ibaPDA-Data-Store-InfluxDB

Data streaming into InfluxDB

Manual

Issue 1.0

Measurement Systems for Industry and Energy

www.iba-ag.com

Manufacturer

iba AG

Koenigswarterstr. 44

90762 Fuerth

Germany

Contacts

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

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

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

Version	Date	Revision - Chapter / Page	Author	Version SW
1.0	04-2022	First issue	st	8.0.0

Content

1 About this manual

This documentation describes the function and application of the data store *ibaPDA-Data-Store-InfluxDB*.

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

You will find basic information about data storage in *ibaPDA* in the *ibaPDA* manual part 5.

1.1 Target group and previous knowledge

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

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

- Windows operating system
- Basic knowledge of *ibaPDA*
- Basic knowledge of databases, cloud or cluster storage technology

1.2 Notations

In this manual, the following notations are used:

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

1.3 Used symbols

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

Danger!

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

- Observe the specified measures.

Warning!

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

- Observe the specified measures.

Caution!

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

- Observe the specified measures

Note

A note specifies special requirements or actions to be observed.

Tip

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

Other documentation

Reference to additional documentation or further reading.

2 Introduction

Different types of data stores are available in *ibaPDA* for different purposes and methods of data storage. Depending on the licenses enabled, different types of data stores are available for configuration in the dialog.

This documentation describes the “InfluxDB timebased data store” type of recording. This recording type writes timebased data to an InfluxDB database management system.

Chapter ↗ *Signal selection*, page 18 describes the selection of the signals that are to be recorded.

The data can be continuously recorded or recorded by trigger, see chapter ↗ *Trigger mode*, page 19.

2.1 System requirements

The following system requirements are necessary when using data storage in an InfluxDB server:

- *ibaPDA* v8.0.0 or higher
- License for *ibaPDA-Data-Store-InfluxDB*
- InfluxDB v2.x or higher

The licenses are staggered according to the number of signals that should be written in the InfluxDB server. The number of used data stores is unlimited.

Order no.	Product name	Description
30.671060	ibaPDA-Data-Store-InfluxDB-64	Data streaming into an InfluxDB server, max. 64 signals
30.671061	ibaPDA-Data-Store-InfluxDB-256	Data streaming into an InfluxDB server, max. 256 signals
30.671062	ibaPDA-Data-Store-InfluxDB-1024	Data streaming into an InfluxDB server, max. 1024 signals
30.671065	upgrade-ibaPDA-Data-Store-InfluxDB-64 to 256	License for extension from 64 to 256 signals
30.671066	upgrade-ibaPDA-Data-Store-InfluxDB-256 to 1024	License for extension from 256 to 1024 signals

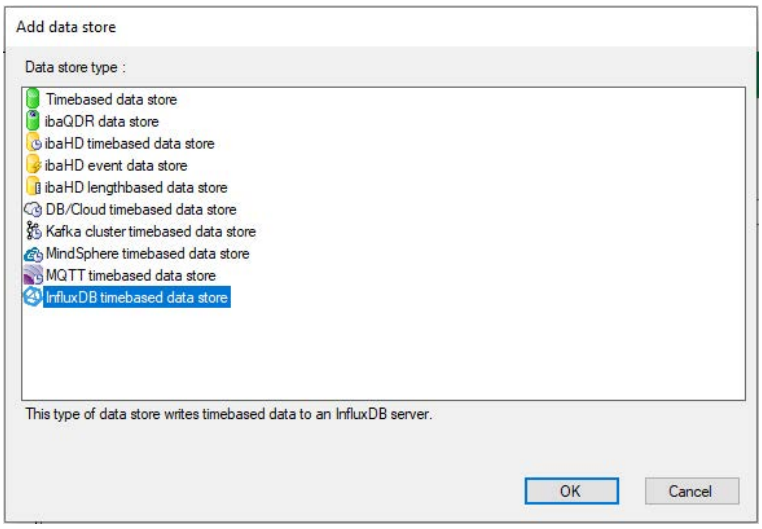
3 Data store configuration

3.1 Add a data store

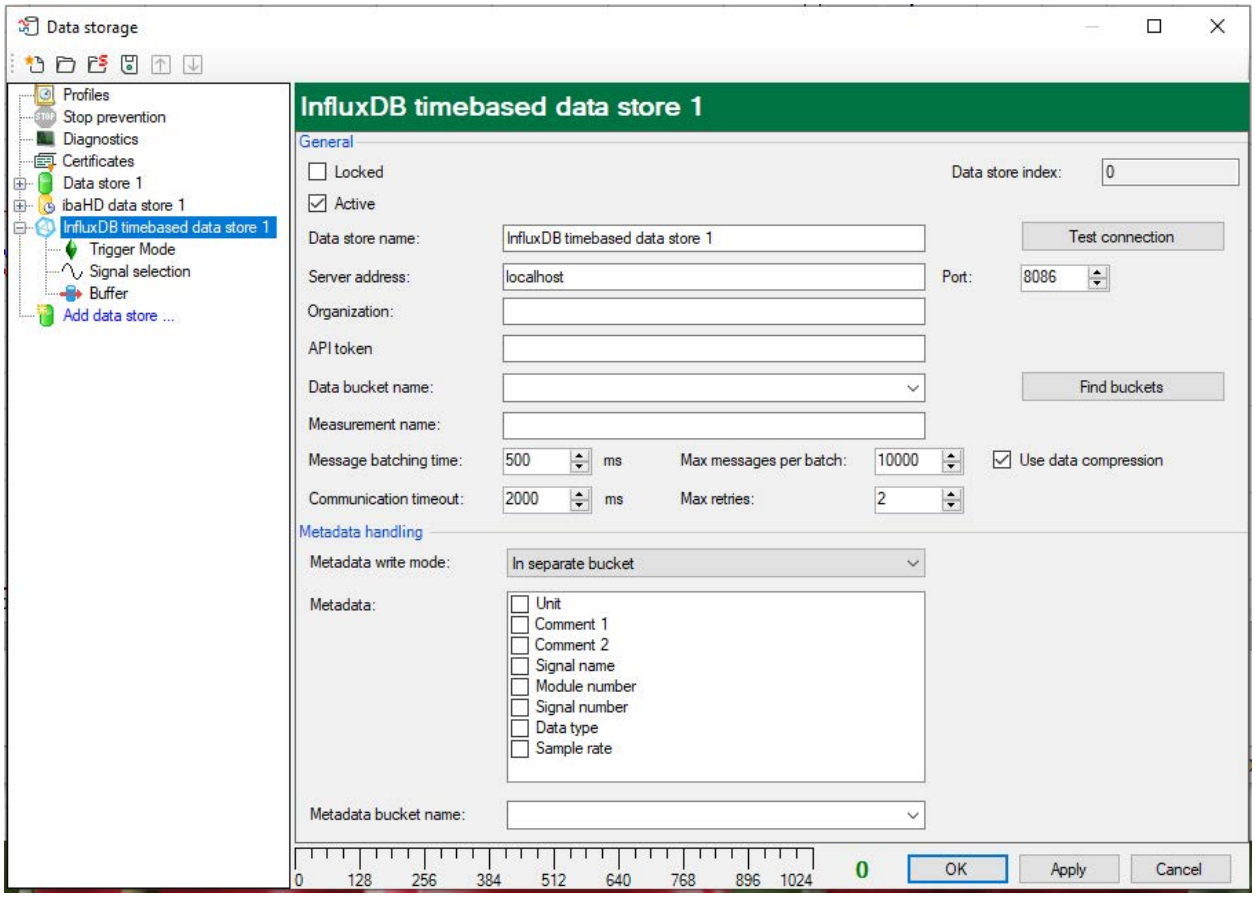
The dialog for data storage configuration can be opened in the *Configure – Data storage* main menu or by clicking on the button in the main toolbar.

In order to add a new data store, click on the blue link *Add data store* in the tree structure. You can also right-click on the data store node in the tree structure and choose *Add data store* from the context menu.

Select *InfluxDB timebased data store* for the streaming of timebased data into an InfluxDB server.



3.2 Data store InfluxDB



Locked
A data store can be locked in order to prevent an accidental or unauthorized change of settings.

Active
A data store must be enabled in order to work. However, you can configure various data stores and disable data stores that are not required.

Data store index
Unique index of all existing InfluxDB data stores. You need to reference this index e.g. in the virtual function *DataStoreInfoInflux()* for generating diagnostic data for a specific InfluxDB data store.

Data store name
You can enter a name for the data store here.

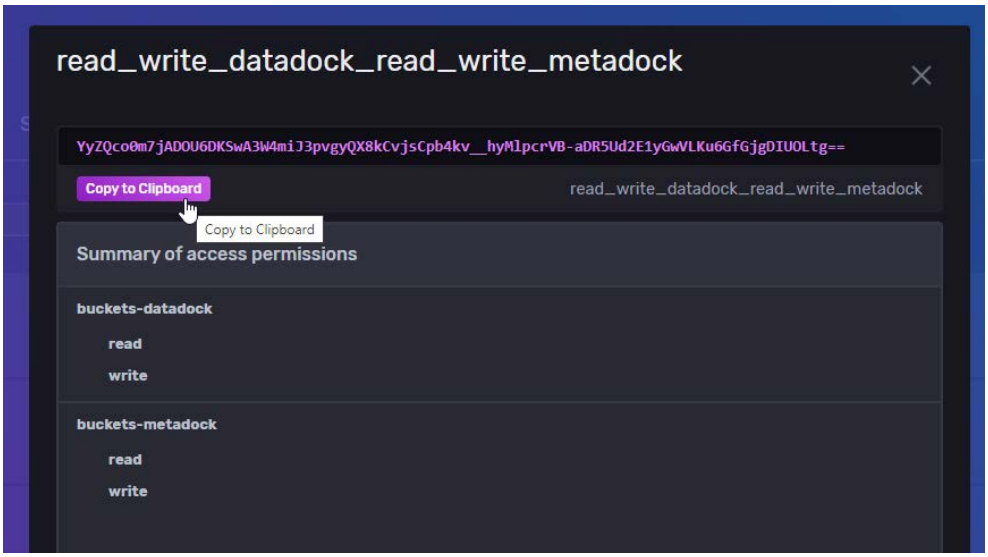
Server address
The IP address or hostname of the InfluxDB server.

Port
Port to use for the connection. The default port is 8086.

<Test connection>
Click on the button <Test connection> to verify if *ibaPDA* can establish a connection to your InfluxDB server using the server address and port number.

Organization
The organization in your InfluxDB you want to use.

API token
The API token you enter here needs to have read and write rights to the data bucket and, if used, the metadata bucket. To do this, log in to the InfluxDB web interface and go to API token configuration. Use the <Copy to clipboard> button to copy & paste your API token into the *ibaPDA* configuration.



Data bucket name
The name of the bucket you want to store the data in.

Use the button <Find buckets> to fill the dropdown list with all available buckets. Then select the data bucket you want to use from the dropdown list.

Only buckets will be found (and can be used) which belong to the configured organization and have read rights assigned for the API token in use.

Measurement name
Name for the current measurement

Message batching time, Max messages per batch
To reduce the number of single telegrams sent to the InfluxDB server, multiple data points are collected and sent as a packet. This is done when either the message batching time expires, or the number of messages exceeds the max messages per batch number.

Use data compression
You can enable data compression to reduce the size of the transmitted data packets.

Communication timeout
Time until a telegram sent to the server is regarded as not successfully delivered.

Max retries
Number of retries for a not successfully delivered message until the connection to the InfluxDB server is regarded as broken.

In the lower section it is configured how metadata should be handled. For examples of the different data models, see chapter ↗ *Examples for data models*, page 12.

Select a metadata write mode.

No metadata
Only the data bucket is used. No signal metadata is written, except for the *ibaPDA* signal ID.

The signal ID is written as a tag key not as a field key. Tag keys are indexed in InfluxDB. Therefore queries on tags are faster than queries on fields. In this write mode the least data is transmitted via the line protocol and the least space is required for data storage in InfluxDB. The filtering options in queries are limited on the other hand, since only the signal ID is available as metadata.

In data bucket
Here you can choose to write additional metadata (the signal ID by default is always written). All additional metadata will be written into the data bucket as extra tag keys. Choose via checkboxes which metadata you want to include.

- Unit
- Comment 1
- Comment 2
- Signal name
- Module number
- Signal number
- Data type
- Sampling rate in ms

The metadata needs to be transmitted via the line protocol for each data point in this write mode. Also more space is required for data storage in InfluxDB. The advantage is that time series data as well as all metadata is kept in one single bucket.

In separate bucket
The additional metadata and the signal ID is written into a separate bucket. Metadata is written only once when either the IO configuration or the data store configuration is applied. Via the signal ID and the timestamp the metadata and the time series data from both buckets can be correlated.

This way less data needs to be transferred and stored compared to the write mode “In data bucket”. The disadvantage is that the analysis e.g. via Flux is more complex.

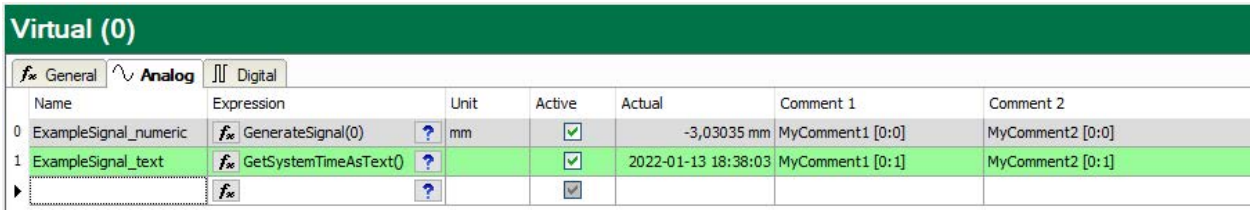
Metadata bucket name
The name of the bucket you want to store the metadata when using the write mode “In separate bucket”. Use the button <Find buckets> to fill the dropdown list with all available buckets. Then select the metadata bucket you want to use from the dropdown list.

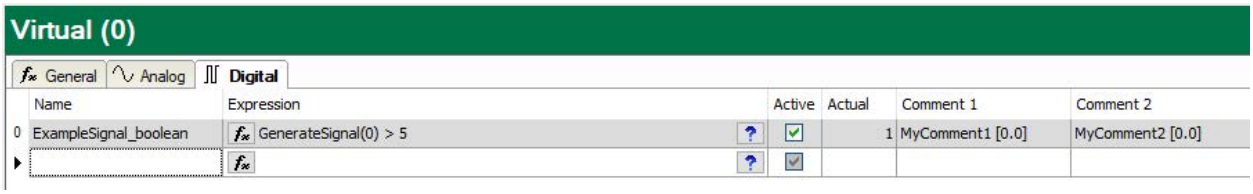
3.3 Examples for data models

The following example demonstrates what the data in InfluxDB will look like when using the different metadata write modes.

Three example signals are configured in a module with number 0 and will be written to InfluxDB applying the different metadata write modes. The three signals are:

- The signal with ID [0:0] has a numeric value
- The signal with ID [0:1] has a text value
- The signal with ID [0.0] has a boolean value





Metadata write mode 'No metadata'

The column *_value* contains the signal value for each written timestamp. The field key (column *_field*) shows an entry *value*, *value_t* or *value_b* depending on if it is a numeric, text or boolean signal value. There is just one tag key *SignalId* filled with the signal ID. A tag key is indexed for faster query execution.

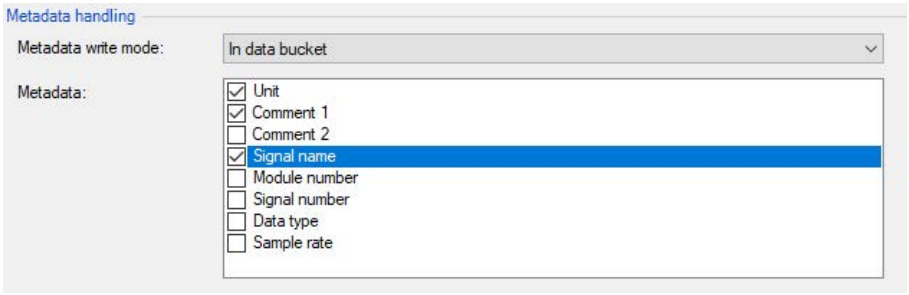
_time	_value	SignalId	_field	_measurement
13/01/2022 18:47:33.810	-2.36499	[0:0]	value	MyMeasurement
13/01/2022 18:47:33.910	-7.624425	[0:0]	value	MyMeasurement
13/01/2022 18:47:34.010	-9.971589	[0:0]	value	MyMeasurement
13/01/2022 18:47:34.110	-8.509945	[0:0]	value	MyMeasurement
13/01/2022 18:47:34.210	-3.797791	[0:0]	value	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 18:47:33.810	2022-01-13 18:47:33	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:33.910	2022-01-13 18:47:33	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:34.010	2022-01-13 18:47:34	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:34.110	2022-01-13 18:47:34	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:34.210	2022-01-13 18:47:34	[0:1]	value_t	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 18:47:33.810	false	[0.0]	value_b	MyMeasurement
13/01/2022 18:47:33.910	false	[0.0]	value_b	MyMeasurement
13/01/2022 18:47:34.010	false	[0.0]	value_b	MyMeasurement
13/01/2022 18:47:34.110	false	[0.0]	value_b	MyMeasurement
13/01/2022 18:47:34.210	false	[0.0]	value_b	MyMeasurement

Metadata write mode ‘In data bucket’

As an example three additional metadata values are written in the data bucket (*Unit, Comment 1, Signal name*):



Compared to the first example additional tag key columns for the metadata values appear: *Unit, Comment1, SignalName*. InfluxDB writes to the data bucket cyclically. Even though metadata rarely changes, this data is still transmitted and stored repetitively.

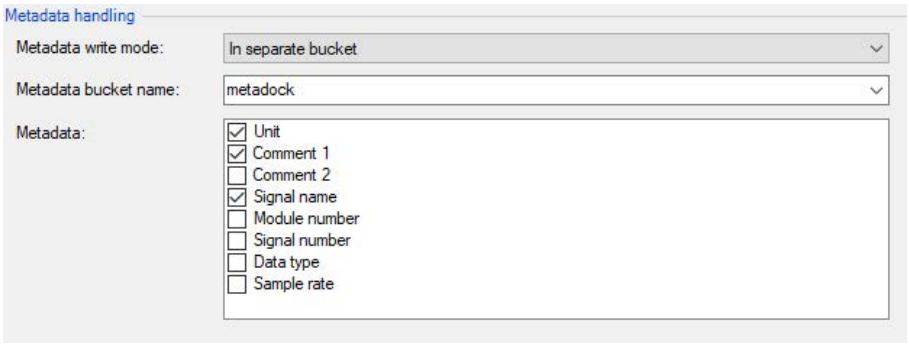
_time	_value	Comment1	SignalId	SignalName	_field	_measurement	Unit
13/01/2022 18:58:...	-8.509945	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	-3.797791	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	2.36499	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	7.624425	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	9.971589	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 18:58:22.1...	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22....	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22....	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22....	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22....	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 18:58:22.1...	false	MyComment1 [0.0]	[0.0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22....	false	MyComment1 [0.0]	[0.0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22....	false	MyComment1 [0.0]	[0.0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22....	true	MyComment1 [0.0]	[0.0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22....	true	MyComment1 [0.0]	[0.0]	ExampleSignal_boole...	value_b	MyMeasurement

Metadata write mode ‘In separate bucket’

Again three additional metadata values are written, but this time in a separate metadata bucket (*Unit, Comment 1, Signal name*):



The data bucket now again looks the same as when using the write mode ‘No metadata’:

_time	_value	SignalId	_field	_measurement
13/01/2022 19:08:57.950	-7.624425	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.050	-9.971589	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.150	-8.509945	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.250	-3.797791	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.350	2.36499	[0:0]	value	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 19:08:57.950	2022-01-13 19:08:57	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.050	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.150	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.250	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.350	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 19:08:57.950	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.050	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.150	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.250	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.350	false	[0.0]	value_b	MyMeasurement

The metadata bucket shows now a single metadata entry per signal for the point of time where the acquistion was started. Metadata can change only when the configuration was changed.

The column *_value* contains the signal ID. The field key (column *_field*) is always *SignalId*. The tag key *SignalId* always exists. The other tag key columns depend on the metadata that was configured. In this example again three additional tag key columns *Unit, Comment1, SignalName* exist.

_time	_value	Comment1	SignalId	SignalName	_field	_measurement	Unit
13/01/2022 19:0...	[0:0]	MyComment1 [0:0]	[0:0]	ExampleSignal_...	SignalId	MyMeasurement	mm

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 19:08:4...	[0:1]	MyComment1 [0:1]	[0:1]	ExampleSignal_text	SignalId	MyMeasurement

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 19:08:4...	[0.0]	MyComment1 [0.0]	[0.0]	ExampleSignal_bool...	SignalId	MyMeasurement

Less metadata is transmitted this way but the analysis is more complex sind the data from two buckets needs to be correlated.

3.4 Buffer

The data storage uses a memory buffer and additionally a file buffer that can be enabled optionally.

The description applies to all types of data stores that transfer data to external systems and where temporary accessibility and available bandwidth issues may occur, such as:

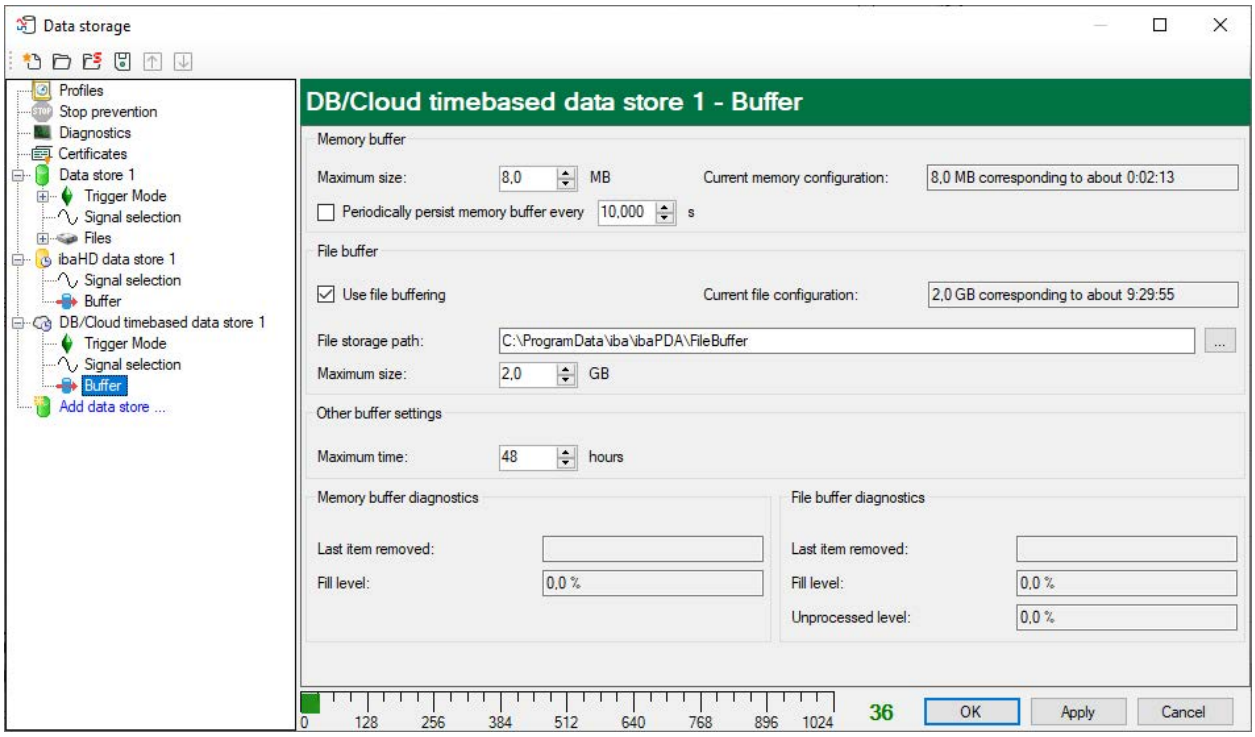
- ibaHD timebased/event/lengthbased
- DB/Cloud timebased
- Kafka cluster timebased
- MQTT timebased
- MindSphere timebased
- InfluxDB timebased

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 from the memory buffer immediately. If the connection is lost, or the data cannot be sent out fast enough, the data remain 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 in a local drive of the *ibaPDA* server. When the file buffer is enabled, data is transferred from the overflowing memory buffer to the file buffer. If the acquisition is finished or restarted (e.g. by applying a modified IO 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 transferred 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. The data is saved in the format that was configured in the data store at the time of buffering and it is also sent in this format when the connection is established again.

You configure the buffering in the *Buffer* node of the respective data store. The figure below shows the buffer configuration using the example of DB/Cloud data store.



Memory buffer

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

Maximum size

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

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

Periodically persist memory buffer every ... s

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

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

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

Current memory configuration

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

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

The same file directory can be used for several 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 a data store. 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 can enter a value between 1 and 1000 hours.

Memory buffer / File buffer diagnostics

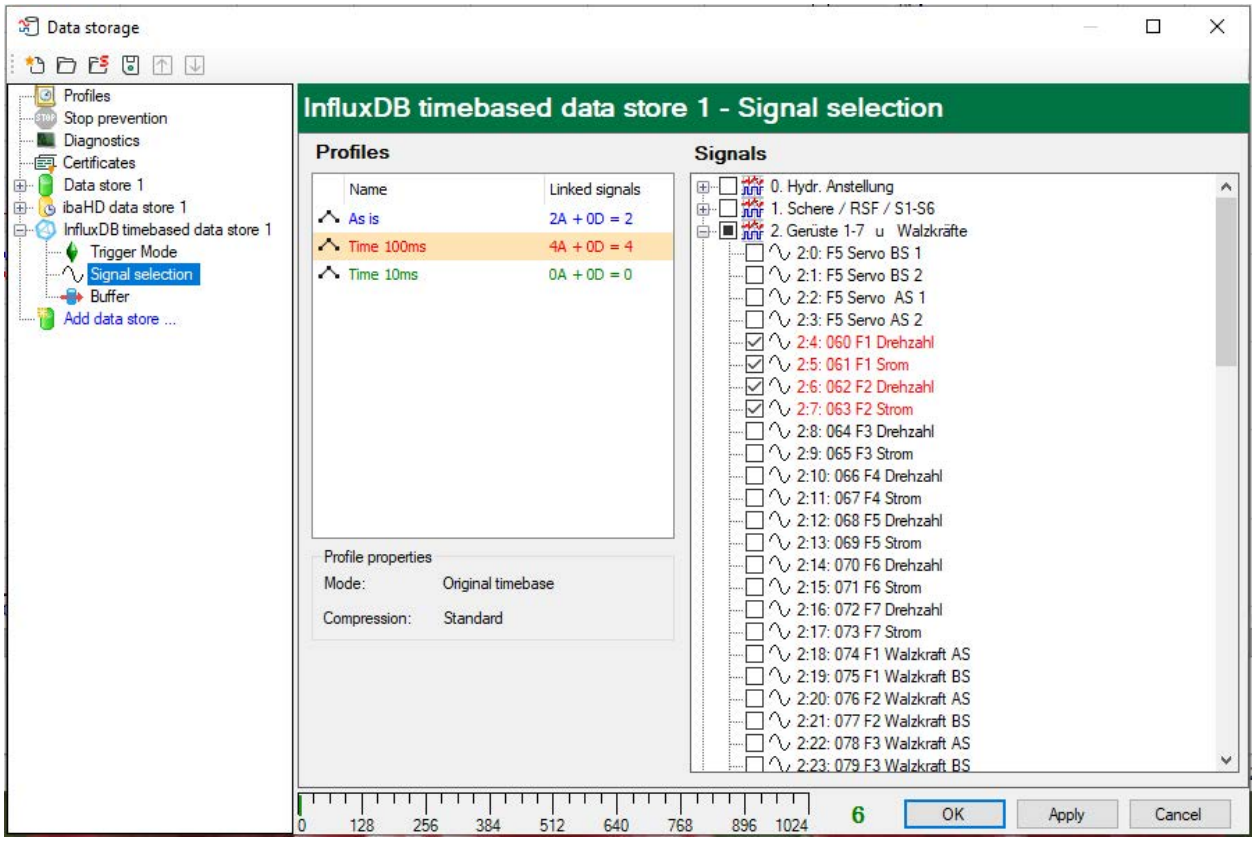
Last item removed
Indicates when the last item was taken from this part of the buffer.

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.

4 Signal selection

To enable signals to be recorded, they must be assigned to a storage profile of type *Time*. Select the signal selection node below your *InfluxDB timebased data store* to open the signal selection dialog.



In the profile list, select the storage profile to which you want to assign certain signals. Set a check mark in the selection fields next to the signals which you would like to assign to this profile. A signal can only be assigned to one profile per data store. The *Profile properties* section displays some information about the configured timebase, filtering and column naming of the selected profile.

InfluxDB data stores are staggered according to the number of signals written to the database. The current number of selected signals in all InfluxDB data stores is shown at the bottom of the dialog, similar to the number of configured signals in the I/O Manager.

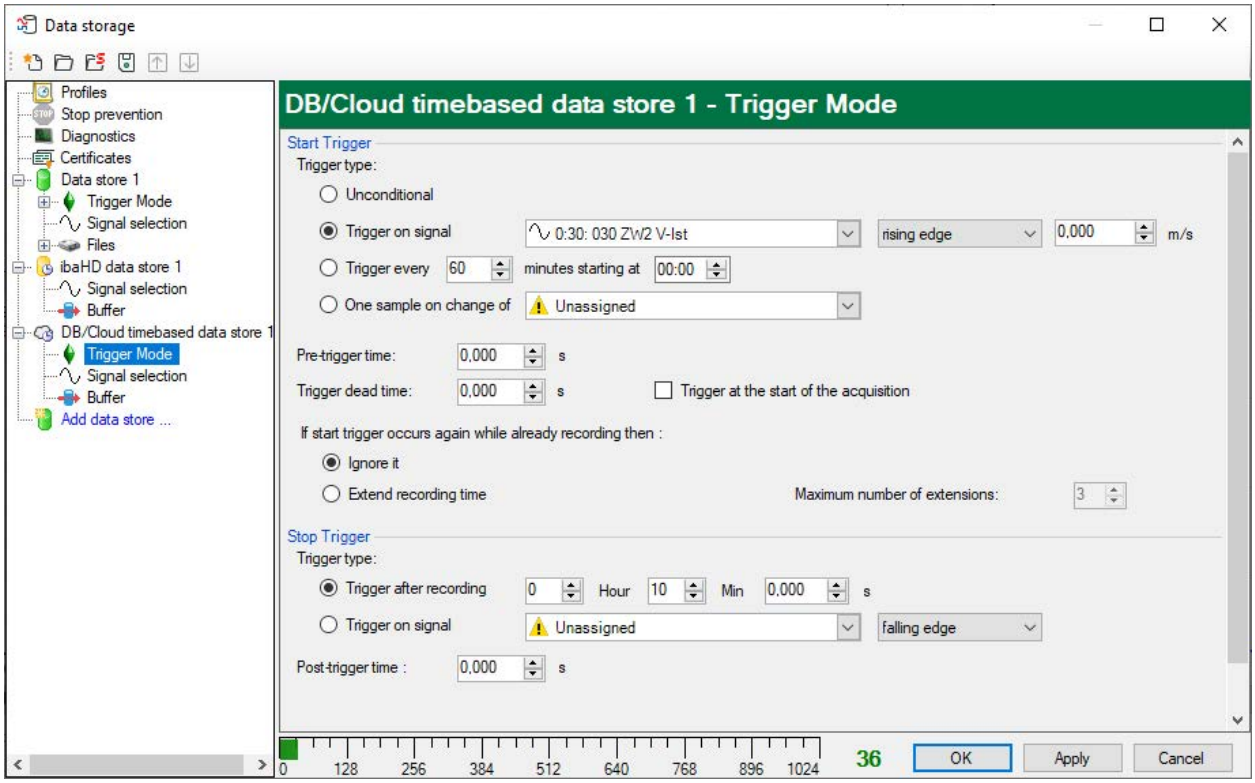
The licensed number of signals is indicated by the length of the signal strip. In the above example, it is possible to write up to 1,024 signals in several InfluxDB data stores. Currently 6 signals are enabled.

5 Trigger mode

The description applies to all types of data stores that transfer data to external systems, such as:

- ibaHD time/event/lengthbased
- DB/cloud timebased
- Kafka cluster timebased
- MQTT timebased
- MindSphere timebased
- InfluxDB timebased.

In the *Trigger Mode* node, you determine when data is recorded, here using the example of DB/Cloud timebased data store.



Start trigger

You initially choose whether you would like to continuously record or it should be fired by a trigger.

Unconditional

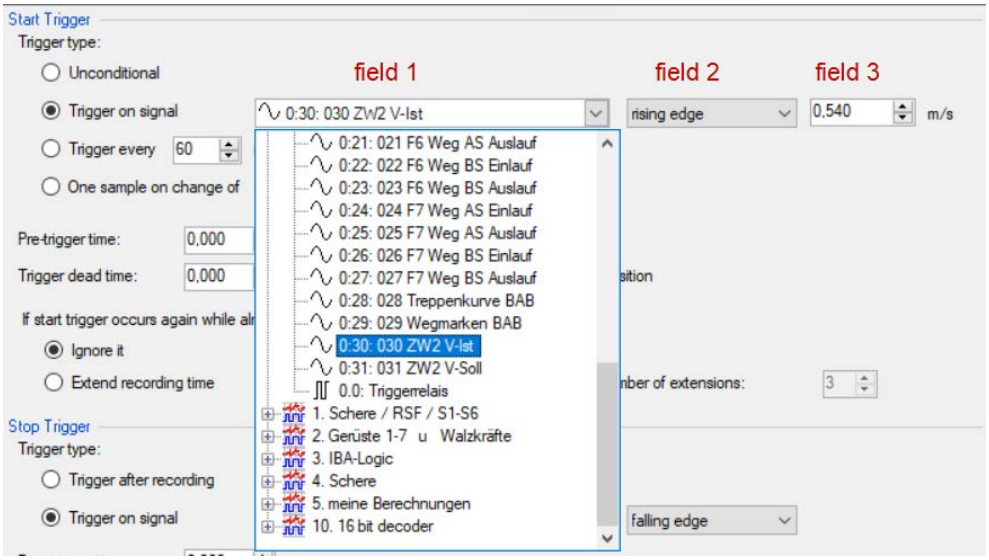
The data is continuously recorded with this selection. In this case, the recording will start immediately at the start of the measurement or when pressing the "GO" button.

Trigger on signal

If you want the trigger to fire on a measured signal or a virtual signal, you need to check *Trigger on signal* in the option field. In the fields next to this, define the properties of the trigger signal.

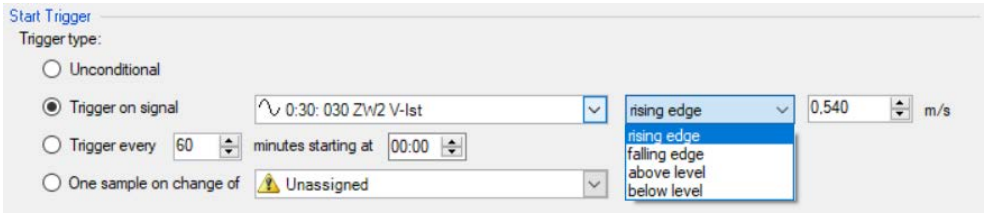
- Field 1: Drop-down list for signal selection (available analog and digital signals)
- Field 2: Drop-down list for selecting edges or levels
- Field 3: Drop-down list for selecting the trigger level value given in the specific physical unit (field 3 is only enabled in case of analog trigger signals)

Both analog and digital signals can serve as triggers. The signal to trigger on is to be selected from the drop-down lists (see picture below, field 1). In the drop-down list, you will find the well-known signal tree containing available signals. Select the signal you want to use as trigger signal.

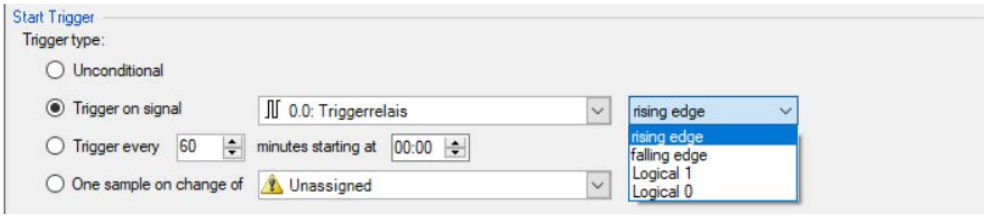


Depending on whether a digital or an analog signal was selected, the fields 2 or 3, respectively, are offered allowing the trigger event to be defined more specifically.

As for analog signals, you can choose between level or edge triggers including a predefined level (field 3).



As for digital signals, you can choose between level or edge triggers including the 2 levels logical 0 (FALSE) and logical 1 (TRUE).



Trigger every ...

If you want to use a start trigger always at a certain time regularly, you can check the “Trigger every ... minutes starting at …” option. Enter the period given in minutes, or select it from the input field. Value range is from 0 to 1440, which equals one day. Then enter or select the start time for the first trigger. Value range is from 00:00 to 23:59, which equals one day.

One sample on change of
When the value of the selected signal changes, a sample is recorded. The recording will 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.

Pre-trigger time
You can configure a pre-trigger time and then the recording begins by the pre-trigger time before the trigger event. If the trigger condition is met, the incoming data is added to the data buffered during the pre-trigger time.

Trigger dead time
This property is available for the start triggers “Trigger on signal”, “Trigger every ...” and "One sample on change of". The trigger dead time determines the time of suppressing subsequent triggers after a trigger occurred.

If the dead time, for instance, is set to 5 seconds, all other triggers are ignored for the duration of 5 seconds after the first trigger occurrence.

Trigger at the start of the acquisition
If you want the recording to start immediately at acquisition start or as soon as you apply a new data storage configuration, you also need to check the *Trigger on acquisition start* option. If you do not enable this option, the recording first starts once the trigger is fired.

If start trigger occurs again while file is already recording, then:
You can determine here what should happen if a new start trigger occurs while a recording is already running.

- Ignore it:
  Selecting this option will cause the system to ignore any new start trigger during a running recording for as long as the stop trigger occurs
- Extend recording time:
  If this option is enabled, it extends the duration of the running recording upon occurrence of another start trigger during an ongoing recording. This occurs as often as set in the "Maximum number of extensions on single file" field. If the max. number of extensions is reached, all subsequent start triggers will be ignored. Of course, the recording is stopped immediately by any stop trigger.

Stop trigger

The settings for the stop trigger are made in the same way as those for the start trigger. Here, both analog and digital signals can also be used as triggers.

Trigger after recording of x hours x minutes x seconds
Here you can configure a time span according to which the recording is ended - after the occurrence of the start trigger.

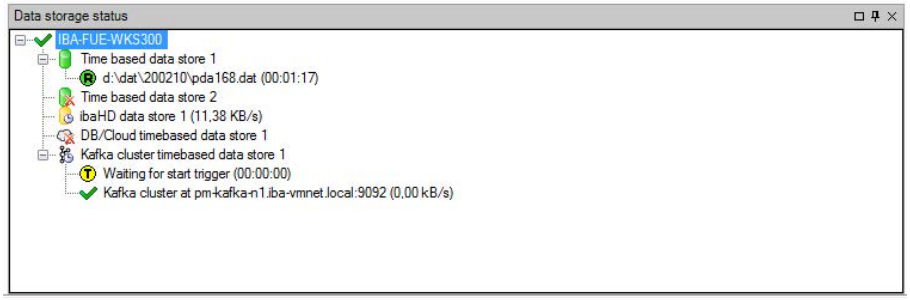
Trigger on signal
See explanation for start trigger above.

Post trigger time
You can configure a post trigger time and then the recording ends by the post trigger time after the stop trigger event.

6 Diagnostics

6.1 Data storage status

The data storage status window shows the current status of the data stores.



All defined data stores and their respective status are displayed here, depending on the data store, with server address, acquisition duration, write speed, etc.

The icon in front of the name indicates the current status of the storage:

Wait for the start trigger (only for triggered recording)

Recording in progress

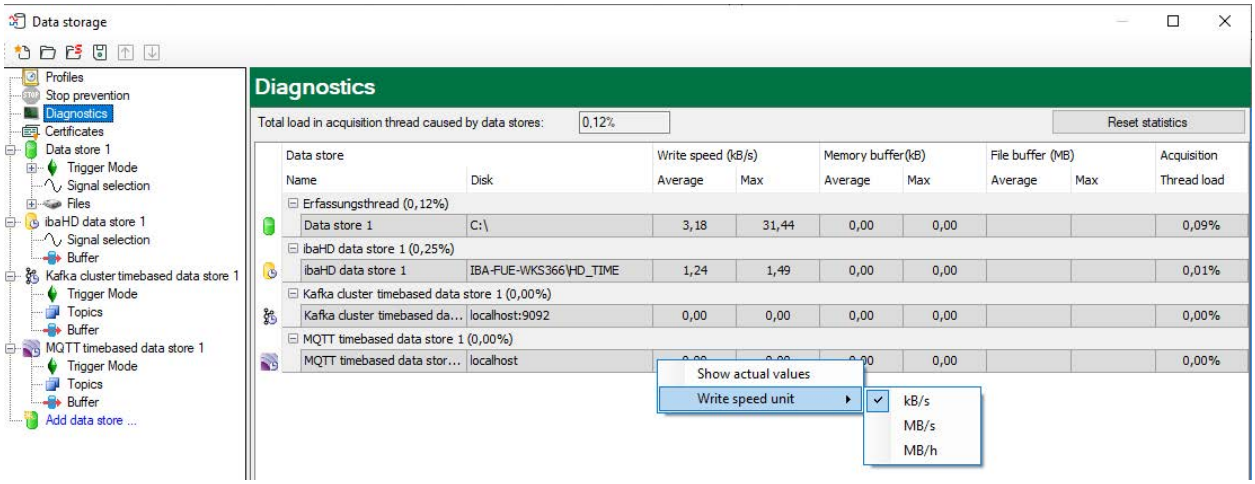
Post-trigger phase; stop trigger occurred, but acquisition continues until the post-trigger time is over

Disabled or faulty data store is indicated by a red cross in the data store icon.

Right-clicking on this node allows you to manually send a start or stop trigger.

6.2 Diagnostics of data stores

The *Diagnostics* node in the data storage tree offers information about the system load by the data stores. The measurement must be running.



The performance values of all data stores are shown in the table. There is one row per data store. The rows are grouped according to the threads that write the data.

In each group row is the name of the thread and (in brackets) its share of the load. The load average is displayed by default. But, you can switch between the average and actual value using the context menu.

The *Disk* column indicates the respective target to which the data is written, for example a hard disk partition, the address of the database, the address of the Kafka cluster, etc. The *Write speed* indicates how fast the data is written.

The *Memory buffer (kB)* columns indicate how much data is buffered in *ibaPDA*. The columns *File buffer (MB)* indicate how much data is buffered in the file buffer.

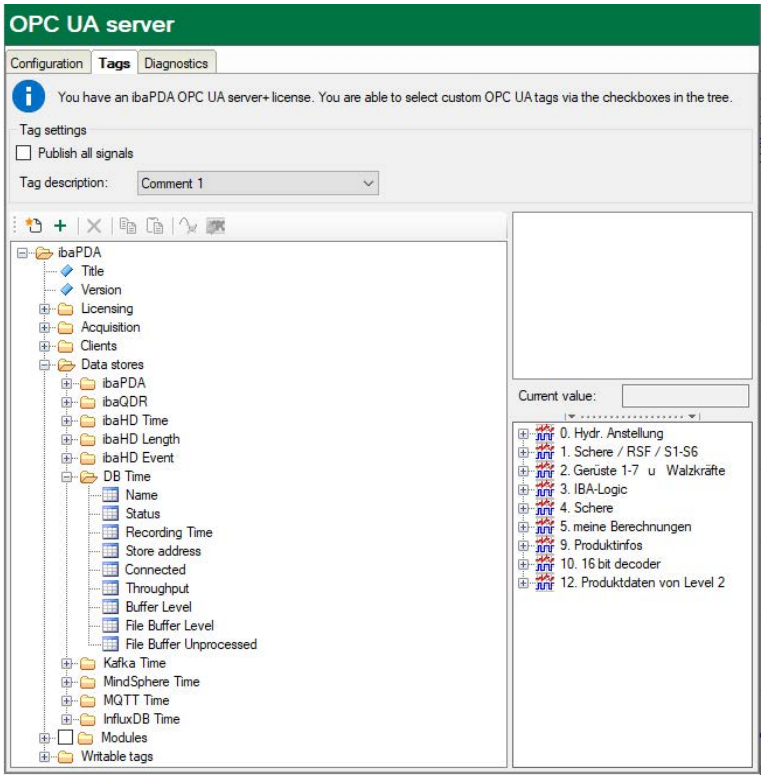
The *Acquisition Thread load* column indicates various information depending on the data stores. For timebased data stores, the *Acquisition Thread load* column indicates the amount of time needed for the run length encoding and writing to a disk. For DB/Cloud, MQTT, Kafka Cluster, InfluxDB and MindSphere data stores, the column indicates the load caused by the analysis of the triggers and creation of the row data.

For HD data stores, the partial processing time will be displayed, that is used for the creation of the data to be written on the HD server. These values already contain the run length encoding for timebased stores, event trigger calculation for event stores and the calculation of the length-based data for lengthbased stores.

Additional information about diagnostics can be found in the *ibaPDA* manual, part 5.

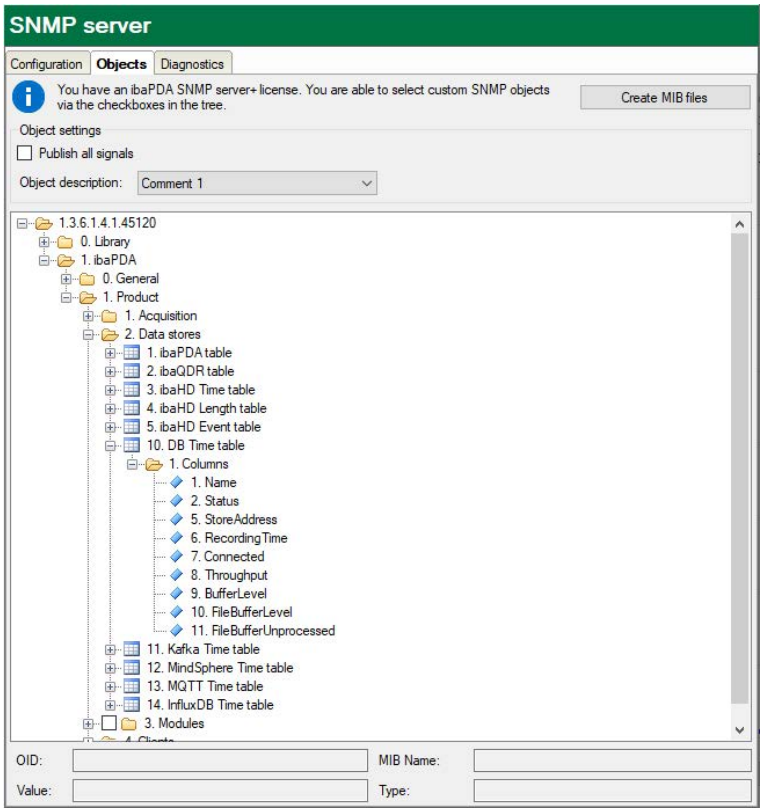
6.3 OPC UA Server

The *ibaPDA* internal OPC UA server provides diagnostic data regarding the data stores. Diagnostic data can always be used without having an additional OPC UA server license.



6.4 SNMP

The *ibaPDA* internal SNMP server provides diagnostic data regarding the data stores. Diagnostic data can always be used without having an additional SNMP server license.



6.5 Virtual functions

Specific virtual functions generate diagnostic data related to the different data stores as signal data for further processing within *ibaPDA*.

- DataStoreInfoDB
- DataStoreInfoInflux
- DataStoreInfoKafka
- DataStoreInfoMindSphere
- DataStoreInfoMQTT

Example for diagnostic function DB/Cloud data store

DataStoreInfoDB('DatastoreIndex*', 'InfoType*')

Diese Funktion liefert Informationen über die ausgewählte DB/Cloud-Datenaufzeichnung.
'DatastoreIndex' >= 0

Die folgenden Informationstypen werden unterstützt:
0: Aufzeichnungsstatus:
0=Angehalten
1=Warten auf Trigger
2=Aufzeichnung läuft
3=Aufzeichnung im Triggerlauf
1: Datendurchsatz in kB/s
2: Ist der Server verbunden?
3: Aufzeichnungsdauer seit dem letzten Starttrigger in Sekunden. Dies ist konstant 0 bei kontinuierlicher Aufzeichnung.
5: Aktuelle Puffernutzung (in %)
6: Aktuelle Dateipuffernutzung (in %)
7: Unverarbeitete Bytes im Dateipuffer (in %)

Parameters ending with * are only evaluated once at the start of the acquisition.

Note

Additional information about the functions can be found in the manual *ibaPDA*, part 4.

7 Support and contact

Support

Phone: +49 911 97282-14

Fax: +49 911 97282-33

Email: support@iba-ag.com

Note

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

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Phone: +49 911 97282-0

Fax: +49 911 97282-33

Email: iba@iba-ag.com

Mailing address

iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address

iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

Regional and Worldwide

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

www.iba-ag.com.