The SQL Database Professional plugin
PRINTED MANUAL

SQL Database Professional plugin

Printed: 28.11.2016

Publisher

AGG Software

Production

© 1999-2016 AGG Software
http://www.aggsoft.com

Table of Contents

1 Introduction

The "SQL Database Professional" data publishing module (hereinafter called "module") for our data loggers (for example, Advanced Serial Data Logger) is used for recording/saving variables the parser receives and extracts from the stream of bytes into SQL-compatible databases, such as SQLBase, Oracle, Microsoft SQLServer, Sybase, DB2, Informix, Interbase, Firebird, MySQL, PostgreSQL.

To access databases, the direct driver access methods provided by the developer of the corresponding database are used. It makes it possible to reduce system requirements, lower the traffic between database clients and servers, and, what is more important, use features unique for every database (for example, stored procedures in Microsoft SQLServer or Oracle).

If our module does not support direct access to some database, it is possible to work through the ODBC driver that is also supported by our module. Unlike our other module for ODBC Databases, this module provides broader opportunities for controlling the database connection and the process of executing databases operations and implements quite a new approach to data publishing.

In this module the user has to create SQL queries himself. If necessary, the user can form a queue of SQL queries that should be executed one after another.

To do it, the user must have certain skills, but it allows him to implement ideas and tasks unavailable in our program before, such as:

- Using stored procedures and other programmable database objects for recording data;
- Using functions and custom data types;
- Recording data into the tables that have links to other tables.

2 System requirements

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

Operating system: Windows 2000 SP4 and above, including both x86 and x64 workstations and servers. A 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.

Notes for Microsoft Vista and above:

Since our software saves data to the registry and installs to the Program Files folder, the following requirements must be met:

1. You need Administrator rights to run and install our software
2. The shortcut icon of our software will be located on the desktop;
3. Windows Vista will ask for your confirmation to continue the installation.

NOTE: You can configure the user account only once in order not to see the above dialog box any more. Search Google for the solution of this problem.

3 Installing SQL 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.

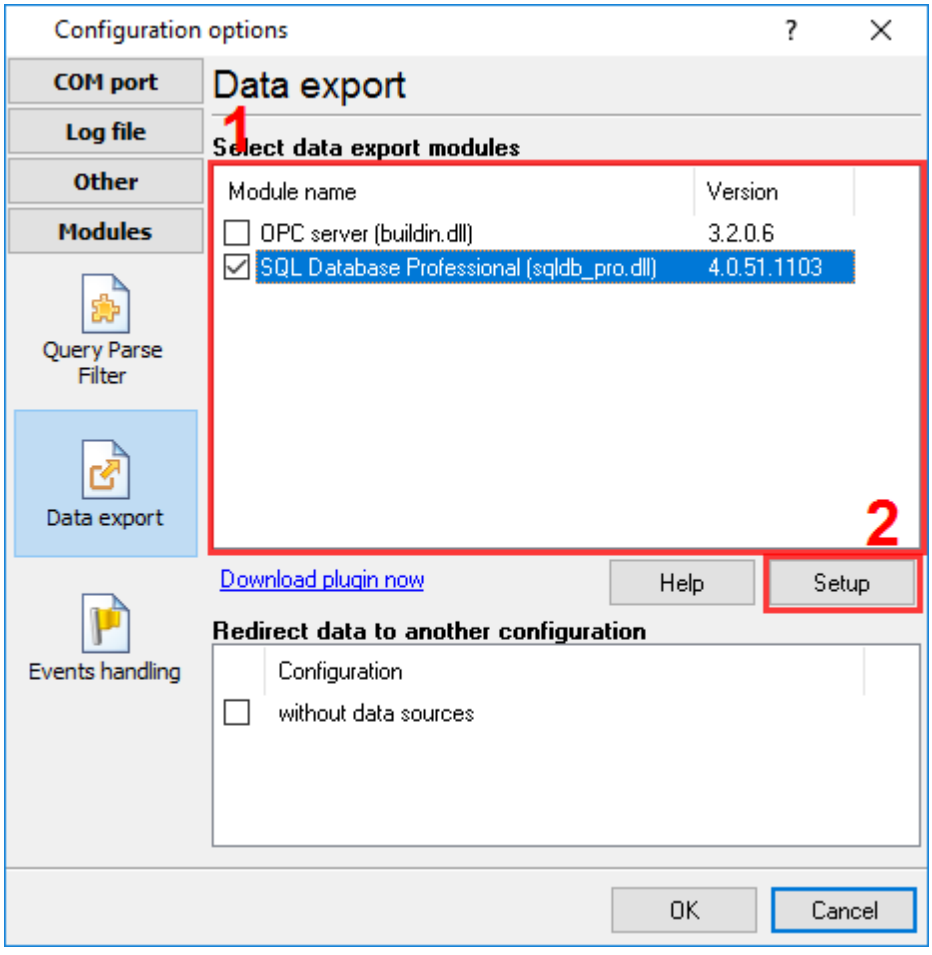


Fig.1. Example of installed module

4 Glossary

Plug-in - module

Main program – the program shell that uses this module. For example: Advanced Serial Data Logger

Parser – the module that processes the data flow singling out data packets from it and 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 "SQL Database Professional" module from the list of publishing modules on this tab. If there is no such module, go to section 1 and make sure you have done everything correctly to install the module;
5. Click the Setup button.

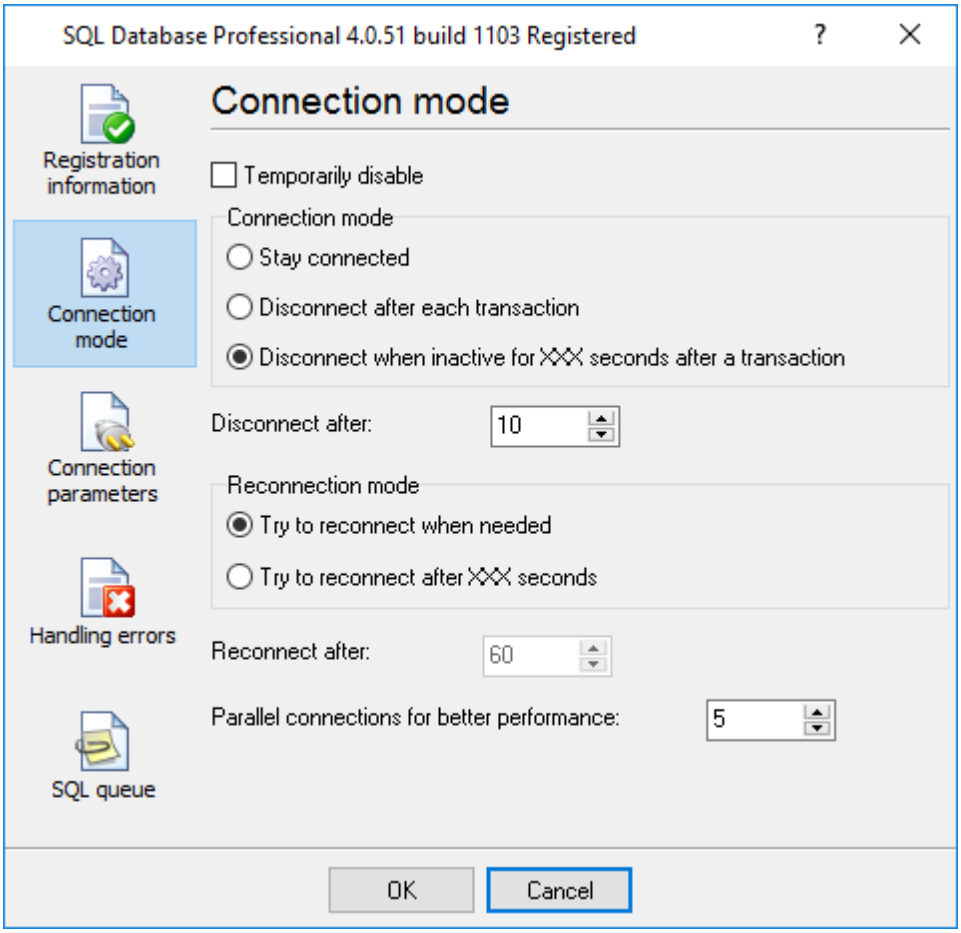
5.1 Connection options

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 (pic. 1)

For the module to be activated, the "**Temporarily disable**" check box must be cleared. You can select it to temporarily pause all operations with the database. 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 maintain 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 terminate 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.



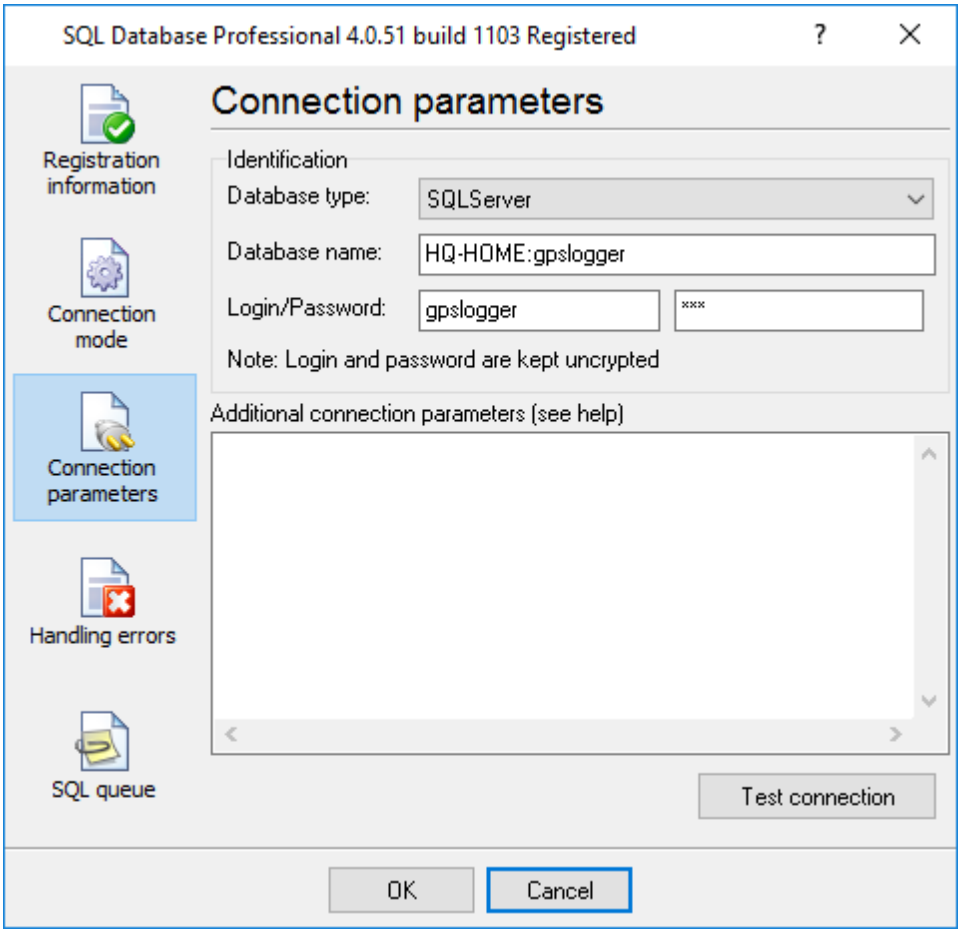
Pic. 1. Connection options

If a connection last for a long period of time, 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 make an attempt to reconnect each time it is called;
- **Try to reconnect after XXX seconds** - the module will make an attempt to reconnect each time it is called but only after the number of seconds specified in the "**Reconnect after**" field.

5.2 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 (pic. 2).



Pic.2. Connection parameters

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

Depending on the selected database type, use the "**Database name**" field to specify the following:

- **DB2, Informix and ODBC** – specify DSN set in the "ODBC Administrator" or the complete description of the data source with the parameters supported by the selected server. You can see an example of this field for the Informix server below:

  SERVICE=ids_srv;HOST=yourhost;PROTOCOL=OLSOCTCP;SERVER=ids_srv;
  DATABASE=sysmaster;UID=informix;PWD=informix.

- **Interbase** – specify the path to the necessary database and the network protocol (see the examples below).

Value	Protocol
<server_name>:<filename>	TCP
\\<server_name>\<filename>	NetBEUI
<server_name>@<filename>	SPX

- **Oracle** – specify host name/service name.
- **MySQL, MS SQL Server or Sybase SQL Server** – if you establish a connection with a remote server, specify its name and the database name separated with a colon. For example, remsrv:dbname points to the DBNAME database situated on the REMSRV sever. Specify **(local)** for a local database.

Then specify username and password to access the database in the "Login" and "Password" fields respectively.

Use the "Additional connection attributes" field to specify connection parameters unique for each server.

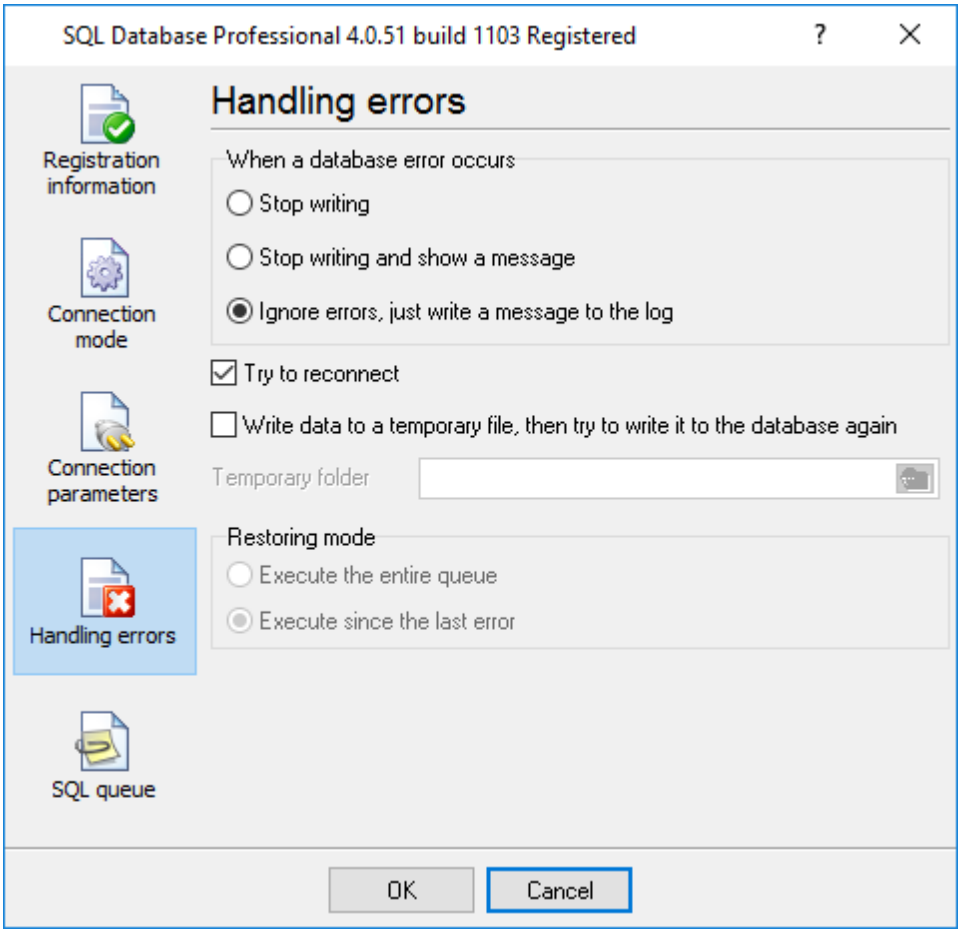
Value	Description	Note
AUTOCOMMIT	Use autocommit	
APPLICATION NAME	The name of the application that will be sent to the server	Only for MSSQL and Sybase
HOST NAME	The name of the workstation that will be send to the server	Only for MSSQL and Sybase
COMMAND TIMEOUT	The number of seconds to wait until any operation is finished	Only for MSSQL, ODBC, SQLBase, Sybase
COMPRESSED PROTOCOL	Use compression while exchanging data between the client and the server. By default, the value is TRUE	Only for MySQL
ENABLE BCD	Change the NUMERIC data type into the BCD data type before sending data to the server	Only for Oracle, Interbase
ENABLE INTEGERS	Change the NUMERIC data type into the INTEGER data type before sending data to the server	Only for Oracle, Interbase
ENABLE MONEY	Change the NUMERIC data type into the CURRENCY data type with the precision (1-4) before sending data to the server	Only for MySQL
ENCRYPTION	Use encrypted passwords when accessing the database. By default, this value is FALSE	Only for Sybase
FIELD REQUIRED	Display an error message if any field has the NULL value when a query is executed	
FORCE OCI7	Use OCI7 (SQL*Net 2.x - Oracle7 interface) to access the Oracle server	Only for Oracle
LOCAL CHARSET	Set the encoding character set	Only for Interbase
LOGIN TIMEOUT	The number of seconds to wait for user authorisation	Only for DB2, Informix, ODBC, MSSQL, MySQL, Sybase
MAX CURSORS	The maximum number of simultaneously opened cursors	Only for MSSQL and Sybase

MAXCHARPARAMLEN	The maximum line length. By default, it is 255	
MAXFIELDNAMELEN	The maximum length of a field name. By default, it is 50	Only for Oracle
MAX STRING SIZE	Limit the size of strings to this value. Longer strings will be considered a blob	Only for Firebird, Interbase, ODBC
NEW PASSWORD	Use this value when the server returns the 'Password expired' message	Only for Oracle8
QUOTED IDENTIFIER	Use identifiers in quotes	Only for MSSQL and Sybase
PREFETCH ROWS	The number of rows to be prefetched in order to minimize network traffic (Oracle8: this option does not work if SELECT contains fields of the LONG type)	Only for DB2, Informix, ODBC, Oracle8
ROLE NAME	Specifies the role the server should assign to the client when it is connected	Only for Interbase and Oracle (SYSDBA/SYSOPER roles)
SERVER PORT	Specifies the server port for connecting via TCP/IP	Only for MySQL, PostgreSQL
SINGLE CONNECTION	Specified whether to use a single process/ connection. By default, it is FALSE	Only for MSSQL and Sybase
SQL DIALECT	Installs SQL Dialect (1,2,3) for the client	Only for Interbase
TDS PACKET SIZE	Specifies the size for a TDS packet. If the server does not support this size, a "Login failed" error will occur in the process of connecting	Only for Sybase
TRANSACTION LOGGING	If it is FALSE, transaction logging will be disabled, so rollback will be unavailable	Only for SQLBase
RTRIM CHAR OUTPUT	Delete spaces on the right for fields of the CHAR type. By default, it is TRUE	Only for DB2, Informix, Interbase, Oracle, ODBC and Sybase
XA CONNECTION	Indicates that it is necessary to connect to the TM service with the name specified in the "Database name" field. By default, it is FALSE	Only for Oracle8i
XXX API LIBRARY	Specifies the interface library type to use for connecting, where XXX stands for server type. For example, Oracle, SQLServer, Interbase, etc.	

After you set up your database connection, you can immediately test it by clicking the "**Test connection**" button. The program will try to connect to the database. The process can take rather long (up to three minutes) depending on the database type. A message will be displayed as the test result. If an error occurs, the message will contain the server response that will help you find out what has caused the error.

5.3 Handling errors

When the module is performing its tasks, some errors may occur in the interaction with the database. It can be such errors as breaking the uniqueness while data are recorded into a table (PRIMARY KEY), breaking the data integrity limitation (FOREIGN KEY), losing the connection to the database, etc. You can set the behaviour of the module when such errors occur. These settings do not influence handling internal module errors, handling which is strictly programmed to provide maximum stability for the module. The parameters of this group are set on the "Errors handling" tab (pic. 3)



Pic. 3. 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. Besides, all the data coming for publishing after the error occurs will be recorded NEITHER to the temporary file NOR to the database. This option can be useful when you are configuring the module.
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; it stops publishing data until the dialog box is closed. All data coming for publishing after the error occurs will be recorded NEITHER to the temporary file NOT to the database. If you click

"Yes" in the dialog box, the module will resume its work. This option can be useful when you are configuring the module.

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 can be useful if the database connection is not stable (due to database or connection failures), because sometimes it is difficult to tell the reason for a disconnection and perform the necessary actions judging from the response of the server.

The last two options allow you to save the data to a temporary file created in the "**Temporary**" folder to avoid losing the data for publishing. The data will be placed into the temporary file only if the "**Write data to temporary file**" option is enabled. If any errors occur in the process of publishing some data, such data will be placed into this file. When the module is called next time, it will try to publish the data in the file into the database. This option is necessary if your database connection is not stable. However, if the queue of SQL queries is not build correctly and an error occurs often or every time when it is executed, the consumption of network and server resources may increase considerably as the result of multiple attempts to publish data from the temporary file. Nevertheless, you can be sure that when the error is fixed, all the data will be move to the database from the temporary file. It should be mentioned that you should use the "Temporary folder" field to specify the path to a folder the user who runs the program save to and reading from. Otherwise, the module will not be able to create a temporary file and the data will be permanently lost. If you leave the "Temporary folder" field empty, the program will create the file in the module installation folder.

Since you can create a queue of SQL queries that will be executed sequentially and an error can occur when any SQL query in the queue is being executed, the module can either execute the entire queue again ("**Execute the entire queue**" option) or continue from the moment when the last error occurred ("**Execute since the last error**" option) when it attempts to publish data from a file.

The first option is necessary when, for example, SELECT queries selecting data from linked tables are first in the queue and an error occurs while inserting data with an INSERT query into another table.

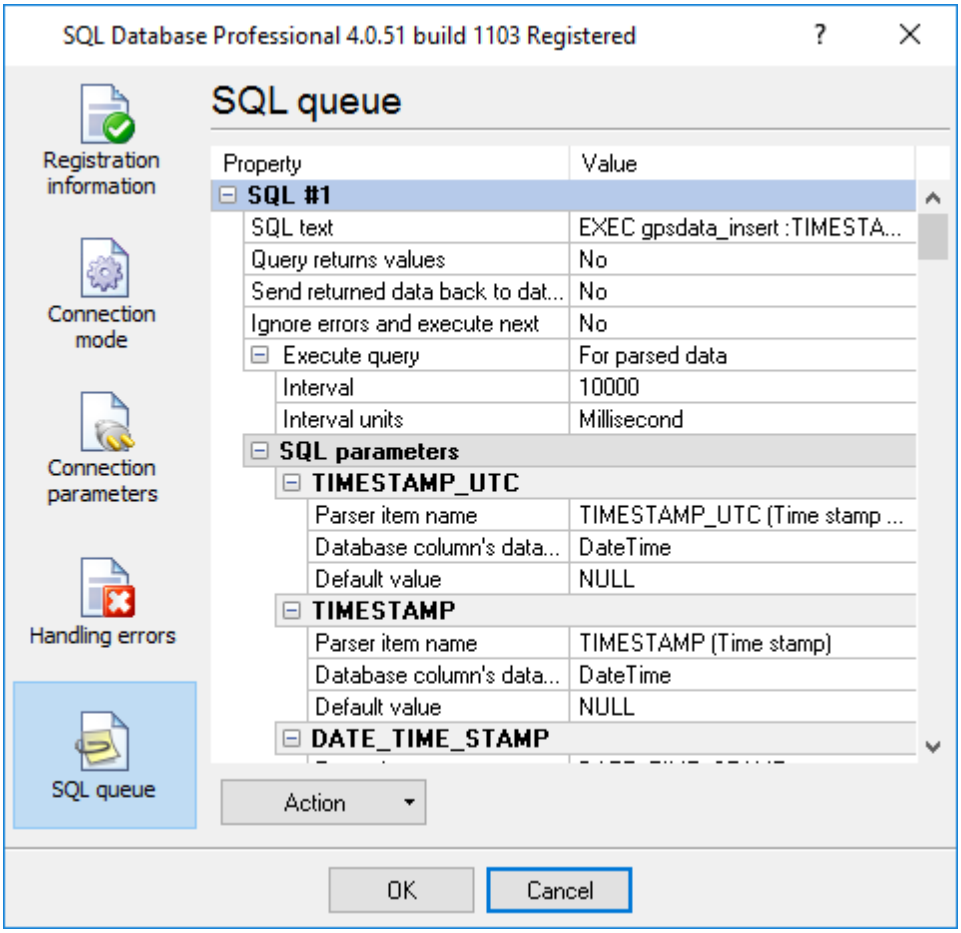
The second option can be used if your queue consists of several INSERT queries and you should skip successful SQL queries to avoid duplicates.

5.4 Module messages

During its work, the module generates three kinds of messages: errors, warnings and information messages. These messages are displayed in the main program window. You can customize how detailed the messages should be using the kernel options of the core engine. To get a more detailed description of this procedure, see the main program user guide. It is worth pointing out that messages the module generates belong to the "Data export" group.

5.5 SQL queue

Very often data is written into a database with the help of INSERT or a stored procedure, and you have to write or read additional data into or from the database to perform INSERT or a stored procedure. In most cases you cannot do it with a single SQL query. With our module, you can create a queue of SQL queries (hereinafter 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 the use of our module because you do not have to create any complex and nested SQL queries or special stored procedures. You can configure the queue on the "**SQL queue** " tab (pic. 4).



Pic. 4. SQL queue

All operations in the queue are applied to the selected SQL query using either the popup menu or the "**Action**" button. To select an SQL query, click either its title (the blue line in pic.4) or any of its parameters.

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

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

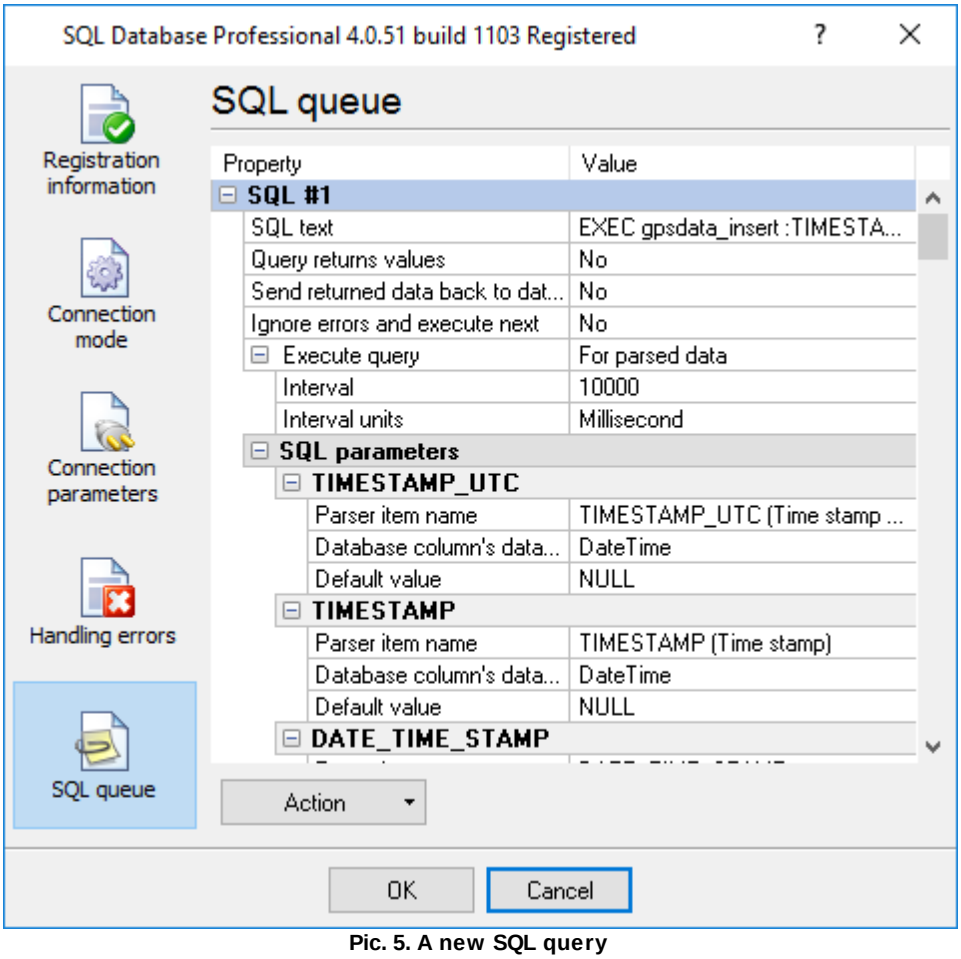
Move SQL up, **Move SQL down** – move the selected SQL query up or down in the queue.

Load an SQL queue from a file – load a new queue from a special 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 special 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.

5.5.1 Creating a new SQL query

To create a new SQL query in the queue, click the "**Action - Add**" button described above. You will see a new SQL query and its parameters in the queue (pic. 5). The query will automatically acquire a name with its number.



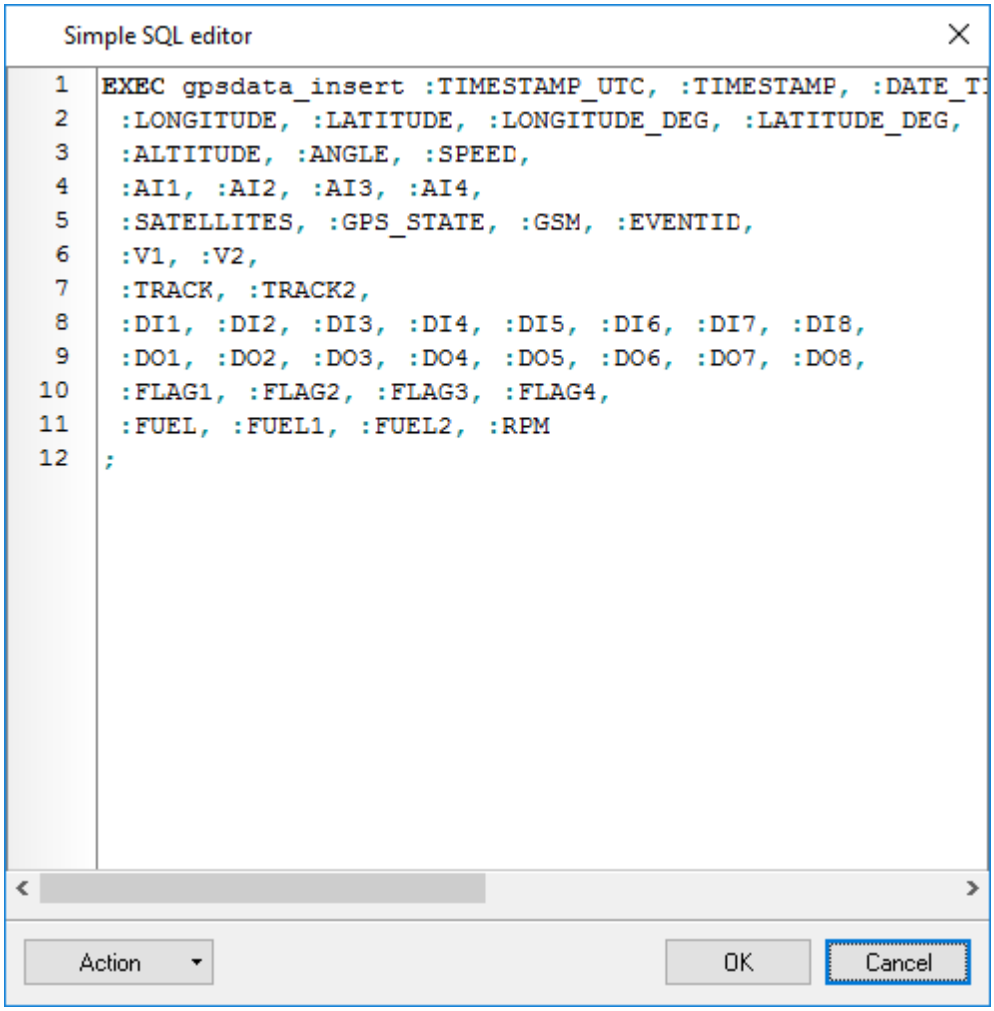
Pic. 5. A new SQL query

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 (pic.6). 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 name "P1". 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 this data as a byte array and sends to the current data source. It allows you to send data to a device from your database. A 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 the execution of this query, it is ignored and 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.



Pic. 6. SQL editor

After you save changes in the SQL editor, its window is closed, and the text of the SQL query is placed into the "SQL text" field (pic. 5). This text is then analyzed and the names of parameters (if there are any) are extracted from it. You can see them in the "**SQL parameters**" group.

Each parameter has three properties:

Parser item name – the name of the variable created by the data parser (you must configure the parser for publishing any data. Read in the main program user guide how to do it). You can either select the of the variable 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.

The names of the variables that are created after SQL queries returning values are executed are entered manually. It is done in the following way. For example, the first SQL query is:

```
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) (pic. 7). 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 (pic.7).

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

Database column data type – determines the data type to write to using the specified parameter. Depending on the specified type, the module converts the data it publishes to the necessary 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.

6 Troubles?

6.1 Possible problems

No data for publication/exporting – no data is passed for exporting. Solution: configure the parser, make sure that one or more variables are declared in the parser.

Error on binding variable with name %s [%s] – the error usually occurs if data does not correspond to the specified format. For example, the date and time format does not correspond to the data.

Unable to disconnect from the database [%s] and **Unable to connect to a database [%s]** – it is impossible to connect/disconnect to/form the database. You should check the parameters of the database connection. The analysis of the additional information will help you locate the error.

Database access error [%s]. Stop operations with the database? – the message appears if an error occurs during an attempt to execute an SQL query if the second variant of reacting to errors is selected. The message implies a "Yes" or "No" answer. The analysis of the additional information will help you locate the error.

Unable to verify your SQL script [%s] – the message appears when an attempt to analyze your SQL query fails. Check if the syntax of your SQL query is correct.

Tested successfully – the message appears if your database connection is successfully tested. It requires no additional actions.

Database isn't used – the message appears if the module is temporarily disabled (the "Temporarily disabled" check box is selected) or the database name field is empty. Check the connection parameters.

Database isn't selected - the message appears if the database type is not selected. Check the connection parameters.

Database: %s – %s contains the database name. The message appears if the database connection is successful. Usually, you see it when you call the module for the first time. It requires no additional actions.

Invalid data block length (columns=%d,length=%d) – an internal application error. It means that the data sent by the parser is in an invalid format. Perhaps, you are using the module incompatible with the version of the Advanced Serial Data Logger kernel. Update the versions of both the kernel and the module.

The time of connection is not due yet (%d,%d) – the message appears during an attempt to connect to the database after the connection to it has been lost and the "Reconnect after" option is enabled. No additional actions are required.

Invalid procedure call. Bad arguments –an attempt to call the module using invalid parameters. Perhaps, you are using the module incompatible with the version of the Advanced Serial Data Logger kernel. Update the versions of both the kernel and the module.

Writing to the database is complete - the message appears if your queue of SQL queries is successfully executed. It requires no additional actions.

Writing to the database is complete with errors – the message appears if the executing your queue of SQL queries was interrupted by an error. It requires no additional actions.

Your SQL is empty. Please, specify some SQL text first – the message appears if you do not enter the text for your SQL query. Check if the options on the "SQL queue" tab are configured correctly.

Invalid temporary path – the path to the temporary file specified by you does not exist. Enter a new path in the "Temporary folder" field on the "Errors handling" tab.

%s, %d – will be replaced by additional information.