ibaPDA-Interface-PostgreSQL

Write/read access to databases

Manual

Issue 1.2

Measurement Systems for Industry and Energy

www.iba-ag.com

Manufacturer

iba AG

Koenigswarterstr. 44

90762 Fuerth

Germany

Contacts

Main office	+49 911 97282-0
Fax	+49 911 97282-33
Support	+49 911 97282-14
Engineering	+49 911 97282-13
E-mail	iba@iba-ag.com
Web	www.iba-ag.com

The content of this publication has been checked for compliance with the described hardware and software. Nevertheless, discrepancies cannot be ruled out, and we do not provide guarantee for complete conformity. However, the information furnished in this publication is updated regularly. Required corrections are contained in the following regulations or can be downloaded on the Internet.

The current version is available for download on our web site www.iba-ag.com.

Version	Date	Revision - Chapter / Page	Author	Version SW
1.2	08-2021	Command timeout (4.3.2), buffer (4.3.3), diagnostic module (5.4)	rm	7.3.0

Content

1 About this Manual

This document describes the function and application of the software interface

ibaPDA-Interface-PostgreSQL

This documentation is a supplement to the *ibaPDA* manual. Information about all the other characteristics and functions of *ibaPDA* can be found in the *ibaPDA* manual or in the online help.

1.1 Target group and previous knowledge

This documentation addresses qualified professionals, who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as a professional if he/she is capable of assessing the work assigned to him/her and recognizing possible risks on the basis of his/her specialist training, knowledge and experience and knowledge of the standard regulations.

This documentation in particular addresses persons, who are concerned with the configuration, test, commissioning or maintenance of supported databases, cloud or cluster storage technology. For the handling of *ibaPDA-Interface-PostgreSQL* the following basic knowledge is required and/or useful:

- Basic knowledge of *ibaPDA*
- Basic knowledge of databases, cloud or cluster storage technology
- Application of SQL statements

1.2 Notations

In this manual, the following notations are used:

Action	Notation
Menu command	Menu *Logic diagram*
Calling the menu command	*Step 1 – Step 2 – Step 3 – Step x* Example: Select the menu *Logic diagram - Add - New function block*.
Keys	<Key name> Example: <Alt>; <F1>
Press the keys simultaneously	<Key name> + <Key name> Example: <Alt> + <Ctrl>
Buttons	<Key name> Example: <OK>; <Cancel>
File names, paths	"Filename", "Path" Example: "Test.doc"

1.3 Used symbols

If safety instructions or other notes are used in this manual, they mean:

Danger!

The non-observance of this safety information may result in an imminent risk of death or severe injury:

- Observe the specified measures.

Warning!

The non-observance of this safety information may result in a potential risk of death or severe injury!

- Observe the specified measures.

Caution!

The non-observance of this safety information may result in a potential risk of injury or material damage!

- Observe the specified measures

Note

A note specifies special requirements or actions to be observed.

Tip

Tip or example as a helpful note or insider tip to make the work a little bit easier.

Other documentation

Reference to additional documentation or further reading.

2 System requirements

The following system requirements are required to use the SQL interface to a database of the PostgreSQL type:

- *ibaPDA* v7.2.2 or higher
- License for *ibaPDA-Interface-PostgreSQL*

For other requirements regarding the used computer hardware and the operating systems, please refer to the *ibaPDA* documentation.

License information

Order no.	Product name	Description
31.001115	ibaPDA-Interface-PostgreSQL	Extension license for an *ibaPDA* system for reading from and writing to databases of the PostgreSQL type. License for 8 SQL modules according to 8 SQL statements.
31.101115	one-step-up-Interface-PostgreSQL	Extension license for additional 8 SQL modules. Max. 3 permissible (32 SQL modules total)

3 The SQL interface

3.1 General information

The SQL interface of *ibaPDA* forms the basis for a series of database interfaces via which *ibaPDA* can read data both from databases as well as write data in databases.

The connections to one or more databases are configured and managed with the SQL interface. One or more connections can be configured per database. All DB connections can be used simultaneously. The databases can be of the same or different types.

Database systems of different vendors are supported:

- Maria DB
- MySQL
- Oracle [1)]
- PostgreSQL
- SAP Hana[2)]
- SQL Server

The data exchange with the databases occurs via corresponding SQL modules, which are configured according to the specific database type.

The data is accessed with SQL statements.

- Input direction (read)
  - For the reading access, custom SQL queries (e.g. SELECT) are used and their results are mapped to input signals in *ibaPDA*.
  - Optional parameters assigned to signals can be used in the queries in order to thus change the queries dynamically during the ongoing acquisition.
- Output direction (write)
  - SQL commands (e.g. UPDATE, INSERT) are used for writing access in order to write data from *ibaPDA* into the database.
  - In the statements, optional parameters assigned to signals can be used in order to therefore write signal actual values.

The execution of queries and commands can be cyclical or triggered.

1) DB client installation "Oracle Data Provider for .NET [ODP.NET]" required on ibaPDA server
2) DB client installation ".NET data provider" required on ibaPDA server

3.2 Comparison to the DB/cloud data storage

In addition to quick measured values that are acquired via other interfaces, the SQL interface serves mainly to acquire additional data about a process, a plant or a product or to output current values, operating statuses or calculated values to a database. The access may occur by request or cyclically, whereby the response time of the database should always be taken into consideration.

The corresponding data storage (e.g. *ibaPDA-Data-Store-SQL-Server*, *-Oracle* etc.) is to be used for faster continuous data writing in databases or cloud systems.

SQL interfaces and DB data storage support the same database types so that you can choose.

SQL interface	DB/cloud data storage
■ Configuration in the I/O manager (read/write)	■ Configuration as data storage (write only)
■ Access to any existing custom table/view is possible ■ The user must provide tables, e.g. by SQL statements	■ Fix table structure, specified by the data storage profile ■ Table is automatically generated by *ibaPDA*.
■ Cyclical/triggered reading with custom SQL queries ■ Cyclical/triggered inserting or updating with custom SQL commands	■ Continuous/triggered writing of signal trends as time series data
	■ Reading the signal trends from the database with SQL trend queries in *ibaAnalyzer* ■ Reading the signal trends from the database with *ibaDaVIS*

Table 1: Differences between SQL interface and DB/cloud data storage

3.3 System Topologies

The connections between databases and *ibaPDA* can be established via standard Ethernet interfaces of the computer.

Note

If highly cyclical accesses or very large data volumes are expected, it is recommended to implement TCP/IP communication on a separate network segment in order to rule out a mutual interference by other network components, such as PLC systems.

3.4 Basic procedure

1. Check whether additional software and/or licenses of the database manufacturer are required for the planned database connection.
2. If necessary, install the required software on the *ibaPDA* computer, e.g. the software "Oracle Data Provider for .NET (ODP.NET)" for Oracle connections
3. In *ibaPDA*, open the I/O manager, *Hardware* area, and configure the desired database connection(s) on the SQL interface.
   See ➚ *Create a database connection*, page 13

For database queries (read access)

1. Under the SQL interface, add a module "SQL query" and select one of the previously configured database connections for the module.
2. Formulate an SQL statement for the database query with or without parameters.
3. If you use parameters: Assign the the corresponding analog and/or digital signals to the parameters whose values you want to use for the query.
4. Configure the analog and/or digital input signals.

For database commands (write access)

1. Switch to the *Outputs* area in the I/O manager.
2. Under the SQL interface, add a module "SQL command" and select one of the previously configured database connections for the module.
3. Formulate an SQL statement as an SQL command with or without parameters.
4. If you use parameters: Assign the the corresponding analog and/or digital signals to the parameters whose values you want to write in the database.

Caution

The network and DB can be overloaded by highly cyclical DB accesses and/or very large data volumes. In the SQL statements, any SQL commands can be used and therefore cause damage (e.g. "delete table," "drop database," etc.).

Therefore...

- Only have qualified users create/change SQL statements (-> user management, security administration).
- Involve your DB administrator for questions about configuring the SQL statements. For example, the DB user used in *ibaPDA* for the DB connection should only have the rights that are actually required.
- Set an appropriate update time (as short as necessary, as long as possible).
- Assess and test the results of your SQL statement.

4 Configuration and engineering ibaPDA

Open the I/O manager, e.g., from the toolbar .

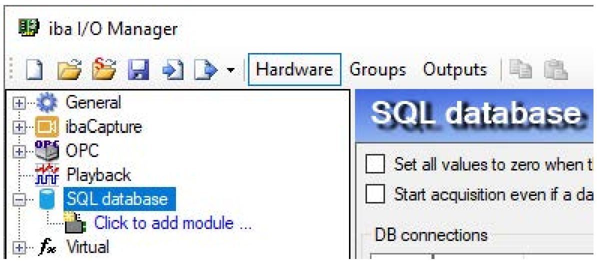


Fig. 1: SQL Interface in the I/O Manager

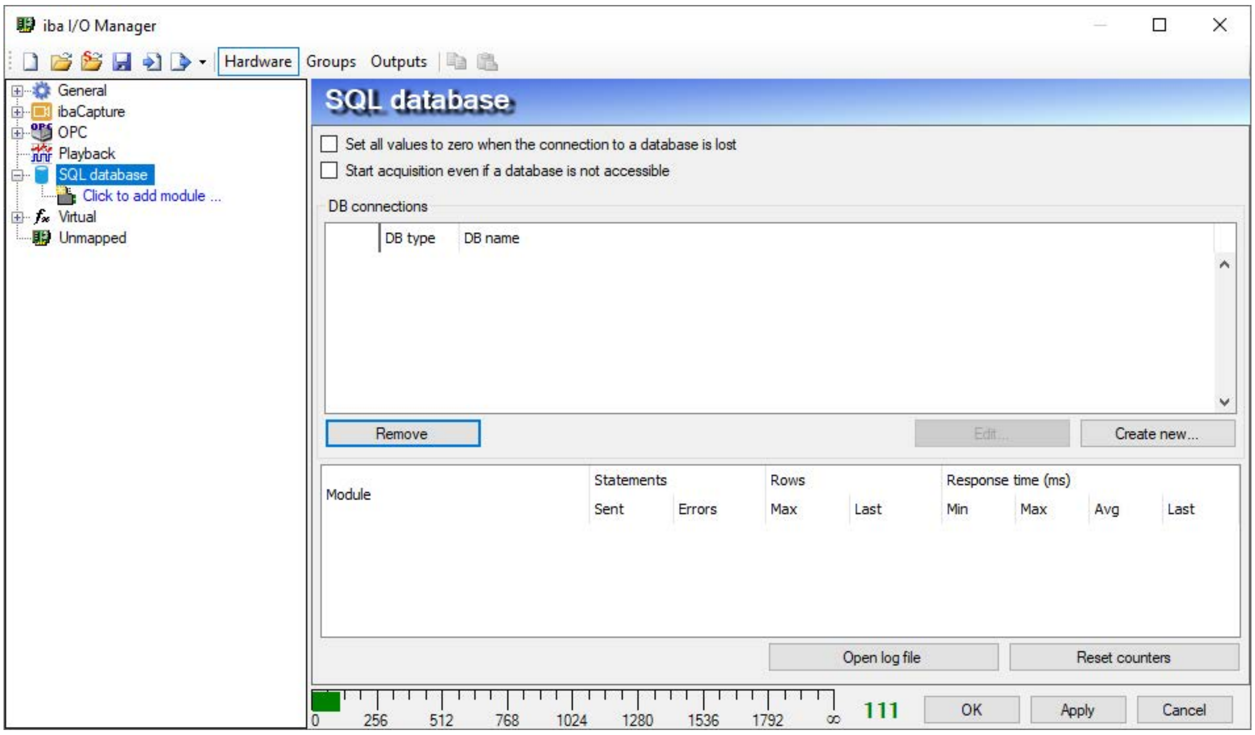
If all system requirements are met, the SQL interface will be displayed in the signal tree.

Note

The SQL interface is visible as soon as one of the SQL interface licenses (SQL Server, Oracle, PostgreSQL, MySQL, SAP HANA) is available on the dongle. Even if only one database type is licensed, all supported DB types appear in the SQL interface configuration. Nevertheless, only the licensed type can be used for data acquisition.

4.1 Basic settings of the interface

The interface has the following functions and configuration options.



Set all values to zero when the connection to a database is lost
Start acquisition even if a database is not accessible

DB connections

DB type	DB name

Module	Statements Sent	Statements Errors	Rows Max	Rows Last	Response time (ms) Min	Response time (ms) Max	Response time (ms) Avg	Response time (ms) Last

Fig. 2: SQL interface settings

Set all values to zero when the connection to a database is lost.
If this option is enabled, all signal values in the modules that are assigned to an interrupted database connection are set to zero. *ibaPDA* regularly attempts to restore the database connection until it either succeeds or the acquisition stops.

If this option is not enabled, then the last values at the time of the connection interruption are preserved.

Note

It may take some time until the signal values are set to zero. A longer response time of a database, even in the range of several seconds, is not necessarily an indication of an interrupted connection.

Start acquisition even if a database is not accessible
If this option is enabled, the acquisition will start even if one or more of the configured databases are not accessed. In case of an error, a warning is output in the validation dialog. If the system has been started without a connection to a database, *ibaPDA* will then regularly try to connect to the database.

DB connections
The configured database connections are clearly displayed in this area. Each database connection is clearly identified with an ID number, the database type and the connected database. You can use the buttons to create, edit and delete database connections. The coming chapters describe the configuration of the database connections.

Table with connection statistics
The bottom area of the dialog contains a table for the purpose of connection diagnostics. Statistical information, such as counter values and times, are displayed for each configured SQL module.

Note

Due to the nature of relational databases, error counters may increment more slowly than success counters. Note this when diagnosing connection problems.

Column	Meaning
Module	Name of the SQL module (query and command modules)
Statements	Sent: Number of executed SQL statements Error: Number of SQL statements where failures occurred. Either SQL statements failed (database responds with error) or statements could not be executed, because the database was not accessible and the buffer was full. An empty result for a statement is not considered an error.
Rows	Max: The highest number of rows that was returned since the last start of acquisition or the last reset of the counter after a statement. Last: The number of rows that was returned after the last statement.
Response time	Time values for the duration between the execution of the SQL statement and the confirmation of the execution by the database. Min, Max, Avg: These values relate to the executed statements since the start of the acquisition or the last counter reset. Last: Response time of the last executed statement.

Table 2: Columns of the connection diagnostics

<Open log file> button
If connections to databases have been established, all connection-specific operations are logged in a text file. Using this button, you can open and see this file. In the file system on the hard disk, you will find the log file in the program path of the ibaPDA server (...\Programs\iba\ibaPDA\Server\Log\). The file name of the current log file is `sqlInterfaceLog.txt`; the name of the archived log files is `sqlInterfaceLog_yyyy_mm_dd_hh_mm_ss.txt`.

Button <Reset counters>
If you would like to reset the counters for all modules and connections, click on the <Reset counters> button.

4.2 Create a database connection

Before you can exchange data with a database, you must first configure a connection. To do this, select the interface *SQL Database* in the interface tree and click on the button <Create new...>.

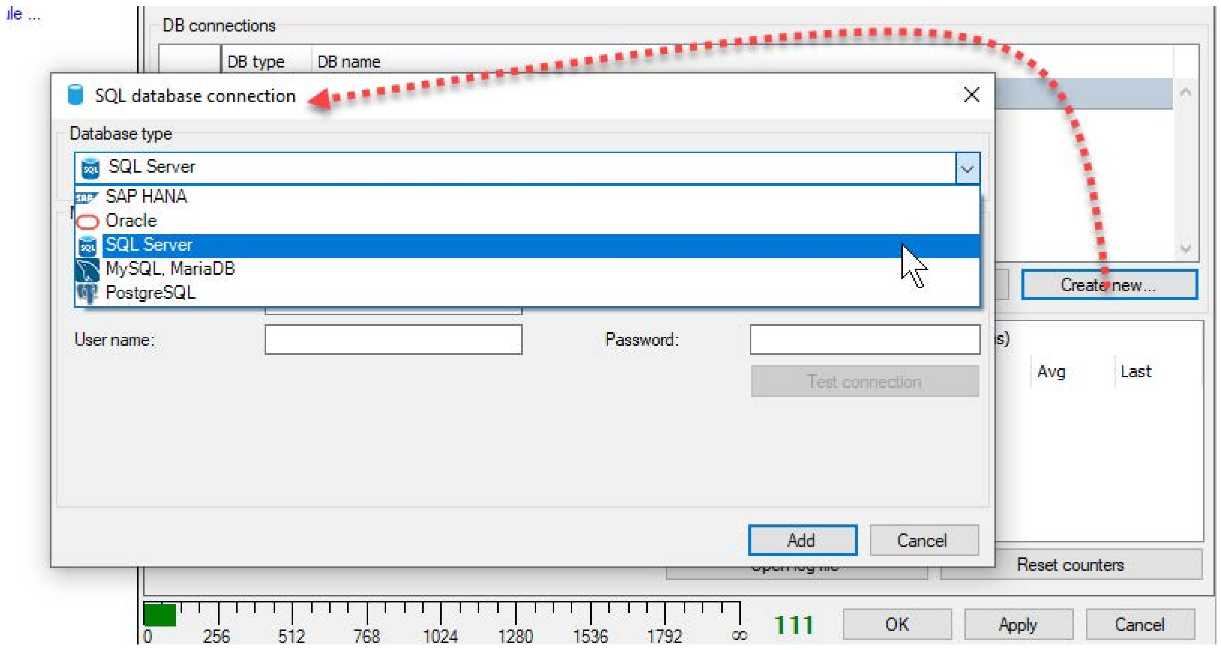


Fig. 3: Selection of the database type to create a new database connection

First of all, select the the desired database type.

The lower part of the dialog changes depending on which type you select.

Then in the lower part of the dialog, enter the information about the database or select it (e.g. DB server, database name, authentication method, possibly user name and password).

You can find more details in the next chapter.

By specifying this information, a name is automatically generated for this connection and displayed in the field *Connection name* (*user name@database*). You can overwrite the name.

In the case of multiple connections to the same type, you can distinguish the connection using this name. In addition, you select the DB connection later using this name during the module configuration.

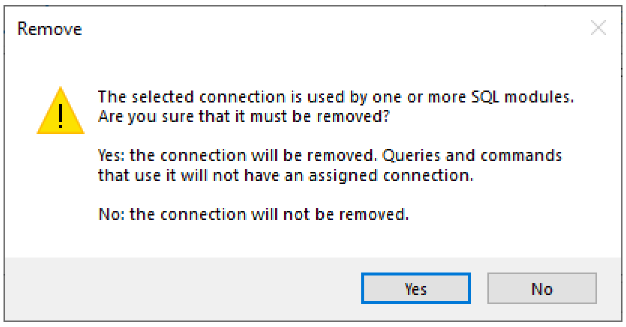
The connection name is displayed in the table with the DB connections in the “DB name” column.

You can edit DB connections at any time once they have been created by selecting the respective connection in the table and clicking on the <Edit...> button.

Use the <Remove> button to delete the selected DB connection.

Note

If a database connection is to be deleted that is already assigned to one or more modules, a warning appears:



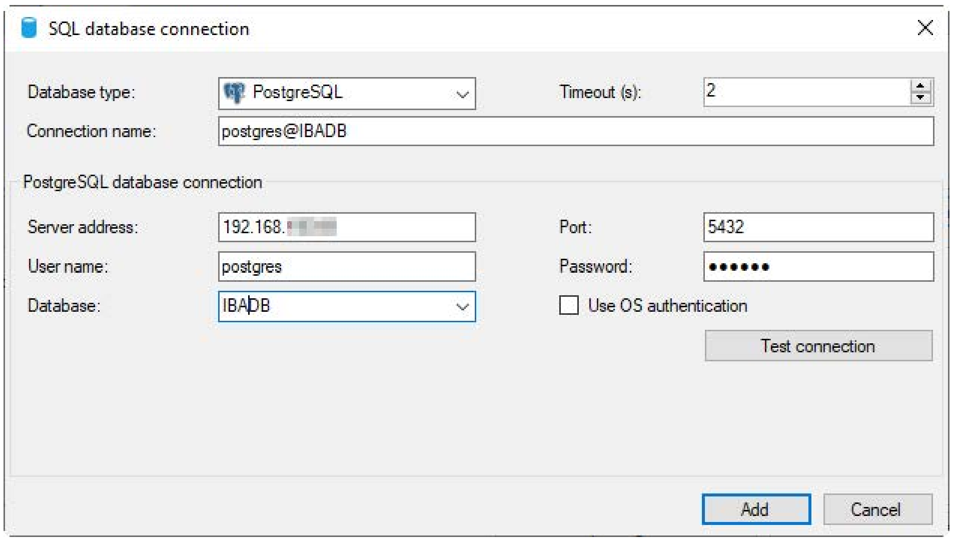
The selected connection is used by one or more SQL modules.
Are you sure that it must be removed?

Yes: the connection will be removed. Queries and commands that use it will not have an assigned connection.

No: the connection will not be removed.

Since SQL modules are linked to a DB connection in the SQL interface, it is important to pay attention to the fact that the modules do not lose their reference to the connection. Otherwise the SQL statements will have no effect.

4.2.1 PostgreSQL DB connection



SQL database connection

Database type:	PostgreSQL	Timeout (s):	2
Connection name:	postgres@IBADB		

PostgreSQL database connection

Server address:	192.168.[redacted]	Port:	5432
User name:	postgres	Password:	••••••
Database:	IBADB	☐ Use OS authentication	

Fig. 4: Creating/editing a DB connection to PostgreSQL

Database type
In the drop-down menu, select your database type. Here it is PostgreSQL.

Timeout
Waiting time for a database connection to be established. If the connection to the database cannot be successfully established within the time set here, the connection attempt is aborted and an error message is output.

This concerns connection setups in the following situations:

- Testing the connection
- Test SQL statement
- Validation when starting the acquisition
- Database accesses during acquisition

Connection name
This line contains the name of the database connection. It will be automatically formed and entered according to the scheme *username@database* as soon as you have filled in the fields below. You can also overwrite the connection name.

You can use this name to identify the database connection in the overview table at the interface level. The connection name is there in the "DB name” column.

Server address
Enter the IP address or the host name of the database server.

Port
This is the port utilized by the PostgreSQL server for communication. Usually, the default value 5432 can remain. If another port is required you may enter it here.

User name/Password
These fields are only available if the option *OS-Authentication* is not selected. Enter here the required login data for the database. If necessary, inquire the correct data from your DB administrator.

Database
Enter the database here. If the server address, the authentication or the user name and password have been entered, the drop-down list shows all available databases for the connection and one can be selected.

Use OS authentication
If you enable this option, the user account the *ibaPDA* server is running on is used. This is normally the system account. Please check with your DB administrator whether this method is appropriate or not.

<Test connection>
By clicking on <Test connection> *ibaPDA* checks if an access to the selected database on the set DB server is possible. If the connection is successful, the version of the database is also output in the message box.

4.3 SQL modules

A distinction is made between input and output modules in SQL modules, which can be configured under the SQL interface.

An SQL module is either a reading module with an SQL query or a writing module with an SQL command. That is why the terms SQL query module and SQL command module are used in the following text as well.

A module cannot be both simultaneously. That is why there is also no SQL module that can be seen simultaneously in the I/O manager in both areas of *Hardware* and *Outputs*.

However, the modules are very similar in their structure and components.

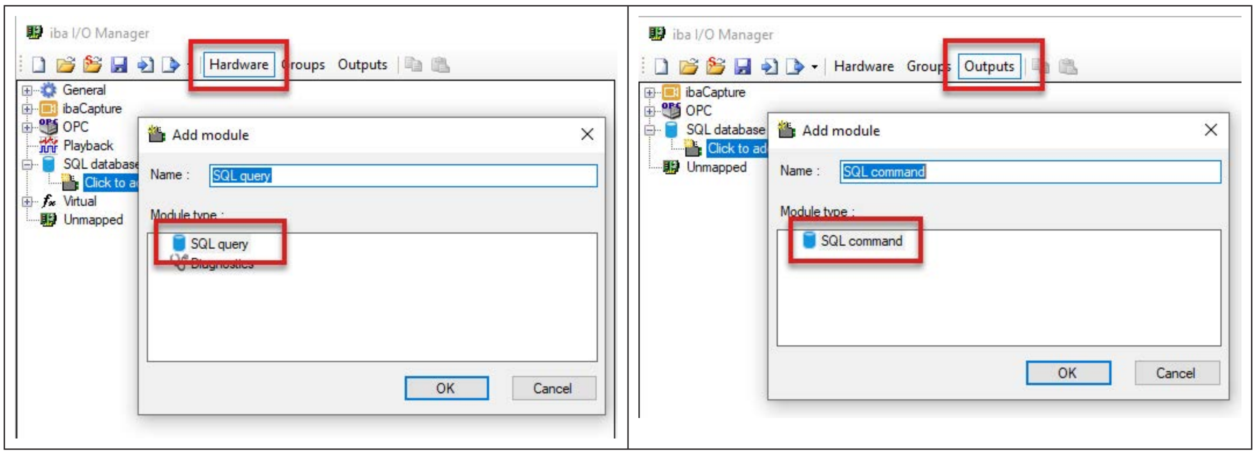
All SQL modules...

- ... have a reference to a database connection in the SQL interface.
- ... contain an SQL statement (query or command),
- ... offer an assignment table for parameters used in the SQL statements and
- ... can work cyclically or triggered.

4.3.1 Add SQL module

In the I/O manager, select the area *Hardware* or *Outputs*, depending on whether you want to add a reading or writing module.

Under the SQL interface, click on *Click to add module...*



Name : SQL query

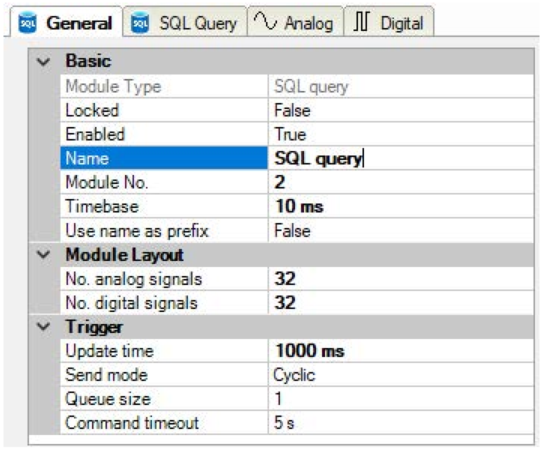
Module type : SQL query

Name : SQL command

Module type : SQL command

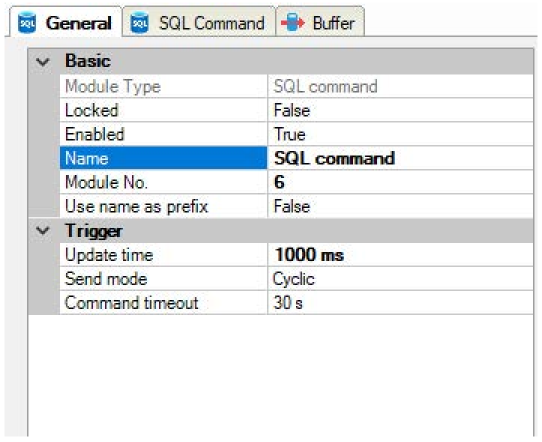
Table 3: Add an SQL query module (left) or SQL command module (right)

4.3.2 General module settings



General | SQL Query | Analog | Digital

Basic	
Module Type	SQL query
Locked	False
Enabled	True
Name	**SQL query**
Module No.	**2**
Timebase	**10 ms**
Use name as prefix	False
Module Layout	
No. analog signals	**32**
No. digital signals	**32**
Trigger	
Update time	**1000 ms**
Send mode	Cyclic
Queue size	1
Command timeout	5 s



General | SQL Command | Buffer

Basic	
Module Type	SQL command
Locked	False
Enabled	True
Name	**SQL command**
Module No.	**6**
Use name as prefix	False
Trigger	
Update time	**1000 ms**
Send mode	Cyclic
Command timeout	30 s

Table 4: General module settings for input modules (left) and output modules (right)

Basic settings

Module Type (information only)
Indicates the type of the current module.

Locked
A module can be locked to avoid unintentional or unauthorized changing of the module settings.

Enabled
Disabled modules are excluded from signal acquisition.

Name
The plain text name should be entered here as the module designation.

Module No.
Internal reference number of the module. This number determines the order of the modules in the signal tree of *ibaPDA* client and *ibaAnalyzer*.

Timebase
All signals of the module will be sampled on this time base.

Use name as prefix
Puts the module name in front of the signal names.

Module structure (only input modules)

Number of analog signals, number of digital signals
You can increase or reduce the number of signals for this module here (the standard setting is 32). You may enter any value between 0 and 10000. The signal tables will be adjusted accordingly.

Trigger

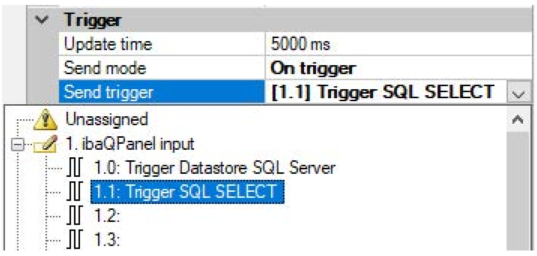
Update time
You can use the update time to determine ...

- ... in trigger mode "cyclic" the execution interval of the SQL statements
- ... in the "on change" and "on trigger" mode the dead time after executing a statement

Enter the value in milliseconds.

Send mode
Set here how the SQL statements (queries and commands) should be executed.

- Cyclic: In this simplest mode, the SQL statement is periodically executed. The update time specifies the period here. The SQL statement is executed again whenever the update time has expired.
- On change: When selecting this mode, another line "Send trigger" appears. You can select any analog or digital signal here that is to serve as the trigger. In case of a change to the signal value, the SQL statement is executed. After a triggering, all other triggering value changes are ignored until the update time has expired. Only then can the SQL statement be executed again.
- On trigger: When selecting this mode, another line "Send trigger" appears. You can select any digital signal here that is to serve as the trigger. In case of a rising edge of the signal, the SQL statement is executed. After a triggering, all other triggering triggers are ignored until the update time has expired. Only then can the SQL statement be executed again.


Trigger

Update time	5000 ms
Send mode	On trigger
Send trigger	[1.1] Trigger SQL SELECT

Fig. 5: Example of a DB query depending on a QPanel input

Send trigger (if send mode "On change" or "On trigger" is selected)
In the "On change" mode: Select an analog or digital signal here for which the SQL statement is to be executed if it changes.

In the "On trigger" mode: Select a digital signal here for which the SQL statement is to be executed in case of its rising edge.

Queue size
The queue serves as a buffer for SQL statements in the following cases:

- The database connection is interrupted.
- The execution of an SQL statement takes long while new SQL statements are already triggered in *ibaPDA* before the execution has been finished.

Use the queue size to determine how many SQL statements should be intermediately stored as a maximum. If this number is exceeded, the oldest SQL statements are deleted.

- With input modules, the queue is 1 by default:
  Since the results of an SQL query are mapped to signals, in most cases it does not make sense to make up for queries that were not executed due to an interrupted connection. The SQL queries buffered in the queue would then be executed in quick succession and the results transmitted to the signals. With the queue size 1, only the last triggered SQL query is always kept available.
- With output modules, the queue size is 1000 by default:
  Production data, for example, can be written via SQL commands. It is important here that these are also buffered and made up for in case of a connection breakdown.

Command timeout
By setting the command timeout you determine the maximum time a statement can take to be executed. If the query or command is not completed before the command timeout has elapsed, the execution is aborted. The default settings of query and command differ:

- SQL query: 5 s
- SQL command: 30 s

4.3.3 Buffer

In output direction the interface uses a memory buffer and additionally a file buffer that can be enabled optionally.

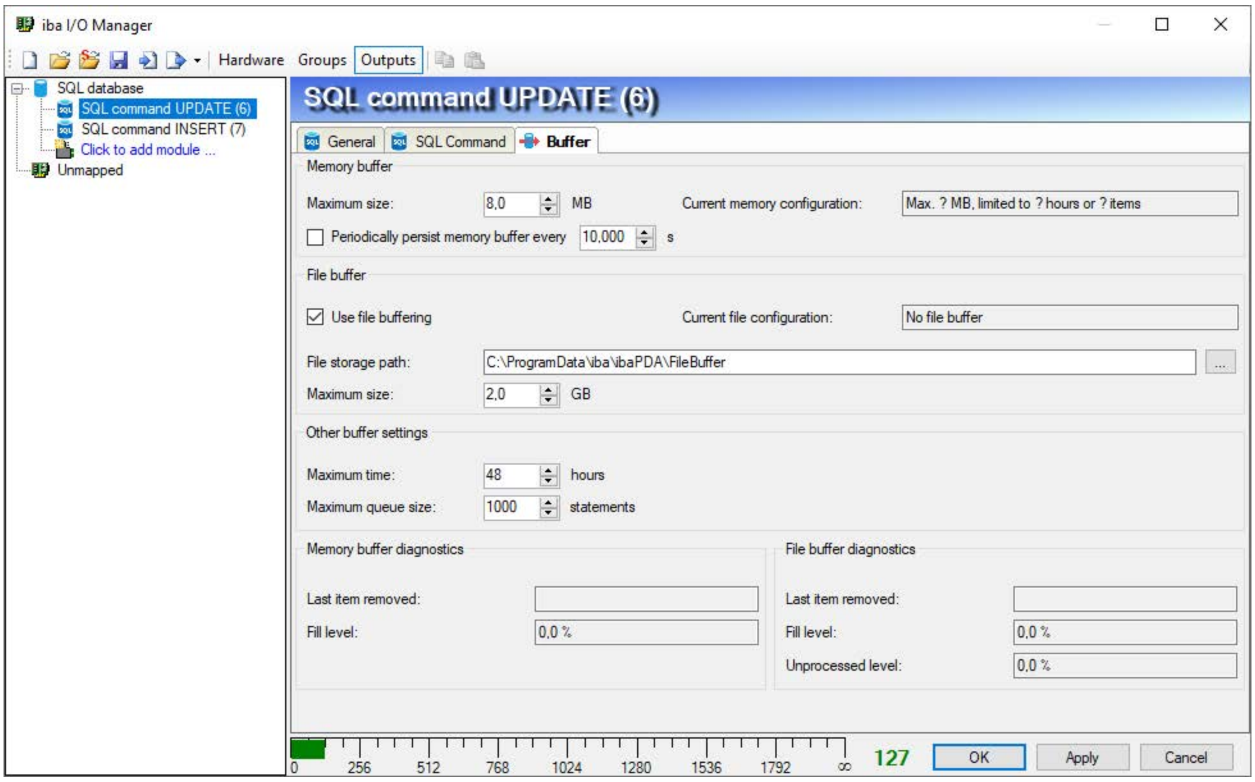
Data to be sent to the target system always passes through the internal *ibaPDA* memory buffer. If the connection to the target system exists, the data is sent there immediately. If the connection is lost, or the data cannot be sent out fast enough, the data is temporarily stored in the memory buffer. The memory buffer is located in the RAM of the *ibaPDA* computer and is therefore limited and volatile. If, for example, the acquisition is restarted, the buffered data will be lost. If the memory buffer grows beyond the configured size during ongoing acquisition, the oldest values are deleted and thus lost.

To improve this, a file buffer can additionally be enabled, which can buffer much larger amounts of data. The data is stored in files in a directory on a local drive of the *ibaPDA* computer. When the file buffer is enabled, data is transferred from the overflowing memory buffer to the file buffer. If the acquisition is stopped or restarted (e.g. by applying a modified I/O configuration), data that may be in the memory buffer at this time is also transferred to the file buffer.

After reconnecting to the target system, the oldest data is always sent first. Newer values are added to the buffer in the meantime. If there is still buffered data in the file buffer when the acquisition is started, it will be handled and processed in the same way.

In terms of SQL statements this means that the statements are buffered according to the configuration which was valid at the time of buffering. If the SQL statement was changed during an ongoing buffering (change and apply in the I/O Manager), the system will still execute the old statements as generated at the time of buffering after the connection has been reestablished.

You configure the buffering on the *Buffer* tab in the *Outputs* section of the SQL interface.



Memory buffer

Maximum size: 8,0 MB

Current memory configuration: Max. ? MB, limited to ? hours or ? items

☐ Periodically persist memory buffer every 10.000 s

File buffer

☑ Use file buffering

Current file configuration: No file buffer

File storage path: C:\ProgramData\iba\ibaPDA\FileBuffer

Maximum size: 2,0 GB

Other buffer settings

Maximum time: 48 hours

Maximum queue size: 1000 statements

Memory buffer diagnostics

Last item removed:

Fill level: 0,0 %

File buffer diagnostics

Last item removed:

Fill level: 0,0 %

Unprocessed level: 0,0 %

Fig. 6: Configuration of buffering, example SQL-Server

Memory buffer

The memory buffer is always enabled. It cannot be deactivated, since data to be transmitted always passes through the buffer before being forwarded to the target system.

Maximum size
Enter here the maximum total size for items buffered in memory. If the maximum size is exceeded, there are 2 options:

- When file buffering is disabled, the oldest item in memory is deleted (and is lost forever).
- When file buffering is enabled, the oldest part of the buffer memory is moved to a buffer file.

Periodically persist memory buffer every ... s
This option can be enabled only if file buffering is enabled. If the option is enabled, the entire memory buffer is periodically swapped to a buffer file.

Enter a duration after which the memory buffer is periodically stored. It must be between 10 s and 600 s.

With this option you can ensure that as little data as possible is lost in case of a system failure.

Current memory configuration
Display of the approximate time period that can be temporarily stored in the memory buffer with the configured settings. Specification in d.hh:mm:ss.

File buffer

Use file buffering
By default, the file buffer is not used. Here you can enable file buffering.

Current file configuration
Display of the approximate time period that can be temporarily stored in the file buffer with the configured settings. Specification in d.hh:mm:ss.

File storage path
In the *File storage path* field you can select a location for the files. You can enter the directory directly into the text field, or select it via the browse button <...>. The configured file directory must be located on a local hard disk of the *ibaPDA* server computer.

The same file directory can be used for several SQL command modules, because the buffer files have a unique name. Files from different SQL command modules can thus be distinguished by their name.

Maximum size
You can configure the maximum total size of the buffer files of an SQL command module. The buffer files themselves have the file extension *.buf*, the index file for managing the buffer files has the extension *.info*. The maximum size is the total size of all these files. If the maximum buffer size is exceeded, the oldest buffer file is deleted.

Other buffer settings

Maximum time
Stored data older than the maximum time will not be transferred to the target system. Files older than the maximum time can be deleted. You may enter any value between 1 and 1000 hours.

Maximum queue size
Here you can set the limit for the number of buffered SQL statements. Depending on the application it may be more reasonable to give a limited number of statements instead of a time limit. However, the buffer size is determining.

Memory buffer diagnostics / File buffer diagnostics

Last item removed
Indicates when the last item was taken from this part of the buffer.

Fill level
The fill level indicates what percentage of the buffer size is currently filled with buffered data.

Unprocessed level
Items transferred to the target system are not deleted immediately in the file buffer. Only when a buffer file is completely read, it is deleted. Therefore, it is possible that only a part of a buffer file contains data that has not yet been transferred. The fill level refers to the existing buffer files, while the "unprocessed level" indicates the percentage of data in the file buffer that has not yet been transferred.

Tip

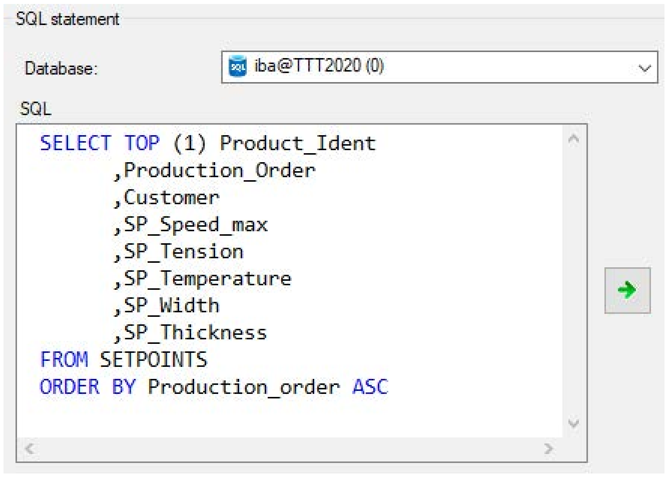
If you configure a diagnostic module for an SQL command module in the I/O configuration, you can monitor and record the fill level values of the memory buffer and file buffer.

See also ➚ *Diagnostic modules*, page 33

4.4 SQL statements

SQL statements used for SQL queries are primarily SELECT instructions and for SQL commands INSERT, UPDATE or DELETE instructions.

Enter the SQL statement in the left window of the *SQL query* tab or *SQL command* tab of the module. You can write the statement directly in the window or paste using copy & paste.



```
SELECT TOP (1) Product_Ident
      ,Production_Order
      ,Customer
      ,SP_Speed_max
      ,SP_Tension
      ,SP_Temperature
      ,SP_Width
      ,SP_Thickness
FROM SETPOINTS
ORDER BY Production_order ASC
```

Fig. 7: Example of an SQL query

Only use instructions that can also be executed by the connected database server. The syntax of the SQL statement must be correct and written in the SQL dialect of the database type. *ibaPDA* does not offer any plausibility check of the SQL statement!

ibaPDA highlights some syntax elements in color to provide support:

- Keywords/instructions/operations: blue
- Comments: green
- Optional SQL parameters: magenta

The highlighting differs from DB type to DB type, because every SQL dialect has different key-words or usage forms of parameters. For example, parameters are marked with a colon ":" for Oracle and they are marked with an at sign "@" for SQL Server.

One SQL statement is permitted per module.

A distinction is made between SQL statements with and without parameters. With an SQL query without parameters, you get the current values of the respective queried columns from the da-tabase as the result and assign this directly to the module signals.

You can use parameters to dynamically link the query to certain conditions in order to create a differentiated query.

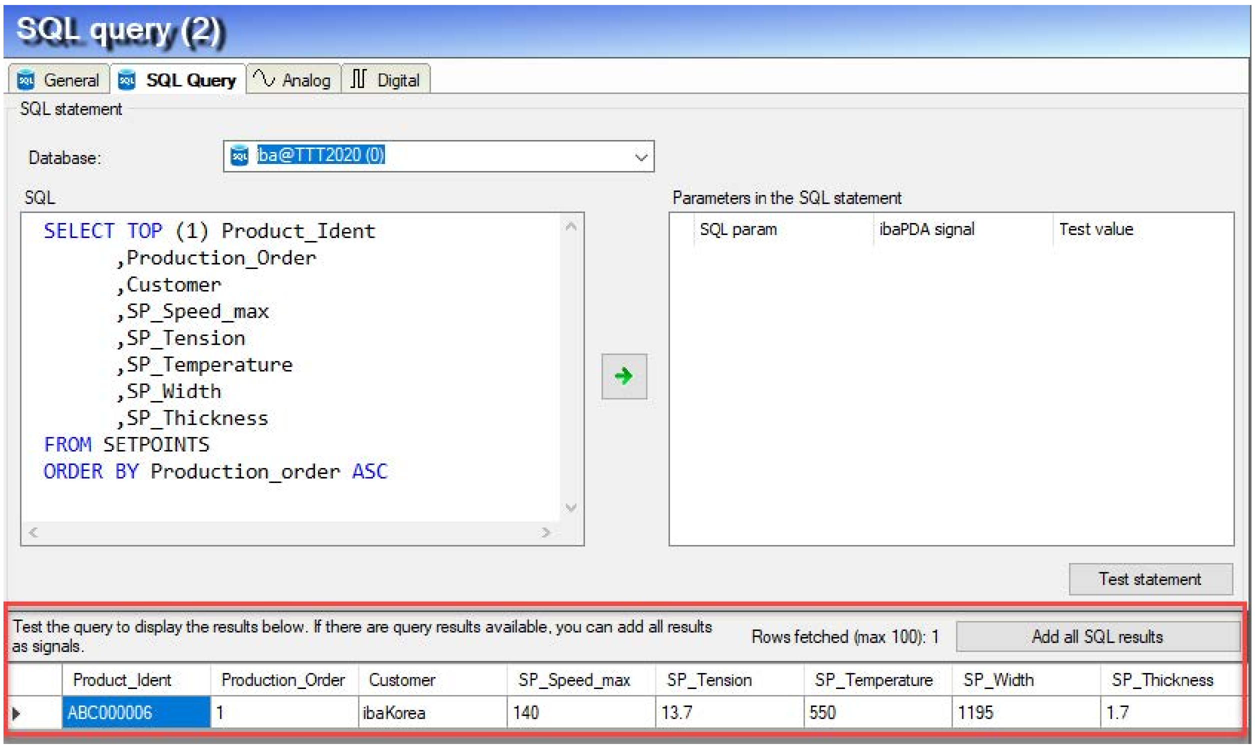
4.4.1 Test SQL statements

Both SQL statements for queries as well as for commands can be tested by clicking on the <Test statement> button.

Technically, the statement is executed in a transaction during the test, which is then undone subsequently (rollback). In a normal case, this avoids the tested execution of an SQL statement leading to changes in the database. This is especially important for writing SQL commands.

Depending on the complexity of the SQL statement and the structure of the database, however, it cannot be completely ruled out that changes remain in the database after the rollback.

Unlike SQL command modules (outputs), there is an area with SQL query modules at the bottom in the *SQL query* tab where the test results are displayed.



Database: iba@TTT2020 (0)

SQL

```
SELECT TOP (1) Product_Ident
      ,Production_Order
      ,Customer
      ,SP_Speed_max
      ,SP_Tension
      ,SP_Temperature
      ,SP_Width
      ,SP_Thickness
FROM SETPOINTS
ORDER BY Production_order ASC
```

Parameters in the SQL statement

SQL param	ibaPDA signal	Test value

Test the query to display the results below. If there are query results available, you can add all results as signals.

Rows fetched (max 100): 1

Product_Ident	Production_Order	Customer	SP_Speed_max	SP_Tension	SP_Temperature	SP_Width	SP_Thickness
ABC000006	1	ibaKorea	140	13.7	550	1195	1.7

Fig. 8: Result of the test query

At this point, a maximum of the first 100 rows of the query result is displayed.

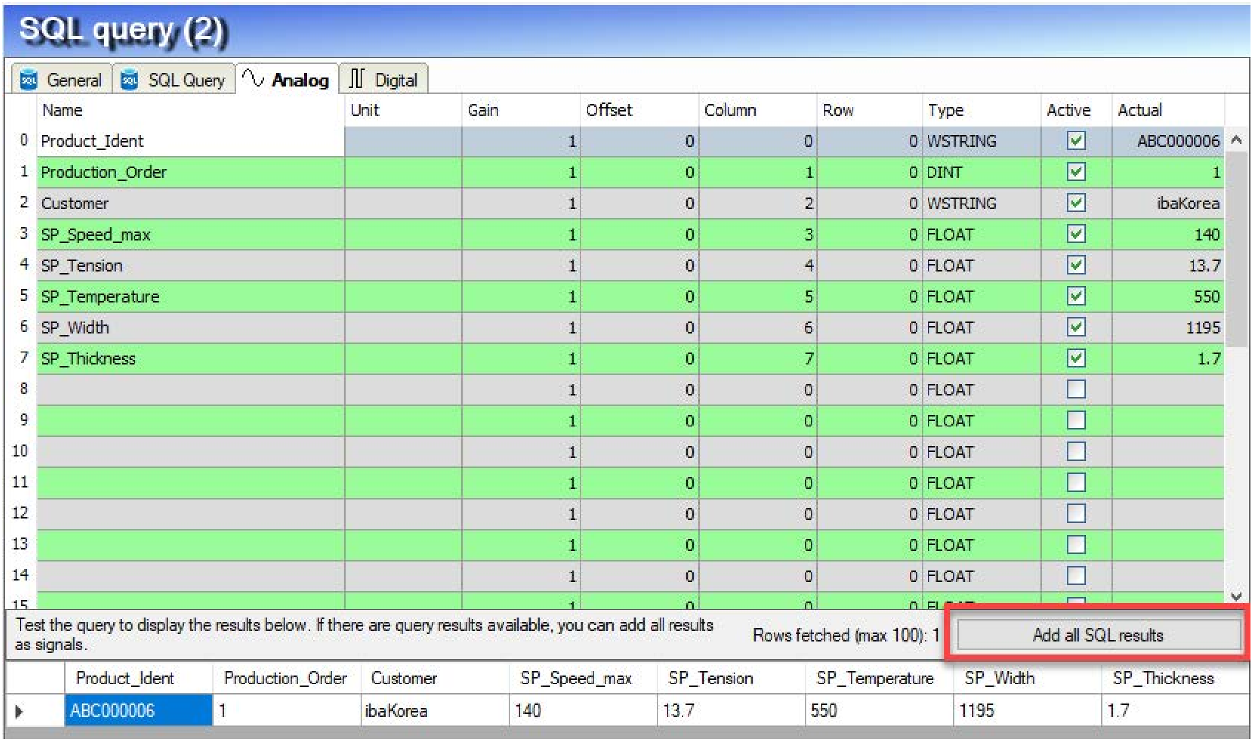
You can also test the statement for SQL commands. However, you only receive feedback as to whether the execution of the statement was successful or not.

4.4.2 Link SQL queries with input signals

For SQL queries, analog and digital signals are always linked with the query. The queried values are thus available as signals for visualization and recording.

You can see the signals in the *Analog* and *Digital* tabs of the SQL query module.

If the query test has been successfully executed, you can click the button <Add all SQL results> to enter the queried values in the signal tables.



	Name	Unit	Gain	Offset	Column	Row	Type	Active	Actual
0	Product_Ident		1	0	0	0	WSTRING	✓	ABC000006
1	Production_Order		1	0	1	0	DINT	✓	1
2	Customer		1	0	2	0	WSTRING	✓	ibaKorea
3	SP_Speed_max		1	0	3	0	FLOAT	✓	140
4	SP_Tension		1	0	4	0	FLOAT	✓	13.7
5	SP_Temperature		1	0	5	0	FLOAT	✓	550
6	SP_Width		1	0	6	0	FLOAT	✓	1195
7	SP_Thickness		1	0	7	0	FLOAT	✓	1.7
8			1	0	0	0	FLOAT	☐	
9			1	0	0	0	FLOAT	☐	
10			1	0	0	0	FLOAT	☐	
11			1	0	0	0	FLOAT	☐	
12			1	0	0	0	FLOAT	☐	
13			1	0	0	0	FLOAT	☐	
14			1	0	0	0	FLOAT	☐	

Test the query to display the results below. If there are query results available, you can add all results as signals.

Rows fetched (max 100): 1

Add all SQL results

	Product_Ident	Production_Order	Customer	SP_Speed_max	SP_Tension	SP_Temperature	SP_Width	SP_Thickness
▸	ABC000006	1	ibaKorea	140	13.7	550	1195	1.7

Fig. 9: Example of the query of a value series (the first) from a table with 8 columns

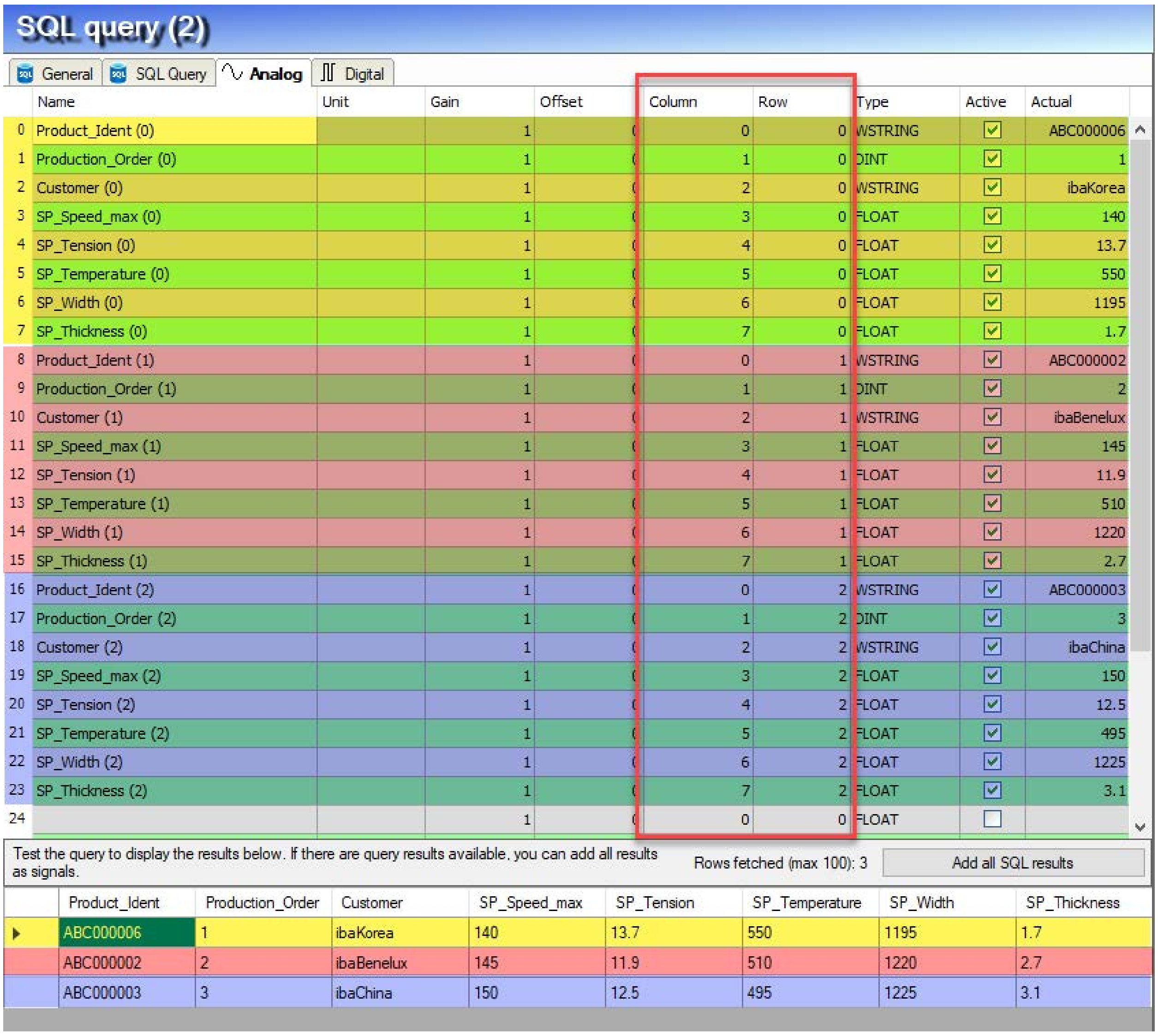
If there are not enough signals available in the module, you will be asked whether you want to increase the number of signals.

The relation of the added signals to the query results will be established via the column and row numbers, which are automatically entered in the signal table when you click on the button <Add all SQL results>. If the query provides more than one result, then the result sets are inserted into the signal table one after the other by row number.

An SQL query of the first 3 rows of a table with the following statement

```
SELECT TOP (3) Product_Ident
      ,Production_Order
      ,Customer
      ,SP_Speed_max
      ,SP_Tension
      ,SP_Temperature
      ,SP_Width
      ,SP_Thickness
FROM SETPOINTS
ORDER BY Production_order ASC
```

would, for example, provide such a result:



	Name	Unit	Gain	Offset	Column	Row	Type	Active	Actual
0	Product_Ident (0)		1	(	0	0	WSTRING	✓	ABC000006
1	Production_Order (0)		1	(	1	0	DINT	✓	1
2	Customer (0)		1	(	2	0	WSTRING	✓	ibaKorea
3	SP_Speed_max (0)		1	(	3	0	FLOAT	✓	140
4	SP_Tension (0)		1	(	4	0	FLOAT	✓	13.7
5	SP_Temperature (0)		1	(	5	0	FLOAT	✓	550
6	SP_Width (0)		1	(	6	0	FLOAT	✓	1195
7	SP_Thickness (0)		1	(	7	0	FLOAT	✓	1.7
8	Product_Ident (1)		1	(	0	1	WSTRING	✓	ABC000002
9	Production_Order (1)		1	(	1	1	DINT	✓	2
10	Customer (1)		1	(	2	1	WSTRING	✓	ibaBenelux
11	SP_Speed_max (1)		1	(	3	1	FLOAT	✓	145
12	SP_Tension (1)		1	(	4	1	FLOAT	✓	11.9
13	SP_Temperature (1)		1	(	5	1	FLOAT	✓	510
14	SP_Width (1)		1	(	6	1	FLOAT	✓	1220
15	SP_Thickness (1)		1	(	7	1	FLOAT	✓	2.7
16	Product_Ident (2)		1	(	0	2	WSTRING	✓	ABC000003
17	Production_Order (2)		1	(	1	2	DINT	✓	3
18	Customer (2)		1	(	2	2	WSTRING	✓	ibaChina
19	SP_Speed_max (2)		1	(	3	2	FLOAT	✓	150
20	SP_Tension (2)		1	(	4	2	FLOAT	✓	12.5
21	SP_Temperature (2)		1	(	5	2	FLOAT	✓	495
22	SP_Width (2)		1	(	6	2	FLOAT	✓	1225
23	SP_Thickness (2)		1	(	7	2	FLOAT	✓	3.1
24			1	(	0	0	FLOAT	☐	

Test the query to display the results below. If there are query results available, you can add all results as signals.

Rows fetched (max 100): 3

	Product_Ident	Production_Order	Customer	SP_Speed_max	SP_Tension	SP_Temperature	SP_Width	SP_Thickness
▸	ABC000006	1	ibaKorea	140	13.7	550	1195	1.7
	ABC000002	2	ibaBenelux	145	11.9	510	1220	2.7
	ABC000003	3	ibaChina	150	12.5	495	1225	3.1

Fig. 10: Query result with three table rows

Tip

For SQL queries with multi-row results, use the “ORDER BY” instruction to sort the results in a defined manner and therefore to achieve a correct referencing of the signals via the row number.

Signals can also be configured manually by entering the correct column and row numbers according to the query result to be expected.

Note

If you do not want to manually enter the query signals to be expected in the signal table, then the operations <Test statement> and then <Add all SQL results> are required. Otherwise the signals will not be created and the module will not be shown later in the signal.

For SQL commands, a connection to *ibaPDA* signals only occurs indirectly via parameters.

Note

If a signal in *ibaPDA* has the value NaN (possible, e.g. when dividing by zero or an incorrect format in the controller) and this value is written in a DB table with an SQL command, then *ibaPDA* replaces this with NULL. This takes into consideration the fact that most databases do not support the value NaN with parameters. The prerequisite for this is that the corresponding column in the DB table is "NULLable," i.e. the value NULL is permitted.

If the value NULL is returned when reading via the SQL interface, *ibaPDA* treats this as NaN. Alternatively, the SQL query can be written so that NULL is converted into other replacement values.

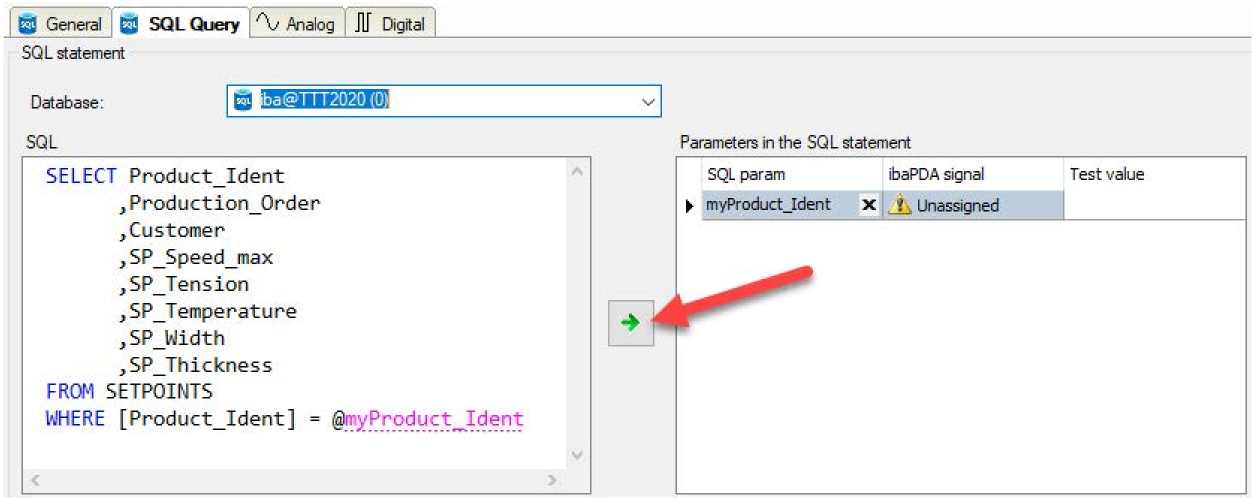
4.4.3 Use parameters

Additional parameters can be used both in SQL queries as well as in SQL commands. You can use these parameters to dynamically design queries or create conditions that expand or narrow the result space. With commands, you can use parameters to write signal values in a database.

Parameters can generally only replace values.

Parameters are defined in the SQL statements and then linked with signals in *ibaPDA*. Proceed as follows here:

1. Write an SQL statement containing parameters. Parameters are marked differently for the different database types with the first character in the parameter name.
   - For SQL Server, PostgreSQL and MySQL, it is an "@".
   - For Oracle and SAP HANA, it is an ":".
2. Then click on the green arrow between the windows.



```
SELECT Product_Ident
      ,Production_Order
      ,Customer
      ,SP_Speed_max
      ,SP_Tension
      ,SP_Temperature
      ,SP_Width
      ,SP_Thickness
FROM SETPOINTS
WHERE [Product_Ident] = @myProduct_Ident
```

SQL param	ibaPDA signal	Test value
myProduct_Ident	Unassigned	

Fig. 11: Example SQL query depending on the product ID with a parameter “myProduct_Ident”

All parameters in the statement are transferred into the table in the right window. If there are already parameters there, you will be asked whether surplus parameters should be deleted.

3. In the “*ibaPDA signal*” column, you can now select a signal that should determine the parameter value. All common signal types are permitted: Analog, digital and text signals.

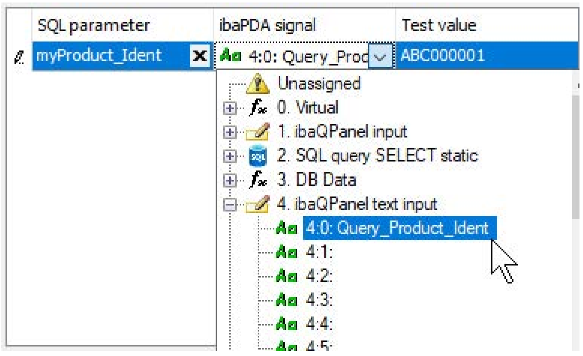
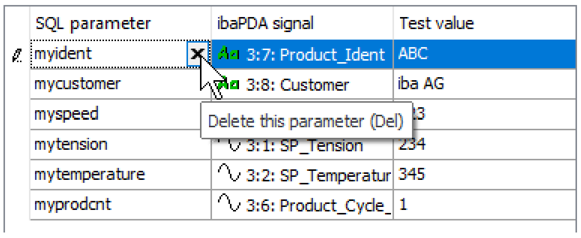


Fig. 12: Assignment of parameter and signal

4. If necessary, you can delete individual parameters from the list. Click on the desired parameter and then on the cross symbol in the “SQL parameter” column.



SQL parameter	ibaPDA signal	Test value
myident	3:7: Product_Ident	ABC
mycustomer	3:8: Customer	iba AG
myspeed	[illegible]	[illegible]3
mytension	3:1: SP_Tension	234
mytemperature	3:2: SP_Temperatur	345
myprodcnt	3:6: Product_Cycle_	1

Fig. 13: Deleting individual parameters

The selected signal provides a value for the parameter when executing the statement.

You can enter a value in the “Test value” column so that the statement can be executed, for example when testing or when validating to start the measurement. Simply enter the value as a text. The test value is correctly interpreted using the type of the assigned signal.

Digital signals have the values 0 or 1.

Note

When testing a statement, the parameter table may contain more parameters than are used in the statement. When testing, surplus parameters are ignored and there is only a warning notice. Before starting the acquisition, however, you need to delete all of the unused parameters from the table, since they are not ignored during acquisition and this may result in errors.

Note

When using parameters in a query condition (WHERE instruction), note that when changing the parameter value during operation more or fewer rows may be in the result than originally intended when configuring the module.

In this case, the number of results and the number of signals no longer match.

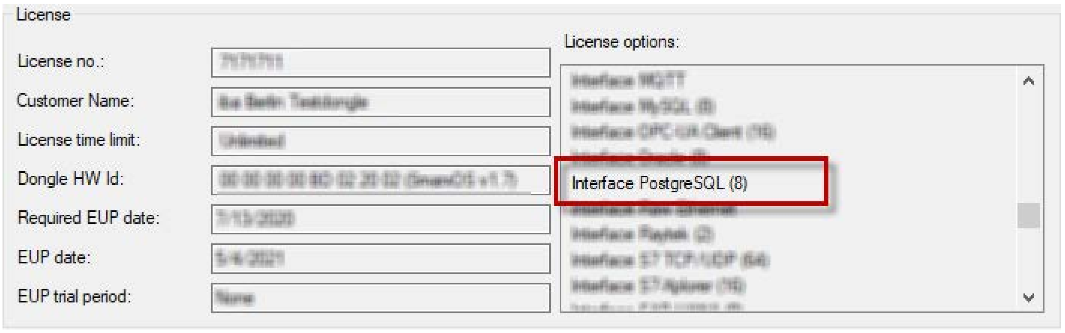
If the result contains more rows than corresponding signals, the surplus results are not taken into consideration.

If the result contains fewer rows than corresponding signals are present, some signals are not refreshed.

5 Diagnostics

5.1 License

If the "SQL database" interface is missing in the signal tree, the system has not detected any license of the supported databases. If the "SQL database" interface is available in the signal tree, at least one license of the supported databases has been detected. Whether it is the license you need or not, you can either check in *ibaPDA* under *General - Settings - License* in the I/O manager or in the *ibaPDA* server status application. In this case the license "Interface PostgreSQL" should be displayed. The number in brackets indicates the number of licensed SQL-modules.



License

License no.:	[redacted]
Customer Name:	[redacted]
License time limit:	[redacted]
Dongle HW Id:	[redacted]
Required EUP date:	[redacted]
EUP date:	[redacted]
EUP trial period:	[redacted]

License options:

[redacted]
Interface PostgreSQL (8)
[redacted]

Fig. 14: License display in the ibaPDA I/O manager

5.2 Log files

If connections to target platforms or clients have been established, all connection-specific actions are logged in a text file. You can open this (current) file and, e.g., scan it for indications of possible connection problems.

The log file can be opened via the button <Open log file>. The button is available in the I/O Manager:

- for many interfaces in the respective interface overview
- for integrated servers (e.g. OPC UA server) in the *Diagnostics* tab.

In the file system on the hard drive, you will find the log files in the program path of the *ibaPDA* server (...\Programs\iba\ibaPDA\Server\Log\). The file names of the log files include the name or abbreviation of the interface type.

Files named `interface.txt` are always the current log files. Files named `Interface_yyyy_mm_dd_hh_mm_ss.txt` are archived log files.

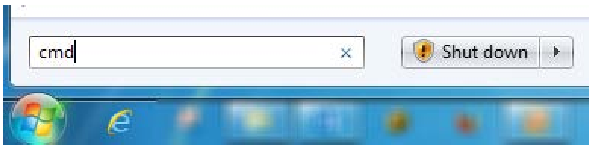
Examples:

- `ethernetipLog.txt` (log of EtherNet/IP connections)
- `AbEthLog.txt` (log of Allen-Bradley Ethernet connections)
- `OpcUAServerLog.txt` (log of OPC UA server connections)

5.3 Connection diagnostics with PING

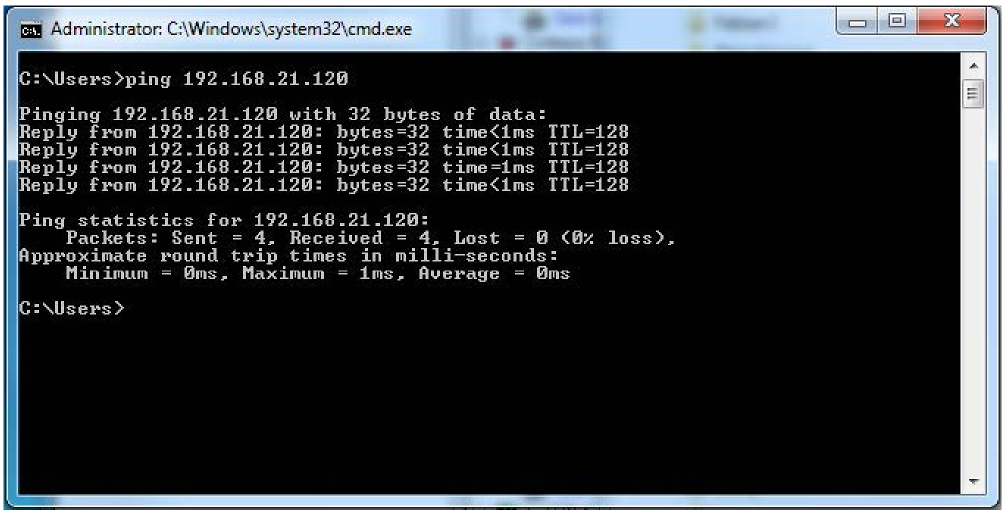
PING is a system command with which you can check if a certain communication partner can be reached in an IP network.

Open a Windows command prompt.



Enter the command “ping” followed by the IP address of the communication partner and press <ENTER>.

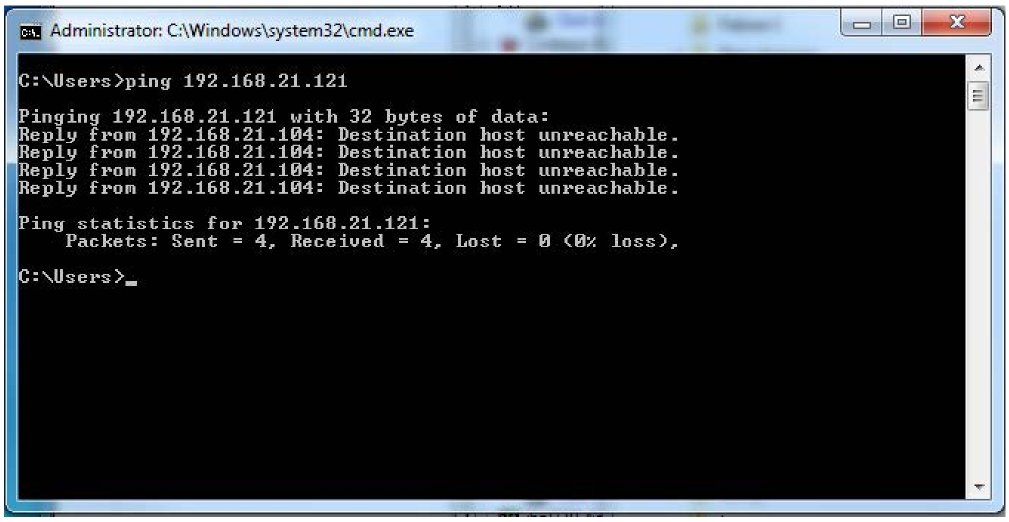
With an existing connection you receive several replies.



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users>ping 192.168.21.120

Pinging 192.168.21.120 with 32 bytes of data:
Reply from 192.168.21.120: bytes=32 time<1ms TTL=128
Reply from 192.168.21.120: bytes=32 time<1ms TTL=128
Reply from 192.168.21.120: bytes=32 time=1ms TTL=128
Reply from 192.168.21.120: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.21.120:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users>
```

Fig. 15: PING successful

With no existing connection you receive error messages.



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users>ping 192.168.21.121

Pinging 192.168.21.121 with 32 bytes of data:
Reply from 192.168.21.104: Destination host unreachable.
Reply from 192.168.21.104: Destination host unreachable.
Reply from 192.168.21.104: Destination host unreachable.
Reply from 192.168.21.104: Destination host unreachable.

Ping statistics for 192.168.21.121:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Users>_
```

Fig. 16: PING unsuccessful

5.4 Diagnostic modules

Diagnostic modules are available for most Ethernet-based interfaces and therefore also for SQL interfaces. A diagnostic module can be used to acquire information from the diagnostic displays (e.g. table with connection statistics of an interface) as signals.

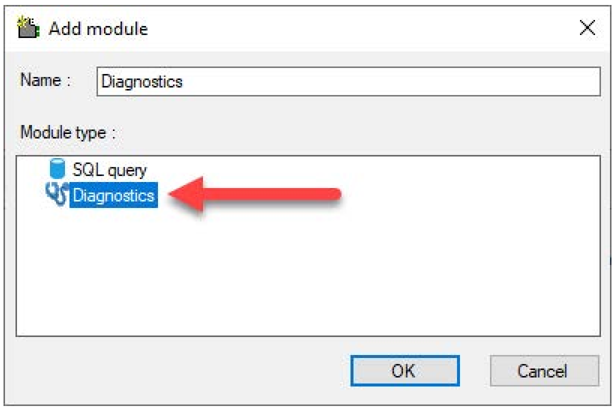
A diagnostic module is always assigned a data acquisition module of the same interface and makes its connection information available. By using a diagnostic module, the diagnostic information can also be consistently recorded and evaluated in the *ibaPDA* system.

Diagnostic modules do not consume any connection of the license, since they do not establish a connection, but rather refer to a different module.

Usage examples for diagnostic modules:

- If the error counter of a communication connection exceeds a certain value or a connection is interrupted, a notification can be generated.
- In case of a fault, the current response times in messaging traffic can be documented in an incident report.
- The status of the connections can be visualized in *ibaQPanel*.
- Diagnostic information can be transferred to superordinate monitoring systems, such as network management tools, via the SNMP server or OPC DA/UA server integrated in *ibaPDA*.

If a diagnostic module is available for an interface, the module type “Diagnose” will be shown in the dialog “Add module.”

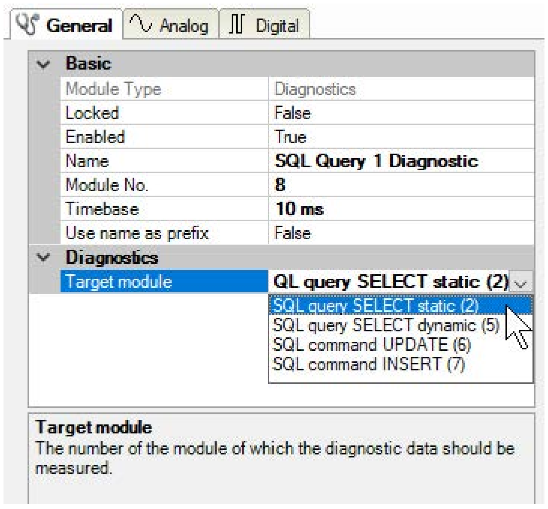


Name : Diagnostics

Fig. 17: Add diagnostic module

Module settings for diagnostic module

The following settings may be made for a diagnostic module:



Basic	
Module Type	Diagnostics
Locked	False
Enabled	True
Name	**SQL Query 1 Diagnostic**
Module No.	**8**
Timebase	**10 ms**
Use name as prefix	False
Diagnostics	
Target module	**QL query SELECT static (2)**

SQL query SELECT static (2)
SQL query SELECT dynamic (5)
SQL command UPDATE (6)
SQL command INSERT (7)

Target module
The number of the module of which the diagnostic data should be measured.

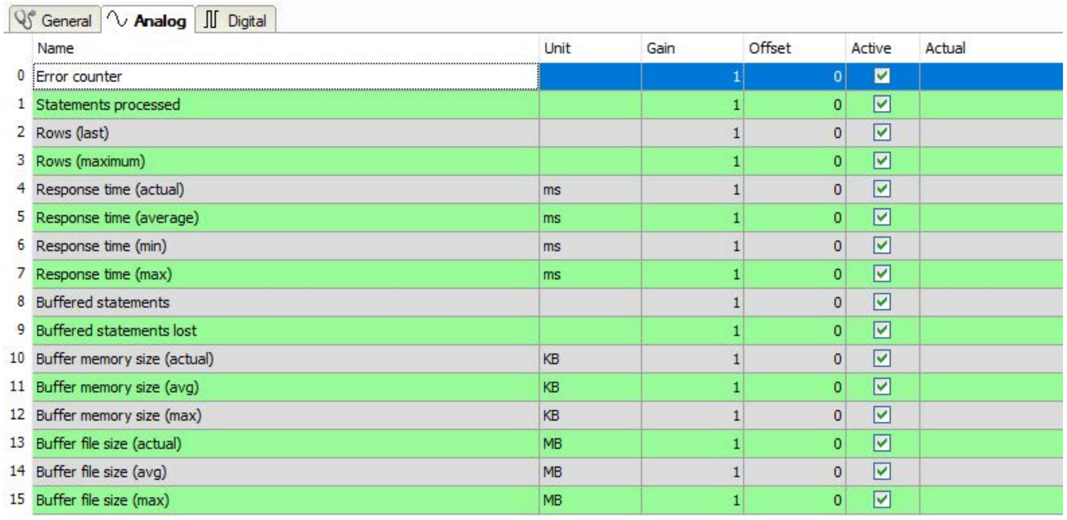
Fig. 18: Module settings for a diagnostic module, selection of the target module

The basic settings of a diagnostic module correspond to those of the other modules.

There is only one specific setting for the diagnostic module, which must be made: the target module.

By selecting the target module, you assign the diagnostic module to the module whose connection information is to be acquired. The supported modules of the same interface are available to choose from in the drop-down list of the setting. Exactly one data acquisition module can be assigned per diagnostic module. Once you have chosen a module, the available diagnostic signals will immediately be added in the *Analog* and *Digital* tabs. The interface type determines which signals these are.

With a diagnostic module for an SQL module, you will find the following information in the *Analog* tab. For the meaning of this data, see ➚ *Basic settings of the interface*, page 10



	Name	Unit	Gain	Offset	Active	Actual
0	Error counter		1	0	✓	
1	Statements processed		1	0	✓	
2	Rows (last)		1	0	✓	
3	Rows (maximum)		1	0	✓	
4	Response time (actual)	ms	1	0	✓	
5	Response time (average)	ms	1	0	✓	
6	Response time (min)	ms	1	0	✓	
7	Response time (max)	ms	1	0	✓	
8	Buffered statements		1	0	✓	
9	Buffered statements lost		1	0	✓	
10	Buffer memory size (actual)	KB	1	0	✓	
11	Buffer memory size (avg)	KB	1	0	✓	
12	Buffer memory size (max)	KB	1	0	✓	
13	Buffer file size (actual)	MB	1	0	✓	
14	Buffer file size (avg)	MB	1	0	✓	
15	Buffer file size (max)	MB	1	0	✓	

Fig. 19: Analog values of the diagnostic module for an SQL module

There is only one signal in the *Digital* tab that shows whether the module is connected to the database or not.

6 Support and contact

Support

Phone:	+49 911 97282-14
Fax:	+49 911 97282-33
Email:	support@iba-ag.com

Note

If you need support for software products, please state the license number or the CodeMeter container number (WIBU dongle). For hardware products, please have the serial number of the device ready.

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Phone:	+49 911 97282-0
Fax:	+49 911 97282-33
Email:	iba@iba-ag.com

Mailing address

iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address

iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

Regional and Worldwide

For contact data of your regional iba office or representative please refer to our web site

www.iba-ag.com.