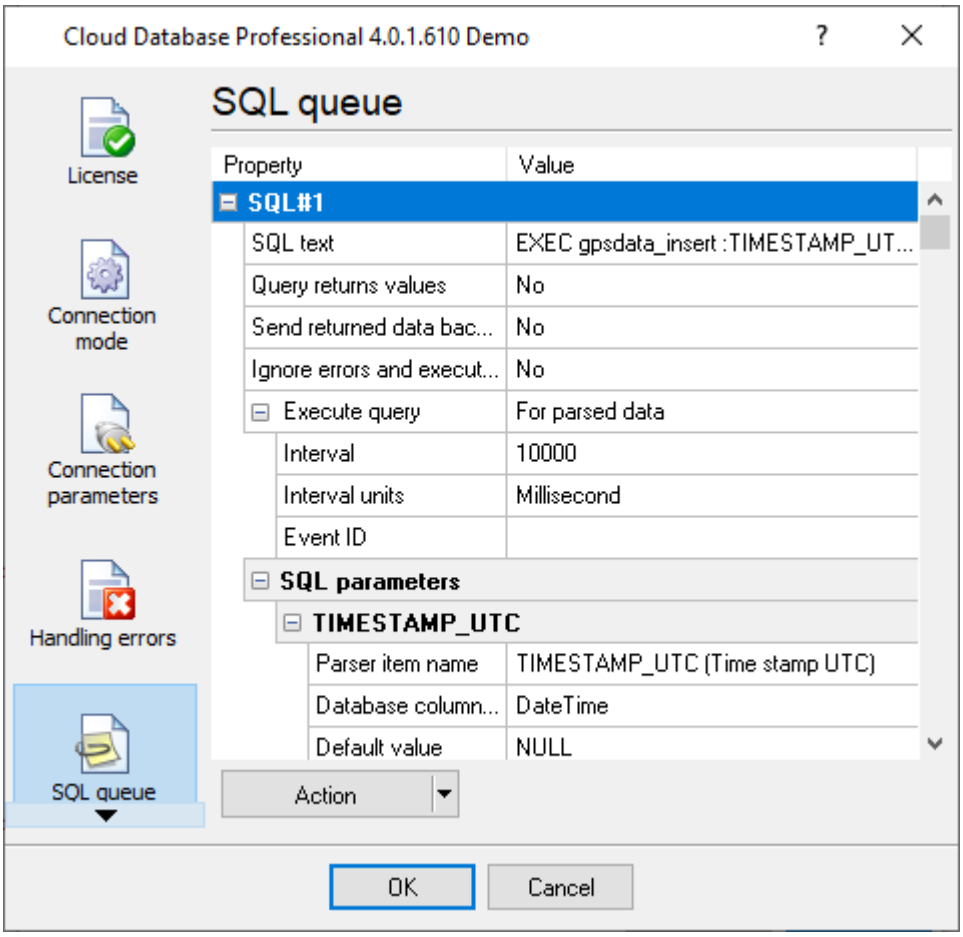

Fig. 12. A new SQL query

SQL text – specify the text of the SQL query in this field. To enter the text, click the "Value" column and click the button you see on the right. You will see a **simple SQL editor** window (fig. 6). Please enter the text of the SQL query into it. You can use parameters when you created an SQL query. Such a string as ":P1" means that a parameter with the "P1" name. Later, you can assign a value created by the parser to this parameter. You can also save or load an SQL query to/from a file. After you finish editing, you can click "OK" to save the changes or "Cancel" to cancel them.

Query returns values – means that the query returns data (for example, SELECT). This parameter can take two values: "Yes" or "No." You can select either of them from the list that appears when you click the "Value" column.

Send returned data back to data source – if the query returns string values, the module interprets that data as a byte array and sends it to the current data source. It allows you to send data to a device from your database. Data should be fully prepared in the database because the module does not change or encode it.

Ignore errors and execute next – if an error occurs during this query execution, it is ignored, and the program continues executing the queue. This parameter can take two values: "Yes" or "No." You can select either of them from the list that appears when you click the "Value" column. You need this option if this SQL query is secondary, and the process of executing other SQL queries in the queue must not depend on it.

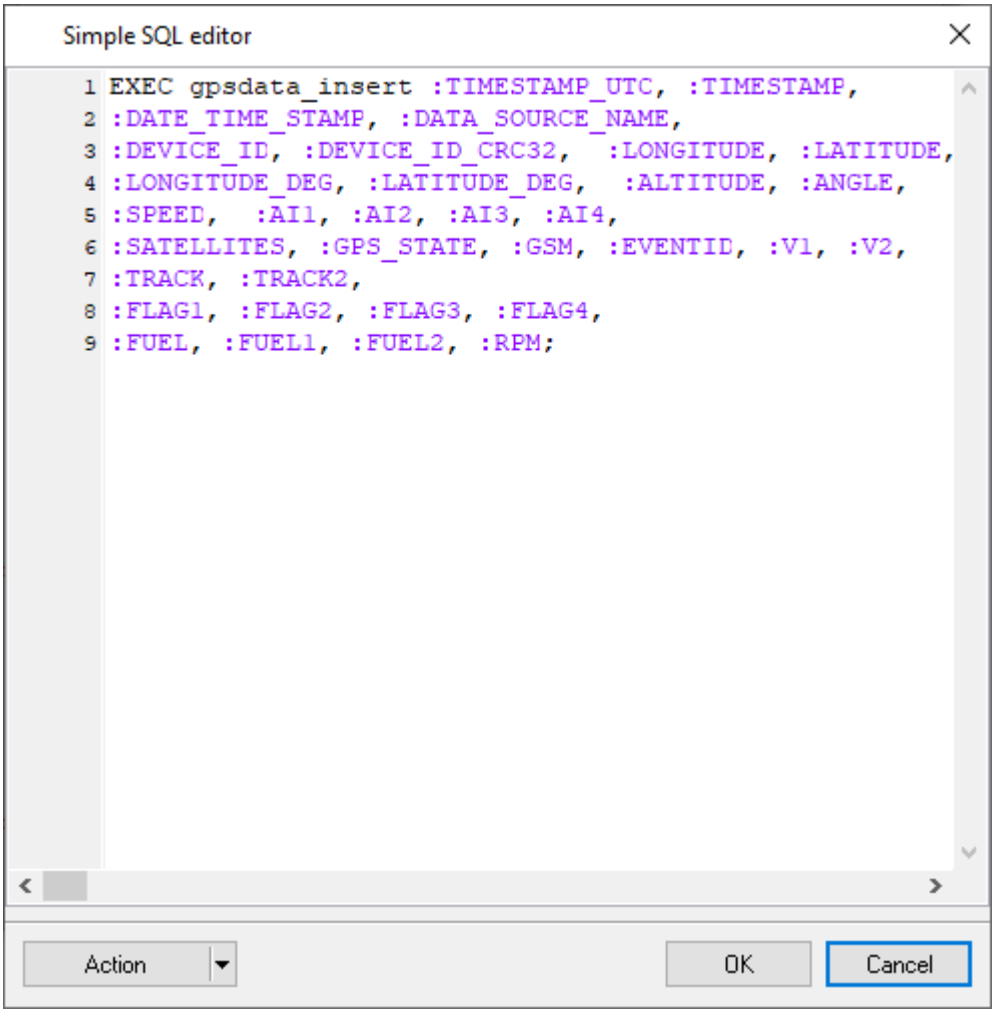

Fig. 13. SQL editor

After you saved changes in the SQL editor, its window is closed, and the text of the SQL query is placed into the "SQL text" field (pic. 13), and the plugin extracts the names of parameters from it (if there are any). You can see them in the "**SQL parameters**" group.

Each parameter has three properties:

Parser item name. The name of the variable created by a data parser plugin that you must configure a parser before. You can either select the variable's name from the list or enter it manually. Besides those created by the parser, there are always two predefined variables in the list: NULL and DEFAULT. They mean that the parameter will always have the NULL value or the default value specified in the "Default value" field.

If your SELECT SQL statement return values, you may define their names in the SQL statement using the "AS" keyword:

```
select (max(id)+1) as max_id from test_datas
```

The result of this query is the maximum value of the ID column that will be given the name *max_id*. Therefore, the MAX_ID variable name is entered manually in all the queries coming next (the letter case does not matter). You can use the MAX_ID name in several SQL queries at a time. If MAX_ID has the null value (for example, if the "test_datas" table has no records), the default value "1" will be used when this variable is used with parameter P1.

You should assign variables to all parameters you specify in your SQL query.

Database column data type – determines the column data type. This data type should match the data type of a parser's variable value. Therefore the parser should extract a value with the corresponding data type. The plugin can convert data between similar data types. For example, the parser extracts integer values, and the database column has the "Float" data type.

Default value – the default value to be used if the variable with the specified name is not sent for publishing or its value is null.