ibaPDA-Interface-MySQL

Write/read access to databases

Manual

Issue 1.3

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
Support	+49 911 97282-14
Engineering	+49 911 97282-13
E-mail	iba@iba-ag.com
Web	www.iba-ag.com

Version	Date	Revision	Author	Version SW
1.3	11-2023	GUI new	rm	8.4.0

Contents

1 About this documentation

This documentation describes the function and application of the software interface *ibaPDA-Interface-MySQL*.

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 is aimed at qualified professionals who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as professional if he/she is capable of assessing safety and recognizing possible consequences and 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-MySQL* 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>
Filenames, paths	`Filename`, `Path` Example: `Test.docx`

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 MySQL or MariaDB type:

- *ibaPDA* v8.4.0 or higher
- License for *ibaPDA-Interface-MySQL*

For further requirements for the used computer hardware and the supported operating systems, refer to the *ibaPDA* documentation.

License information

Order no.	Product name	Description
31.001116	ibaPDA-Interface-MySQL	Extension license for an *ibaPDA* system for reading from and writing to databases of the MySQL or MariaDB type. License for 8 SQL modules according to 8 SQL statements.
31.101116	one-step-up-Interface-MySQL	Extension license for an additional 8 SQL modules. Max. 3 permissible (32 SQL modules total)

3 SQL database 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.

sing the SQL interface you configure and manage the connections to one or more databases. You can configure one or more connections per database. All DB connections can be used simultaneously. The databases can be the same or different systems.

Database systems of different vendors are supported:

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

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

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*.
  - In the queries, you an optionally use parameters, which are assigned to the signals, in order to 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, you can optionally use parameters, which are assigned to the signals, in order to write signal actual values.

You can execute queries and commands cyclically or triggered.

[1)] 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 you must always take the response time of the database into consideration.

The corresponding data storage (e.g. *ibaPDA-Data-Store-SQL-Server*, *ibaPDA-Data-Store-SQL-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 systems 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 "ADO.NET Data Provider" for SAP HANA connections.
3. In *ibaPDA*, open the I/O Manager, *Inputs* tab, and configure one or more database connections on the SQL interface.
   See ↗ *Creating a database connection*, page 12

For database queries (read access)

1. Under the SQL interface on the *Inputs* tab, 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 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. Under the SQL interface on the *Outputs* tab, add a module *SQL command* and select one of the previously configured database connections for the module.
2. Formulate an SQL statement as an SQL command with or without parameters.
3. If you use parameters: Assign 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

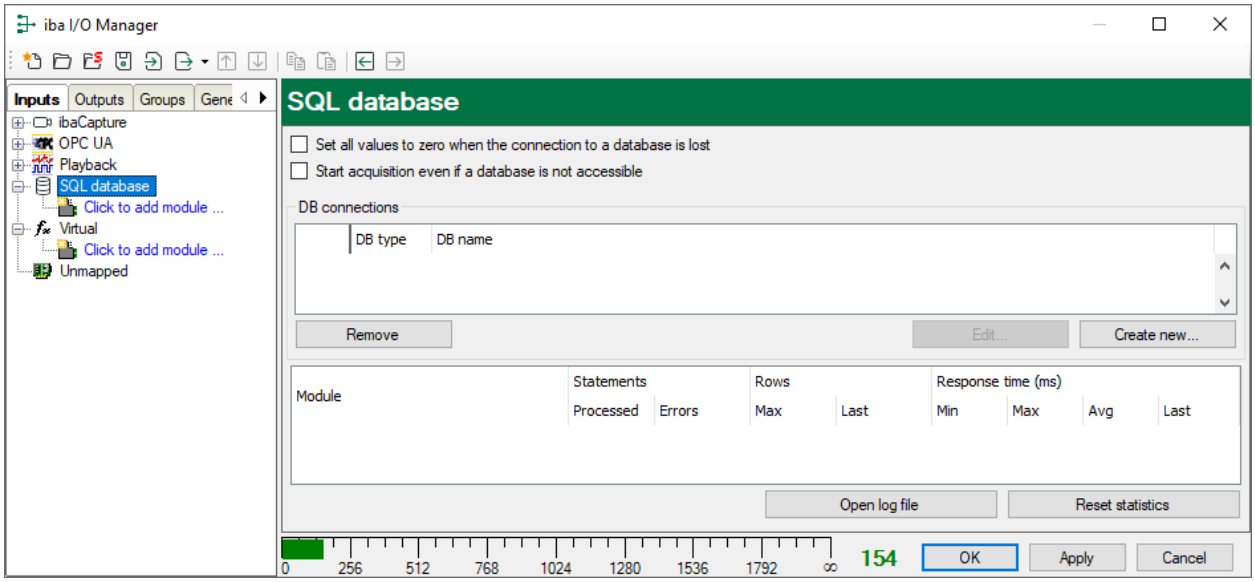
The engineering for *ibaPDA* is described in the following. If all system requirements are fulfilled, *ibaPDA* displays the *SQL database* interface in the interface tree of the I/O Manager.

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 license container (dongle or soft license). Even if only one database system is licensed, all supported DB systems appear in the SQL interface configuration. Nevertheless, you can only use the licensed syst mfor data acquisition.

4.1 Interface settings

The interface has the following functions and configuration options.



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

If this option is disabled, *ibaPDA* keeps the last values in memory when the connection is lost.

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 starts even if one or more of the configured databases are not accessible. A warning is prompted in the validation dialog, not an error. If the system was started without a connection to a database, *ibaPDA* periodically tries to connect to the database.

DB connections
This area displays the configured database connections. Each database connection is clearly identified with an ID number, the database system 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.

<Open log file>
If connections to controllers have been established, all connection specific actions are recorded in a text file. Using this button, you can open and check this file. In the file system on the hard disk, you find the log files of the *ibaPDA* server (…\ProgramData\iba\ibaPDA\Log). The file name of the current log file is InterfaceLog.txt; the name of the archived log files is InterfaceLog_yyyy_mm_dd_hh_mm_ss.txt.

<Reset statistics>
Click this button to reset the calculated times and error counters in the table to 0.

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.

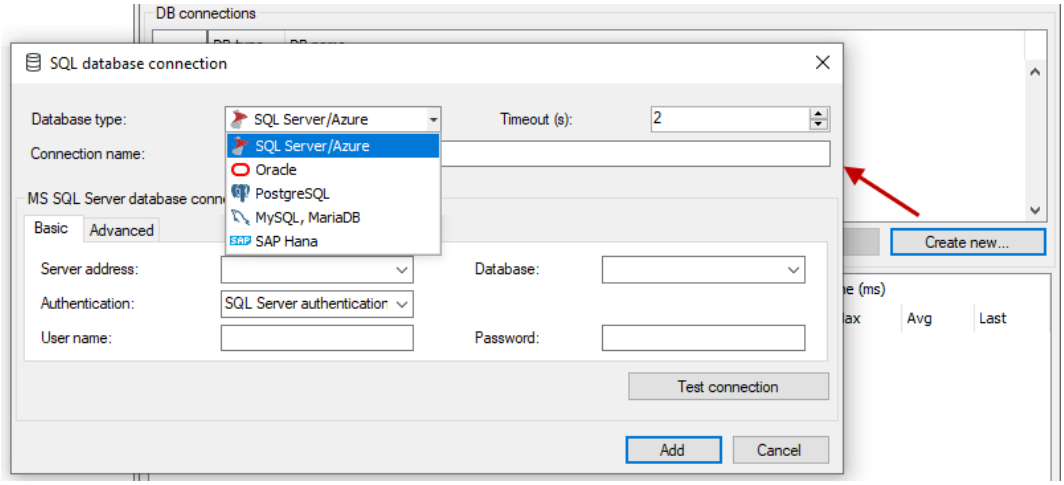
Column	Meaning
Module	Name of the SQL module (query and command modules)
Statements	Processed: Number of executed SQL statements Error: Number of SQL statements where failures occurred. Either SQL statements failed (database responds with error) or statements could not be executed, because the database 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 statistics reset. Last: Response time of the last executed statement.

Table 2: Columns of the connection diagnostics

4.2 Creating a database connection

Before you can exchange data with a database, you must first configure a connection.

1. Select the interface *SQL Database* in the interface tree and click on the button <Create new>.



2. Select the desired database system.

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

3. 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 system 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.

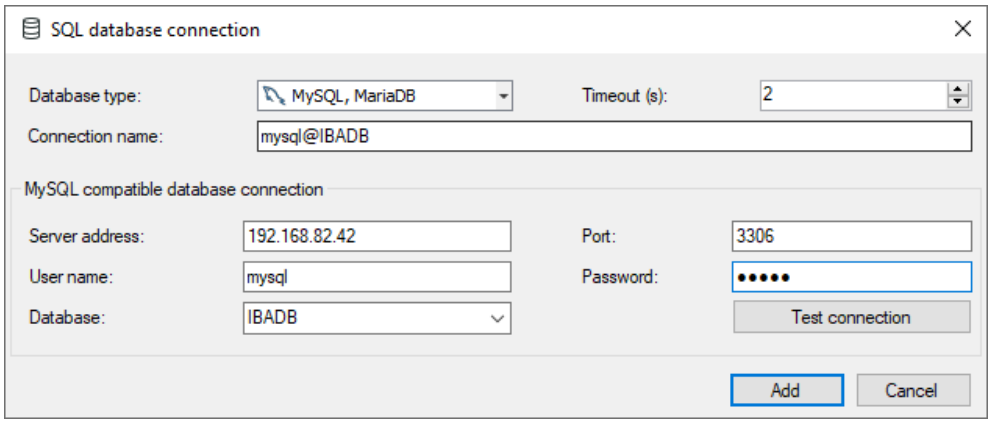
Use the <Remove> button to delete the selected DB connection.

Note

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

Because SQL modules are linked to a DB connection in the SQL interface, you must make sure that the modules do not lose their reference to the connection. Otherwise the SQL statements will have no effect.

4.2.1 MySQL/MariaDB DB connection



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

Timeout
Here you can specify a value for the timeout in seconds for establishing the connection. If the time set here is exceeded, *ibaPDA* aborts the connection attempt and displays an error message.

Note that once the connection is made, each command sent to the database has an additional timeout, which can be set by the application that makes use of it. There is no generic configuration for the command timeout.

Connection name
This line contains the name of the database connection. The name is 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
The DB server communicates via this port. Usually, you can keep the default port 3306.
If another port is required, you can enter this port here.

User name/Password
Enter here the required login data for the database. If necessary, inquire the correct data from your DB administrator.

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

<Test connection>
ibaPDA checks the access to the selected database on the set DB server. If the connection is successful, the message includes the version of the database.

4.3 SQL modules

For the SQL modules under the SQL interface, there are input modules and output modules.

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 on both tabs *Inputs* and *Outputs*.

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

The following applies to all SQL modules:

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

4.3.1 Adding an SQL module

1. Click on the blue command *Click to add module…* located under each data interface in the *Inputs* or *Outputs* tab.
2. Select the desired module type in the dialog box and assign a name via the input field if required.
3. Confirm the selection with <OK>.

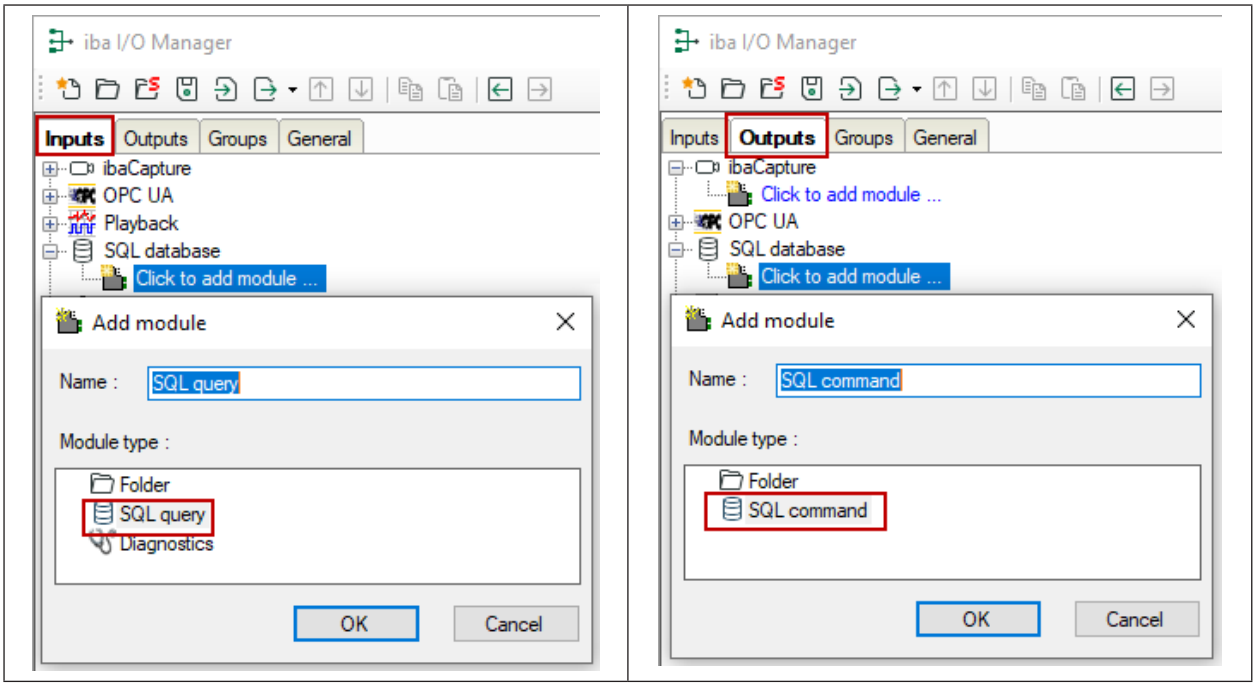


Table 3: Add an SQL query module (left) or SQL command module (right)

4.3.2 General module settings

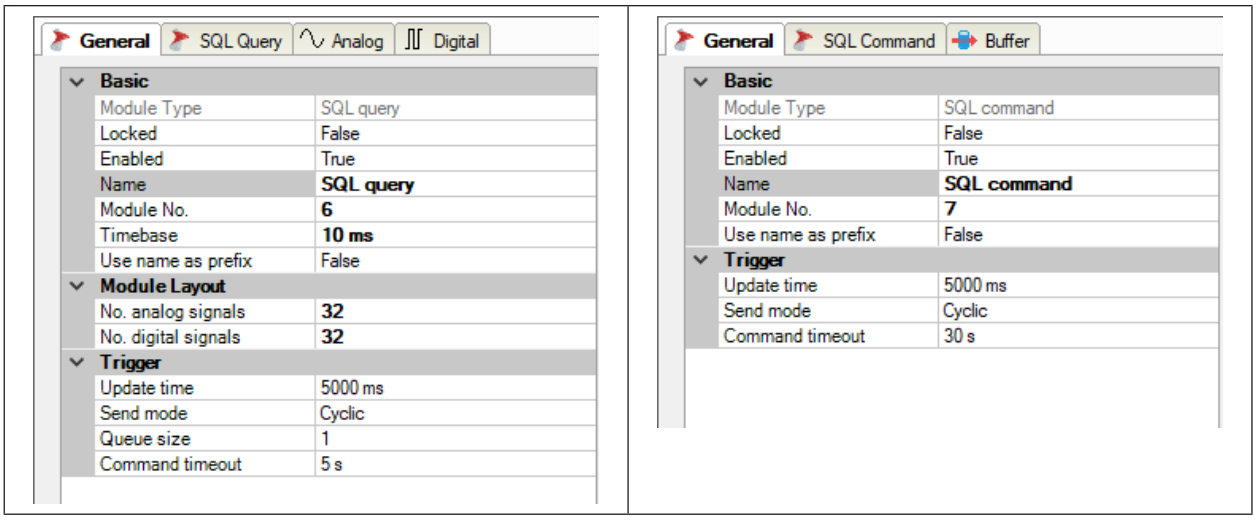


Table 4: General module settings for input modules (left) and output modules (right)

Basic settings

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

Locked
You can lock a module to avoid unintentional or unauthorized changing of the module settings.

Enabled
Enable the module to record signals.

Name
You can enter a name for the module here.

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

Timebase
All signals of the module are sampled on this timebase.

Use module name as prefix
This option puts the module name in front of the signal names.

Module structure (only input modules)

No. of analog signals/digital signals
Define the number of configurable analog and digital signals in the signal tables. The default value is 32 for each. The maximum value is 1000. The signal tables are adjusted accordingly.

Trigger

Update time

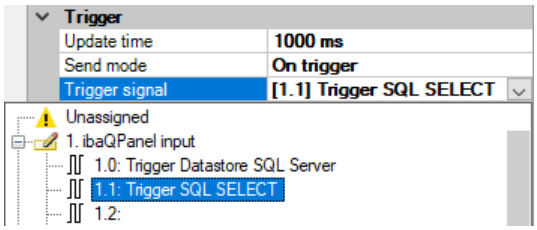
You can use the update time in milliseconds to determine the following:

- In *Cyclic* send mode: execution interval of the SQL statements
- In the *On change* and *On trigger* mode: dead time after executing a statement

Send mode

Set here when to execute the SQL statements (queries and commands).

- *Cyclic*: The SQL statement is executed cyclically at the set update time.
- *On change*: When selecting this mode, the line *Trigger signal* appears. 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, the line *Trigger signal* appears. 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 signal (for On change or On trigger send mode)

- In the *On change* send mode: You can select any analog or digital signal here to serve as the trigger. In case of a change to the signal value, the SQL statement is executed.
- In the *On trigger* send mode: You can select any digital signal here to serve as the trigger. In case of a rising edge of the signal, the SQL statement is executed.

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 are 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:
  Because the results of an SQL query are mapped to signals, in most cases it is not useful to catch up on 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 for a statement 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.

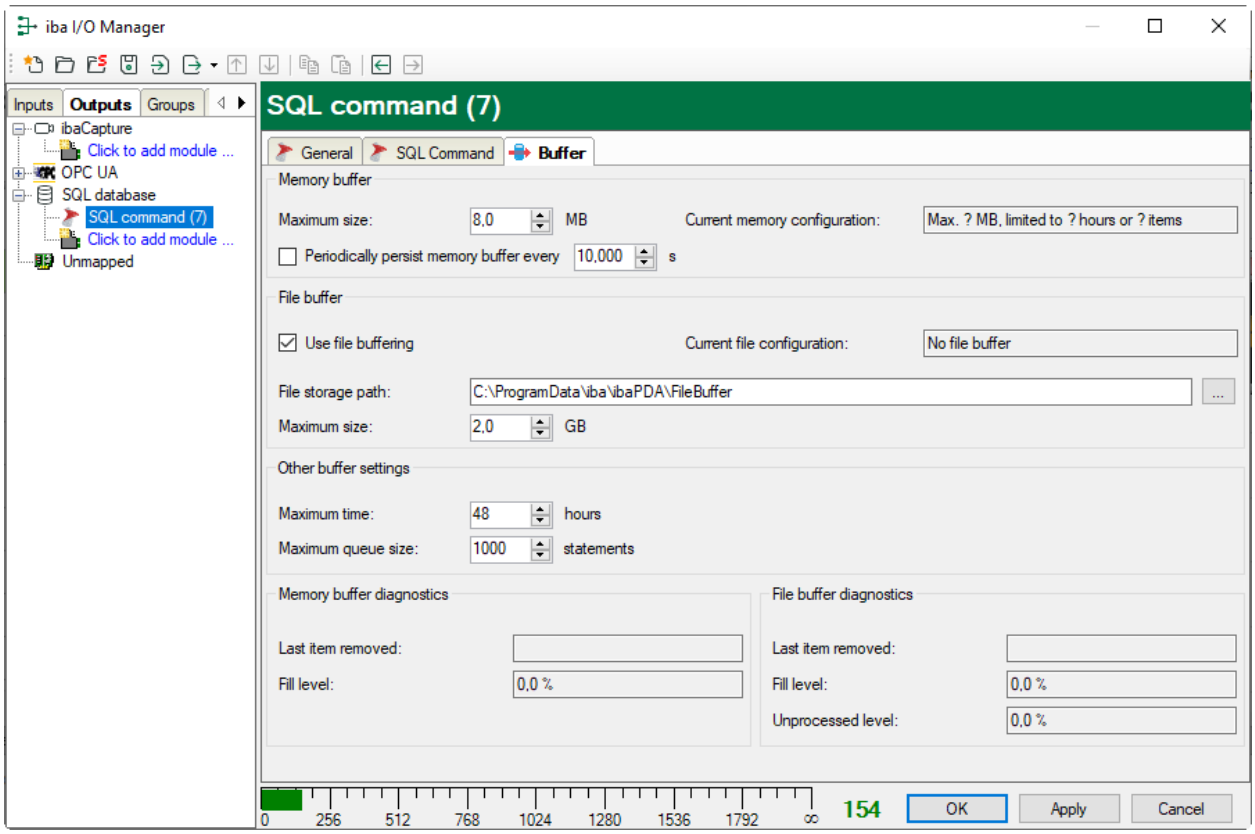
Data to be sent to the target system always passes through the internal *ibaPDA* memory buffer. If the connection to the target system exists, the data is sent there immediately. If the connection is lost, or the data cannot be sent out fast enough, the data is temporarily stored in the memory buffer. 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 you change the SQL statement 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

The memory buffer is always enabled. You cannot deactivate it because 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

You can only enable this option 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.

You can use the same file directory for several data stores because the buffer files of a data store have a unique name. Files from different data stores 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

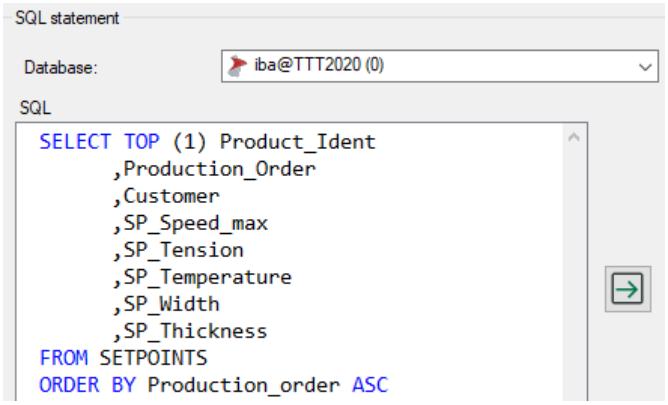
If you configure a diagnostic module for an SQL command module in the I/O configuration, you can monitor and record the fill level values of the memory buffer and file buffer.

See also ↗ *Diagnostic modules*, page 30.

4.4 SQL statements

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

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.



Only use statements that the connected database server can execute. The syntax of the SQL statement must be correct and written in the SQL dialect of the database system. *ibaPDA* does not offer any plausibility check of the SQL statement!

ibaPDA highlights some syntax elements in color to provide support:

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

The highlighting differs for each DB system type because every SQL dialect has different keywords 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 database as the result and assign these values directly to the module signals.

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

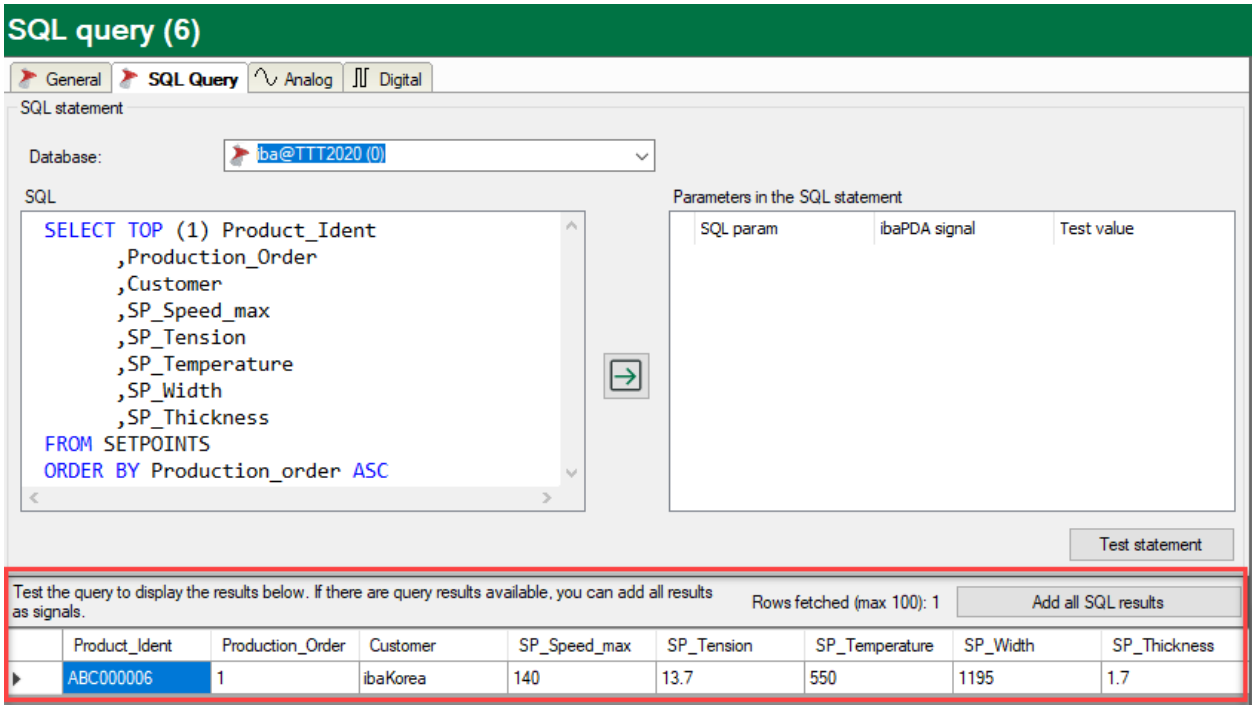
4.4.1 Testing SQL statements

You can test SQL statements for queries as well as for commands by using the <Test statement> button.

Technically, the statement is executed during the test in a transaction, 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.



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.

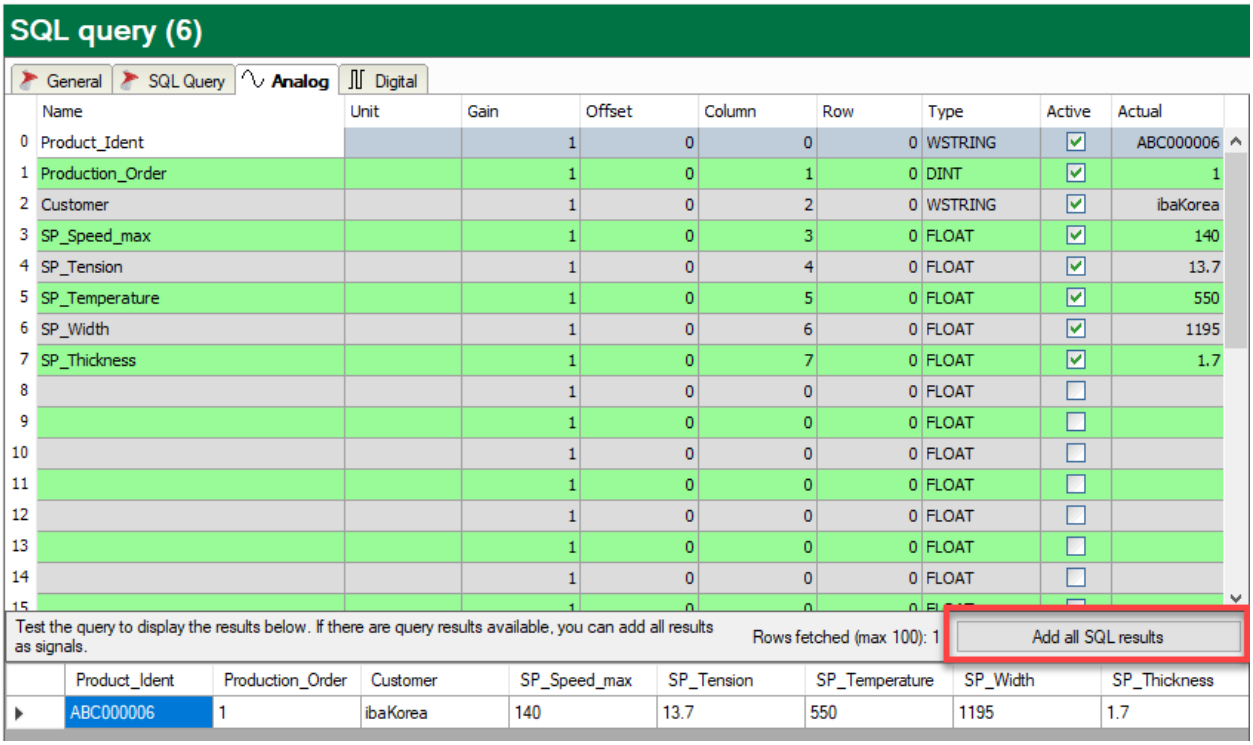
4.4.2 Linking SQL queries with input signals

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 successful, you can enter the queried values in the signal tables using the button <Add all SQL results>.

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



	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	☐	

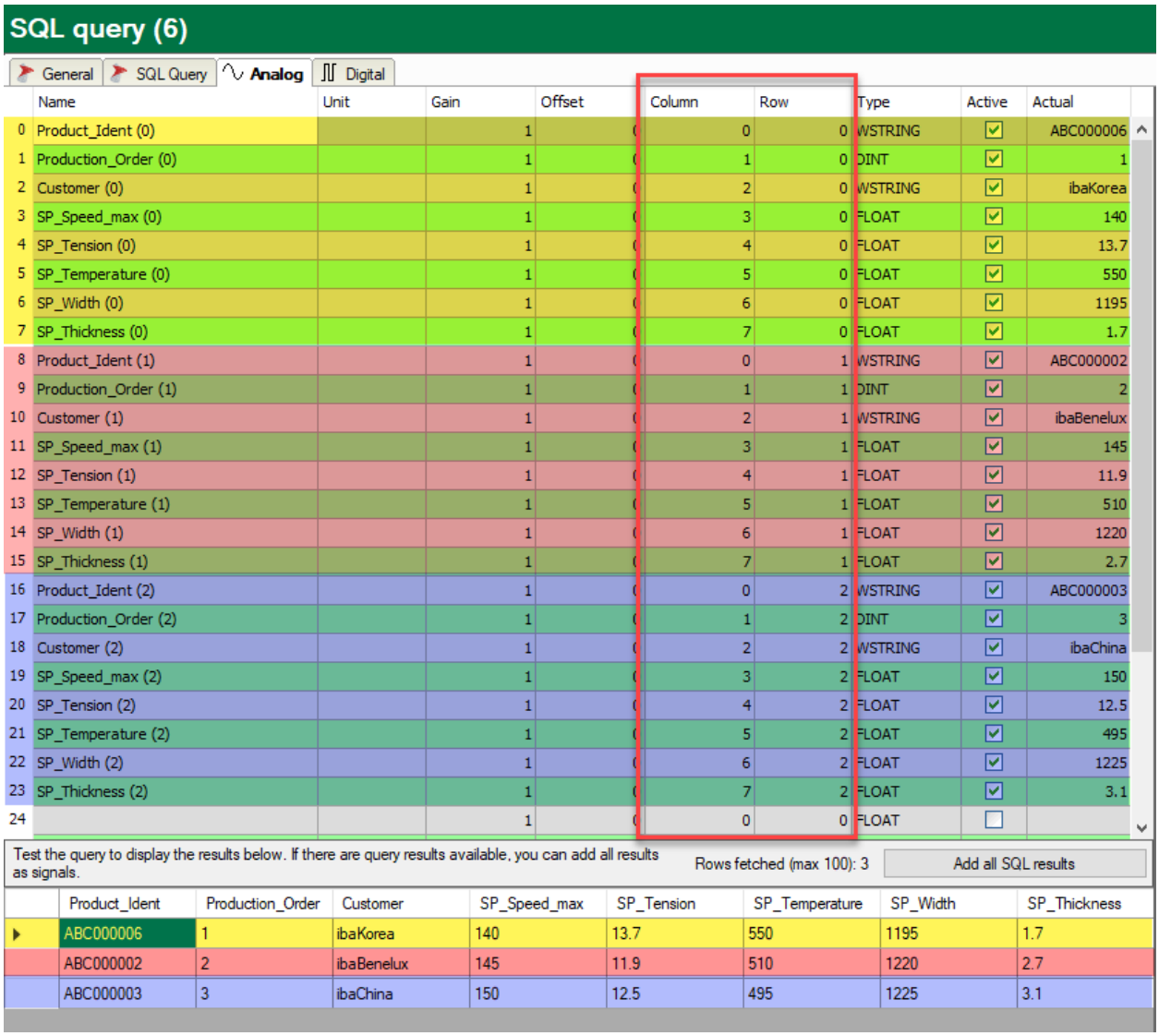
	Product_Ident	Production_Order	Customer	SP_Speed_max	SP_Tension	SP_Temperature	SP_Width	SP_Thickness
▶	ABC000006	1	ibaKorea	140	13.7	550	1195	1.7

If there are not enough signals available in the module, *ibaPDA* asks you whether you want to increase the number of signals.

The relation of the added signals to the query results is 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 would, for example, provide the following result:

```
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
```



Tip

For SQL queries with multi-row results, use the ORDER BY statement 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 are not created and the module is not 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, you can write the SQL query so that "NULL" is converted into other replacement values.

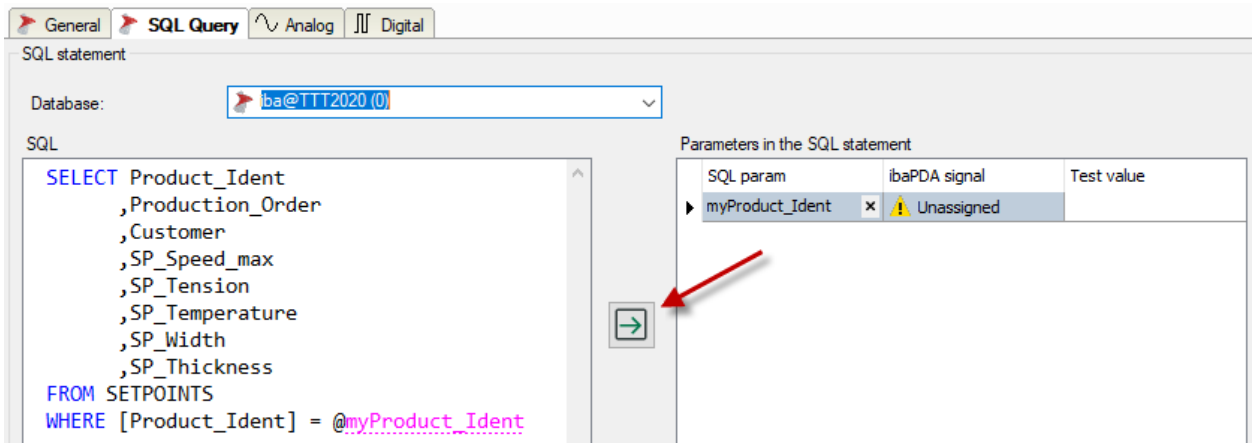
4.4.3 Using parameters

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.

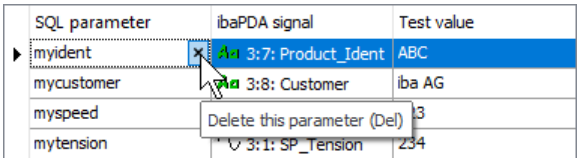
Parameters can generally only replace values.

You defined the parameters 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 systems with the first character in the parameter name.
   - For SQL Server, PostgreSQL and MySQL, it is "@".
   - For Oracle and SAP HANA, it is ":".
2. Then click on the green arrow between the windows. 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 signals , digital signals and text signals.
4. If necessary, you can delete individual parameters from the list. Click on the desired parameter and then on the <X> button in the *SQL parameter* column.



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 because they are not ignored during acquisition and this may result in errors.

Note

When using parameters in a query condition (WHERE statement), 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.

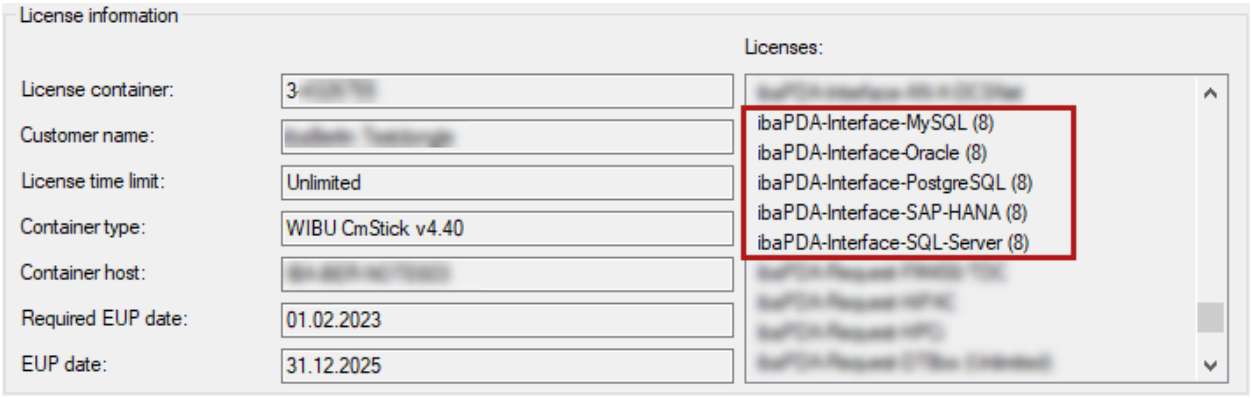
5 Diagnostics

5.1 License

If the *SQL database* interface is not displayed in the signal tree, you can either check in *ibaPDA* in the I/O Manager under *General – Settings* or in the *ibaPDA* service status application whether your license for one or more of the supported databases has been properly recognized. The number of licensed connections is shown in brackets.

You need **one** of the following licenses:

ibaPDA-Interface-MySQL, *ibaPDA-Interface-Oracle*, *ibaPDA-Interface-PostgreSQL*, *ibaPDA-Interface-SAP-HANA*, *ibaPDA-Interface-SQL-Server*



5.2 Visibility of the interface

If the interface is not visible despite a valid license, it may be hidden.

Check the settings in the *General* tab in the *Interfaces* node.

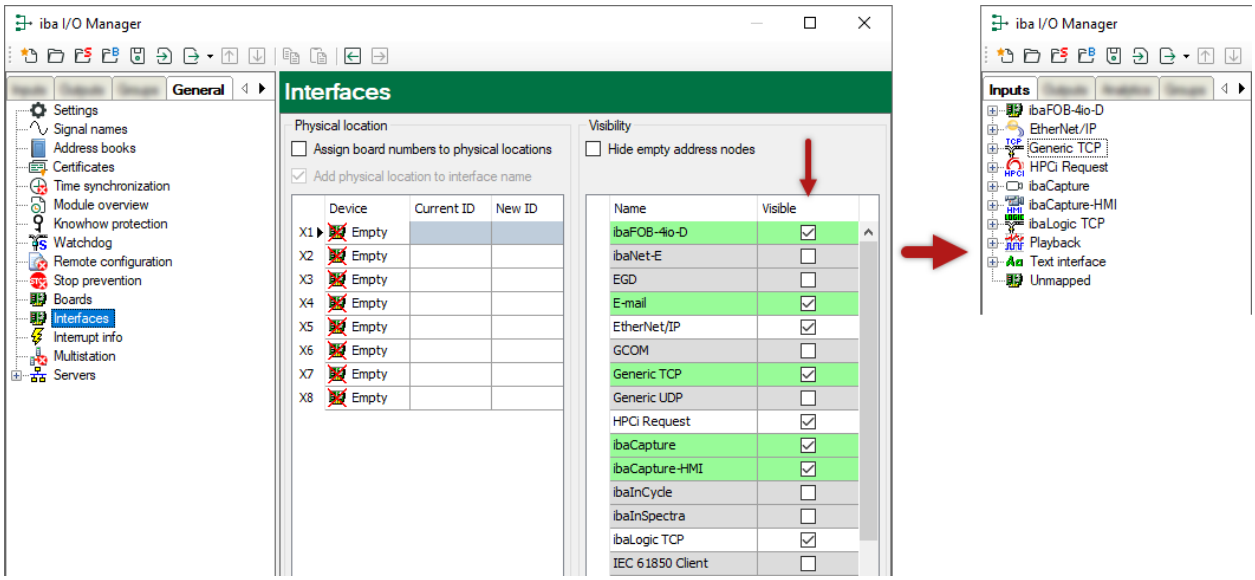
Visibility

The table *Visibility* lists all the interfaces that are available either through licenses or installed cards. These interfaces can also be viewed in the interface tree.

You can hide or display the interfaces not required in the interface tree by using the checkbox in the *Visible* column.

Interfaces with configured modules are highlighted in green and cannot be hidden.

Selected interfaces are visible, the others are hidden:



5.3 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.

You can open the log file 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 can find the log files of the *ibaPDA* server (…`\ProgramData\iba\ibaPDA\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.

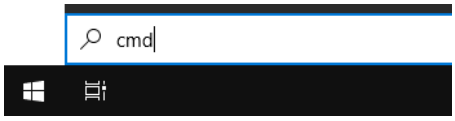
Examples:

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

5.4 Connection diagnostics with PING

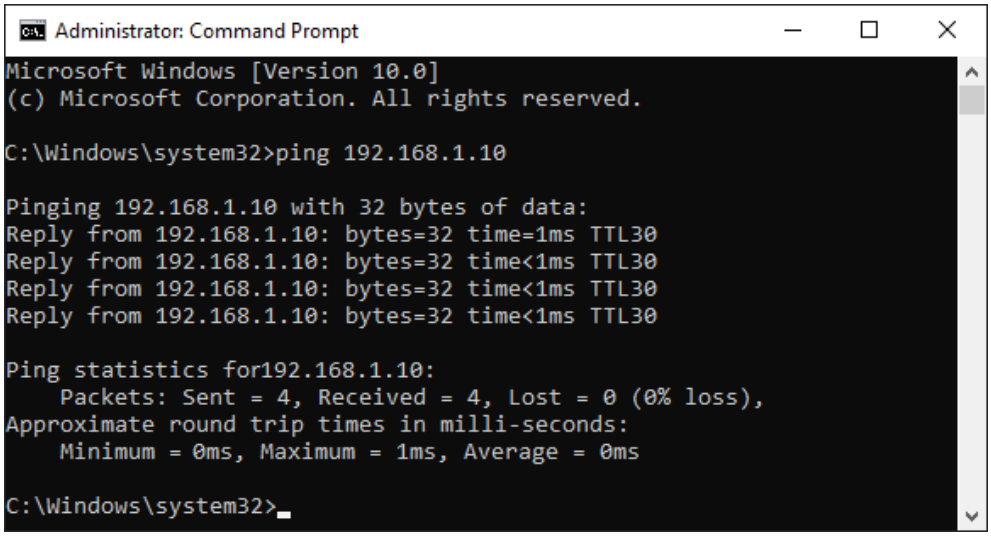
PING is a system command with which you can check if a certain communication partner can be reached in an IP network.

1. Open a Windows command prompt.

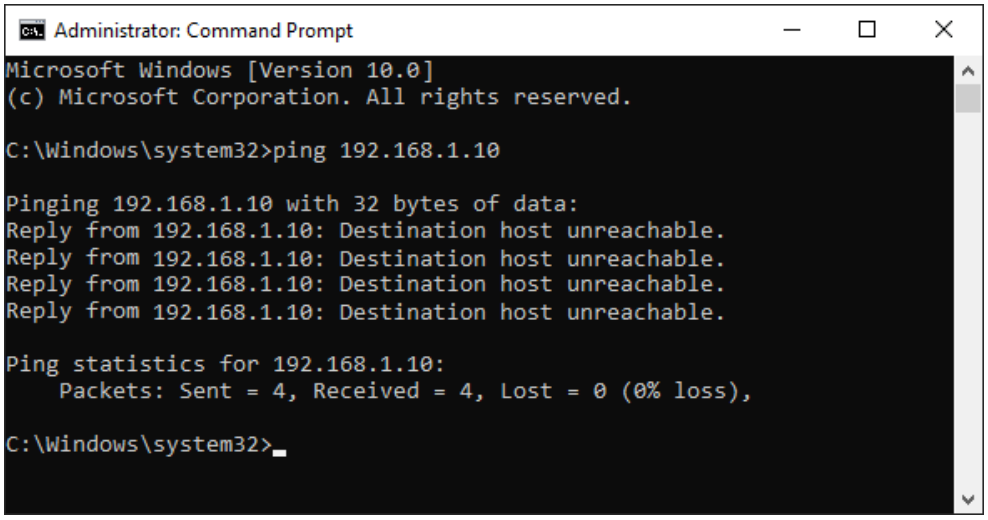


2. Enter the command "ping" followed by the IP address of the communication partner and press <ENTER>.

→ With an existing connection you receive several replies.



→ With no existing connection you receive error messages.



5.5 Diagnostic modules

Diagnostic modules are available for most Ethernet based interfaces and Xplorer interfaces. Using a diagnostic module, information from the diagnostic displays (e.g. diagnostic tabs and connection tables of an interface) can be acquired as signals.

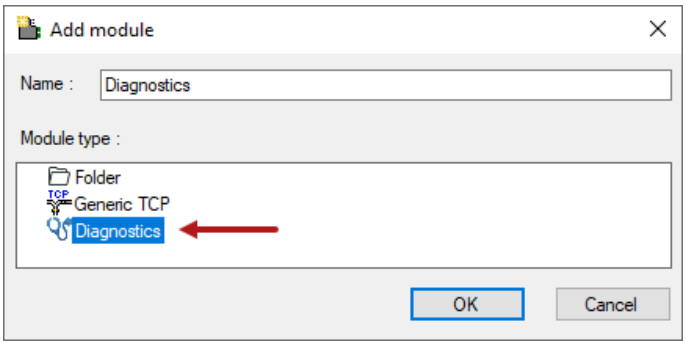
A diagnostic module is always assigned to a data acquisition module of the same interface and supplies its connection information. By using a diagnostic module you can record and analyze the diagnostic information continuously in the *ibaPDA* system.

Diagnostic modules do not consume any license connections because they do not establish their own connection, but refer to another module.

Example for the use of diagnostic modules:

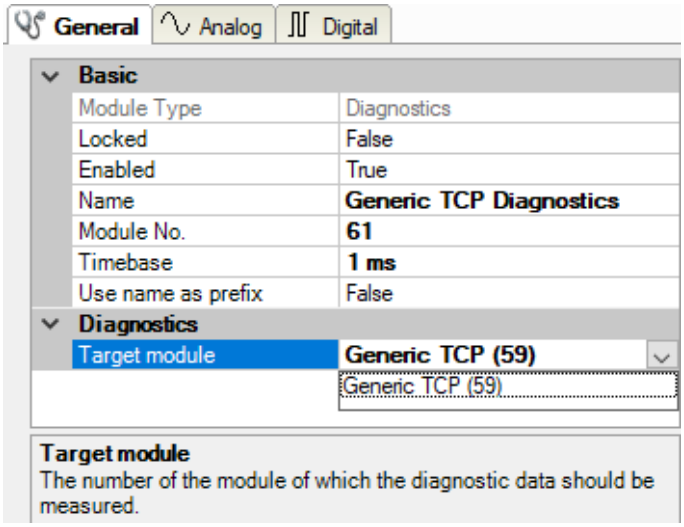
- A notification can be generated, whenever the error counter of a communication connection exceeds a certain value or the connection gets lost.
- In case of a disturbance, the current response times in the telegram traffic may be documented in an incident report.
- The connection status can be visualized in *ibaQPanel*.
- You can forward diagnostic information via the SNMP server integrated in *ibaPDA* or via OPC DA/UA server to superordinate monitoring systems like network management tools.

In case the diagnostic module is available for an interface, a "Diagnostics" module type is shown in the "Add module" dialog (example: Generic TCP).



Module settings diagnostic module

For a diagnostic module, you can make the following settings (example: Generic TCP):



The basic settings of a diagnostic module equal those of other modules.

There is only one setting which is specific for the diagnostic module: the target module.

By selecting the target module, you assign the diagnostic module to the module on which you want to acquire information about the connection. You can select the supported modules of this interface in the drop down list of the setting. You can assign exactly one data acquisition module to each diagnostic module. When having selected a module, the available diagnostic signals are immediately added to the *Analog* and *Digital* tabs. It depends on the type of interface, which signals exactly are added. The following example lists the analog values of a diagnostic module for a Generic TCP module.

General | **Analog** | Digital

	Name	Unit	Gain	Offset	Active	Actual
0	IP address (part 1)		1	0	☑	
1	IP address (part 2)		1	0	☑	
2	IP address (part 3)		1	0	☑	
3	IP address (part 4)		1	0	☑	
4	Port		1	0	☑	
5	Message counter		1	0	☑	
6	Incomplete errors		1	0	☑	
7	Packet size (actual)	bytes	1	0	☑	
8	Packet size (max)	bytes	1	0	☑	
9	Time between data (actual)	ms	1	0	☑	
10	Time between data (min)	ms	1	0	☑	

For example, the IP (v4) address of a Generic TCP module (see fig. above) will always be split into 4 parts derived from the dot-decimal notation, for better reading. Also other values are being determined, as there are port number, counters for telegrams and errors, data sizes and telegram cycle times. The following example lists the digital values of a diagnostic module for a Generic TCP module.

General | Analog | **Digital**

	Name	Active	Actual
0	Active connection mode	☑	
1	Invalid packet	☑	
2	Connecting	☑	
3	Connected	☑	

Diagnostic signals

Depending on the interface type, the following signals are available:

Signal name	Description
Active	Only relevant for redundant connections. Active means that the connection is used to measure data, i.e. for redundant standby connections the value is 0. For normal/non-redundant connections, the value is always 1.
Buffer file size (actual/avg/max)	Size of the file for buffering statements
Buffer memory size (actual/avg/max)	Size of the memory used by buffered statements
Buffered statements	Number of unprocessed statements in the buffer
Buffered statements lost	Number of buffered but unprocessed and lost statements
Connected	Connection is established
Connected (in)	A valid data connection for the reception (in) is available
Connected (out)	A valid data connection for sending (out) is available
Connecting	Connection being established
Connection attempts (in)	Number of attempts to establish the receive connection (in)
Connection attempts (out)	Number of attempts to establish the send connection (out)
Connection ID O->T	ID of the connection for output data (from the target system to *ibaPDA*). Corresponds to the assembly instance number
Connection ID T->O	ID of the connection for input data (from *ibaPDA* to target system). Corresponds to the assembly instance number
Connection phase (in)	Status of the ibaNet-E data connection for reception (in)
Connection phase (out)	Status of the ibaNet-E data connection for sending (out)
Connections established (in)	Number of currently valid data connections for reception (in)
Connections established (out)	Number of currently valid data connections for sending (out)
Data length	Length of the data message in bytes
Data length O->T	Size of the output message in byte
Data length T->O	Size of the input message in byte
Destination IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to *ibaPDA*)
Destination IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from *ibaPDA* to target system)
Disconnects (in)	Number of currently interrupted data connections for reception (in)
Disconnects (out)	Number of currently interrupted data connections for sending (out)
Error counter	Communication error counter
Exchange ID	ID of the data exchange
Incomplete errors	Number of incomplete messages

Signal name	Description
Incorrect message type	Number of received messages with wrong message type
Input data length	Length of data messages with input signals in bytes (*ibaPDA* receives)
Invalid packet	Invalid data packet detected
IP address (part 1-4)	4 octets of the IP address of the target system
Keepalive counter	Number of KeepAlive messages received by the OPC UA Server
Lost images	Number of lost images (in) that were not received even after a retransmission
Lost Profiles	Number of incomplete/incorrect profiles
Message counter	Number of messages received
Messages per cycle	Number of messages in the cycle of the update time
Messages received since configuration	Number of received data telegrams (in) since start of acquisition
Messages received since connection start	Number of received data telegrams (in) since the start of the last connection setup. Reset with each connection loss.
Messages sent since configuration	Number of sent data telegrams (out) since start of acquisition
Messages sent since connection start	Number of sent data telegrams (out) since the start of the last connection setup. Reset with each connection loss.
Multicast join error	Number of multicast login errors
Number of request commands	Counter for request messages from *ibaPDA* to the PLC/CPU
Output data length	Length of the data messages with output signals in bytes (*ibaPDA* sends)
Packet size (actual)	Size of the currently received message
Packet size (max)	Size of the largest received message
Ping time (actual)	Response time for a ping telegram
Port	Port number for communication
Producer ID (part 1-4)	Producer ID as 4 byte unsigned integer
Profile Count	Number of completely recorded profiles
Read counter	Number of read accesses/data requests
Receive counter	Number of messages received
Response time (actual/average/max/min)	Response time is the time between measured value request from *ibaPDA* and response from the PLC or reception of the data. Actual: current value Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Retransmission requests	Number of data messages requested again if lost or delayed

Signal name	Description
Rows (last)	Number of resulting rows by the last SQL query (within the configured range of result rows)
Rows (maximum)	Maximum number of resulting rows by any SQL query since the last start of acquisition (possible maximum equals the configured number of result rows)
Send counter	Number of send messages
Sequence errors	Number of sequence errors
Source IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to *ibaPDA*)
Source IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from *ibaPDA* to target system)
Statements processed	Number of executed statements since last start of acquisition
Synchronization	Device is synchronized for isochronous acquisition
Time between data (actual/ max/min)	Time between two correctly received messages Actual: between the last two messages Max/min: statistical values since start of acquisition or reset of counters
Time offset (actual)	Measured time difference of synchronicity between *ibaPDA* and the ibaNet-E device
Topics Defined	Number of defined topics
Topics Updated	Number of updated topics
Unknown sensor	Number of unknown sensors
Update time (actual/average/ configured/max/min)	Specifies the update time in which the data is to be retrieved from the PLC, the CPU or from the server (configured). Default is equal to the parameter "Timebase". During the measurement the real actual update time (actual) can be higher than the set value, if the PLC needs more time to transfer the data. How fast the data is really updated, you can check in the connection table. The minimum achievable update time is influenced by the number of signals. The more signals are acquired, the greater the update time becomes. Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Write counter	Number of successful write accesses
Write lost counter	Number of failed write accesses

6 Support and contact

Support

Phone: +49 911 97282-14

Email: support@iba-ag.com

Note

If you need support for software products, please state the number of the license container. 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

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