# ibaPDA v7.2.0

## New Features

13/07/2020

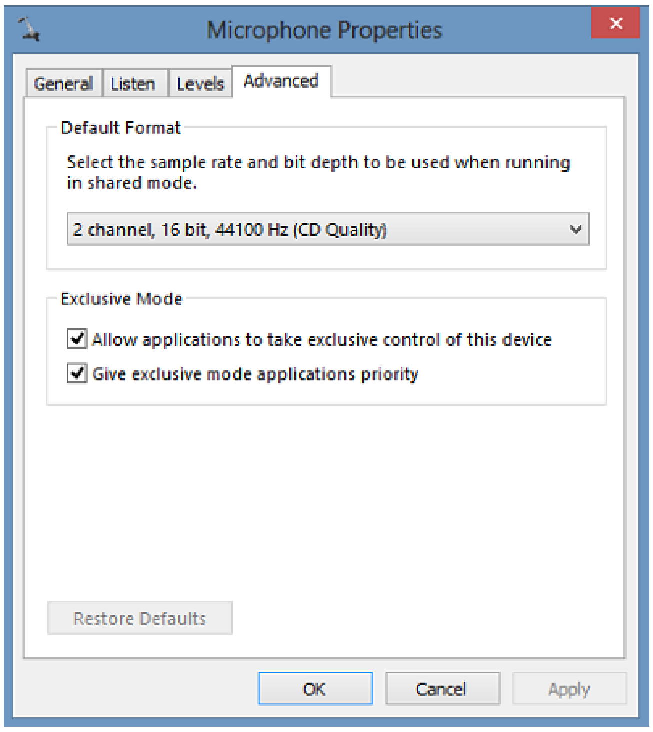
iba AG

# Table of contents

# 1 Audio Interface

The Audio interface in ibaPDA is used to acquire data from a standard sound card, installed and available in the Windows system. By this, it works with built-in devices as well as with USB devices.

The audio device is used in “shared mode”, that means multiple applications can use the same device at the same time. It also implies that for recording, the basic system audio settings cannot be changed. To change any of those options, right-click on the “sound” icon and select “Recording devices”. Double-click on the microphone to be used in ibaPDA and open the “Advanced” tab.


Microphone Properties

Default Format

Select the sample rate and bit depth to be used when running in shared mode.

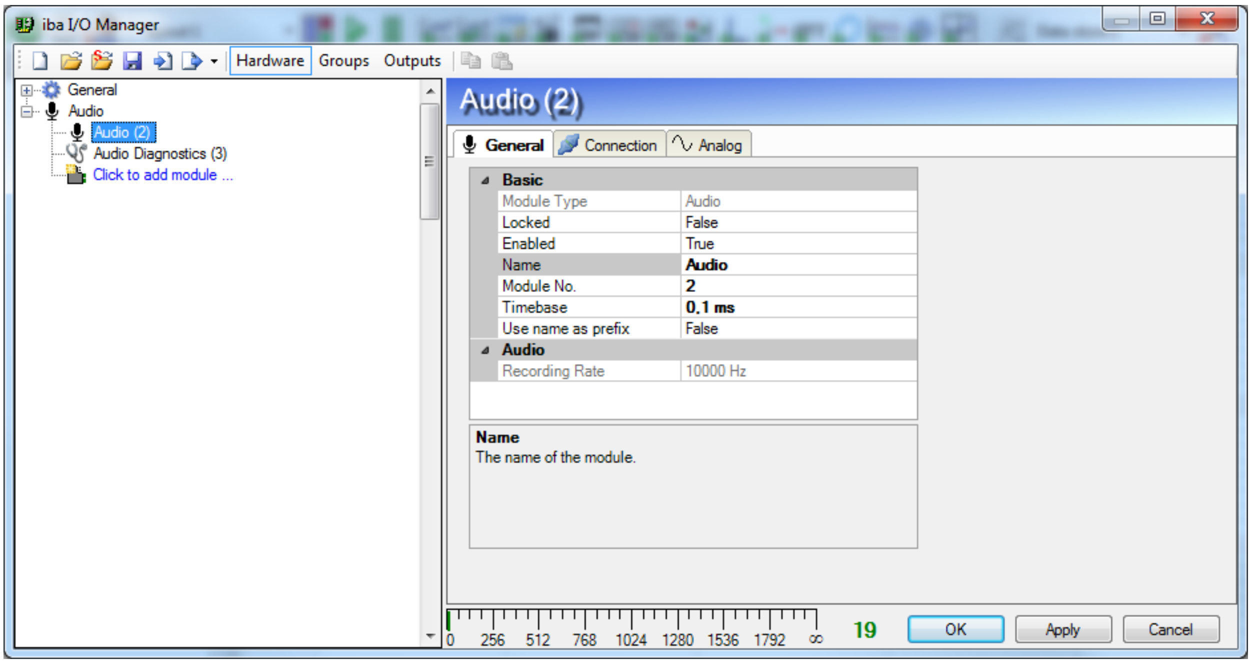
2 channel, 16 bit, 44100 Hz (CD Quality)

Exclusive Mode

[✓] Allow applications to take exclusive control of this device

[✓] Give exclusive mode applications priority

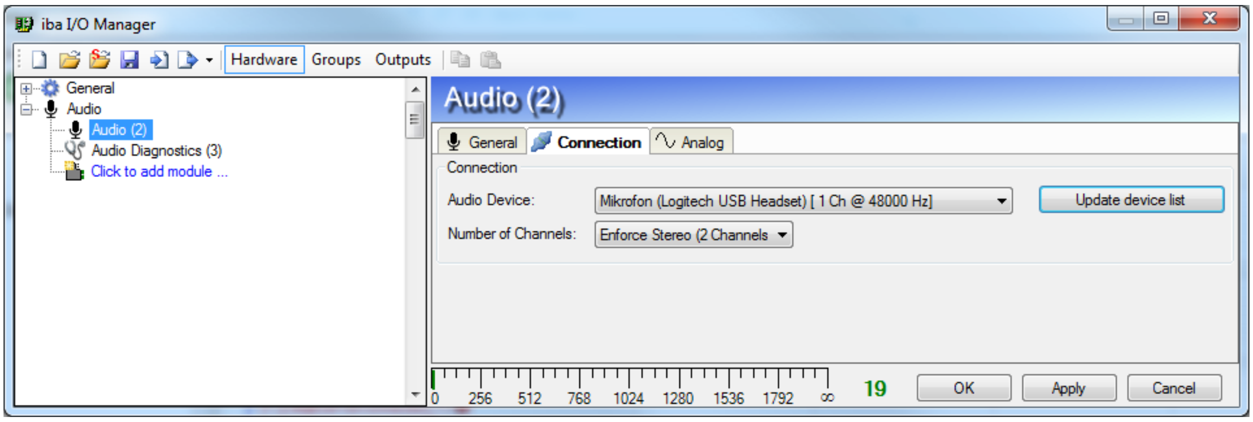
Depending on the installed sound card, you can select now different sample rates and bit depths from the list. For some devices, this is not changeable. When creating a new audio module, the system sample rate can be resampled to the desired recording rate.



| | |
|---|---|
| **Basic** | |
| Module Type | Audio |
| Locked | False |
| Enabled | True |
| Name | **Audio** |
| Module No. | **2** |
| Timebase | **0,1 ms** |
| Use name as prefix | False |
| **Audio** | |
| Recording Rate | 10000 Hz |

**Name**
The name of the module.

In the **General tab**, the Timebase can be changed to the desired rate. It directly relates to the sample frequency shown under “Recording Rate”.

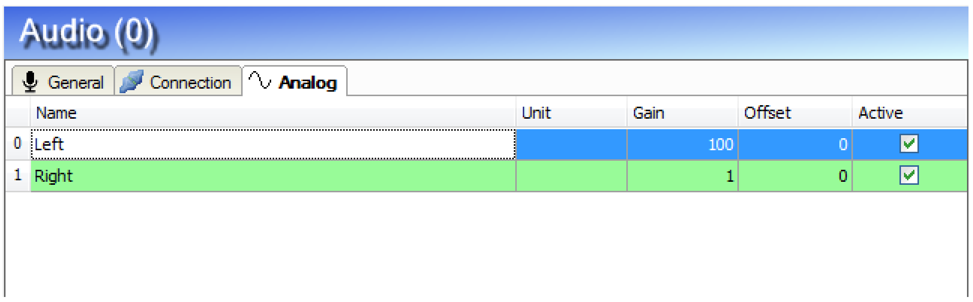


Connection

Audio Device: Mikrofon (Logitech USB Headset) [ 1 Ch @ 48000 Hz]

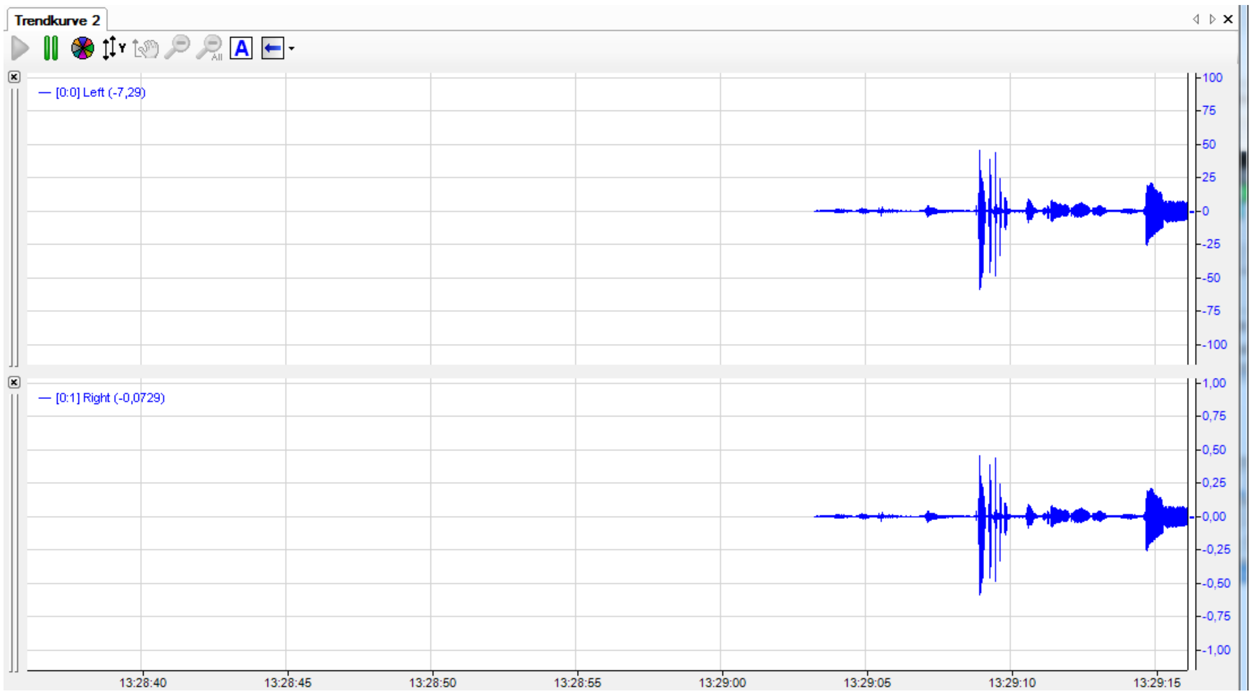
Number of Channels: Enforce Stereo (2 Channels

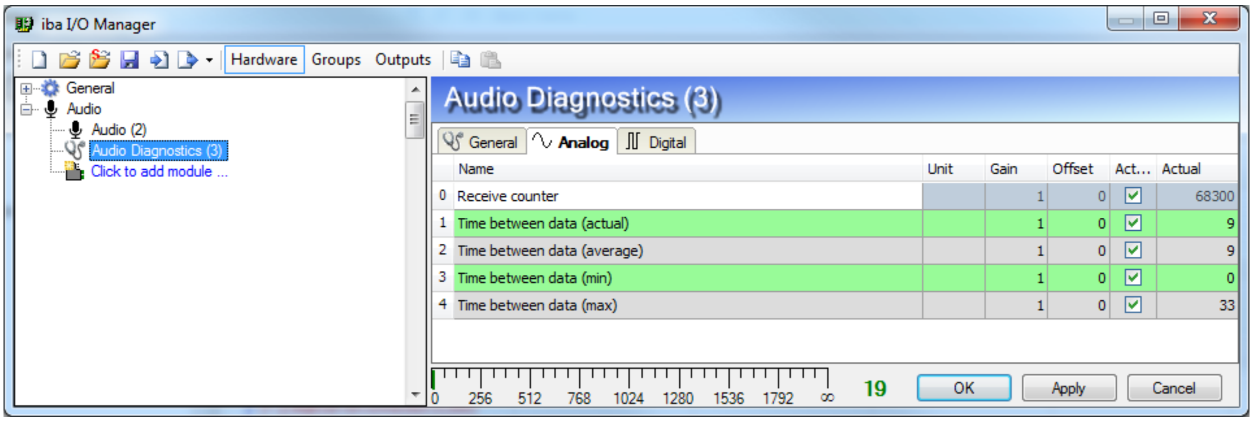
In the **Connection tab**, the desired audio device can be selected from the devices found on the PC hosting the ibaPDA Server. Also the **number of channels** used to record can be changed, the native number of channels can be changed to resample the present channels either to stereo (2 channels) or mono (1 channel).



| | Name | Unit | Gain | Offset | Active |
|---|---|---|---|---|---|
| 0 | Left | | 100 | 0 | ✓ |
| 1 | Right | | 1 | 0 | ✓ |

In the **Analog tab**, the channels can be renamed, and **gain** and **offset** settings can be applied to the values. The samples are received as float values, ranging from -1.0 to 1.0.





Audio Diagnostics (3)

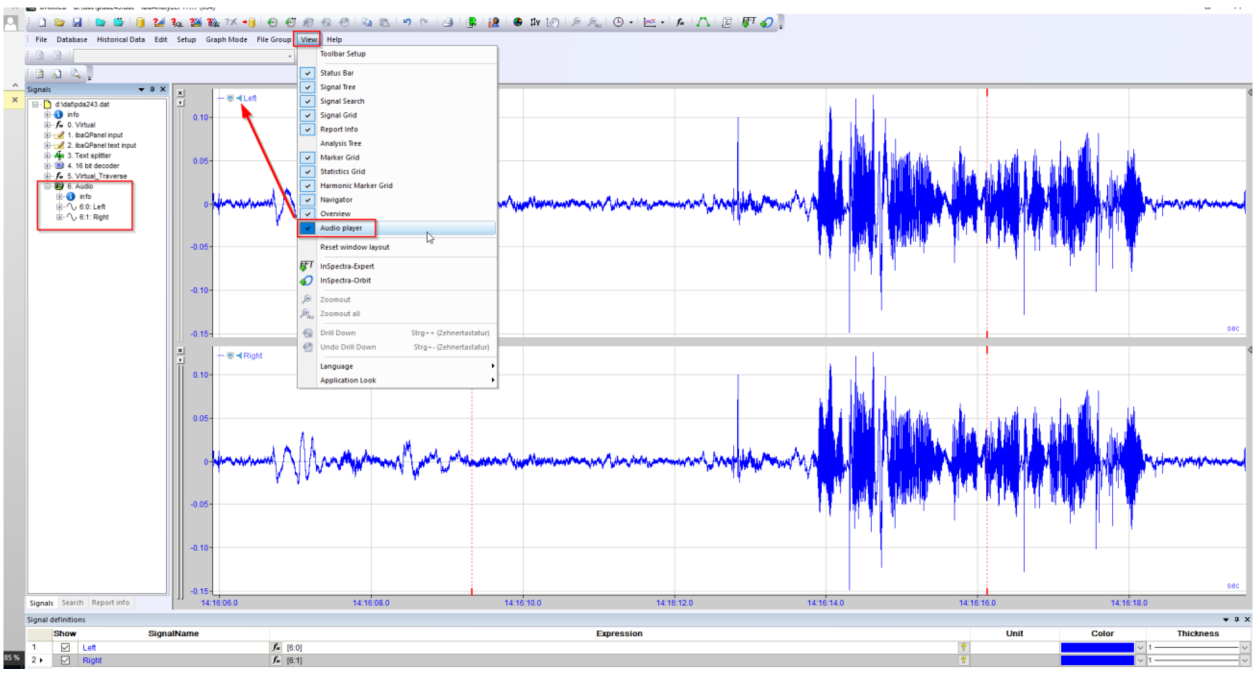
| | Name | Unit | Gain | Offset | Act... | Actual |
|---|---|---|---|---|---|---|
| 0 | Receive counter | | 1 | 0 | ✓ | 68300 |
| 1 | Time between data (actual) | | 1 | 0 | ✓ | 9 |
| 2 | Time between data (average) | | 1 | 0 | ✓ | 9 |
| 3 | Time between data (min) | | 1 | 0 | ✓ | 0 |
| 4 | Time between data (max) | | 1 | 0 | ✓ | 33 |

The corresponding Diagnostics module can be used to show statistic data of the audio channel, and has the “Connected” status as digital signal.

If the audio device is removed while the acquisition is running, the “Connected” signal changes to false. If the same audio device is reconnected during that same session, the connection is recognized and recording continues from that point.

ibaAnalyzer can replay the audio signal recorded in a dat-file:

The audio signals appear as normal signals in the signal tree of your dat-file. Drag an audio signal to the workspace on the right to display it in a chart. When you activate the audio player in the menu View a speaker symbol will appear in the signal legend. When you click on this symbol the signal will be replayed as sound using your standard audio device. The cursor x1 will start to move to indicate the current replay position.



Signal definitions

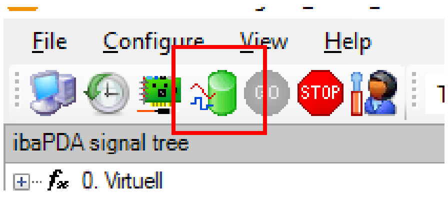
| | Show | SignalName | Expression | Unit | Color | Thickness |
|---|---|---|---|---|---|---|
| 1 | ☑ | Left | [6:0] | | 🟦 | 1 |
| 2 | ☑ | Right | [6:1] | | 🟦 | 1 |

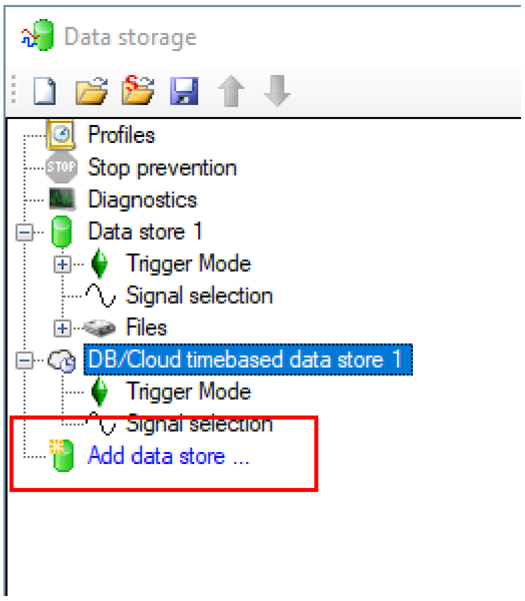
# 2 DB/Cloud timebased data stores

The functionality of the DB/Cloud time-based data store that was introduced in ibaPDA v6.39.0 is extended.

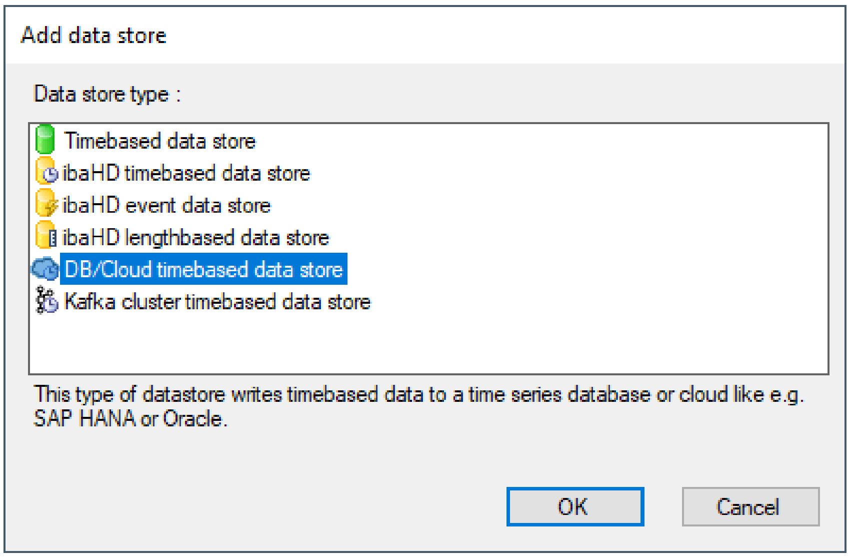
## 2.1 Adding a new data store

To add a new DB/Cloud data store to the data store configuration open the Data storage editor and click on “Add data store…”:





In the popup dialog, the “DB/Cloud timebased data store” item is visible in all installations where at least one license for a DB data store license is active.

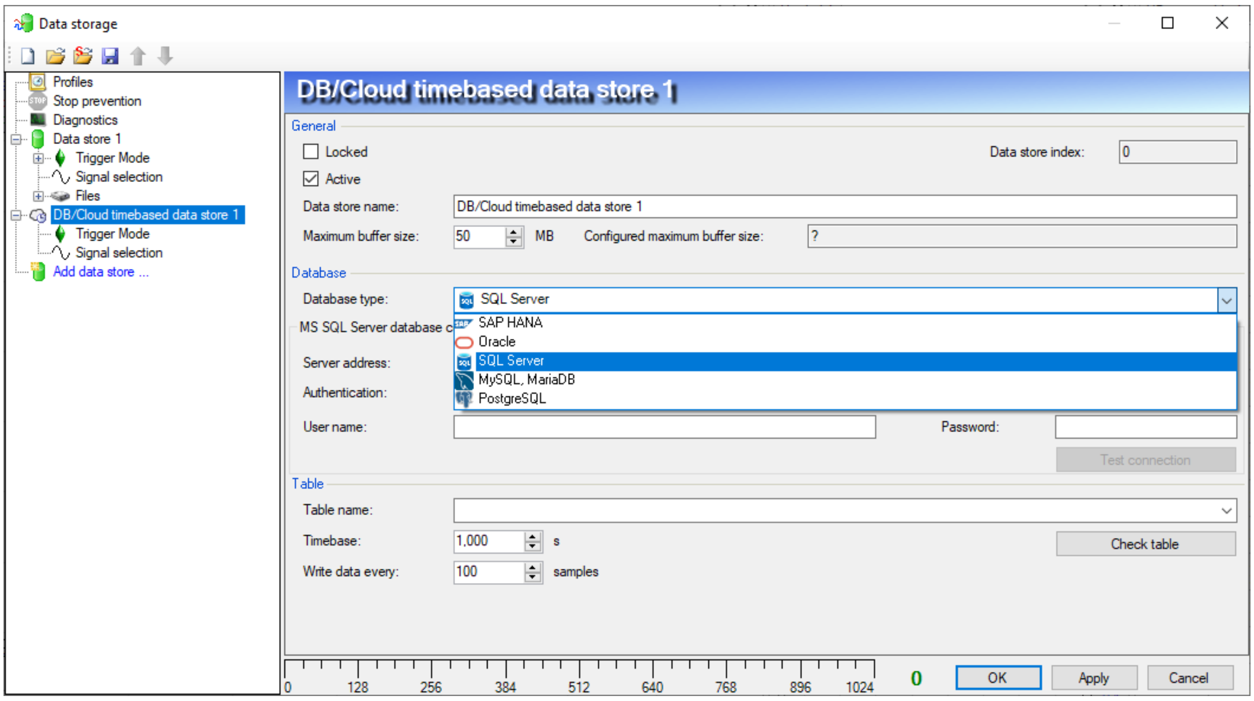

Add data store

Data store type :

- Timebased data store
- ibaHD timebased data store
- ibaHD event data store
- ibaHD lengthbased data store
- DB/Cloud timebased data store
- Kafka cluster timebased data store

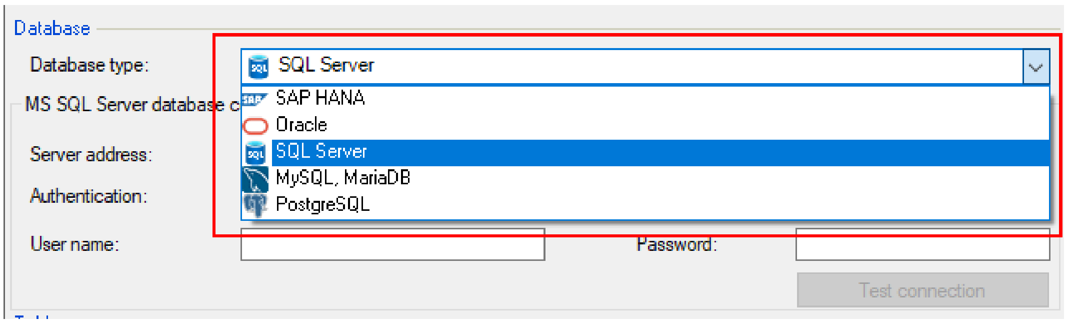
This type of datastore writes timebased data to a time series database or cloud like e.g. SAP HANA or Oracle.

The DB data store is added to the configuration on clicking <OK>.



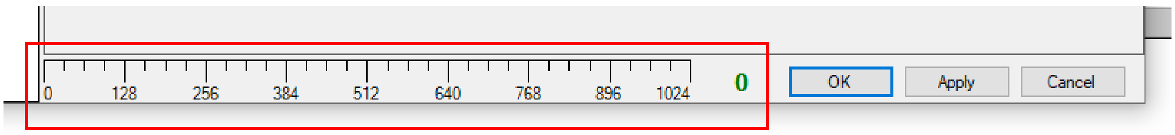
## 2.2 Supported databases and licensing

Five types of databases are now supported: SAP HANA, Oracle, Microsoft SQL Server, MySQL and PostgreSQL. The database type is selected via the drop down menu “Database type” in the data store settings.



Each database type is licensed separately. Per database type, the license allows a total number of signals to be written via DB data stores. Multiple data stores can be used.

The number of signals you can write according to your license is visually displayed in the license bar at the bottom of the data storage configuration dialog.

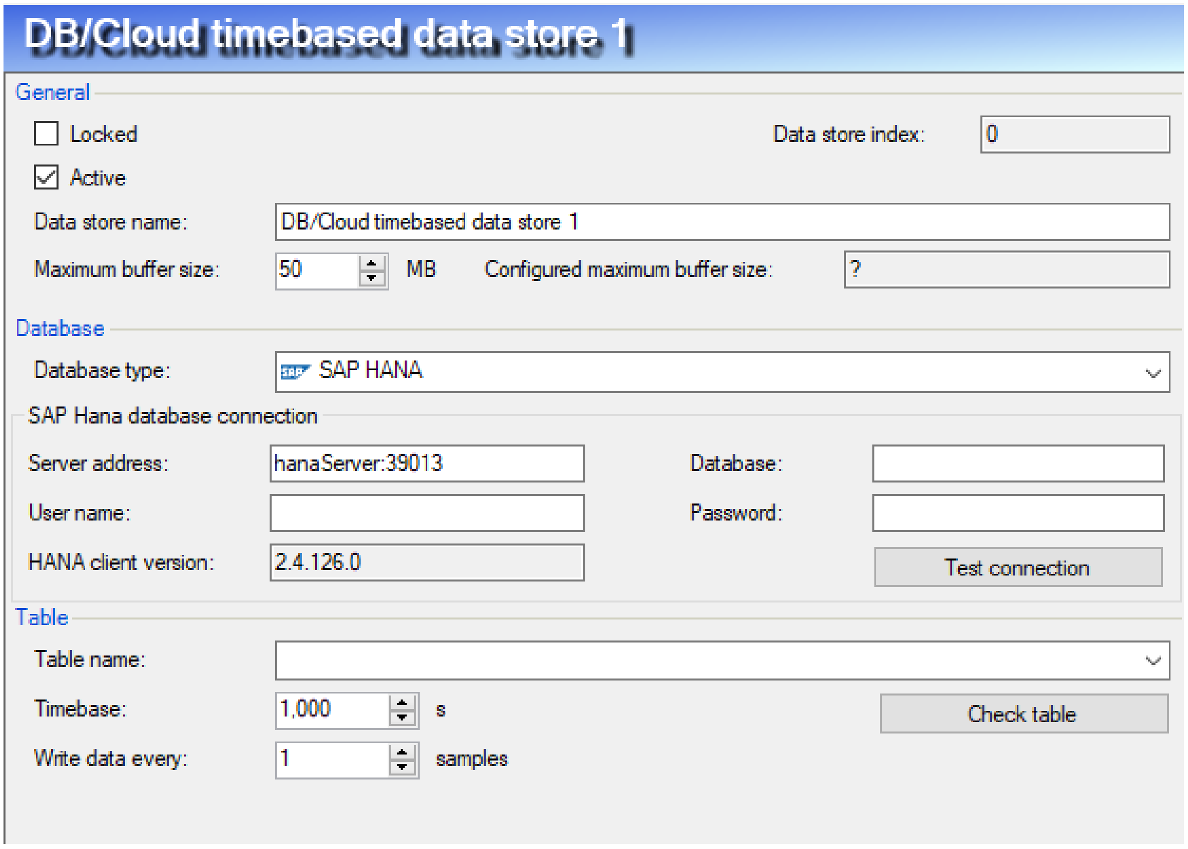


For the database types SAP HANA and Oracle it is required to install a the 32bit version of the corresponding database client. For other supported database types no client installation is required. For SAP HANA the ADO.NET client components are required for ibaPDA They are included in the regular client installation. For Oracle specific client components can be selected during the installation. Here the component “Oracle Data Provider for .NET” is required.

### 2.2.1 SAP HANA

For SAP HANA, nothing has changed on the surface. Below the surface, the interface was made more generic. The intention was to make sure that existing SAP HANA users will be able to use their configuration without modifications.

The client version that is displayed may be different than before. It is now the version of the .NET data provider for SAP HANA that is shown. A SAP HANA client installation is still required.



DB/Cloud timebased data store 1

General

| | | | |
|---|---|---|---|
| ☐ Locked | | Data store index: | 0 |
| ☑ Active | | | |
| Data store name: | DB/Cloud timebased data store 1 | | |
| Maximum buffer size: | 50 MB | Configured maximum buffer size: | ? |

Database

Database type: SAP HANA

SAP Hana database connection

| | | | |
|---|---|---|---|
| Server address: | hanaServer:39013 | Database: | |
| User name: | | Password: | |
| HANA client version: | 2.4.126.0 | | |

Table

| | |
|---|---|
| Table name: | |
| Timebase: | 1,000 s |
| Write data every: | 1 samples |

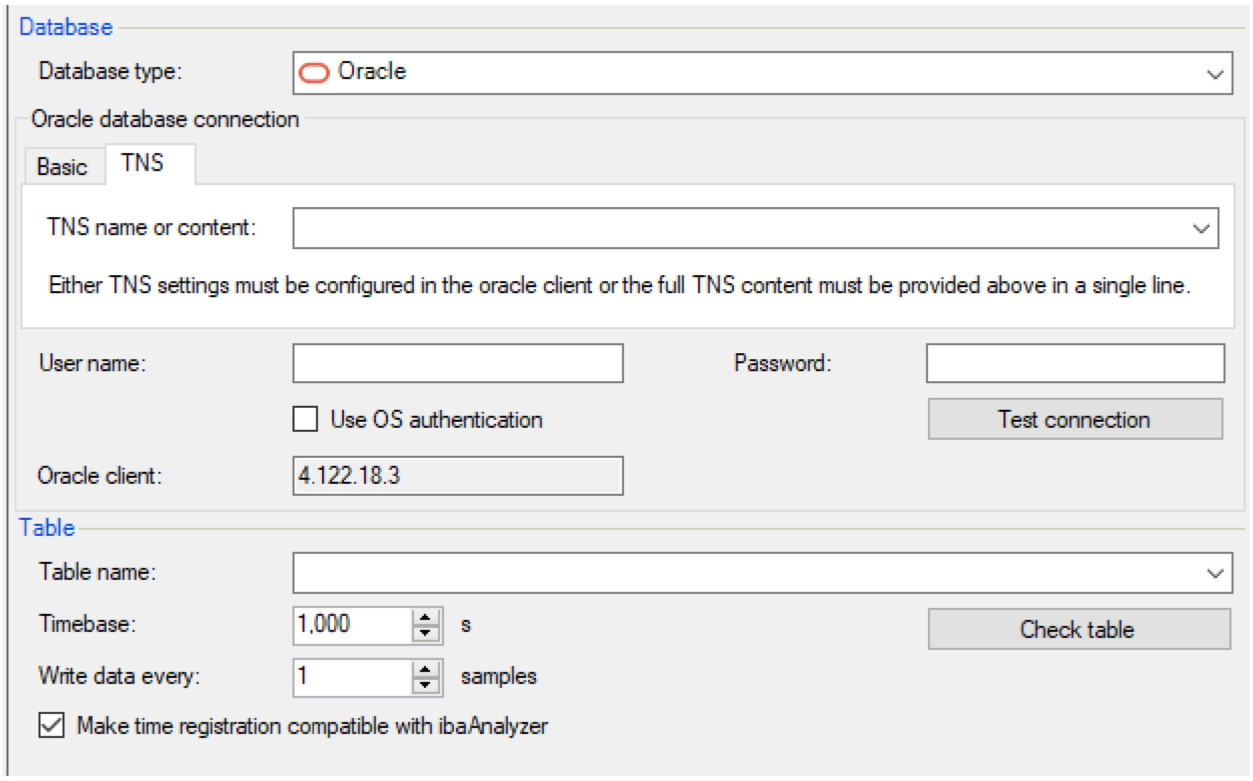
The following configuration options are required to connect to a SAP HANA database.

- **Server address**, including port number. Server address and port number are separated by a colon.
- **Database**: the name of the database to connect to
- **User name**: the user that connects to the database
- **Password**: the password for the connection. The text entered in the password box is not readable.
- **HANA client version**: a SAP HANA client installation is automatically detected. If no version number is displayed here, it will not be possible to connect to the SAP HANA database.
  - The following is required to detect a SAP HANA .NET Data Provider:
    - The client assembly must be installed in the GAC (global assembly cache)

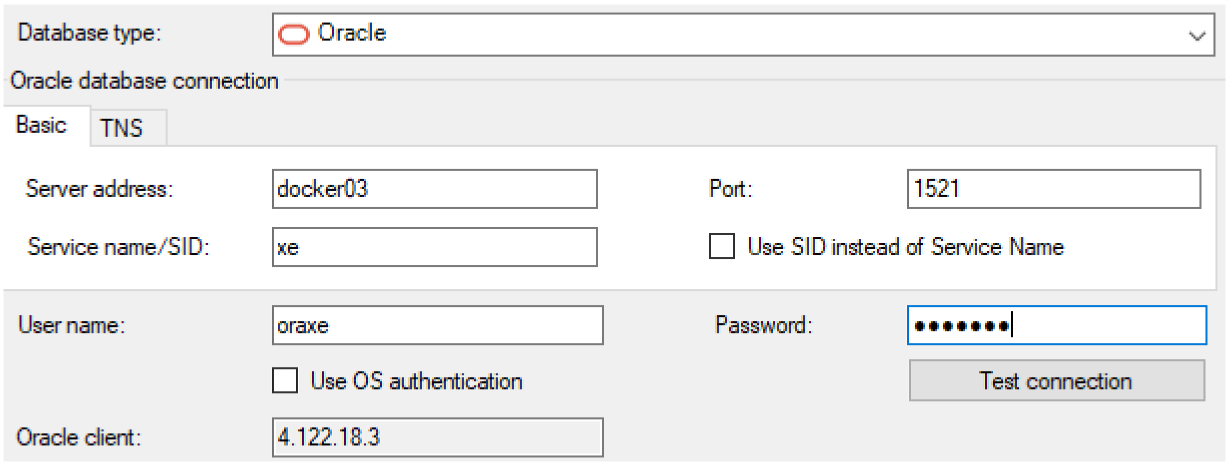
    - The client must be listed in the *machine.config*, a configuration file for the .NET Framework. The *machine.config* file is located here *%windir%\Microsoft.NET\Framework\v4.0.30319\Config*
  - Both requirements normally are handled automatically by the SAP HANA client installation.
  - In case of problems installing a different version of the SAP HANA client can help.
- **Test connection** button: pressing this button starts an attempt to connect to the database.
  - If the connection succeeded, all the table names are fetched. The table names that could be usable will be listed in the dropdown in the field **Table name**

### 2.2.2 Oracle

Oracle can be configured with either **TNS** configuration


Database
Database type: Oracle
Oracle database connection
Basic | TNS
TNS name or content:
Either TNS settings must be configured in the oracle client or the full TNS content must be provided above in a single line.
User name: Password:
☐ Use OS authentication
Test connection
Oracle client: 4.122.18.3
Table
Table name:
Timebase: 1,000 s
Check table
Write data every: 1 samples
☑ Make time registration compatible with ibaAnalyzer

Or with **Basic** connection configuration. The tab that is selected makes a difference in the way that the connection to the Oracle server or cluster is established.



Database type: Oracle

Oracle database connection

Basic | TNS

| | | | |
|---|---|---|---|
| Server address: | docker03 | Port: | 1521 |
| Service name/SID: | xe | ☐ Use SID instead of Service Name | |
| User name: | oraxe | Password: | ••••••• |
| | ☐ Use OS authentication | | Test connection |
| Oracle client: | 4.122.18.3 | | |

In both cases, the following options are required:

- **User name** for the connection
- **Password** for the connection
- Whether or not to **Use OS authentication**
- The **Oracle client** must be configured in order to be able to connect to the Oracle database. The Oracle connectivity in ibaPDA uses ODP.NET to make the connection from .NET to Oracle. ODP.NET is the Oracle .NET Data Provider, which must be installed with your Oracle client installation. Additional action is needed to install the Oracle client.
- **Use OS authentication**: the account of the user with which the ibaPDA service is running is used to make a connection to the Oracle server or cluster. Usually that is the built-in SYSTEM account. But the user that runs the ibaPDA service can also be changed to another user with administrative privileges which is also configured as Oracle database user.

### 2.2.2.1 TNS configuration

TNS has some more possibilities than basic configuration. It enables you to use a failover cluster instead of only a single server setup for the Oralcle database, for example.

If you use TNS configuration, there are two possibilities: either you enter the name of a connection that is available in *tnsnames.ora* (the Oracle client configuration file) in the Oracle client directory. Note that the client directory is the directory where the Oracle client dll is located that is configured in *machine.config*. For the Oracle client usage by ibaPDA it is not required to set environment variables as it is required in other Oracle client use cases.

If you want to use *tnsnames.ora*, ODP.NET searches for that file in the following order:

- First in the Oracle client directory. That is a parent directory of the directory where the *Oracle.ManagedDataAccess.dll* is located. Multiple clients can be installed, but only one ODP.NET installation can be configured machine wide, in *machine.config*.
- Then in the directory from where the application using the client is started. That means if there is no *tnsnames.ora* file in the Oracle client directory, you can place a *tnsnames.ora*

file in the ibaPDA server directory. By default, this is *C:\Program Files (x86)\iba\ibaPDA\Server*.

Instead of entering a TNS name that is configured in *tnsnames.ora*, it is also possible to enter a complete section like it can be found in a *tnsnames.ora* file. For example, you can enter something like

**(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.99.101)(PORT=15210)))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=xe)))**

in one single line. A use case where that would be useful is if you want to keep ibaPDA running while an upgrade of the Oracle server is in progress. ODP.NET will only read *tnsnames.ora* once, so in a case when its content changes, you can paste the specific part of the new content of *tnsnames.ora* in the Oracle database connection setting of your DB data store and apply the new data store configuration.

### 2.2.2.2 Basic configuration

When using the basic configuration, ibaPDA can connect to one Oracle server. A failover cluster can not be configured with these settings.

The following settings can be used:

- **Server address**: The IP address or DNS name of the Oracle server
- **Port**: The TCP port on which the Oracle server listens for new connections.
- **Service name/SID**: The name of the Oracle service, as configured in the Oracle server.
- **Use SID instead of Service name**: in some rare cases this might need to be checked. Please refer to the Oracle documentation for the difference between SID and service name.

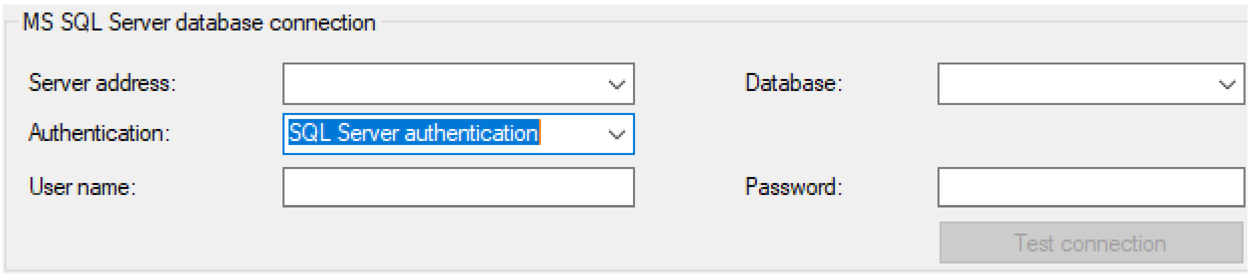
### 2.2.2.3 Oracle client installation

More information about how to install and configure the Oracle client can be found in the separate document

- **Installation instructions - Oracle® Data Provider for .NET (ODP.NET)** (*ibaNotes_Oracle_ODP.NET_Installation_v1.0_en.pdf*)

## 2.2.3 SQL Server, Azure SQL

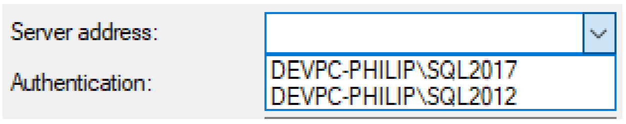
No additional client installation is required to connect to SQL Server. ibaPDA can directly establish a connection to SQL Server or Azure SQL. The client is included in .NET Framework, which is required for ibaPDA to run.


MS SQL Server database connection

Server address:
Database:
Authentication: SQL Server authentication
User name:
Password:

### 2.2.3.1 Server address

The IP address, host name or combination of host name and instance name of the SQL Server database server. The port is always assumed to be the same, the default port.

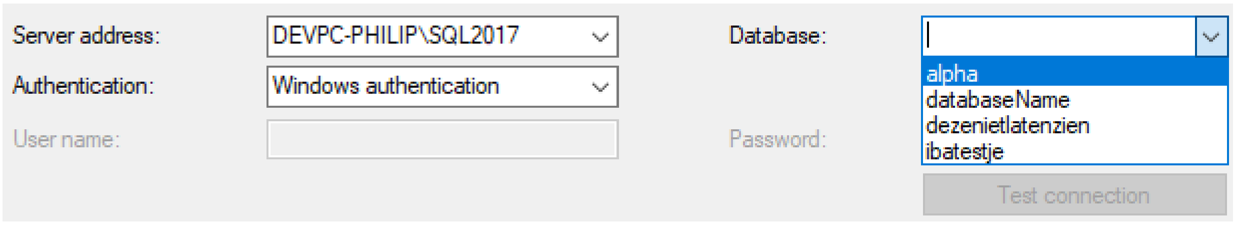


Server address:
Authentication:
DEVPC-PHILIP\SQL2017
DEVPC-PHILIP\SQL2012

The server address can be typed in or can be chosen from the dropdown list. The dropdown shows the database instances that are reachable from the ibaPDA server. Note that no network search for SQL Server instances is performed, since such a search can potentially take a long time to finish. Only the registered SQL Server instances on the computer where ibaPDA server is installed are shown in the dropdown.

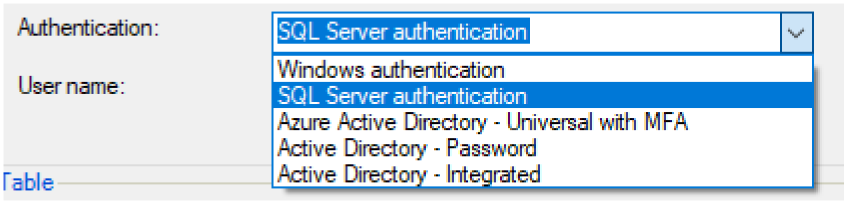
### 2.2.3.2 Database

The database can be typed in here. If the server address, authentication and if required user name and password are filled in, the dropdown shows all the available databases for the connection. Usually for SQL Server installations on premises, it will not be required to type in a database name manually, but one can be selected.



Server address: DEVPC-PHILIP\SQL2017
Authentication: Windows authentication
User name:
Database:
alpha
databaseName
dezenietlatenzien
ibatestje
Password:

### 2.2.3.3 Authentication



Authentication: SQL Server authentication
User name:
Windows authentication
SQL Server authentication
Azure Active Directory - Universal with MFA
Active Directory - Password
Active Directory - Integrated

There are 5 authentication types.

- **Windows authentication**

  The account of the user with which the PDA server is running is used to make a connection to the SQL Server instance. Usually that is the built-in SYSTEM account, but the user that runs ibaPDAServer can also be changed to another user with administrative privileges.

- **SQL Server authentication**

  Authentication that is stored within the database. A user name and password are required to use this authentication method.

- **Azure Active Directory – Universal with MFA**

  Multi-factor authentication, as configured on Azure. More information can be found here: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-aad-authentication-configure?tabs=azure-powershell

- **Active Directory – Password or Integrated**

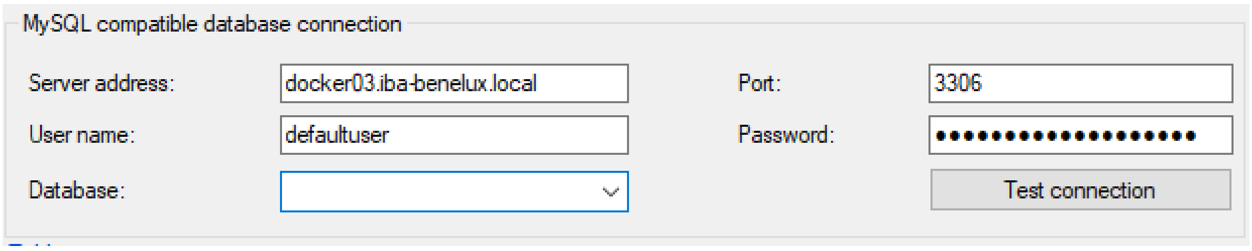
  These authentication methods can be used for members of a user group in AD that has access to the database. A trust relationship must be established between the user and its device and the domain. SSMS (SQL Server Management Studio) version 17 or above can be used to test if this kind of connection should work.

### 2.2.3.4 User name and password

The user name and password can be filled in in these fields, if they are required.

## 2.2.4 MySQL, MariaDB

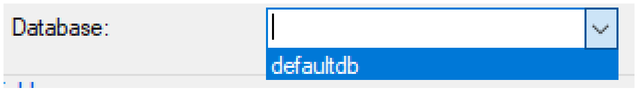
For connectivity to MySQL or MariaDB databases no extra client installation is required.



MySQL compatible database connection

| | | | |
|---|---|---|---|
| Server address: | docker03.iba-benelux.local | Port: | 3306 |
| User name: | defaultuser | Password: | ●●●●●●●●●●●●●●●●●● |
| Database: | | | Test connection |

- **Server address**: the name or IP address of the server to connect to
- **Port**: the TCP port on which the MySQL or compatible server is listening
- **User name**: the name of the user that connects to the database
- **Password**: the password to use for the connection
- **Database**: the name of the database to connect to. If the other fields are correctly filled in and if the database server is online, the available databases are shown in the dropdown list.

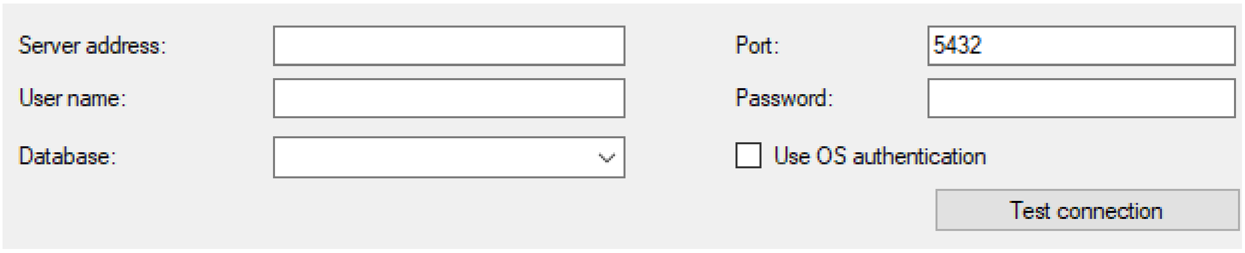


Database:

defaultdb

## 2.2.5 PostgreSQL

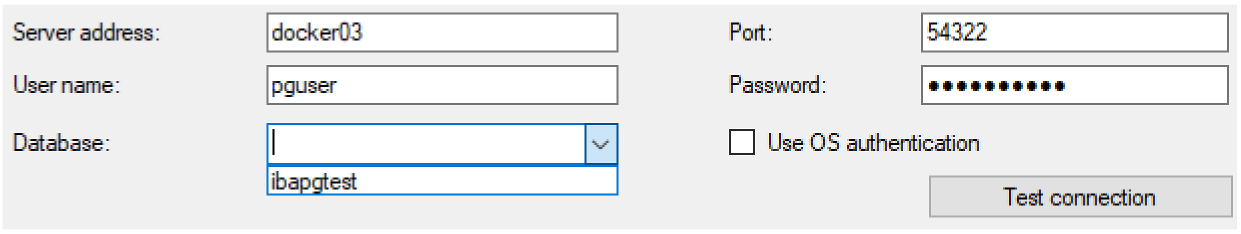
For connectivity to PostgreSQL or compatible databases no extra client installation is required.



| | | | |
|---|---|---|---|
| Server address: | | Port: | 5432 |
| User name: | | Password: | |
| Database: | | ☐ Use OS authentication | |
| | | | Test connection |

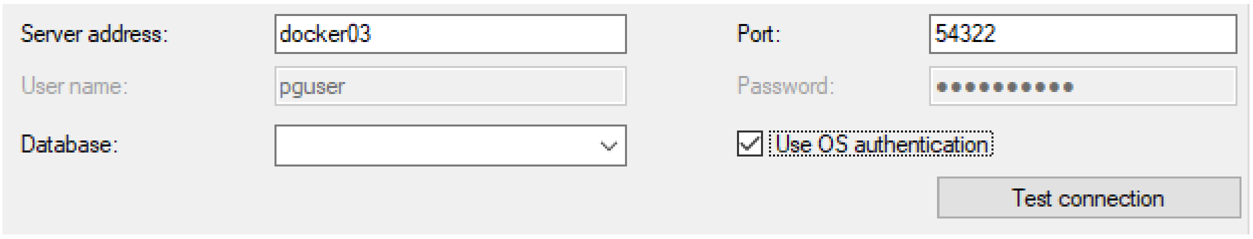
- **Server address**: the name or IP address of the server to connect to
- **Port**: the TCP port on which the PostgreSQL or compatible server is listening
- **User name**: the name of the user that connects to the database
- **Password**: the password to use for the connection

- **Database**: the name of the database to connect to. If the other fields are correctly filled in and if the database server is online, the available databases are shown in the dropdown.



| | | | |
|---|---|---|---|
| Server address: | docker03 | Port: | 54322 |
| User name: | pguser | Password: | ●●●●●●●●●●● |
| Database: | ibapgtest | ☐ Use OS authentication | Test connection |

- **Use OS authentication**: can be enabled to connect using the Windows user that the ibaPDA server is running with. Usually that is SYSTEM, but the service configuration can be adapted so that any user with administrative privileges can be the user of the ibaPDA service.



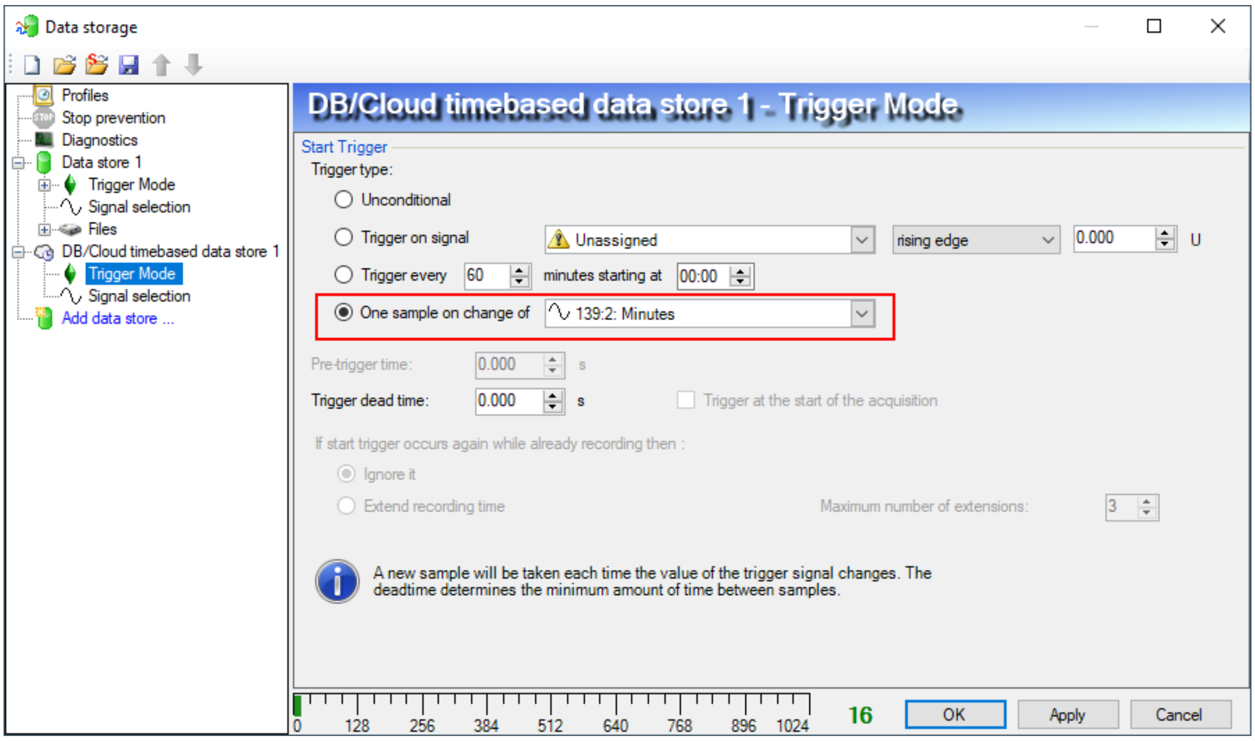
| | | | |
|---|---|---|---|
| Server address: | docker03 | Port: | 54322 |
| User name: | pguser | Password: | ●●●●●●●●●●● |
| Database: | | ☑ Use OS authentication | Test connection |

## 2.3 Trigger Mode

The trigger mode of the time-based datastores is expanded with a new option: **One sample on change**. This trigger mode is applicable for the following data stores:

- DB/Cloud timebased data store
- Kafka cluster timebased data store
- MindSphere timebased data store
- MQTT timebased data store

To select this new trigger mode, select the *Trigger Mode* node of your data store in the data storage configuration, activate the “One sample on change …” trigger type and select a signal that will to be monitored for changes.



DB/Cloud timebased data store 1 - Trigger Mode

Start Trigger
Trigger type:
Unconditional
Trigger on signal: Unassigned, rising edge, 0.000 U
Trigger every 60 minutes starting at 00:00
One sample on change of: 139:2: Minutes
Pre-trigger time: 0.000 s
Trigger dead time: 0.000 s
Trigger at the start of the acquisition
If start trigger occurs again while already recording then :
Ignore it
Extend recording time
Maximum number of extensions: 3

A new sample will be taken each time the value of the trigger signal changes. The deadtime determines the minimum amount of time between samples.

When the value of the selected signal changes, one sample will be recorded in the data store that is being configured. Unlike “Trigger on signal”, the recording will immediately stop after one sample, until the next signal change is detected.

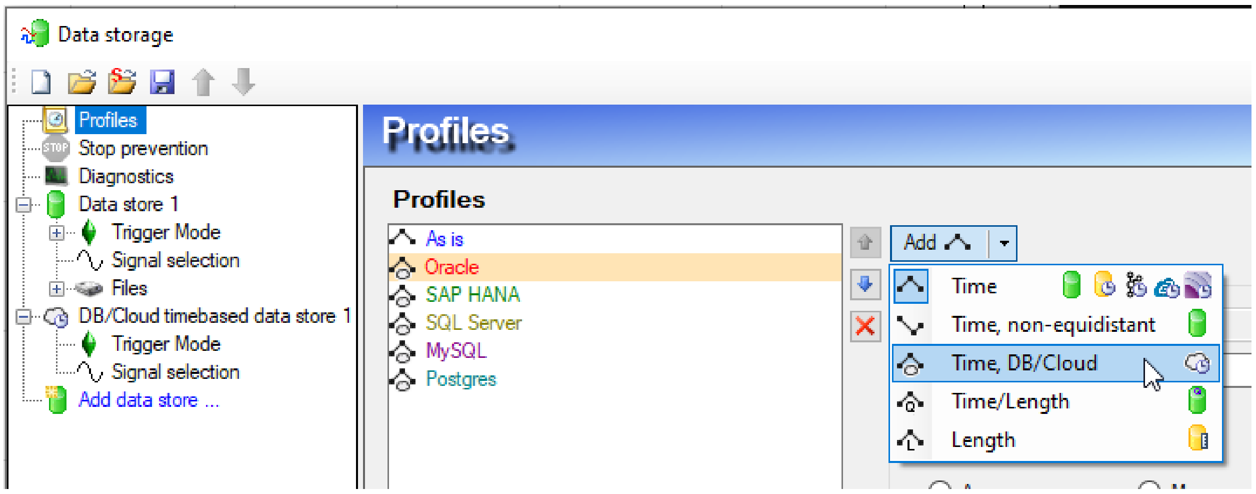
A deadtime can be configured to determine a minimum amount of time between samples. Before the deadtime has elapsed, no new sample will be recorded.

For the SAP Hana data store, a time series table is used. Therefore the timestamp of the trigger will be aligned with the time base of the time series table in SAP Hana. The other data stores use the time stamp of the trigger (or as soon as possible after the trigger).

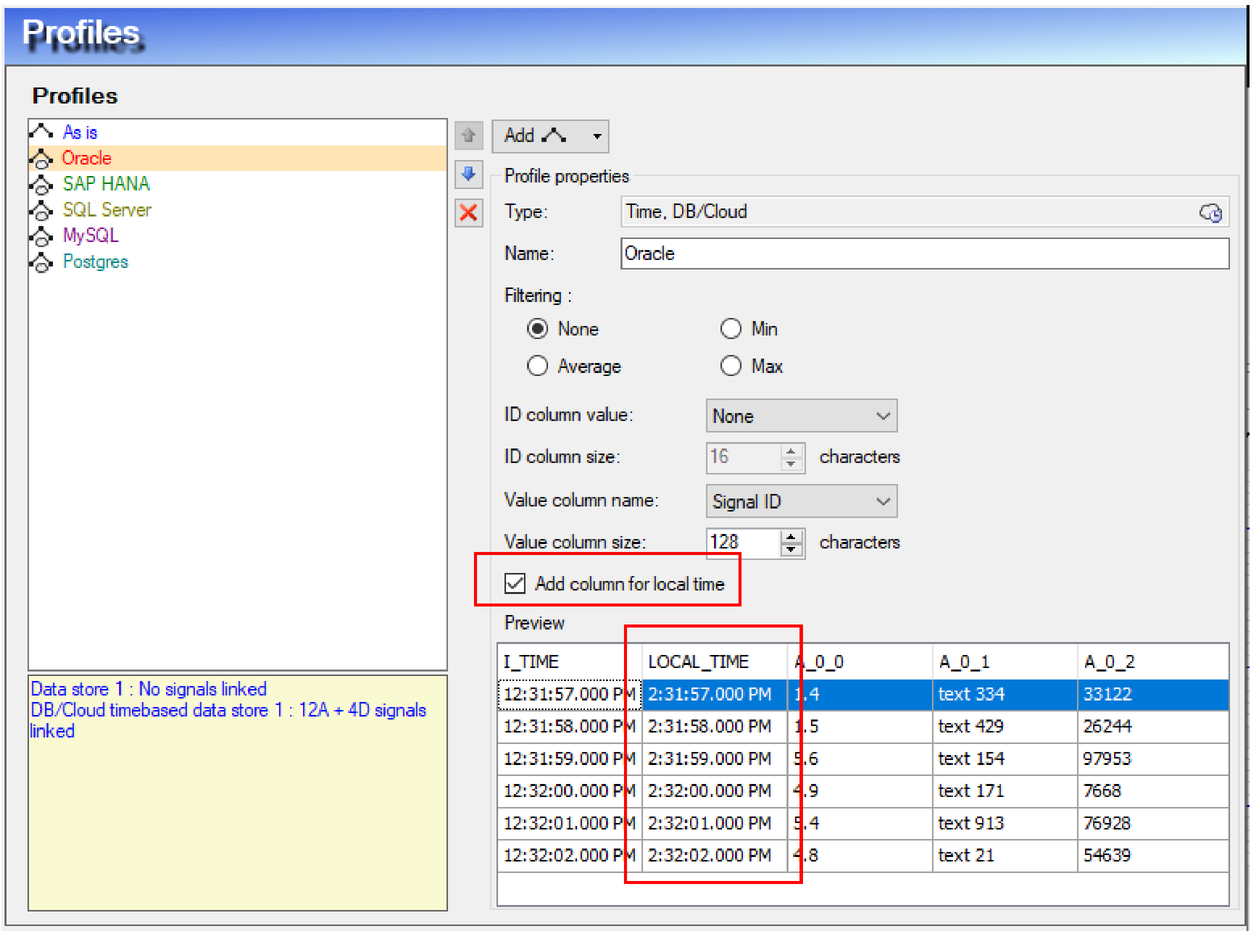
For the rest, the trigger mode setup is the same as for the previously existing time-based data stores.

## 2.4 Profiles

For the signal assignment in a DB/Cloud data store you have to use a profile of the type ***Time, DB/Cloud***. The scheme of the table that will be created by ibaPDA is determined by choosing various settings for the ID column (if required) and the value columns in the profile. The resulting table scheme according to your settings is shown for reference in the preview area.



The profile configuration still is as originally introduced in v6.39.0. A new option "Add column for local time" is available in the profile now. If that option is enabled, an additional column LOCAL_TIME is added to the table. The system time is recorded in that column, alongside the UTC time in column I_TIME.


Profiles

Profiles

As is
Oracle
SAP HANA
SQL Server
MySQL
Postgres

Data store 1 : No signals linked
DB/Cloud timebased data store 1 : 12A + 4D signals linked

Add

Profile properties

Type: Time, DB/Cloud

Name: Oracle

Filtering :

[✓ None] Min
Average Max

ID column value: None

ID column size: 16 characters

Value column name: Signal ID

Value column size: 128 characters

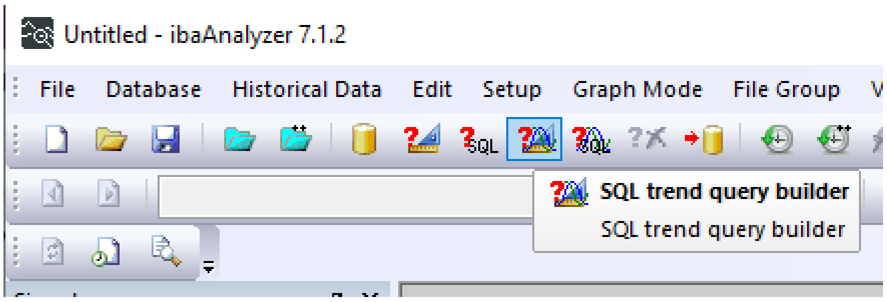
[✓ Add column for local time]

Preview

| I_TIME | LOCAL_TIME | A_0_0 | A_0_1 | A_0_2 |
|---|---|---|---|---|
| 12:31:57.000 PM | 2:31:57.000 PM | 1.4 | text 334 | 33122 |
| 12:31:58.000 PM | 2:31:58.000 PM | 1.5 | text 429 | 26244 |
| 12:31:59.000 PM | 2:31:59.000 PM | 5.6 | text 154 | 97953 |
| 12:32:00.000 PM | 2:32:00.000 PM | 4.9 | text 171 | 7668 |
| 12:32:01.000 PM | 2:32:01.000 PM | 5.4 | text 913 | 76928 |
| 12:32:02.000 PM | 2:32:02.000 PM | 4.8 | text 21 | 54639 |

## 2.5 ibaAnalyzer compatibility

It makes sense to record data in ibaPDA and to be able to read the recorded data in ibaAnalyzer. The **SQL trend query builder** in ibaAnalyzer can be used to read tables recorded with ibaPDA.



ibaPDA attempts to use the best fitting data type for the target database, while ibaAnalyzer has always had its own already existing expectations regarding table names, column names and data types.

Therefore, ibaAnalyzer will be compatible with the DB/Cloud data stores soon after the release of ibaPDA 7.2.0. The compatibility is implemented in ibaAnalyzer for Oracle, SQL Server, PostgreSQL and MySQL compatible databases. Those databases are supported by both ibaPDA and ibaAnalyzer.

A couple of known incompatibilities remain.

Digital signals are stored in PostgreSQL using a boolean type and in SQL Server as a BIT type. Those data types are currently read as a character “1” or “0” in ibaAnalyzer. For those database types, in order to read digital signals as a numeric zero or one a conversion needs to be done, either in SQL or with ibaAnalyzer expressions.

## 2.6 Limitations

Tables and statements have limitations depending on the database type. Oracle for example has a limit of 1000 columns per table. That means for the DB data store that less than 1000 signals can be stored in a table in Oracle.

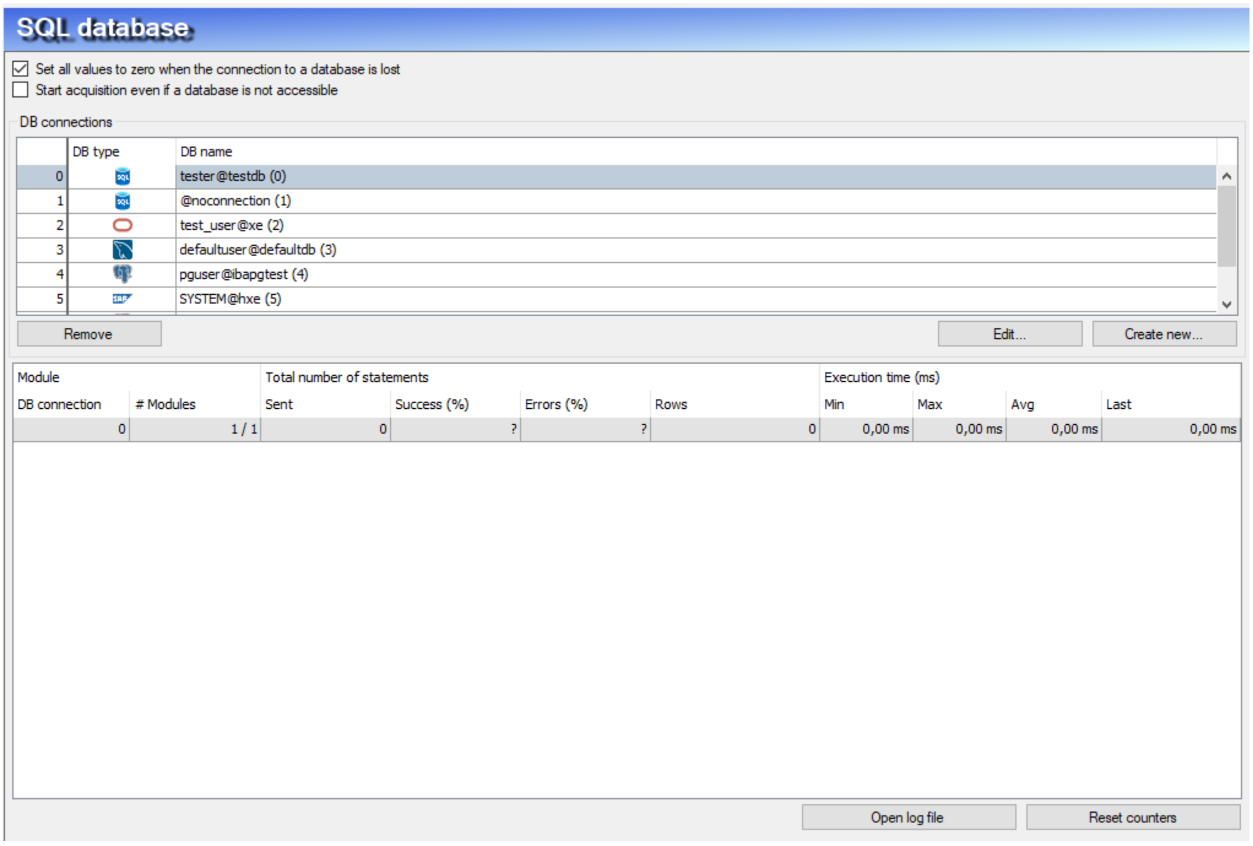
Each DB/Cloud data store module uses exactly one table in the corresponding database. Therefore, the number of signals per data store module is limited to the amount imposed by the database.

# 3 SQL interface

The SQL interface holds the information about connections to databases. The input modules for the SQL interface are called *SQL queries*. *SQL commands* are the output modules. There are also diagnostic modules for SQL modules.

## 3.1 Database configuration in the interface control

Upon selecting the “SQL database” node in iba I/O Manager, the SQL database interface control is shown on the right hand side of the I/O Manager.



[✓] 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 |
|---|---|---|
| 0 | | tester@testdb (0) |
| 1 | | @noconnection (1) |
| 2 | | test_user@xe (2) |
| 3 | | defaultuser@defaultdb (3) |
| 4 | | pguser@ibapgtest (4) |
| 5 | | SYSTEM@hxe (5) |

| Module | | Total number of statements | | | | Execution time (ms) | | | |
|---|---|---|---|---|---|---|---|---|---|
| DB connection | # Modules | Sent | Success (%) | Errors (%) | Rows | Min | Max | Avg | Last |
| 0 | 1 / 1 | 0 | ? | ? | 0 | 0,00 ms | 0,00 ms | 0,00 ms | 0,00 ms |

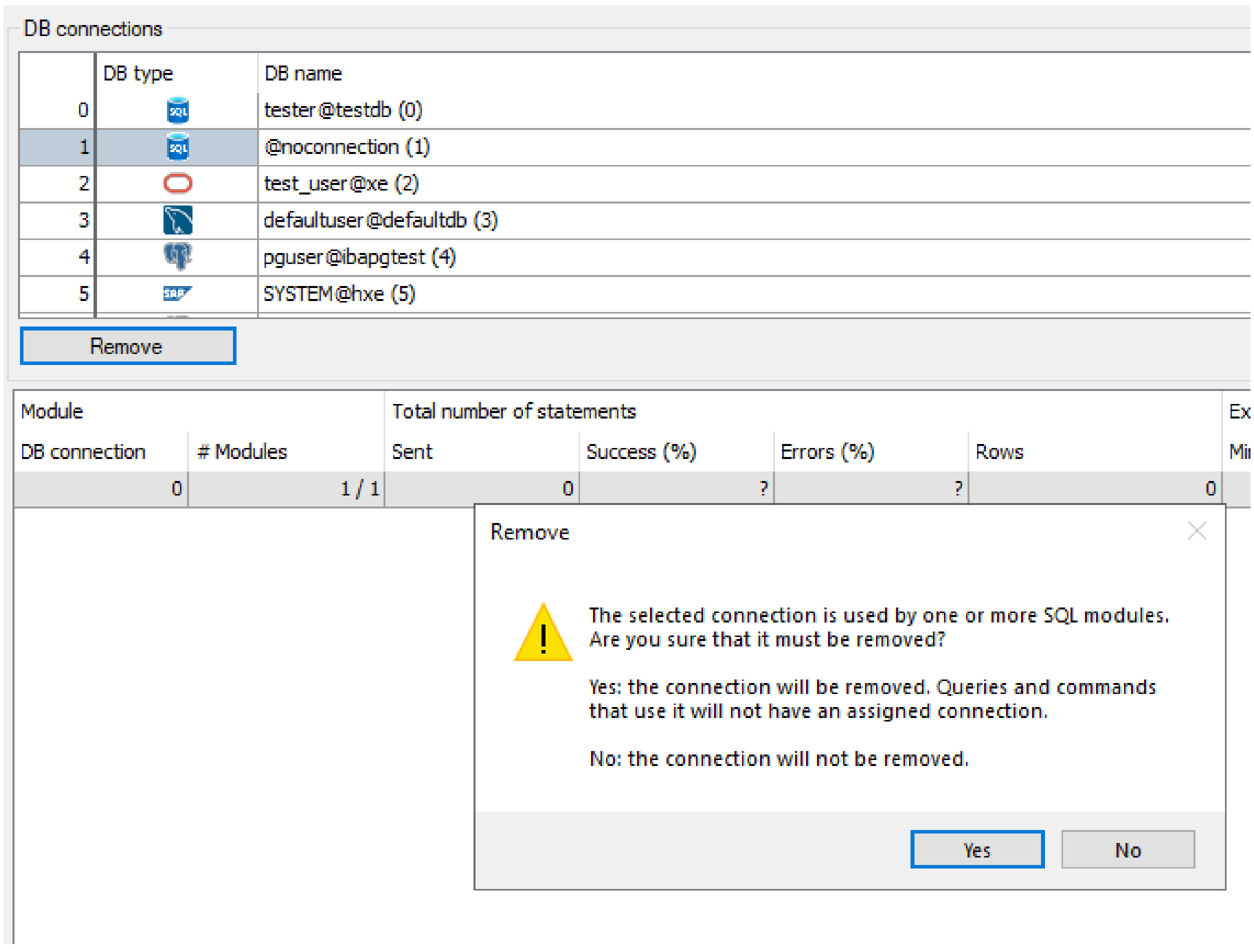
At the top, 2 checkboxes are displayed

- **Set all values to zero…** makes the signal values zero if connection to a database is lost during acquisition. If connection is lost, ibaPDA will keep trying to reconnect to the database until the acquisition is stopped. Note that it can take some time before a signal value is actually set to zero, because a timeout of multiple seconds is not necessarily a sign that there is no longer a connection to a database.
- **Start acquisition even…** ensures that the acquisition can be started, even if a database is not available at the time when the acquisition is started. ibaPDA will cyclically try to establish the connection in parallel to the running acquisition.

### 3.1.1 DB connections

In this part of the control, database connections can be added, removed or changed. Every database connection gets a connection ID, which is displayed at the left of the table. The DB type columns displays an icon that reflects the database type. The third column displays a name by which the DB connection can be recognized in the SQL query and command modules.

If a connection is removed, a warning is shown if modules are using the connection.


DB connections

| | DB type | DB name |
|---|---|---|
| 0 | | tester@testdb (0) |
| 1 | | @noconnection (1) |
| 2 | | test_user@xe (2) |
| 3 | | defaultuser@defaultdb (3) |
| 4 | | pguser@ibapgtest (4) |
| 5 | | SYSTEM@hxe (5) |

| Module | | Total number of statements | | | | Ex |
|---|---|---|---|---|---|---|
| DB connection | # Modules | Sent | Success (%) | Errors (%) | Rows | Mi |
| 0 | 1 / 1 | 0 | ? | ? | 0 | |

Remove

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.

When adding or editing a connection, the same connection settings can be used as for the DB data stores.

Since the SQL modules refer to a connection in the SQL interface, it is important to make sure that the modules don’t become unmapped. Otherwise, the connection in the unmapped module becomes unassigned.

### 3.1.2 Connection diagnose

At the bottom of the interface control, connection diagnostics are shown. Note that due to the nature of relational databases, it is possible that error counts increase much slower than success counts. Take that into account when diagnosing problems with database connections.

## 3.2 Supported databases and licensing

The databases that are supported are the same as the databases that are supported in the DB/Cloud timebased data store.

The same requirements regarding the installation of client components apply as for the DB/Cloud data stores:

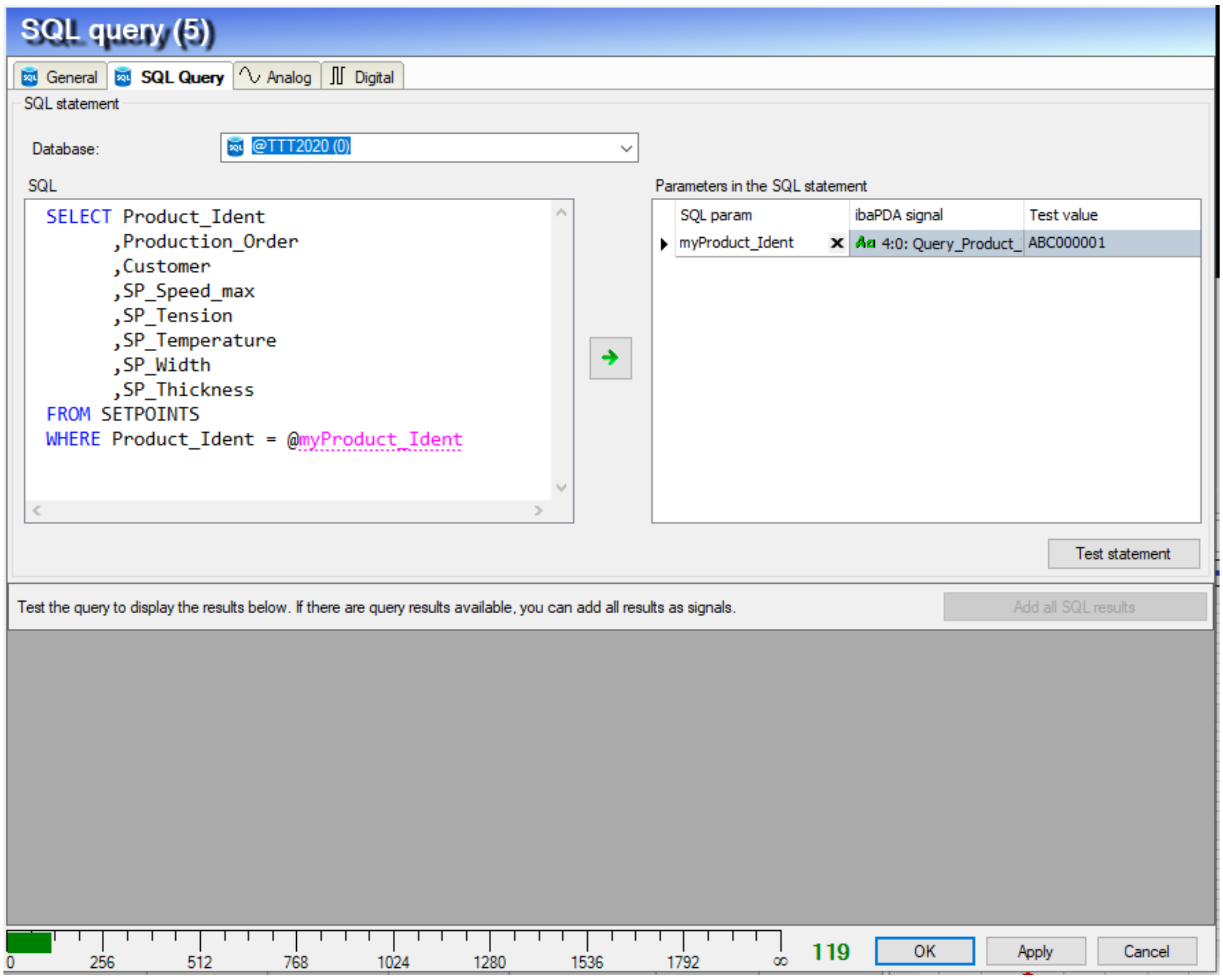
For SAP HANA and Oracle database connections a client installation is required, for SQL Server, MySQL and PostgreSQL no additional installation is necessary. More information about how to install and configure the Oracle client can be found in the separate document

- **Installation instructions - Oracle® Data Provider for .NET (ODP.NET)** (*ibaNotes_Oracle_ODP.NET_Installation_v1.0_en.pdf*)

The SQL interface is visible if at least one database type is licensed. Once a database type is licensed, all the available database types will be visible in the IO manager, but only the databases for which a license is active can be used during acquisition.

## 3.3 SQL modules

There are input modules and output modules. Each module is either an SQL query or an SQL command. There are no modules that are both at the same time. Therefore, you will never see a module both in the inputs and the outputs in I/O Manager.



Database: @TTT2020 (0)

SQL

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

Parameters in the SQL statement

| SQL param | ibaPDA signal | Test value |
|---|---|---|
| myProduct_Ident | 4:0: Query_Product_ | ABC000001 |

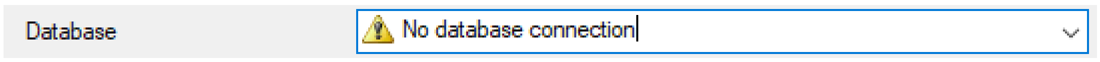
Test the query to display the results below. If there are query results available, you can add all results as signals.

Each SQL module has

- a reference to a database connection in the interface,
- a SQL statement,
- a mapping assignment for parameters used in the SQL statement and
- a trigger mode.

The reference to the database connection is stored as a number that refers to the database connection ID in the interface. If a module becomes unmapped, that number is no longer linked to the SQL interface and is therefore lost. In some cases, there is no database connection selected:

- When the SQL module is new and no database connection is selected yet.
- When the SQL module is unmapped.
- When the connection the SQL module referred to is removed.


Database
No database connection

### 3.3.1 The SQL statement

The SQL statement will in most cases be a SELECT statement for SQL queries and an INSERT, UPDATE or DELETE statement for SQL command modules. The statement can be entered on the left hand side of the module control. Make sure to add statements that can be executed by the database server.

The syntax of the SQL statement must be correct. It must be written in the SQL dialect of the targeted database type. Some help is provided in ibaPDA by coloring the SQL syntax. Keywords are displayed in blue, comments in green, and possible SQL parameters are also shown in a color, like for example “mycounter” in the example above.

The syntax highlighting differs per database type, because each SQL dialect can have different keywords and different ways to use SQL parameters (e.g. in Oracle a colon “:” is used to identify a parameter while in SQL Server an ampersand “@” is used).

One statement is allowed per module. Note that for some database types, a semicolon can be part of a statement while for Oracle, for example, a statement never ends with a semicolon. Make sure that you use the correct SQL dialect for the database that you want to access.

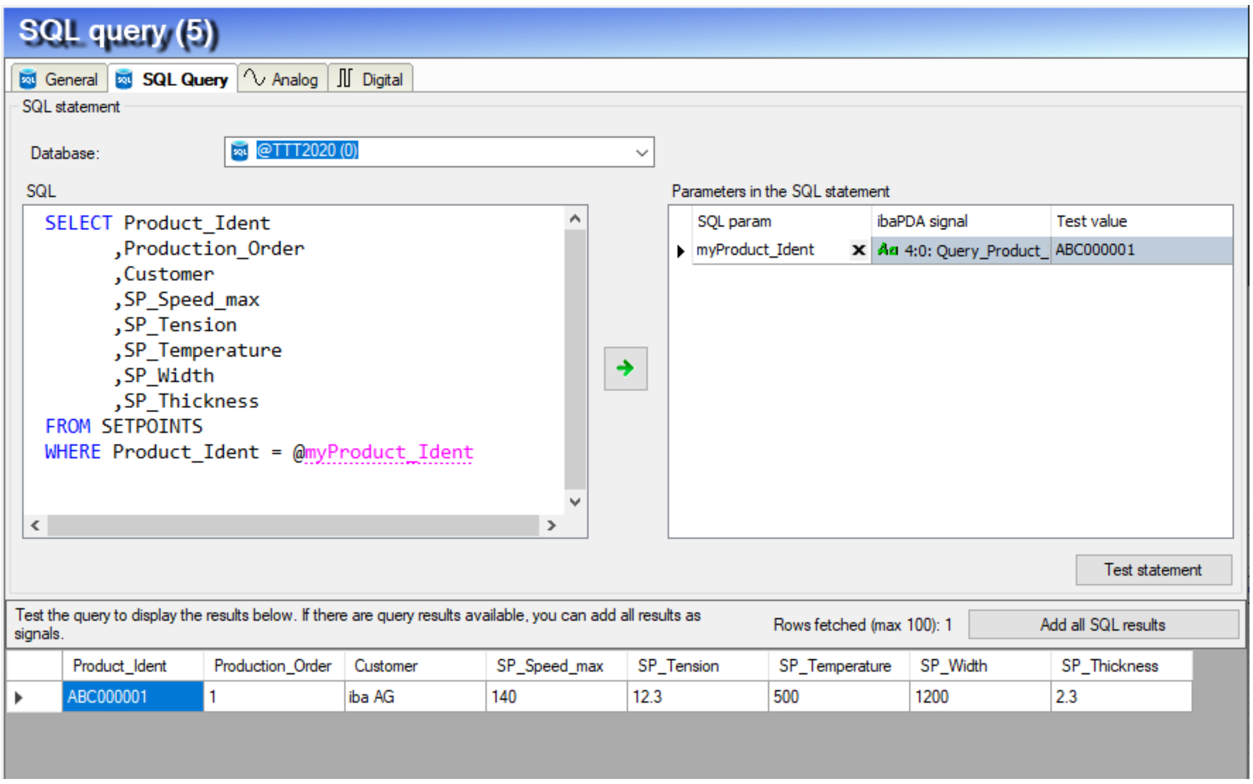
### 3.3.2 Test statements without parameters

It is possible to test statements in ibaPDA. If you test a statement, you can see to a certain extent whether it would yield the desired results. Statements that are tested are executed in a transaction that is rolled back after the execution. That means that no actual change should be made to the data in the database. However, it is impossible to filter out all possible side effects, since ibaPDA doesn’t fully control the database instance it accesses. Under some circumstances, there can be traces of the statements that were tested, even though they were rolled back.

#### 3.3.2.1 Test statements in SQL queries

SQL query modules are displayed differently than SQL command modules. The query modules have an additional area at the bottom of the control where query results can be displayed. Query results can become available after clicking <Test statement>.

A maximum of 100 rows will be fetched when testing a query, to avoid unnecessary memory consumption.



```
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 | 4:0: Query_Product_ | ABC000001 |

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 |
|---|---|---|---|---|---|---|---|
| ABC000001 | 1 | iba AG | 140 | 12.3 | 500 | 1200 | 2.3 |

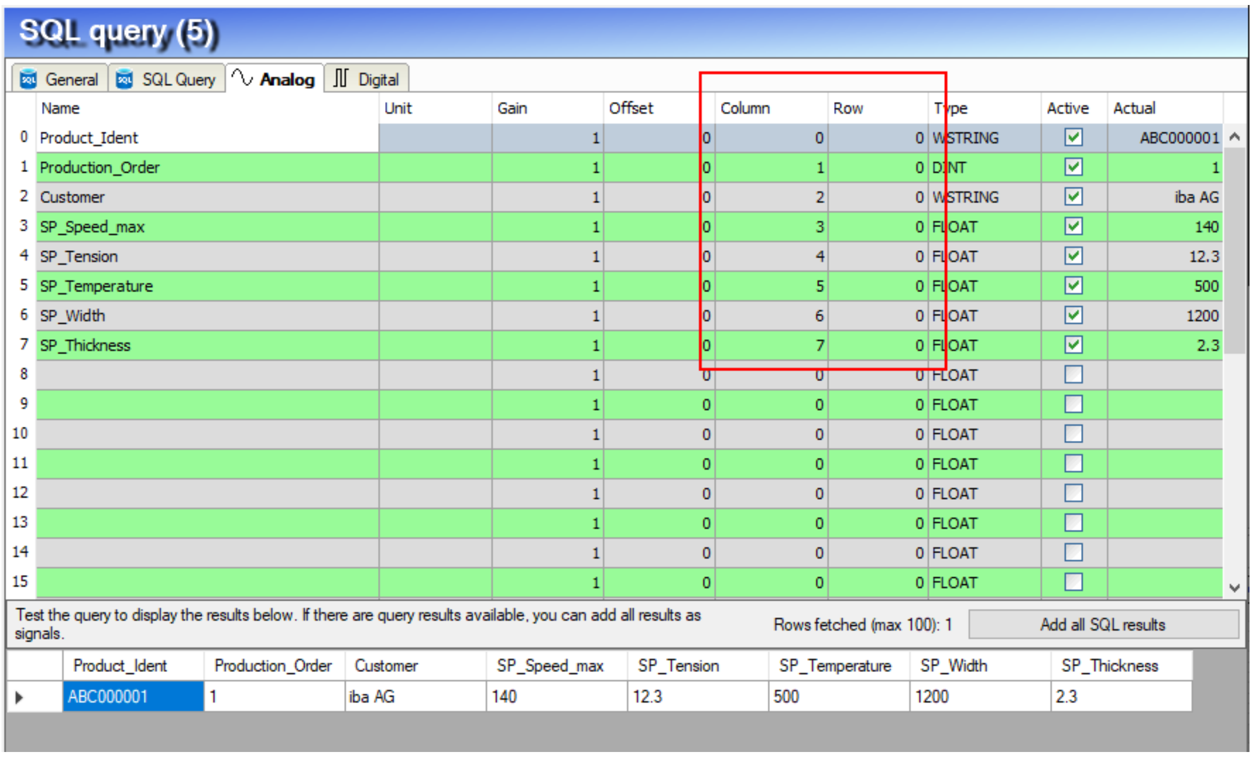
### 3.3.2.2 Test statements in SQL commands

SQL commands can also be tested. The only feedback you get is whether the execution of the statement was successful.

Make sure that a statement can be executed multiple times without rollback, to avoid errors during acquisition that were not yet encountered during testing.

## 3.3.3 Add input signals in SQL query modules

Unlike SQL commands, SQL queries have signals attached. The signals can be viewed in the Analog and Digital tabs of an SQL query module. If a result set is retrieved, all the results displayed at the bottom of the control can be added as input signals by clicking the button <Add all SQL results>. The signals will then be added to the analog or digital signals. If not enough signals are available, the user will be asked whether to extend the number of signals. Added analog and digital signals are referenced within the result set of the query by the column and row parameters which are automatically filled when clicking the button <Add all SQL results>. It is also possible to manually configure signals by configuring the correct column and row parameters according to the expected result set.



SQL query (5)

General | SQL Query | Analog | Digital

| | Name | Unit | Gain | Offset | Column | Row | Type | Active | Actual |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Product_Ident | | 1 | 0 | 0 | 0 | WSTRING | ✓ | ABC000001 |
| 1 | Production_Order | | 1 | 0 | 1 | 0 | DINT | ✓ | 1 |
| 2 | Customer | | 1 | 0 | 2 | 0 | WSTRING | ✓ | iba AG |
| 3 | SP_Speed_max | | 1 | 0 | 3 | 0 | FLOAT | ✓ | 140 |
| 4 | SP_Tension | | 1 | 0 | 4 | 0 | FLOAT | ✓ | 12.3 |
| 5 | SP_Temperature | | 1 | 0 | 5 | 0 | FLOAT | ✓ | 500 |
| 6 | SP_Width | | 1 | 0 | 6 | 0 | FLOAT | ✓ | 1200 |
| 7 | SP_Thickness | | 1 | 0 | 7 | 0 | FLOAT | ✓ | 2.3 |
| 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 | ☐ | |
| 15 | | | 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

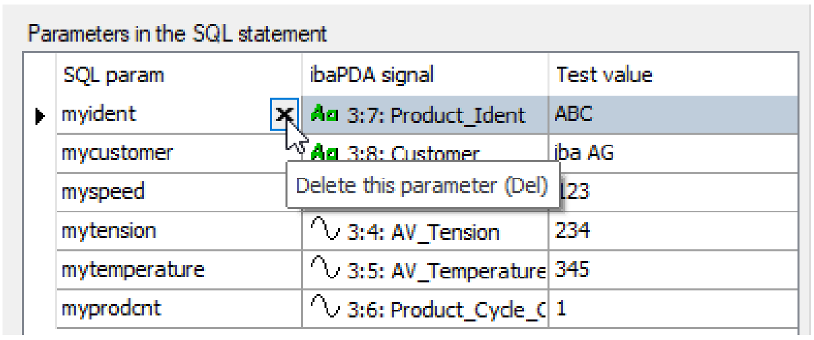
| Product_Ident | Production_Order | Customer | SP_Speed_max | SP_Tension | SP_Temperature | SP_Width | SP_Thickness |
|---|---|---|---|---|---|---|---|
| ABC000001 | 1 | iba AG | 140 | 12.3 | 500 | 1200 | 2.3 |

### 3.3.4 Add parameters

Input parameters can be used in both SQL queries and SQL commands. To make use of a parameter, the proceed as follows.

- Write a SQL statement with parameters (starting with “@” for SQL Server, PostgreSQL and MySQL and starting with “:” for Oracle and SAP HANA). Note that parameters can only replace a value, not other parts of a statement. For example, it is not easy to use a parameter to replace the name of a table.
- Click on the button with the green arrow to find and add all SQL parameters in the statement to the parameter mapping list on the right side. If there are already parameters in the right hand side of the SQL module control, you can be asked if you want to delete surplus parameters.

Parameters can be deleted individually. To do that, select a parameter and click the cross that appears next to the parameter name.



Parameters in the SQL statement

| SQL param | ibaPDA signal | Test value |
|---|---|---|
| myident | 3:7: Product_Ident | ABC |
| mycustomer | 3:8: Customer | iba AG |
| myspeed | | 123 |
| mytension | 3:4: AV_Tension | 234 |
| mytemperature | 3:5: AV_Temperature | 345 |
| myprodcnt | 3:6: Product_Cycle_C | 1 |

Delete this parameter (Del)

Parameters have a name, an assigned ibaPDA signal and a test value. The name is found in the SQL statement and added to the parameter table by clicking on the green arrow button.

The ibaPDA signal is the signal whose value is used when executing the SQL statements. The SQL parameter is replaced with the value from the ibaPDA signal during acquisition. All common types of signals can be used as source for parameter values: analog, digital and text signals.

To enable executing a statement during testing or validation, a test value can be used. The test value is entered in the third column as plain text. The type of the ibaPDA signal is used to derive how the text in the "Test value" column should be interpreted.

Digital signals will always have the value 1 or 0. Although for example PostgreSQL has a special boolean type which is not numeric, in ibaPDA digital signals will still always have value 1 or 0. To enforce using boolean values in this case, the SQL statement can be changed from for example using

```
@param
```

to

```
(@param <> 0)
```

Both queries and commands can also be executed without parameters. Just use a parameterless statement in order to do that.

#### 3.3.4.1 Parameter use during testing and acquisition

When testing a statement, you can keep more parameters than the ones that will be used to test the statement. When testing the statement, you will get a warning that some parameters were not found in the statement. Those surplus parameters can be kept in the parameter table during testing of the configuration, but before starting the acquisition, they must be removed. During the test of the statement, the parameters will be ignored, but during acquisition they will actually be used, which can result in errors.

### 3.3.5 Send mode and update time

The ibaPDA user can control when and how frequent SQL statements will be executed. For that purpose, there are three send modes which can be configured in the "General" tab of an SQL module:

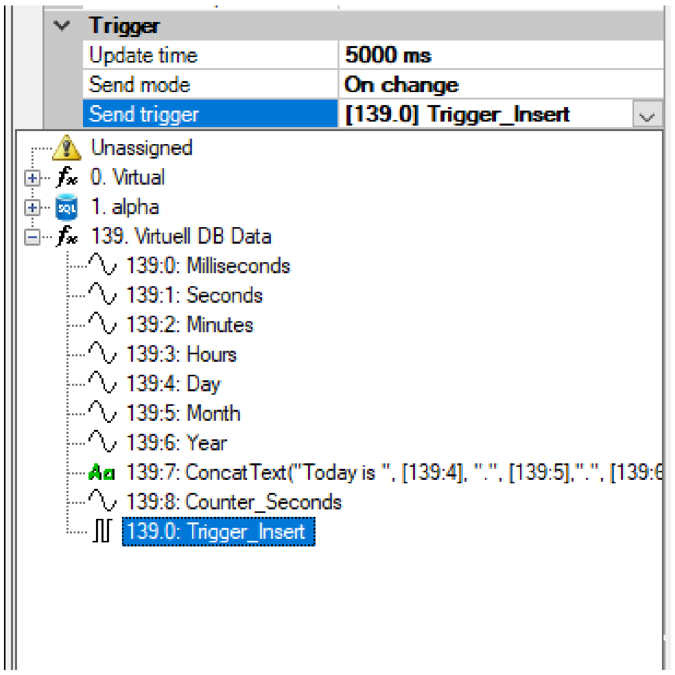
- Cyclic
- On change
- On trigger

#### 3.3.5.1 Cyclic send mode

Cyclic send mode is the simplest use case. In this send mode, a statement is executed periodically. The update time is the period. Every time when the update time has expired, a new statement execution will be performed.

#### 3.3.5.2 On change

In the on change send mode, any analog or digital signal can be chosen to serve as trigger signal. The update time now serves as deadtime. No new statement will be executed before the deadtime has expired.

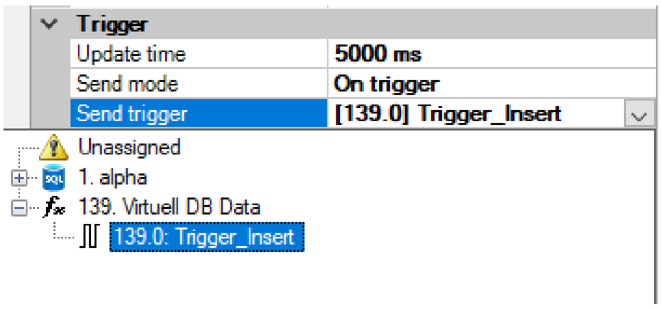


| Trigger | |
|---|---|
| Update time | 5000 ms |
| Send mode | On change |
| Send trigger | [139.0] Trigger_Insert |

```
Unassigned
0. Virtual
1. alpha
139. Virtuell DB Data
    139:0: Milliseconds
    139:1: Seconds
    139:2: Minutes
    139:3: Hours
    139:4: Day
    139:5: Month
    139:6: Year
    139:7: ConcatText("Today is ", [139:4], ".", [139:5],".", [139:6
    139:8: Counter_Seconds
    139.0: Trigger_Insert
```

### 3.3.5.3 On trigger

In this send mode, a digital signal must be selected as trigger signal. A statement is sent each time a rising edge is detected on the trigger signal. The update time again serves as a dead time: New triggers are ignored until the update time has elapsed.



| Trigger | |
|---|---|
| Update time | 5000 ms |
| Send mode | On trigger |
| Send trigger | [139.0] Trigger_Insert |

```
Unassigned
1. alpha
139. Virtuell DB Data
    139.0: Trigger_Insert
```

### 3.3.6 Input signals

Input signals for the SQL interface are only available for SQL query modules and have two additional properties that not all signals have:

- Row
- Column

These properties refer to the row and column in the result set of the SQL query. The value found in the result set becomes the value of the corresponding input signal.

Input signals also have a data type, just like all other signals. One specialty for the data types is that strings can be either fixed size or variable size in ibaPDA.

There are no output signals for the SQL interface. In order to output data from ibaPDA to the database the mapping of signals into parameters in the statement as described above has to be used.

#### 3.3.6.1 Actual values

During acquisition the actual values of the signals are displayed in the rightmost column of the analog and digital signal tabs.

When a statement is tested, the actual values are no longer updated with the new values fetched by the PDA server. Instead, the new values are filled in when the <Add all SQL results> button is clicked.

### 3.3.7 Diagnose module

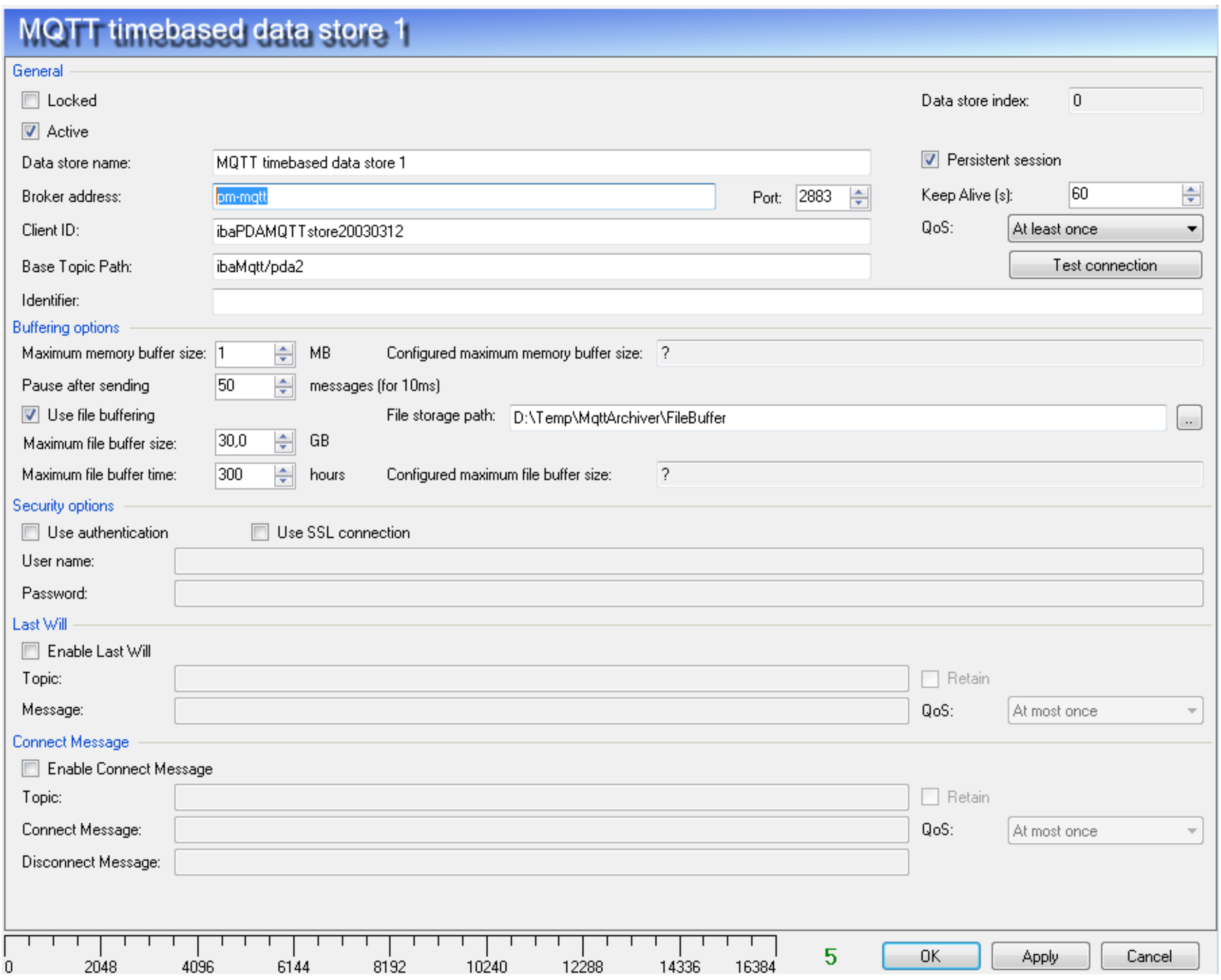
A diagnose module can be added. Both input and output modules can be selected as the target module for the diagnose module.

SQL statements can sometimes take some time to execute. The required time can depend on external factors determining the responsiveness of the database. Therefore, when a statement fails or a database is temporarily not available, it may take some time (almost always less than a minute) before a statement execution is recognized as failed. The number of successful and failed statements may therefore be difficult to act upon. It may seem that there are only few failed attempts, while the database is just not connected.

## 4 MQTT timebased data store with longtime file buffer

The MQTT timebased data store is extended to use a file-based buffering, additionally to the standard in-memory buffering. This feature allows ibaPDA to keep all samples in case of a longer connection loss to the MQTT broker.

Keeping all data only works if connection from ibaPDA to the MQTT broker and the connection from the MQTT consumer client to the broker are set to use a persistant session, with QoS 1 or QoS 2. If the sessions are not marked as persistant, the client and the broker will remove all values that are not sent on connection loss. And messages with QoS 0 are note buffered at all, so if the broker gets multiple messages to one topic before sending out data to the consumer clients, only the newest message is sent out at all. As MQTT messages themselves contain no time information, it is recommended to use the JSON format and add the timestamp metadata, so the consuming client can retrieve that information together with the value.


MQTT timebased data store 1

General
Locked: ☐
[✓ Active]
Data store index: 0
Data store name: MQTT timebased data store 1
[✓ Persistent session]
Broker address: pm-mqtt
Port: 2883
Keep Alive (s): 60
Client ID: ibaPDAMQTTstore20030312
QoS: At least once
Base Topic Path: ibaMqtt/pda2
Test connection
Identifier:

Buffering options
Maximum memory buffer size: 1 MB
Configured maximum memory buffer size: ?
Pause after sending: 50 messages (for 10ms)
[✓ Use file buffering]
File storage path: D:\Temp\MqttArchiver\FileBuffer
Maximum file buffer size: 30,0 GB
Maximum file buffer time: 300 hours
Configured maximum file buffer size: ?

Security options
Use authentication: ☐
Use SSL connection: ☐
User name:
Password:

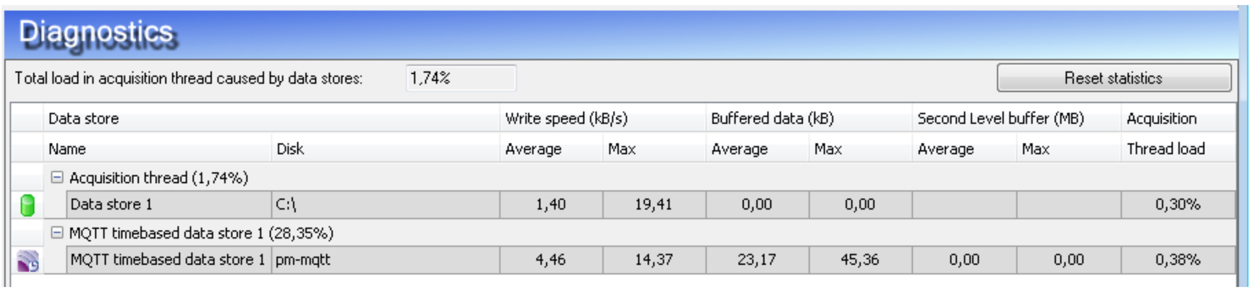
Last Will
Enable Last Will: ☐
Topic:
Retain: ☐
Message:
QoS: At most once

Connect Message
Enable Connect Message: ☐
Topic:
Retain: ☐
Connect Message:
QoS: At most once
Disconnect Message:

The buffering options are extended for this.

- **Pause after sending**: A new general option is to add a pause when sending data to the broker, as many brokers seem to have problems with a continuous stream of messages. To avoid this, ibaPDA can pause sending if a certain number of messages is sent to the broker without interrupt. Each topic is handled as a separate message, so the number set here should take that into respect. Set the value to 0 to disable the feature.

- **Use file buffering**: Set this option to activate file based buffering. When activating this, the **Maximum memory buffer size** can be reduced to 1MB instead of the standard 4MB
- **File storage path**: The path used for storing the files. This should be on a local physical hard disc, to avoid data loss in case of a power outage
- **Maximum file buffer size**: The maximal hard disc space in GB the buffering files are allowed to use. If that size is exceeded, the oldest data is removed from storage.
- **Maximum file buffer time**: The maximal age of the buffered data in hours. Data older than this is removed from storage.
- **Configured maximum file buffer size**: As with the memory buffer, an estimate of the time the values can be buffered according to the message size and the available disc space is shown when ibaPDA is online.



Diagnostics

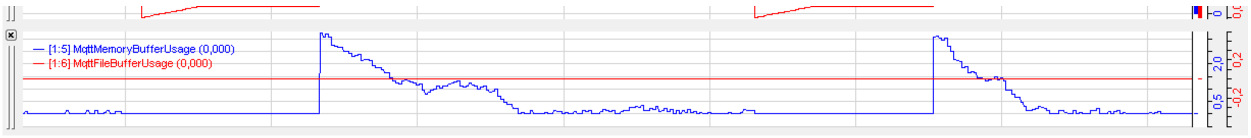
Total load in acquisition thread caused by data stores: 1,74%

Reset statistics

| Data store | | Write speed (kB/s) | | Buffered data (kB) | | Second Level buffer (MB) | | Acquisition |
|---|---|---|---|---|---|---|---|---|
| Name | Disk | Average | Max | Average | Max | Average | Max | Thread load |
| ⊟ Acquisition thread (1,74%) | | | | | | | | |
| Data store 1 | C:\ | 1,40 | 19,41 | 0,00 | 0,00 | | | 0,30% |
| ⊟ MQTT timebased data store 1 (28,35%) | | | | | | | | |
| MQTT timebased data store 1 | pm-mqtt | 4,46 | 14,37 | 23,17 | 45,36 | 0,00 | 0,00 | 0,38% |

The diagnostic window is extended to show an additional Second Level buffer, which reflects the usage of the file buffering in MB.

If a pretrigger is used, the Buffered data might exceed the limitations set for the memory buffering, as the data shown here includes both the pretrigger data and the memory buffer.



The virtual function DataStoreInfoMQTT()for the MQTT data store is extended to show the file buffer usage in % when using InfoType=6.

When data is collected by ibaPDA, it is stored in the memory buffer. If a connection to the configured broker exists, this data is immediately sent to the broker. If the connection to the broker is lost, or the data cannot be sent out fast enough, the memory buffer starts to grow. If it grows beyond the configured size, the data is removed from the memory buffer and (if activated) stored in the file buffer. When the file buffer grows beyond the configured size, or the stored values are older than the configured maximum time, the oldest values are removed.

When sending data, always the oldest data is sent out first. So when a connection is reestablished after some time, ibaPDA starts sending the oldest values, while the newer values are still stored in the buffer.

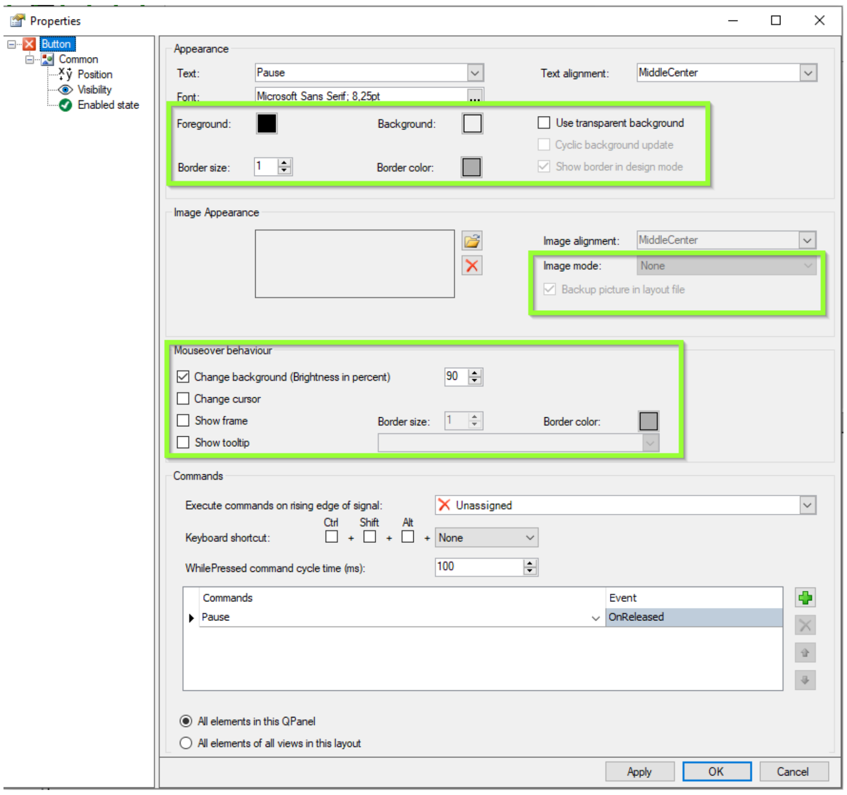
When the acquisition is stopped, if a connection to a broker exists and the file buffering is not currently used, ibaPDA tries to send out the remaining data to the broker. If this takes more than 30 seconds, the process is stopped. If file buffering is currently used, this step is skipped, as sending out all data would take too long anyway. If file buffering is activated, all remaining values are stored in the file buffer.

When the acquisition is started and there are still files left from the previous session, these values are put into the queue again and sent to the broker. If the configuration was changed, the stored values are still sent in the format defined by the configuration they were recorded with. If this is not desired, the user has to manually remove the files from the storage path before starting the acquisition.

# 5 ibaQPanel

## 5.1 Button (advanced settings)

There are new advanced settings for the button in ibaQPanel:


Properties

- Button
  - Common
  - Position
  - Visibility
  - Enabled state

Appearance

Text: Pause
Text alignment: MiddleCenter
Font: Microsoft Sans Serif; 8,25pt
Foreground: ■ Background: □ ☐ Use transparent background
☐ Cyclic background update
Border size: 1 Border color: ▨ ☑ Show border in design mode

Image Appearance

Image alignment: MiddleCenter
Image mode: None
☑ Backup picture in layout file

Mouseover behaviour

☑ Change background (Brightness in percent) 90
☐ Change cursor
☐ Show frame Border size: 1 Border color: ▨
☐ Show tooltip

Commands

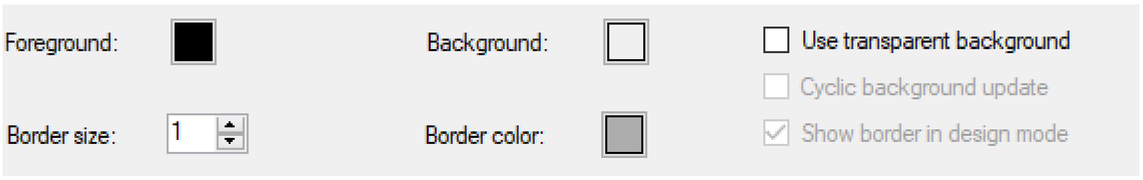
Execute commands on rising edge of signal: Unassigned
Keyboard shortcut: Ctrl ☐ + Shift ☐ + Alt ☐ + None
WhilePressed command cycle time (ms): 100

| Commands | Event |
|---|---|
| Pause | OnReleased |

(● All elements in this QPanel)
○ All elements of all views in this layout

With these settings it is possible to create invisible button areas if there is no picture, no text, border size = 0 and transparent background.

The new settings are:


Foreground: ■ Background: □ ☐ Use transparent background
☐ Cyclic background update
Border size: 1 Border color: ▨ ☑ Show border in design mode

Now the **foreground color, background color, the border color**, and the **border size** can be set.

In addition, a **transparent background** is possible. If you enable the "Use transparent backgroud" option two further options become available:

Show border in design mode: If the button has no border and is transparent a border will be shown in the design mode to indicate that there is a button area.

Cyclic background update: If there is a moving/changing element behind the transparent button, it will be also shown. For that, the cyclic update is necessary. If there is a moving/changing element in the background and this option is disabled, then the background will be frozen at the time when applying the settings.

**Remarks: Use this option rarely, because it needs a lot of calculation power.**

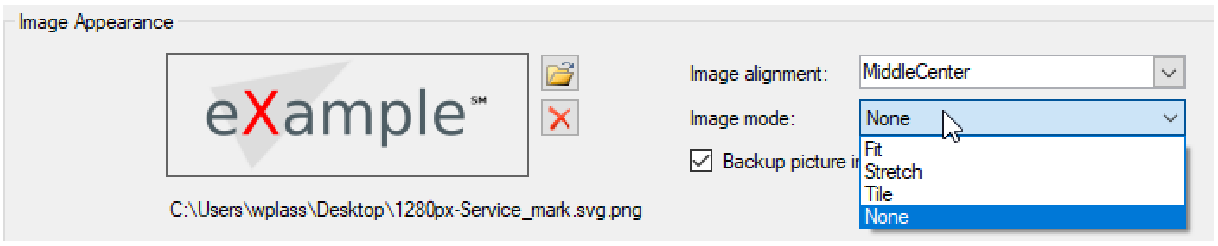


Image Appearance

C:\Users\wplass\Desktop\1280px-Service_mark.svg.png

Image alignment: MiddleCenter

Image mode: None

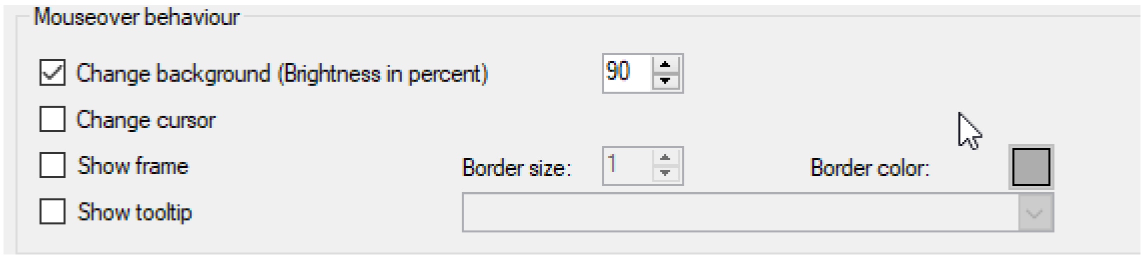
☑ Backup picture i

A selected picture will be shown in the preview pane on the left side with its path below. There is a change in the settings for the text and picture position. The former options like “Picture above text”, “Picture below text” etc. have been replaced by the setting of **Image alignment** and the text alignment in combination.

Furthermore, an image mode can be set.

- **Fit**: Fit the original size with his ratio into the button size
- **Stretch**: Stretch the picture in width and height to the button size
- **Tile**: The original picture will be shown as often as it has place in choosen button size.
- **None**: Original picture size is used

**Backup picture in layout**: The picture is stored in the layout


Mouseover behaviour

☑ Change background (Brightness in percent) 90

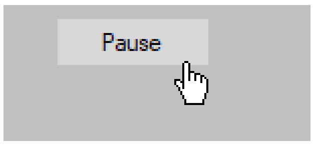
☐ Change cursor

☐ Show frame Border size: 1 Border color:

☐ Show tooltip

Now also **mouseover behavior** can be set:

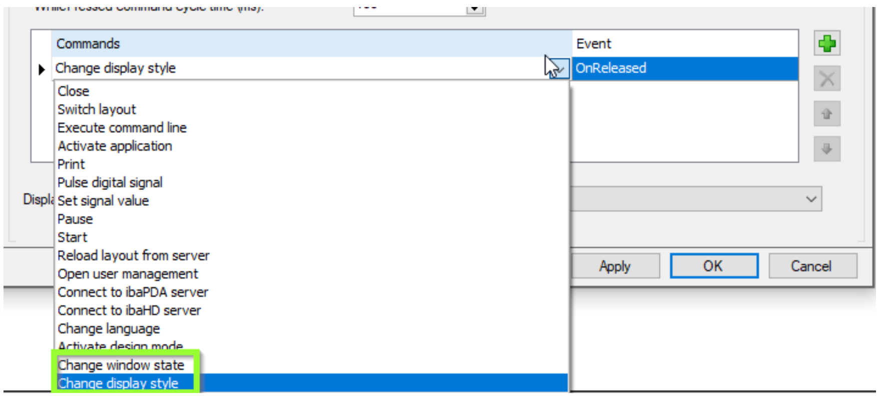
- **Change background (Brightness in percent):** That is default selection because the former button has this standard behavior. With mouseover the brightness of the button dims down to 90 % of the full brightness. A mouse click on the button dims out another 5 %.
- **Change cursor**: The cursor changes from the mouse arrow to a finger-pointing symbol



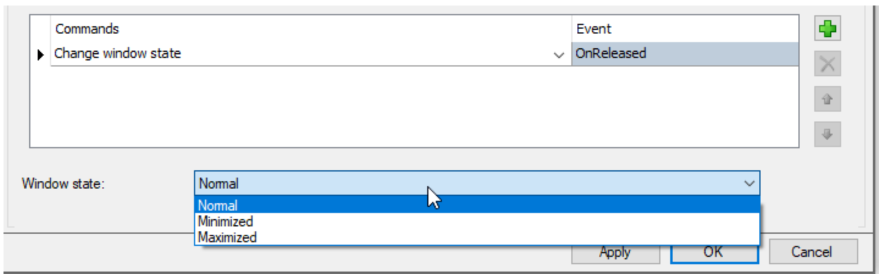
- **Show frame**: A frame can be shown in a certain border size and color.
- Show tooltip:  A fixed text can be shown as a tooltip

## 5.2 New button commands for windows/display style

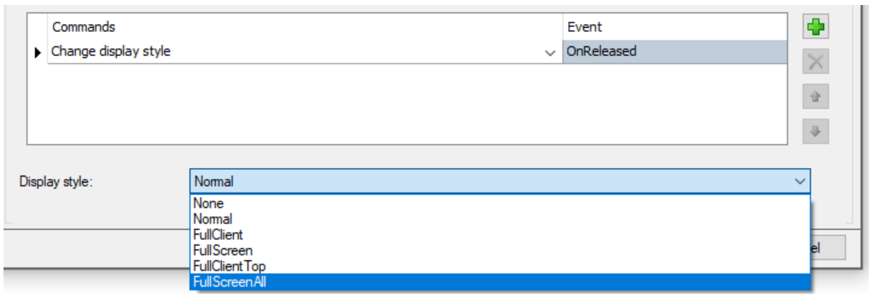
There are new commands in the button available:



- The window state can be switched to normal/minimized/maximized.

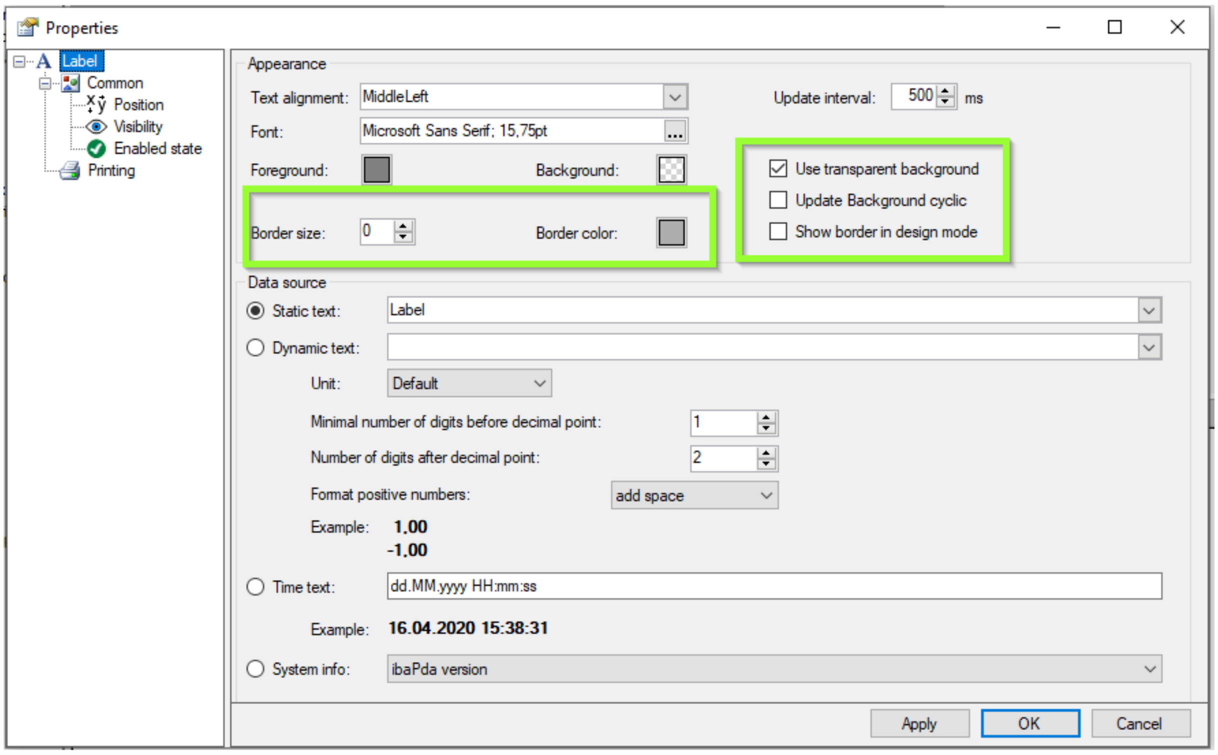


- The display style can be set to different style modes.



## 5.3 Label (advanced settings)

The label has new settings for border and transparency:



Appearance
Text alignment: MiddleLeft
Update interval: 500 ms
Font: Microsoft Sans Serif; 15,75pt
Foreground:
Background:
[✓] Use transparent background
[ ] Update Background cyclic
[ ] Show border in design mode
Border size: 0
Border color:

Data source
(●) Static text: Label
( ) Dynamic text:
Unit: Default
Minimal number of digits before decimal point: 1
Number of digits after decimal point: 2
Format positive numbers: add space
Example: 1,00 -1,00
( ) Time text: dd.MM.yyyy HH:mm:ss
Example: 16.04.2020 15:38:31
( ) System info: ibaPda version

**Boder size** and **border color** and also a **transparent mode** can be set.

**Show border in design mode:** In case of a combination of transparent background and no border, a border will be shown in the design mode to see that there is a button area.

**Update background cyclic**: If there is a moving/changing element behind the transparent button, it will be also shown. For that, the cyclic update is necessary. If there is a moving/changing element in the background and this option is NOT set, then the background in the moment when apply the settings will be frozen.

**Remarks: Use this option rarely, because it needs a lot of calculation power.**

## 5.4 File picker was replaced by file selector

The file picker is now replaced by the file selector.

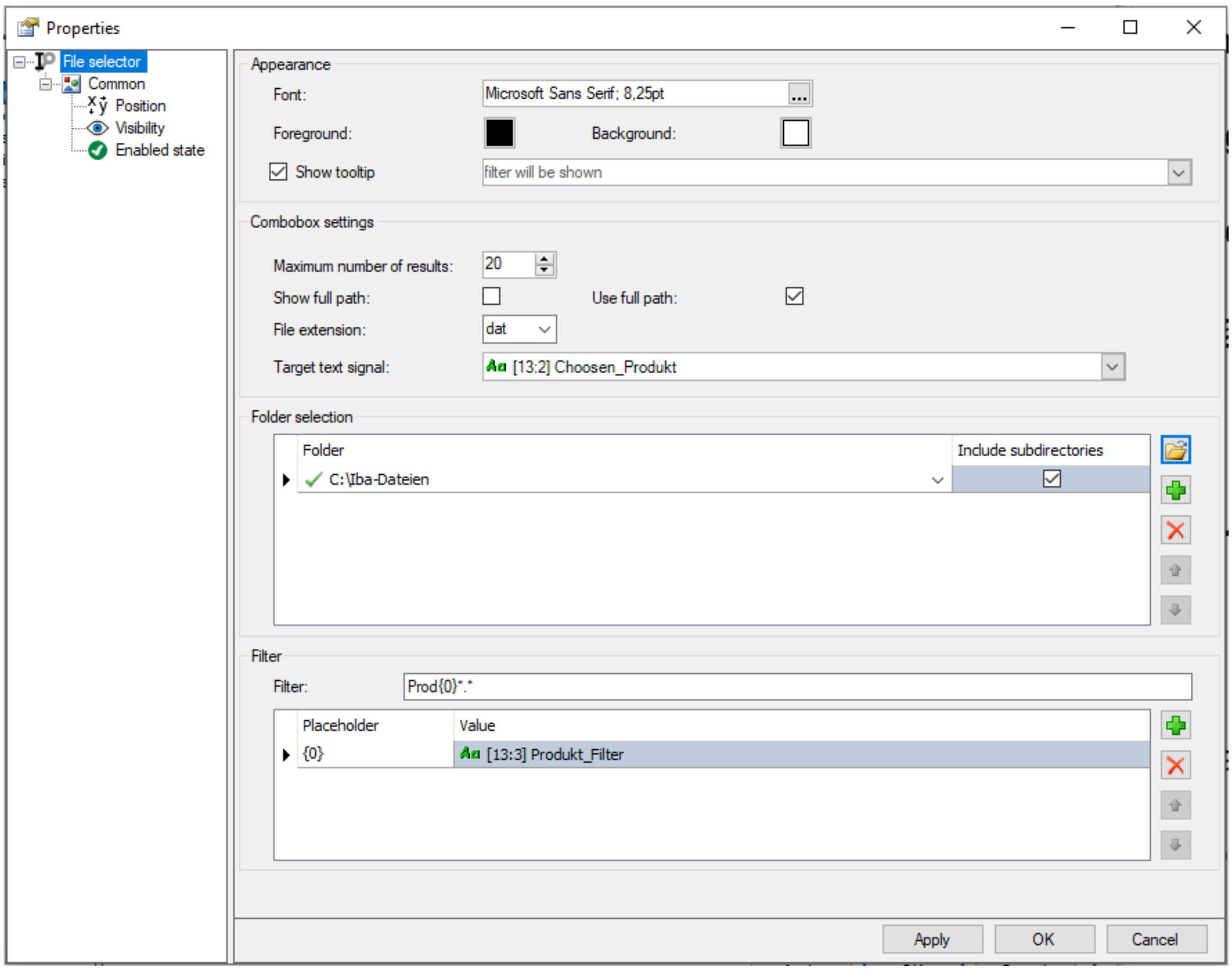
If a file picker is used inside an old layout it will be automatically replaced/converted to the file selector.

The behavior of the file selector is similar but with a lot of more functions.

The general function is that you can choose a file from a folder and the path and filename is written to an ibaQPanel text input signal. That text signal can e.g. now be used in an offline trend graph to show the file.

With the advanced settings of the file selector the user is now able to:

- Select one ore more folders
- Use dynamic folder switching by using text signals for folder names
- Use one and more placeholders for filter the files of the folder selection



Appearance

Font: Microsoft Sans Serif; 8,25pt

Foreground: ■ Background: □

☑ Show tooltip: filter will be shown

Combobox settings

Maximum number of results: 20

Show full path: ☐ Use full path: ☑

File extension: dat

Target text signal: [13:2] Choosen_Produkt

Folder selection

| Folder | Include subdirectories |
|---|---|
| C:\Iba-Dateien | ☑ |

Filter

Filter: Prod{0}*.*

| Placeholder | Value |
|---|---|
| {0} | [13:3] Produkt_Filter |

**Appearance**:

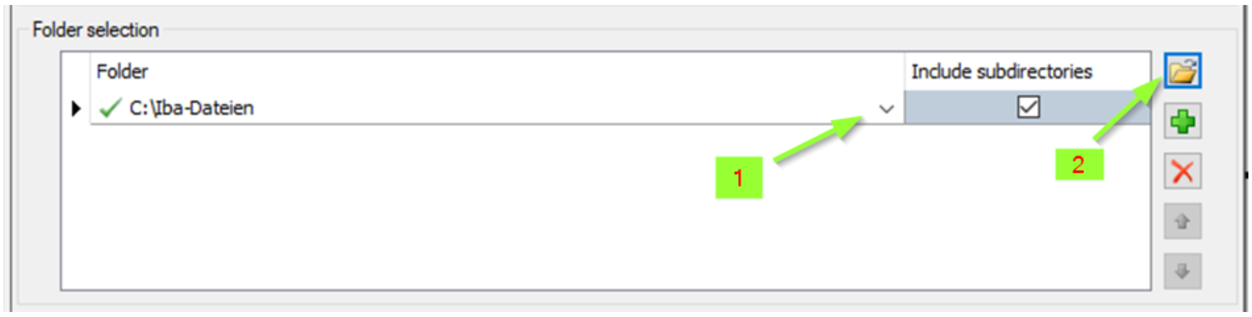
- **Font / Foreground / Background** can be set
- **Show tooltip**: A tooltip can be set which will be shown by mouse over. If there is no tooltip text then the filter criteria will be shown

**Combo box settings**:

- **Maximum numbers of result**: This defines the size of the combo box. This is the visible area of results before scroll bars are shown. The maximum is set to 100.
- **Show full path**: The files are shown with their path in the combo box and not only the file name
- **Use full path**: The target text signal will also have the path and not only the file name.
- **File extension**: all / dat / txt / csv The user can filter by these file extension.

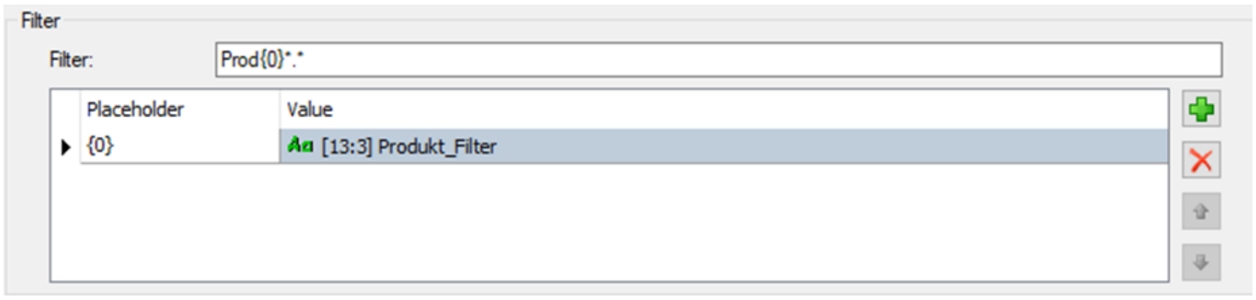
  Remark: With ALL and some filter criteria the user is able to filter also to other file extensions.
- **Target signal**: The filename selected by the user will be written to this ibaQPanel text input signal.

**Folder selection**:



- : The user can add folders
- 1: The user can select a text signal for a dynamic folder
- 2: The user can select a static folder for the selected line
- The user can delete or move an entry

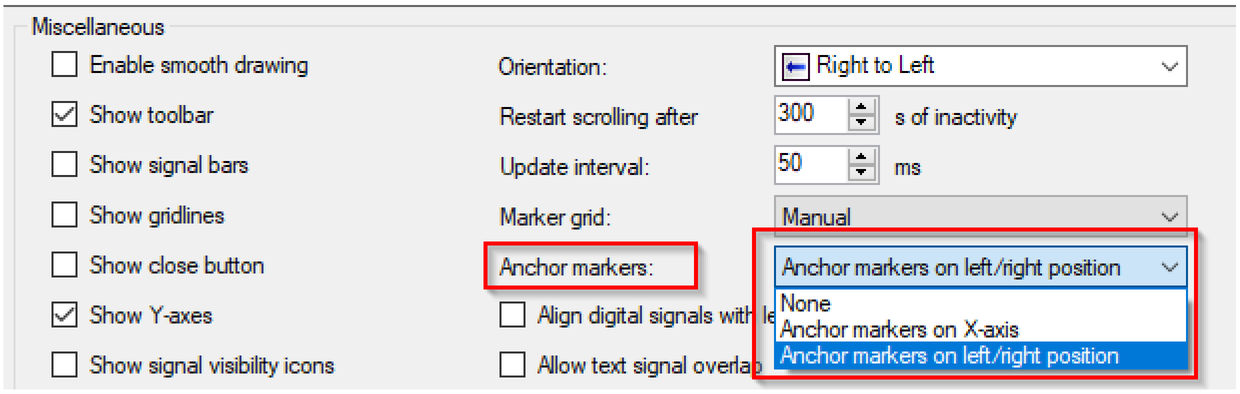
**Filter selection**:



- **Filter**: static and dynamic parts of the filter can be combined. Every dynamic placeholder from the placeholder list can be used. : The user can add filters and use a text signal for it
- The user can delete or move an entry

## 5.5 New marker setting in trend graphs

The trend graphs (trend graph, HD trend graph, Offline trend graph) have a new marker setting.



Miscellaneous

- ☐ Enable smooth drawing
- ☑ Show toolbar
- ☐ Show signal bars
- ☐ Show gridlines
- ☐ Show close button
- ☑ Show Y-axes
- ☐ Show signal visibility icons
- ☐ Align digital signals with le
- ☐ Allow text signal overlap

Orientation: Right to Left
Restart scrolling after 300 s of inactivity
Update interval: 50 ms
Marker grid: Manual
Anchor markers: Anchor markers on left/right position

**Anchor markers on left/right position**

The marker is always set to the left (x1) and right (x2) position when there is an interaction like PAUSE / ZOOM etc.

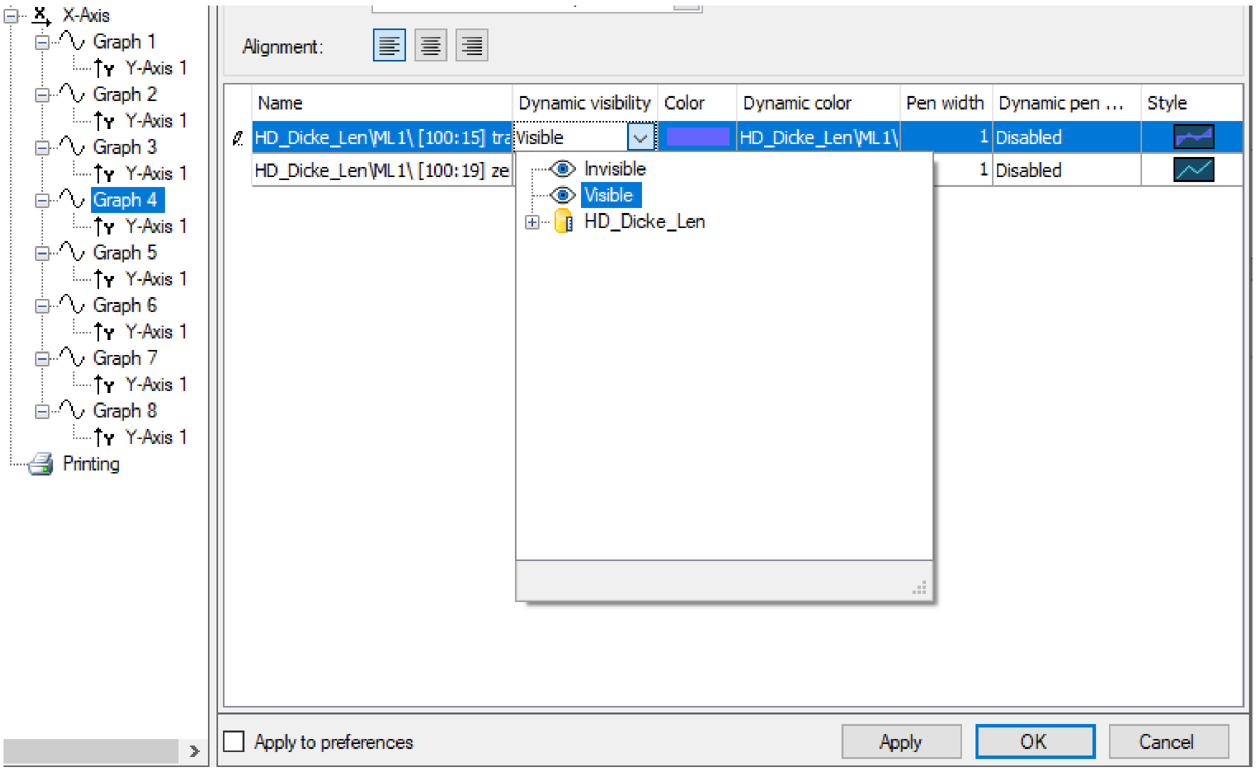
This enables the user in addition with the marker grids to see the min/max/avg values from the actual area.

The markers can still be moved but the next zoom action will set them back to the end positions.

## 5.6 New dynamic signal visibility in trend graphs

The trend graphs (trend graph, HD trend graph, Offline trend graph) have a new option to set the dynamic visibility of a signal.

It is similar to the manual visibility by the monitor icon.

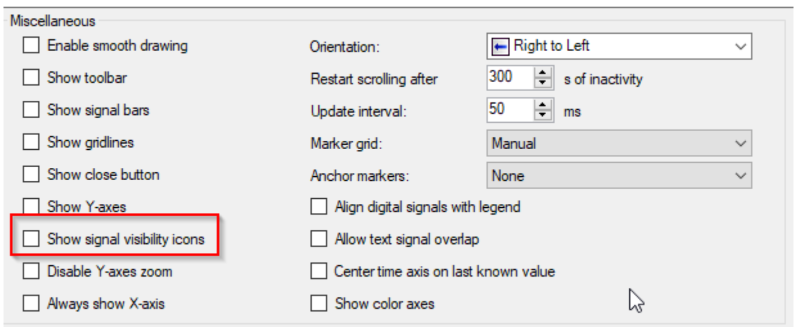


| Name | Dynamic visibility | Color | Dynamic color | Pen width | Dynamic pen ... | Style |
|---|---|---|---|---|---|---|
| HD_Dicke_Len\ML1\ [100:15] tra | Visible | 🟦 | HD_Dicke_Len\ML1\ | 1 | Disabled | |
| HD_Dicke_Len\ML1\ [100:19] ze | | | | 1 | Disabled | |

With this option the user can make signals visible or invisible by a signal.

E.g. uses checkboxes to show certain signals in a pool of signals of a trend graph

It is similar to the manual visibility by the monitor icon.



Miscellaneous

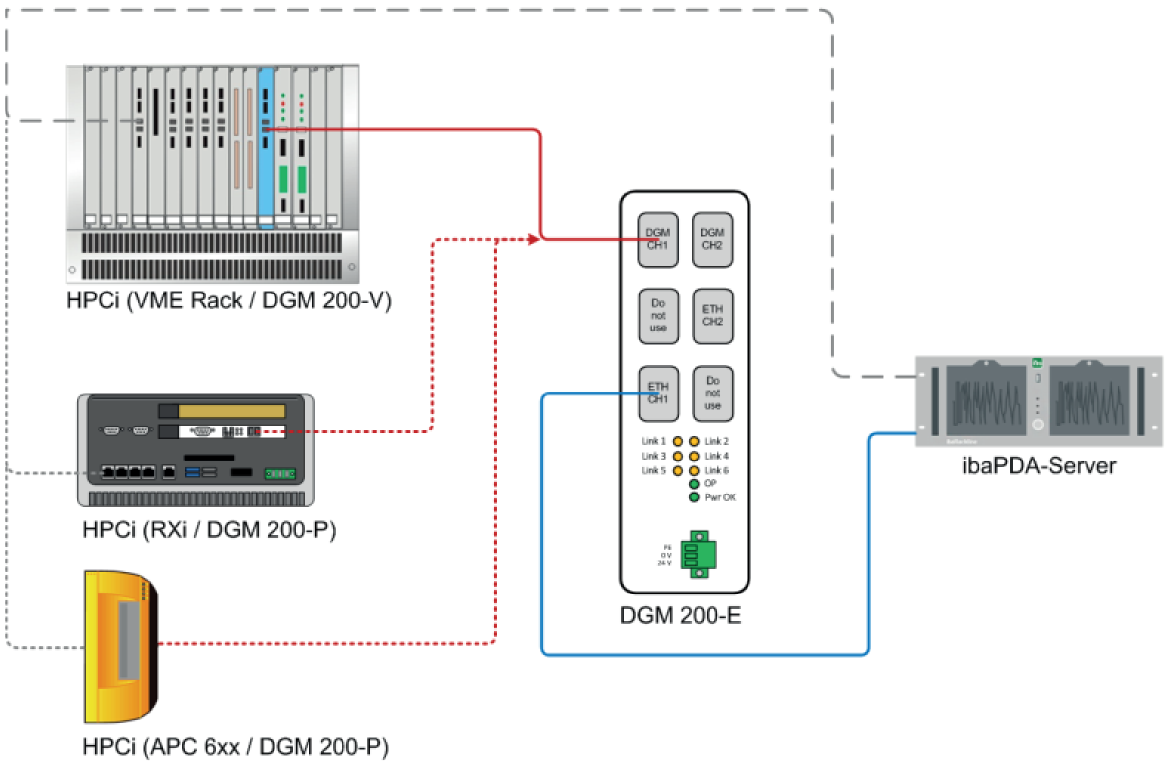
- ☐ Enable smooth drawing
- ☐ Show toolbar
- ☐ Show signal bars
- ☐ Show gridlines
- ☐ Show close button
- ☐ Show Y-axes
- ☐ Show signal visibility icons
- ☐ Disable Y-axes zoom
- ☐ Always show X-axis

Orientation: Right to Left
Restart scrolling after 300 s of inactivity
Update interval: 50 ms
Marker grid: Manual
Anchor markers: None

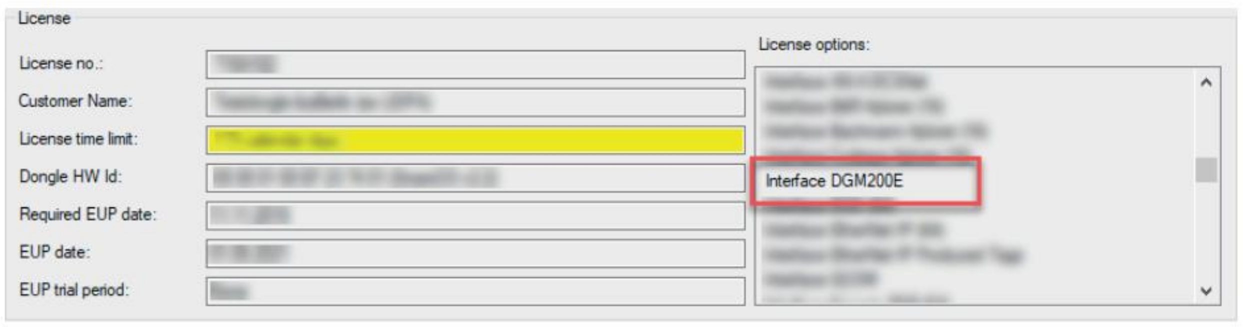
- ☐ Align digital signals with legend
- ☐ Allow text signal overlap
- ☐ Center time axis on last known value
- ☐ Show color axes

# 6 DGM200E

The DGM200E interface allows for measuring data from a CC100/DGM200 communication network of the HPCi automation system by GE Power Conversion.



As illustrated in the above schema, the DGM200-E hardware is an external gateway that converts data from the CC100/DGM200 network into Ethernet-compatible packets which are sent to a standard network interface in the ibaPDA server. The DGM200-E converter serves as the successor of the DGM200-P PCI board.

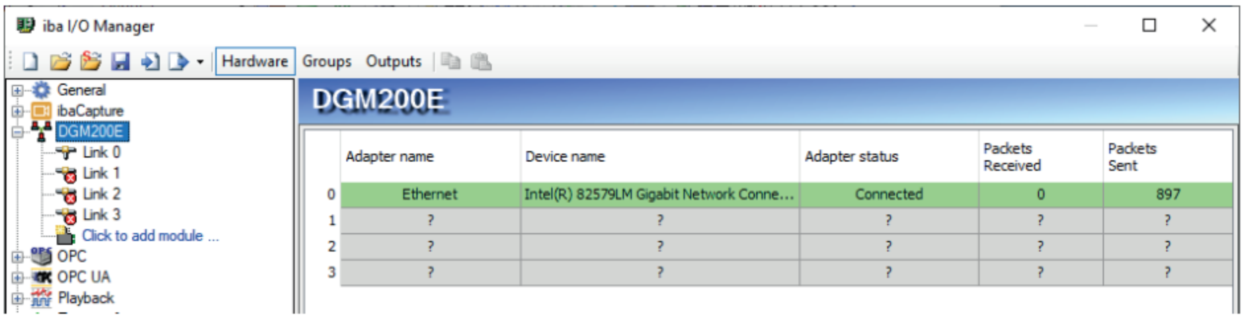

License

| | |
|---|---|
| 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 DGM200E
[redacted]

If an ibaPDA-Interface-DGM200E license is available in the ibaPDA server it should be displayed in the *Settings* tab of the *General* node in the I/O manager.

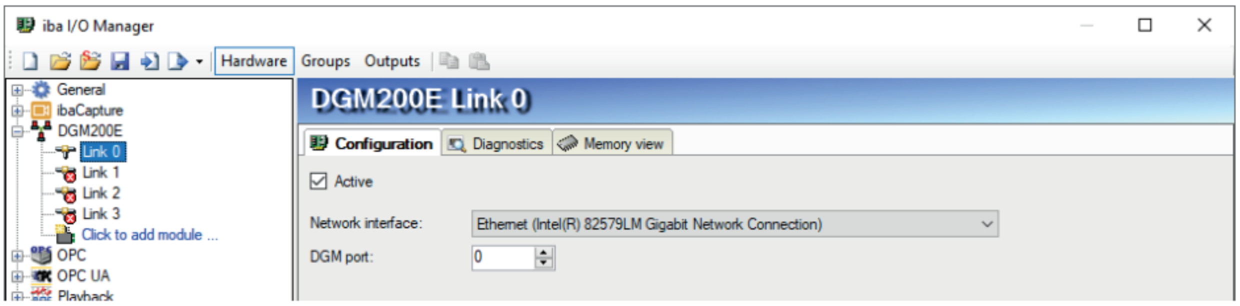


| | Adapter name | Device name | Adapter status | Packets Received | Packets Sent |
|---|---|---|---|---|---|
| 0 | Ethernet | Intel(R) 82579LM Gigabit Network Conne... | Connected | 0 | 897 |
| 1 | ? | ? | ? | ? | ? |
| 2 | ? | ? | ? | ? | ? |
| 3 | ? | ? | ? | ? | ? |

Apart from that, an interface node called **DGM200E** will be available in the I/O manager with 4 links. Each link corresponds to a different network interface on the ibaPDA server so data can be measured from up to 4 different CC100/DGM200 networks.

When selecting the main interface node a grid is shown with 4 rows (corresponding to the 4 links). The following columns are available:

- **Adapter name**: the name of the network interface as displayed e.g. in the Network Connections dialog of the Windows Control Panel.
- **Device name**: the name of the underlying device of the network interface.
- **Adapter status**: a general status of the network interface. Indicates a.o. whether a cable is connected or not. When a cable is not connected the corresponding row in the diagnostic grid will have a red back color indicating an error.
- **Packets Received**: the total number of DGM200E packets received by ibaPDA.
- **Packets Sent**: the total number of DGM200E packets sent by ibaPDA.



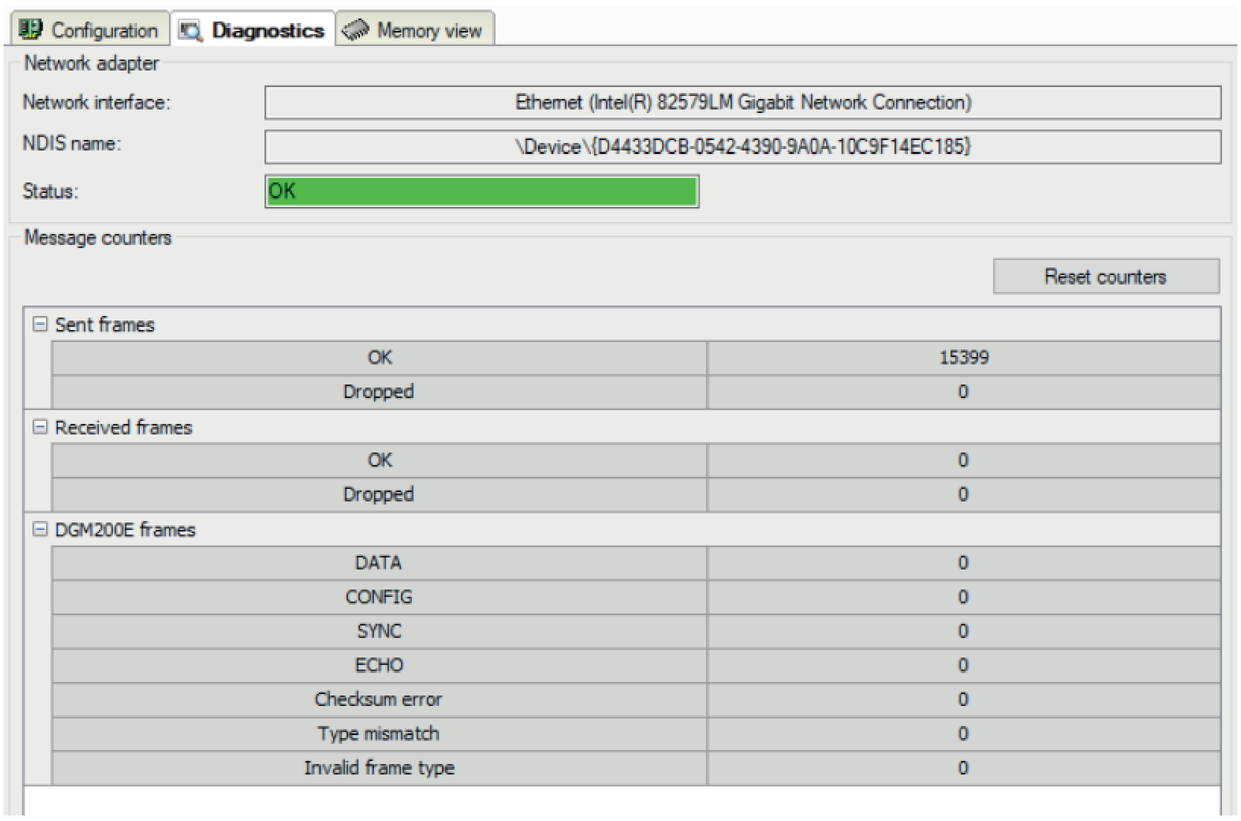
☑ Active

Network interface: Ethernet (Intel(R) 82579LM Gigabit Network Connection)

DGM port: 0

When clicking on a DGM200E link the above **Configuration** tab will be displayed where the following parameters can be configured:

- **Active**: check this if you want to use this link and the associated network interface.
- **Network interface**: the network interface you want to use to receive DGM200E data on this link. The format of the network interface is *Adapter name (device name)*.
- **DGM port**: the port number that will be used in DGM200E messages sent by ibaPDA. Possible values are 0 to 19.



Network adapter

| | |
|---|---|
| Network interface: | Ethernet (Intel(R) 82579LM Gigabit Network Connection) |
| NDIS name: | \Device\{D4433DCB-0542-4390-9A0A-10C9F14EC185} |
| Status: | OK |

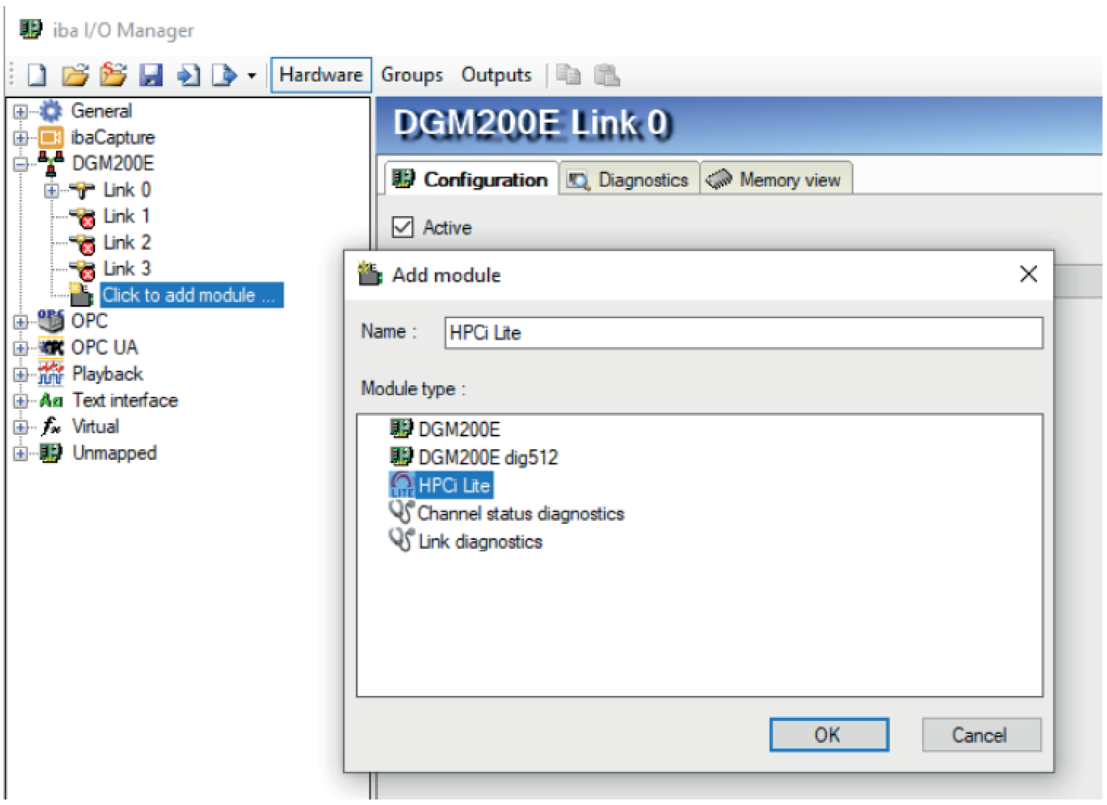
Message counters

| | |
|---|---|
| **Sent frames** | |
| OK | 15399 |
| Dropped | 0 |
| **Received frames** | |
| OK | 0 |
| Dropped | 0 |
| **DGM200E frames** | |
| DATA | 0 |
| CONFIG | 0 |
| SYNC | 0 |
| ECHO | 0 |
| Checksum error | 0 |
| Type mismatch | 0 |
| Invalid frame type | 0 |

The **Diagnostics** tab displays the following information:

- **Network interface**: the network interface that was configured in the **Configuration** tab.
- **NDIS name**: an internal identifier of the network interface used by the Windows networking stack.
- **Status**: status of the DGM200E interface link in the ibaPDA server. The status refers to the communicaton between the ibaPDA driver and the network adapter only. It does not indicate whether a connecton to a DGM200 network is working or not. The status is “OK” when the communicaton between ibaPDA and the network adapter is running, even if no DGM200 network or DGM 200-E device is connected.
- **Sent frames OK/dropped**: the number of sent DGM200E frames that were resp. processed correctly or dropped by the ibaPDA server.
- **Received frames OK/dropped**: the number of received DGM200E frames that were resp. processed correctly or dropped by the ibaPDA server.
- **DGM200E frames**: further diagnostics based on the actual contents of correctly processed DGM200E frames is available here. For more information we refer to the DGM200E interface’s manual.

Finally the **Memory view** tab can be used to examine the raw contents of ibaPDA’s internal buffer reserved for this DGM200E link.



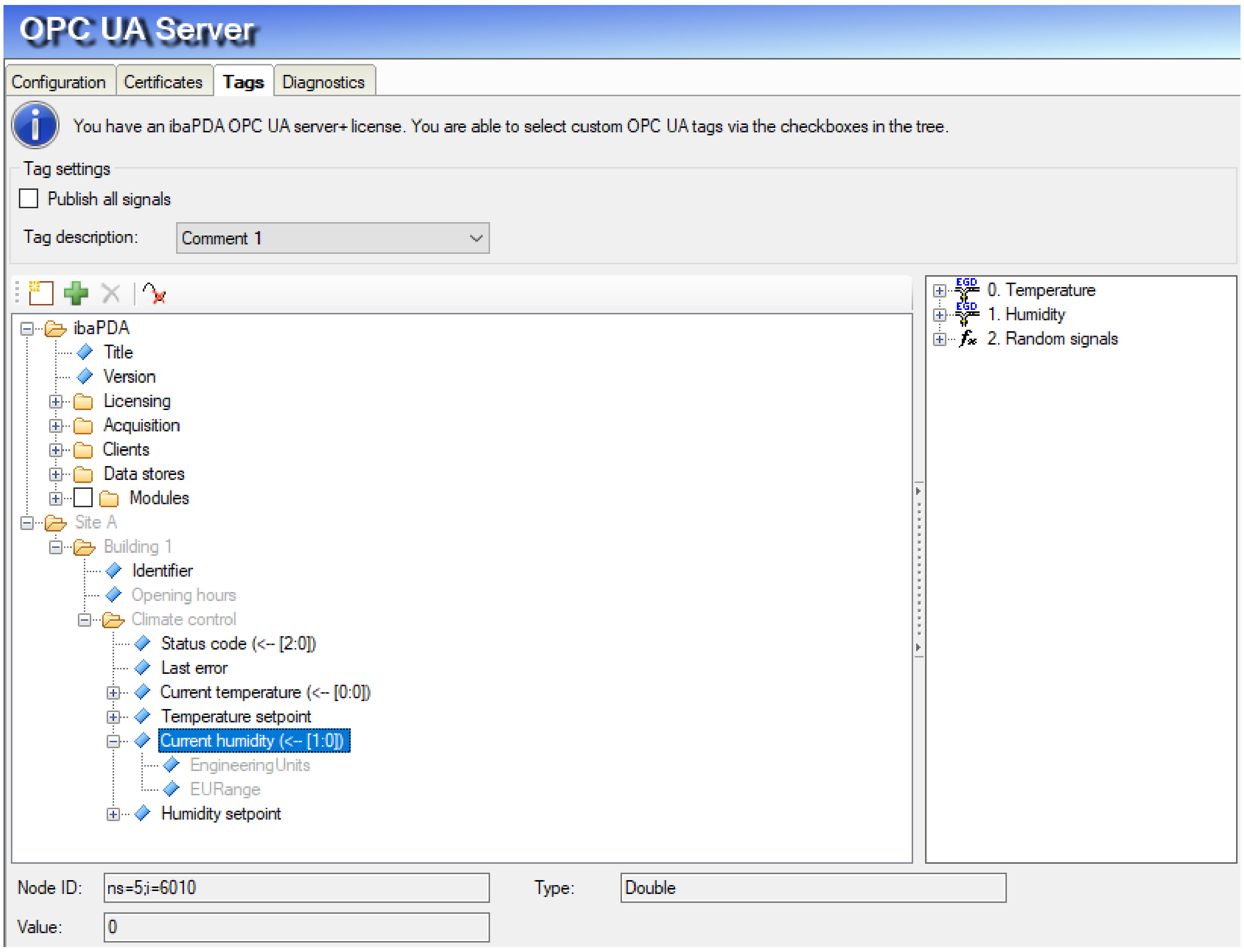
Modules can be added to a DGM200E link by right-clicking on a link or by clicking the **Click to add module …** node in the I/O manger. The following module types are supported by the DGM200E interface:

- **DGM200E**: used to read data from specific offsets in incoming DGM200E data frames.
- **DGM200E**: used to read digital signals packed in 32-bit integers at specific offsets in incoming DGM200E data frames.
- **HPCi Lite**: dynamically request data from an HPC/HPCi system. For more information we refer to the HCPi request manual.
- **Channel status diagnostics**: contains a general status of the DGM200E network and a specific status for each of the 20 possible DGM200 channels. This information is maintained by the DGM200E stack in the ibaPDA server based on the incoming DGM200E frames.
- **Link diagnostics**: allows the user to measure the values displayed in the *Diagnostics* tab of the link as signals.

Apart from these modules the DGM200E interface can also be used in combination with the HPCi request interface.

# 7 OPC UA Server: Support for user defined information models

It is now possible to import custom UA NodeSets into ibaPDA's OPC UA server and assign signals to the imported tags (if the tag's data type is compatible with the signal's data type). The standard tags automatically generated by ibaPDA will always be available in a dedicated namespace.



You have an ibaPDA OPC UA server+ license. You are able to select custom OPC UA tags via the checkboxes in the tree.

Publish all signals

Tag description: Comment 1

Node ID: ns=5;i=6010

Type: Double

Value: 0

To import an UA NodeSet click the ➕ button in the toolbar. After selecting a valid XML file the tags will be added in the tree structure shown on the left hand side, below the standard ibaPDA root tag.

Imported tags that cannot be linked to a signal are displayed in grey (i.e. folders and tags with unsupported data types). To link a signal to a tag select it from the signal tree on the right hand side and drag it to the tag in the tree structure on the left hand side. The signal ID will be added to the tag's display name to indicate that a signal is currently linked to the tag.

When selecting a tag while the acquisition is not running only the **Type** is displayed. If the tag's data type is an OPC UA primitive type (e.g. integers, floats, strings) a human readable type name will be displayed. For complex types (e.g. AnalogItemType) the node ID will be displayed. Note that the **Node ID** cannot be displayed while the OPC UA server is not running because the namespace index of the nodes is adjusted upon server start. The actual **Node ID** and **Value** are displayed one the server is running.

Tags with the AnalogItemType definition are treated in a special way: as can be seen in the tree structure it is not possible to assign individual signals to the EngineeringUnits and EURange properties (since they are not standard data types). However, ibaPDA automatically fills in the EngineeringUnits property when linking a signal to the parent AnalogItemType. The UnitId of the EngineeringUnits property is determined using the list made available by the OPC foundation:

http://www.opcfoundation.org/UA/EngineeringUnits/UNECE/UNECE_to_OPCUA.csv

If the signal unit configured in ibaPDA is found in the above list the UnitId is automatically set; if the signal unit is not recognized the UnitId is set to -1. Note that several UNECE codes may correspond to the same unit display name (e.g. UNECE M43 and UNECE 77). Since in ibaPDA only the display name of the unit is configured and no further information is available, the last entry in the list will be used (so in the case of *mil* it will be UNECE 77).

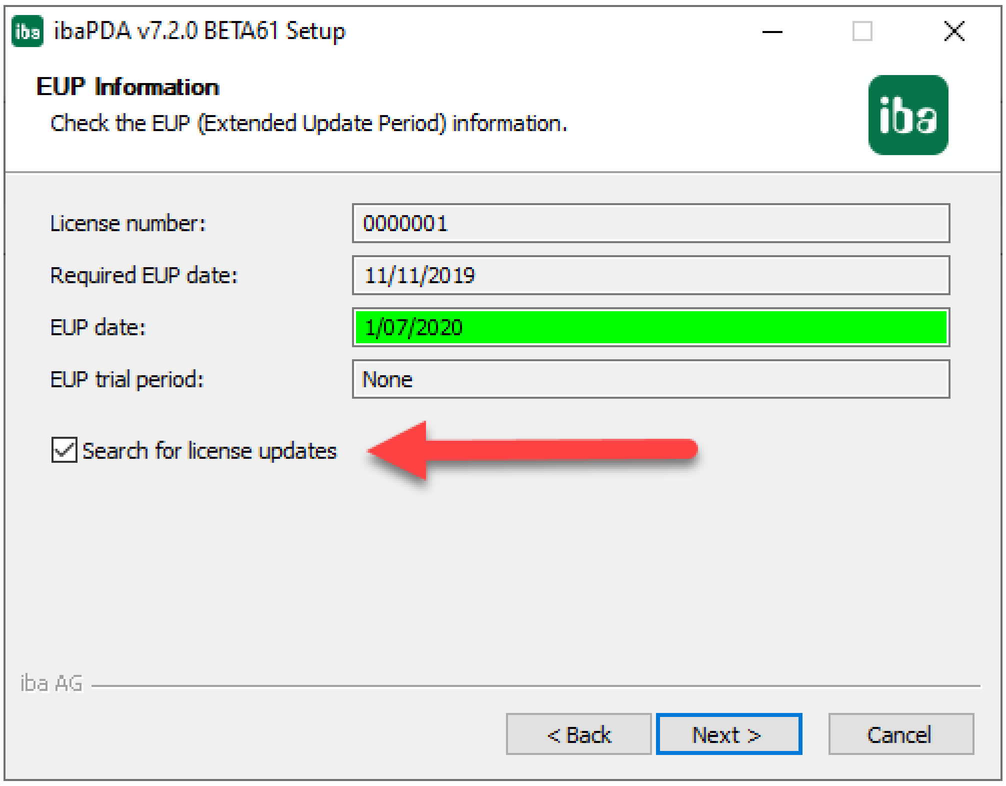
If you want to replace the signal linked to a tag simply drag a new signal to the tag; if you want to remove a signal linked to a tag (without assigning a new signal to it), select the tag and click the button in the toolbar. You can clear multiple tags at once by using multiselect or by selecting a parent tag.

To delete a single imported root tag, select the root tag and click the button (note that subnodes cannot be removed; only the entire imported nodeset can be removed).

Clicking the button will create a fresh tag configuration (all other OPC UA server settings will remain unaltered) so all imported tags will be deleted and all configured signals in the ibaPDA root tag will be cleared.

# 8 ibaLicenseUpdater

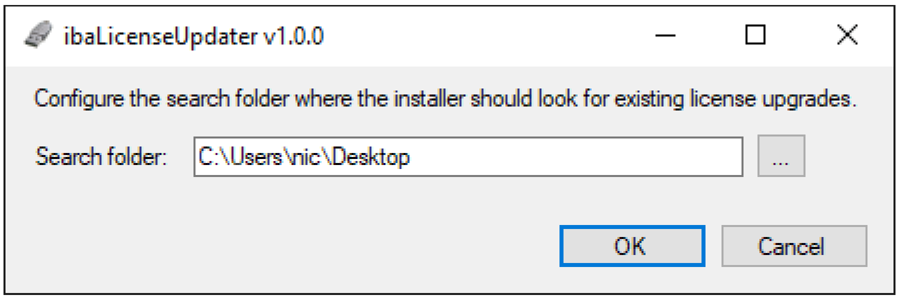
When you install the ibaPDA server then the EUP information page is shown in the installer. It contains a new option to search for license updates.


ibaPDA v7.2.0 BETA61 Setup
EUP Information
Check the EUP (Extended Update Period) information.
iba AG
< Back
Next >
Cancel

| | |
|---|---|
| License number: | 0000001 |
| Required EUP date: | 11/11/2019 |
| EUP date: | 1/07/2020 |
| EUP trial period: | None |

☑ Search for license updates

When you select this option then the ibaLicenseUpdater is started. The first time it runs you can configure the folder it should search for license upgrades. The folder can be a network folder. The folder should contain ibaDongleUpgrade_xxx.exe files.



Configure the search folder where the installer should look for existing license upgrades.

Search folder: C:\Users\nic\Desktop

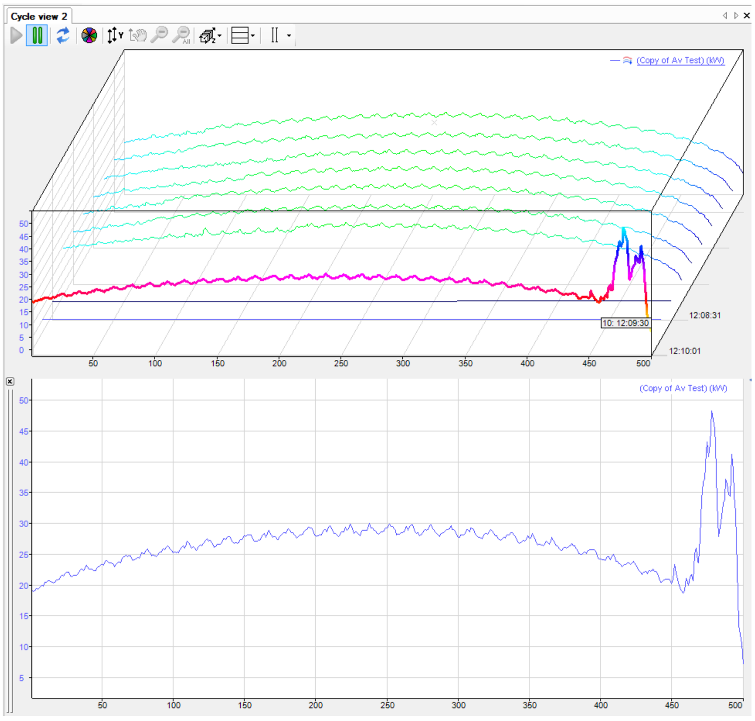
If the ibaLicenseUpdater finds an ibaDongleUpdgrade_xxx.exe file where the xxx matches the serial number of the connected dongle then it will execute the dongle upgrade. DongleUpgrade will run interactively. If the dongle upgrade succeeds then the ibaDongleUpgrade_xxx.exe file is moved to a subfolder “Applied” of the configured folder.

# 9 ibaInCycle

ibaInCycle is a new Add On in ibaPDA for anomaly detection in cyclic and rotating processes. Any signal with repeating cyclic behavior can be monitored with ibaInCycle. There are two types of modules available. The Expert-Module offers an individual analysis of the cycles and the Auto-Adapting-Module offers a self learning functionality taking process conditions into account.

For further information on ibaInCycle, please refer to the manual.

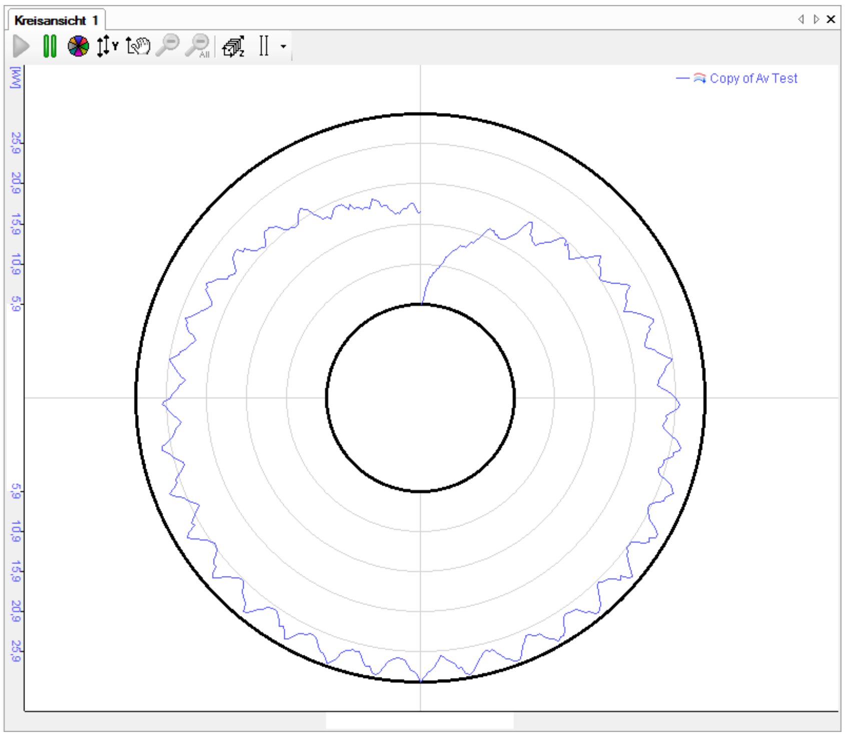
## 9.1 Cycle view



For results of ibaInCycle the new cycle view offers the option to display single curve, waterfall and contour plots.

For further information, please refer to the manual.

## 9.2 Circle view

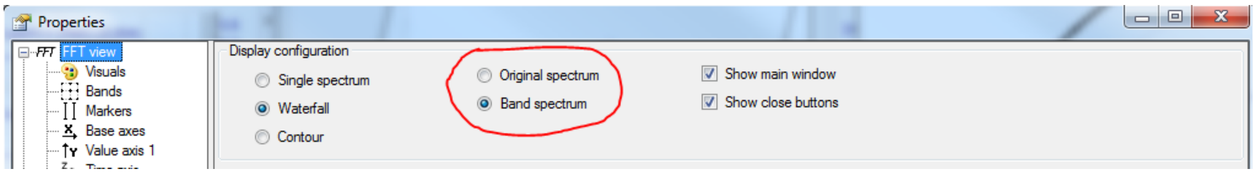


For results of ibaInCycle the new circle view displays the time signal of one cycle (revolution) drawn in a circle.

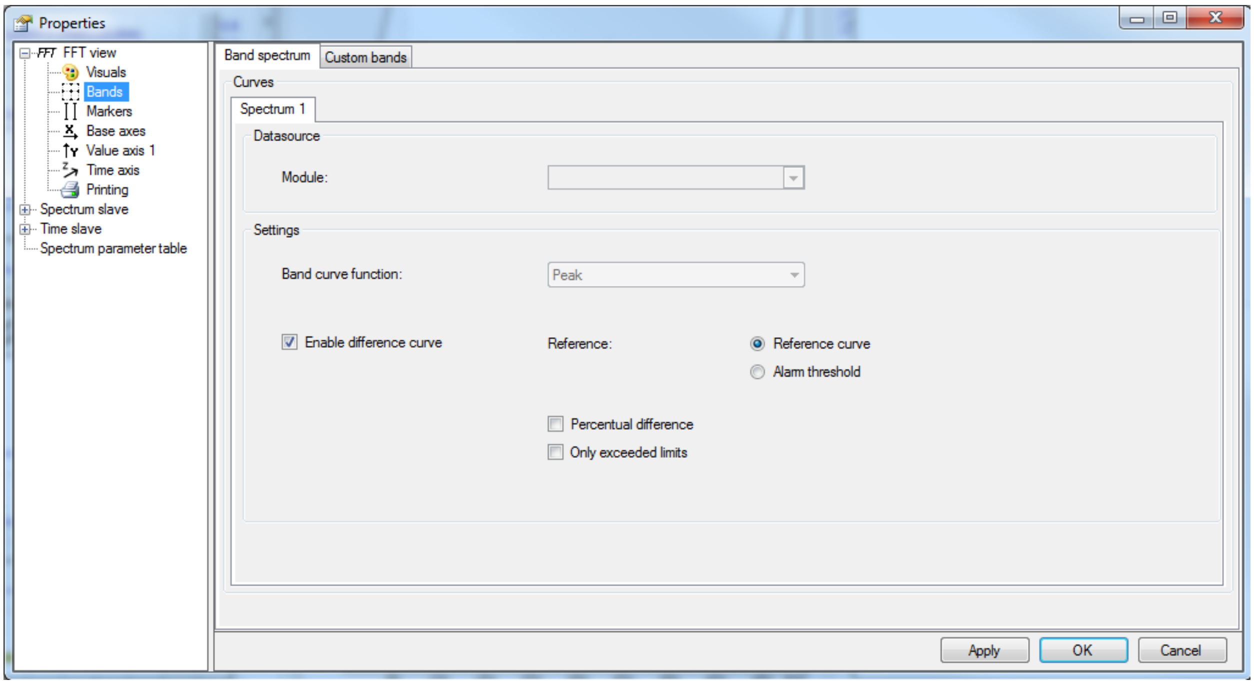
For further information, please refer to the manual.

# 10 FFT-view: Band and Difference Spectrum for ibaInSpectra

The FFt-view now offers the option to display a band-spectrum instead of the original spectrum.



Instead of showing the raw spectrum the peak, peak frequency or RMS values for each ibaInSpectra band are displayed in this spectrum. In case the FFT-view shows an Auto-Adapting Module, a difference spectrum can be shown, which displays the difference between the reference spectrum or alarm thresholds for each band.



- **Percentual difference** shows all values in percent to the reference values
- When **only exceeded limits** is active, for all bands where the limit is not exceeded the y-value is set to 0
- **Absolute difference** displays all differences in positive y-direction