ibaPDA-Data-Store-MQTT

Data streaming into MQTT broker

Manual
Issue 1.6

Measurement Systems for Industry and Energy
www.iba-ag.com

Manufacturer

iba AG

Koenigswarterstrasse 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

Version	Date	Revision	Author	Version SW
1.6	06-2023	MQTT Sparkplug B	st	8.3.0

Content

1 About this manual

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

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 can 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-MQTT* 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>
Filenames, paths	`Filename`, `Path` Example: `Test.docx`

1.3 Used symbols

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

Danger!

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

■ Observe the specified measures.

Warning!

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

■ Observe the specified measures.

Caution!

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

■ Observe the specified measures

Note

A note specifies special requirements or actions to be observed.

Tip

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

Other documentation

Reference to additional documentation or further reading.

2 Introduction

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

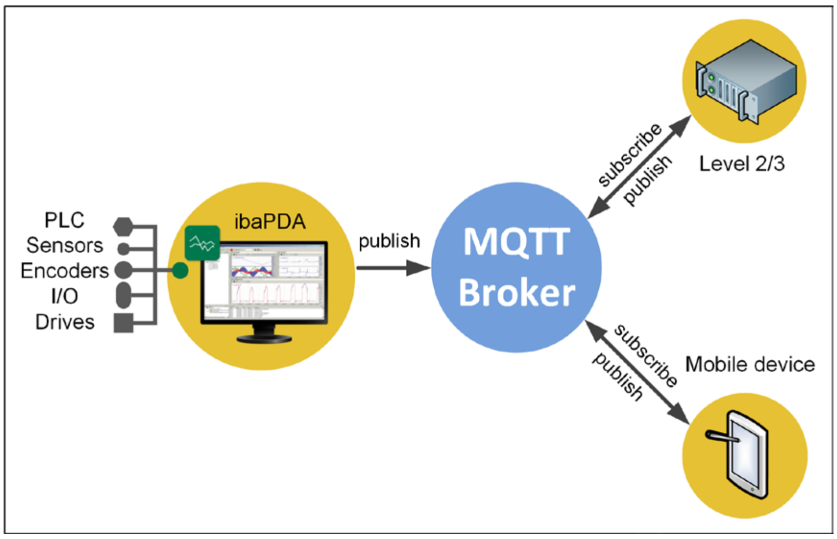
This documentation describes these type of recording:

- "MQTT timebased data store" to stream unformatted data
- "MQTT Sparkplug B data store" to stream Sparkplug B formatted data

This type of data store sends timebased signal data to an MQTT broker. The broker publishes these values to clients who have subscribed to these signals.

MQTT is based on an event-driven publish/subscribe architecture. The core is a central server (broker) to which the transmitter as well as the receiver connect. The data is sent (published) and received (subscribed) via so-called topics. Topics are basically communication channels in which the transmitters, e.g. sensors, write their data. The broker checks which receivers (clients) have opened a channel for this data and passes on the data to the clients.

This means for *ibaPDA* that *ibaPDA* acts as MQTT client. *ibaPDA* sends (publishes) topics with measured values to the MQTT broker, which distributes these further.



Chapter ↗ *Signal selection*, page 23 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 25.

Note

The MQTT broker is not part of *ibaPDA*. This must be purchased, installed and configured separately. The transmission behavior is decisively influenced by the configuration and the performance of the broker.

2.1 System requirements

The following system requirements are necessary when using MQTT timebased data store:

- *ibaPDA* v8.3.0 or higher
- License for *ibaPDA-Data-Store-MQTT*

With this license, both the *MQTT timebased data store* and the *MQTT Sparkplug B data store* can be used.

The licenses are staggered according to the number of signals that should be sent to an MQTT broker. The number of used data stores is unlimited.

Order No.	Name	Description
30.671000	ibaPDA-Data-Store-MQTT-16	Data streaming into MQTT broker, 16 signals
30.671001	ibaPDA-Data-Store-MQTT-64	Data streaming into MQTT broker, 64 signals
30.671002	ibaPDA-Data-Store-MQTT-256	Data streaming into MQTT broker, 256 signals
30.671003	ibaPDA-Data-Store-MQTT-1024	Data streaming into MQTT broker, 1024 signals
30.671010	upgrade-ibaPDA-Data-Store-MQTT-16 to 64	Signal extension 16 signals to 64 signals
30.671011	upgrade-ibaPDA-Data-Store-MQTT-64 to 256	Signal extension 64 signals to 256 signals
30.671012	upgrade-ibaPDA-Data-Store-MQTT-256 to 1024	Signal extension 256 signals to 1024 signals

3 Data store configuration

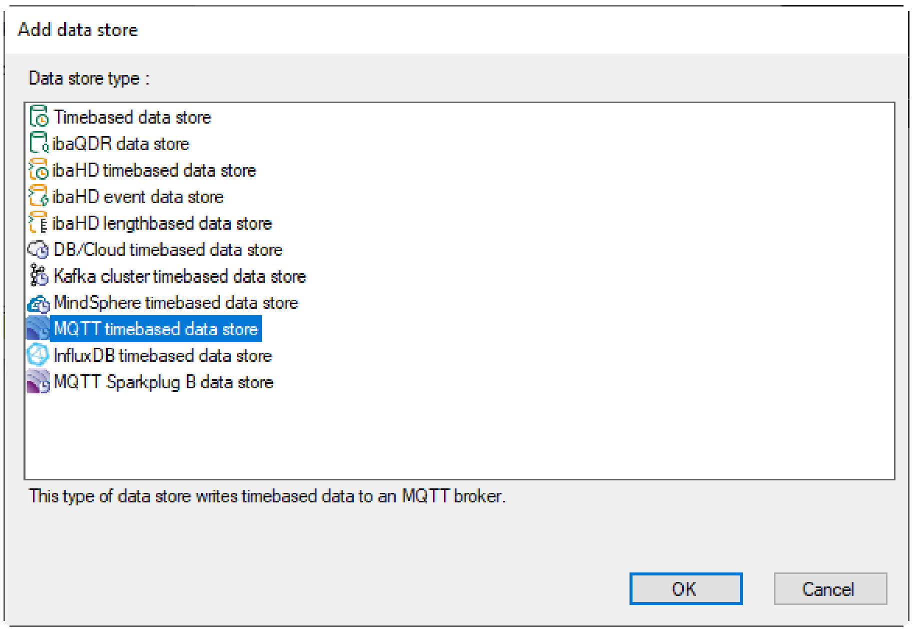
3.1 Add a data store

Open the dialog for data storage configuration 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

- *MQTT timebased data store* to stream timebased data to an MQTT broker.
- *MQTT Sparkplug B data store* to stream Sparkplug B formatted data to an MQTT broker.



Add data store

Data store type :

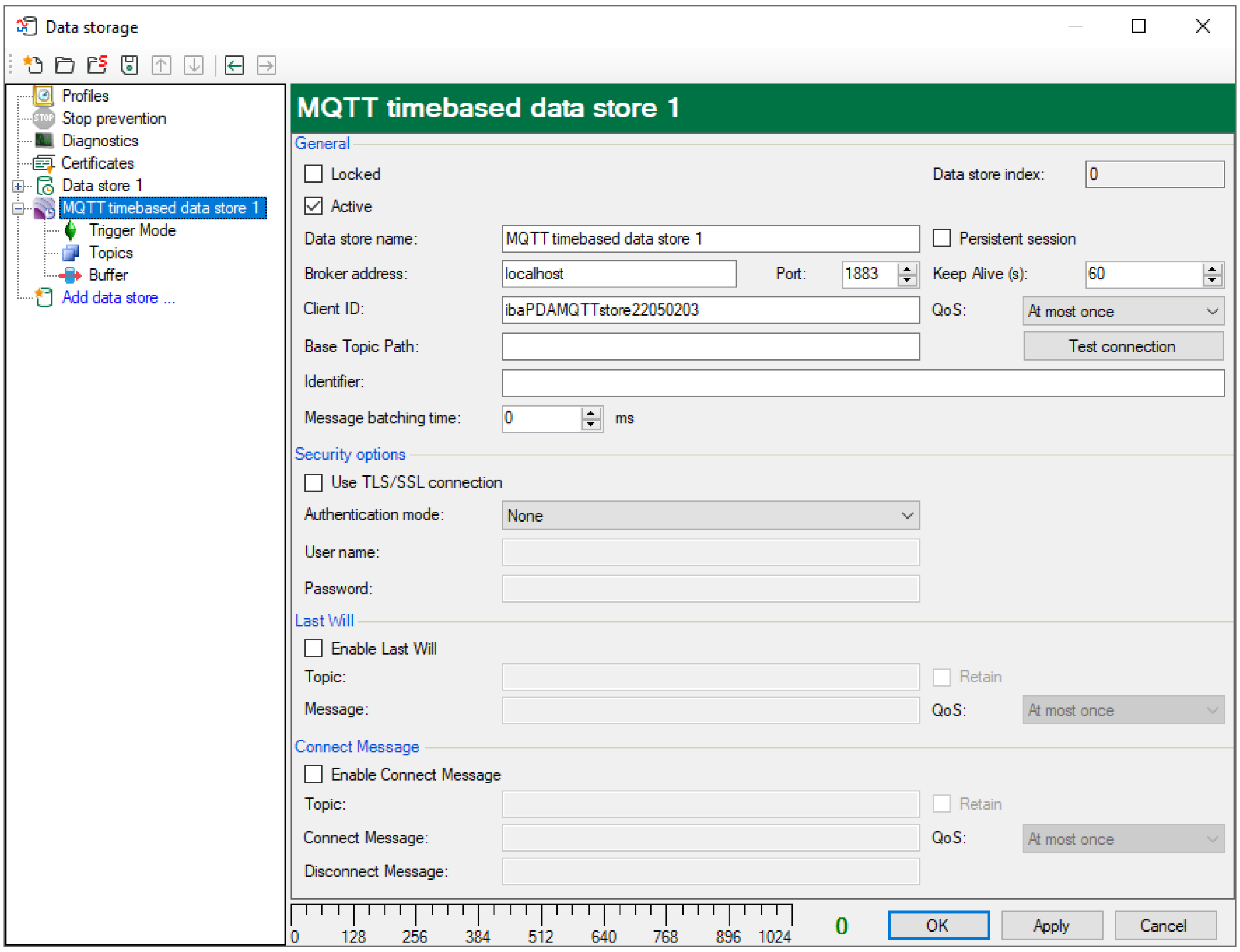
- Timebased data store
- ibaQDR data store
- ibaHD timebased data store
- ibaHD event data store
- ibaHD lengthbased data store
- DB/Cloud timebased data store
- Kafka cluster timebased data store
- MindSphere timebased data store
- MQTT timebased data store
- InfluxDB timebased data store
- MQTT Sparkplug B data store

This type of data store writes timebased data to an MQTT broker.

3.2 MQTT timebased data store

This chapter describes the configuration of the *MQTT timebased data store*. The topics are configured in the *Topics* subnode, see chapter ↗ *Configuring topics*, page 14.

The configuration of the *MQTT Sparkplug B data store* is described in chapter ↗ *MQTT Sparkplug B data store*, page 17.



MQTT timebased data store 1

General

Locked	☐
Active	☑
Data store index:	0
Data store name:	MQTT timebased data store 1
Persistent session	☐
Broker address:	localhost
Port:	1883
Keep Alive (s):	60
Client ID:	ibaPDAMQTTstore22050203
QoS:	At most once
Base Topic Path:	
Identifier:	
Message batching time:	0 ms

Security options

Use TLS/SSL connection	☐
Authentication mode:	None
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:	

General

Locked
You can lock a data store in order to prevent an accidental or unauthorized change of settings.

Active
Activate the data storage in order to record data. However, you can configure various data stores and disable data stores that are not required.

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

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

Broker address
Enter the address of the broker here. The address can be an IP address, a host name or a URL.

Port
Port that is to be used for the connection. The default MQTT port is 1883, or 8883 when using SSL.

Client ID
When connecting to a broker, each client must choose a unique name that is used only once for this broker. The client ID is pre-set by *ibaPDA* automatically.

Base Topic Path
This path is added in front of each registered topic (optional setting). The topic configuration can be made easier if all topics reside within a common path.

Note: Do not use a leading forward slash "/" in a topic, because this would add an empty path entry to the topic. Two separators next to each other ("//") are also not allowed.

Identifier
The identifier is a text-based value that can be included in the MQTT messages sent to the broker. For the subsequent processing of data, this value may be useful for distinguishing between several *ibaPDA* systems that write to the same broker.

Persistent session
If the client is disconnected, it receives the last values sent to the broker when reconnecting with the same client ID as before. For this purpose, the QoS must be set to "Exactly once" or "At least once" not only for *ibaPDA*, but also for the sender, which delivers the values to the broker.

Keep Alive
The time for sending a *Keep Alive* telegram to the broker to make sure that the connection is still online.

QoS
Quality of Service, which is used when registering to a topic on the broker. The possible values are:

- At most once (messages can be dropped)
- At least once (messages are repeated if an acknowledgement is not sent within a certain time)
- Exactly once (a secured handshake protocol for each message sent)

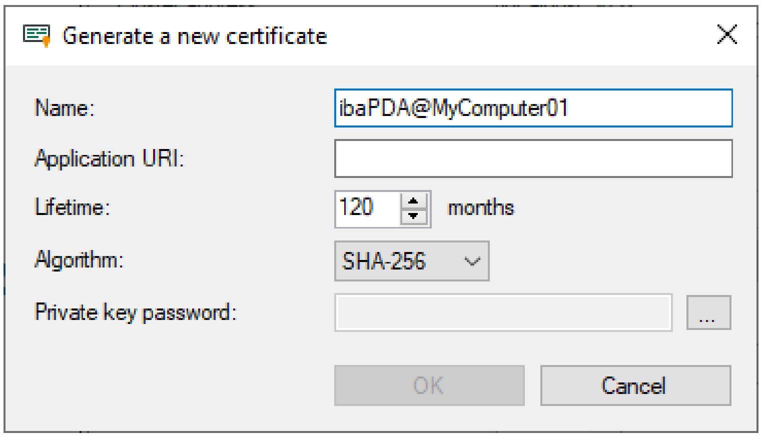
<Test connection>
Click on the button <Test connection> to verify if *ibaPDA* can establish a connection to the MQTT broker using the selected security settings.

Security options

Use TSL/SSL connection
Use TSL or SSL instead of plain TCP connection. The port must be set to 8883 by default. This option has to be configured accordingly on the broker, otherwise the connection will fail.

Authentication mode
The following are available:

- *None*: without authentication
- *Login*: Enter the user name and password as login when connecting to the broker. This option has to be configured accordingly on the broker, otherwise the connection will fail.
- *Certificate*: The certificates that can be used are available for selection in the drop-down menu. In addition more options can be selected:
  - *No certificate*: No certificate is used. However, this usually causes the validation to fail.
  - *Manage certificates*: Opens the central certificate store, where certificates can be managed centrally in *ibaPDA*.
  - *Create new certificate*: *ibaPDA* creates a new self-signed certificate. Enter the necessary settings in the opening dialog. When the certificate is successfully created, the new certificate is selected.


Generate a new certificate

Name:	ibaPDA@MyComputer01
Application URI:	
Lifetime:	120 months
Algorithm:	SHA-256
Private key password:	...

Enter a name for the certificate. You can change the default name.

Entering an Application URI is optional. Set the lifetime and select the algorithm. Available for selection are SHA-256, SHA-384 and SHA-512.

You have to assign a password in order to create a certificate. To enter the password for the private key, click the <…> button. For security reasons, you must enter the password twice in the following dialog. The password can be assigned arbitrarily, it does not have to meet any other requirements.

Last Will

Enable Last Will
If this option is enabled, the absence of this client is announced to other clients, if the connection breaks without regular disconnect.

Topic
Enter a path to the topic used for the last will message. The base topic path is not applied to this topic.

Message
Enter a message which should be sent in the text field.

Retain
If this option is enabled, the message in this topic is stored on the broker until it is overwritten.

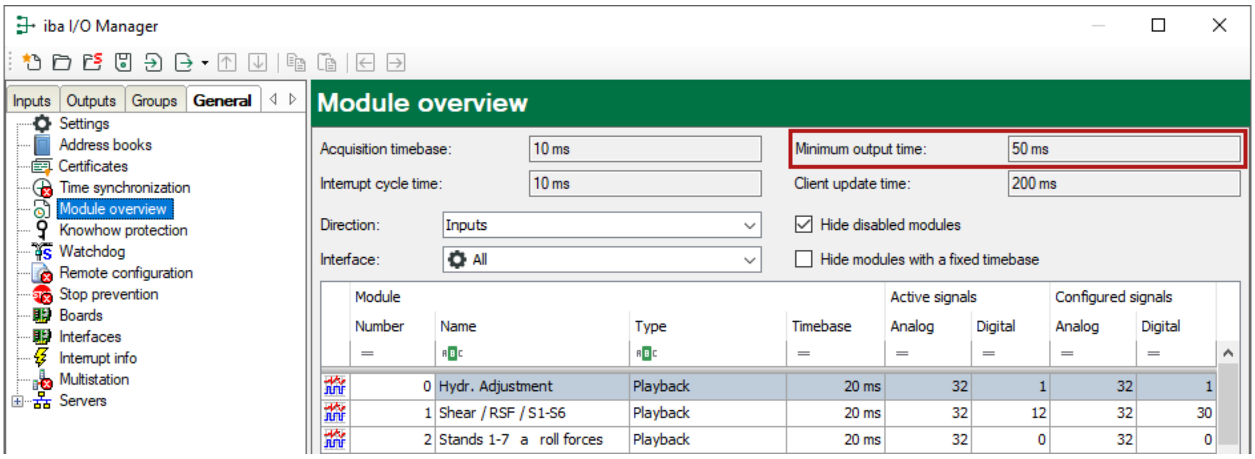
QoS
The *Quality of Service* used for sending the last will message to other clients. The *QoS* settings are as described above.

Connect Message

Connect Message has the same options as *Last Will*, but the *Connect Messages* are sent on a regular connect or disconnect.

3.2.1 Time behavior

The processing of the data written via the MQTT data store, is cyclical. The duration of the cycle corresponds to the "minimum output time".



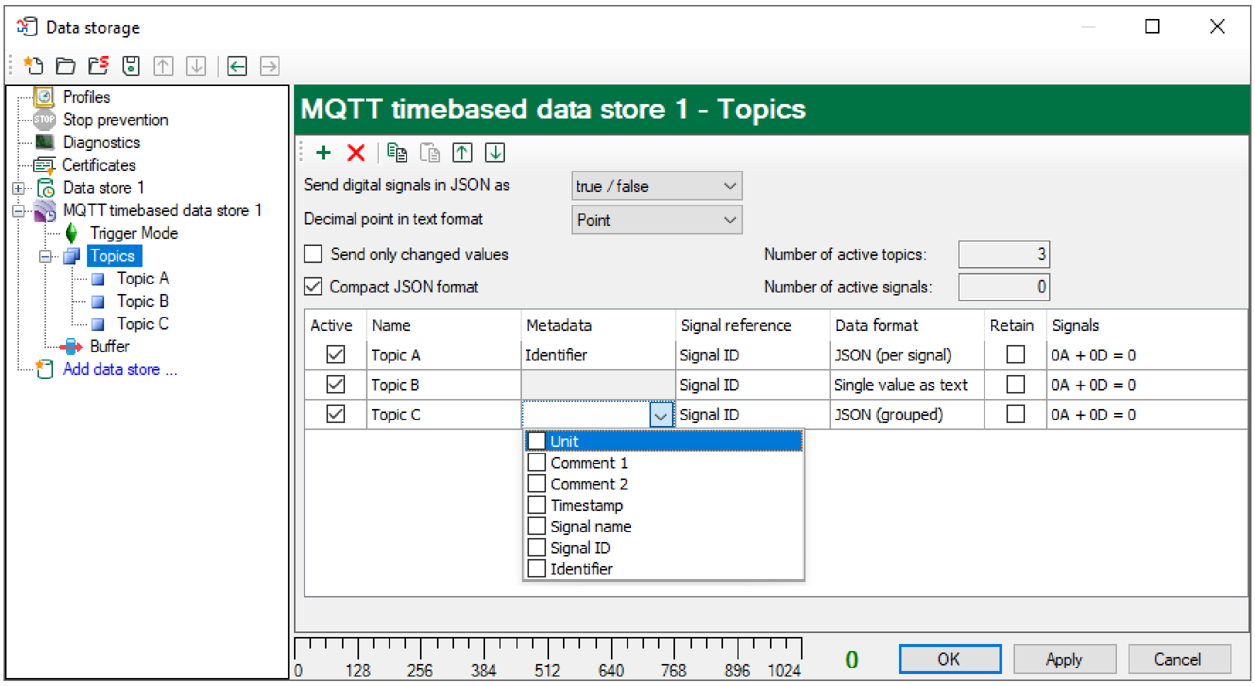
Module overview

Acquisition timebase: 10 ms
Interrupt cycle time: 10 ms
Minimum output time: 50 ms
Client update time: 200 ms
Direction: Inputs
Interface: All
☑ Hide disabled modules
☐ Hide modules with a fixed timebase

Module				Active signals		Configured signals	
Number	Name	Type	Timebase	Analog	Digital	Analog	Digital
0	Hydr. Adjustment	Playback	20 ms	32	1	32	1
1	Shear / RSF / S1-S6	Playback	20 ms	32	12	32	30
2	Stands 1-7 a roll forces	Playback	20 ms	32	0	32	0

As a result, signal samples that are published faster than the minimum output time are sent bundled in the cycle of the minimum output time.

3.2.2 Configuring topics



Data storage

- Profiles
- Stop prevention
- Diagnostics
- Certificates
- Data store 1
- MQTT timebased data store 1
  - Trigger Mode
  - Topics
    - Topic A
    - Topic B
    - Topic C
  - Buffer
- Add data store ...

MQTT timebased data store 1 - Topics

Send digital signals in JSON as: true / false

Decimal point in text format: Point

☐ Send only changed values — Number of active topics: 3

☑ Compact JSON format — Number of active signals: 0

Active	Name	Metadata	Signal reference	Data format	Retain	Signals
☑	Topic A	Identifier	Signal ID	JSON (per signal)	☐	0A + 0D = 0
☑	Topic B		Signal ID	Single value as text	☐	0A + 0D = 0
☑	Topic C		Signal ID	JSON (grouped)	☐	0A + 0D = 0

☐ Unit
☐ Comment 1
☐ Comment 2
☐ Timestamp
☐ Signal name
☐ Signal ID
☐ Identifier

If you highlight the *Topics* node, you can define the topics that are sent to the broker. Different formats are available, depending on the information required for subscribers.

Buttons for the configuration of the topics:

＋	Manually add a new topic
✗	Remove selected topic
⧉	Copy cell content to the clipboard
📋	Paste data from the clipboard into cells
↑	Move selected topic up
↓	Move selected topic down

Note

Use <Shift> and <Ctrl> to select multiple cells and copy/paste/delete them.

If you select JSON as data format, you can set whether digital signals are sent as 1/0 or true/false.

You can reduce the amount of data to be transmitted if you enable the *Send only changed values* option.

The *Compact JSON format* option also reduces the amount of data transmitted and removes filler characters, such as multiple spaces, line breaks, etc., from the telegram.

The fields to the right show the number of active topics and active signals.

For the text format, a period or comma can be selected as floating point separator.

Meaning of the columns:

Active
Here you enable/disable writing to a topic.

Name
Enter an unambiguous name for the topic here.

Metadata
If you chose one of the JSON formats as the data format for the topic, you can choose metadata in the drop-down menu that is saved with the topic.

Available for selection are: Unit, comment 1 + 2, timestamp, signal name, signal ID, identifier. Select the desired metadata with a check mark.

Signal reference
In the drop-down menu, select whether the signal ID or the signal name should be used as a signal reference.

Data format
Choose a data format in the drop-down menu:

- Single value as text: A single signal is written under this topic into the broker and formatted as plain text.
- Single value as binary: A single signal is written under this topic into the broker and saved in binary format.
- JSON (per signal): An individual signal is written under this topic as a JSON-formatted line into the broker and provided with optional metadata.
- JSON (grouped): Several signals are written under this topic as a JSON-formatted line into the broker and provided with optional metadata.
- Protobuf (grouped): Multiple values are written into a block, starting with a timestamp. The time base is defined by the profile used. Optionally, metadata can be selected. The metadata timestamp and identifier cannot be selected because they are a fixed part of the protobuf definition.

  The Protobuf definition file, which can be used for decoding the values, is installed with the *ibaPDA* server under:
  `C:\Program Files (x86)\iba\ibaPDA\Server\AuxiliaryFiles\Protobuf\ibaPDA_data.proto`

Examples for the description of the data formats JSON (per signal) and JSON (grouped) can be found in ➚ *Description of the data formats*, page 30.

Combine values (ms)
For JSON and Protobuf data formats, multiple values can be combined over a period of time in one message.

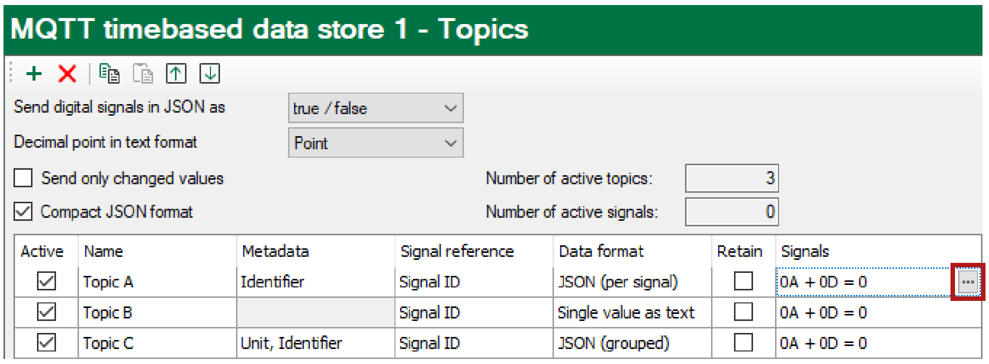
If you enter a value greater than 0 in the *Combine Values (ms)* column, data from that time period is written to the message. A value of 0 disables this function.

Retain

If you enable this option, the messages in this topic are stored on the broker until they are overwritten.

Signals

In order to configure the signals that are to be written in a topic to configure, select the topic in the structure tree or click on the <...> button in the *Signals* column.



MQTT timebased data store 1 - Topics

Send digital signals in JSON as: true / false

Decimal point in text format: Point

☐ Send only changed values

☑ Compact JSON format

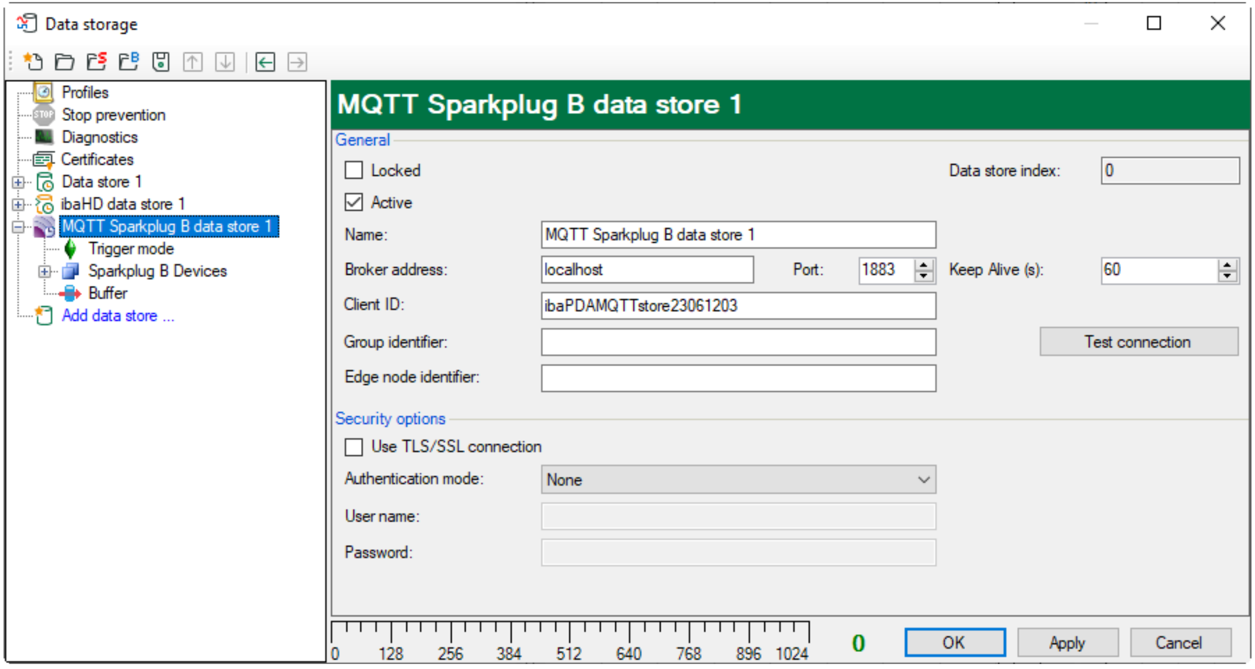
Number of active topics: 3

Number of active signals: 0

Active	Name	Metadata	Signal reference	Data format	Retain	Signals
☑	Topic A	Identifier	Signal ID	JSON (per signal)	☐	0A + 0D = 0
☑	Topic B		Signal ID	Single value as text	☐	0A + 0D = 0
☑	Topic C	Unit, Identifier	Signal ID	JSON (grouped)	☐	0A + 0D = 0

In the following dialog, you assign the desired signals to the *Topics* using the storage profiles. See chapter ↗ *Signal selection*, page 23.

3.3 MQTT Sparkplug B data store



☐ Locked
☑ Active
Data store index: 0
Name: MQTT Sparkplug B data store 1
Broker address: localhost
Port: 1883
Keep Alive (s): 60
Client ID: ibaPDAMQTTstore23061203
Group identifier:
Edge node identifier:
☐ Use TLS/SSL connection
Authentication mode: None
User name:
Password:

The basic settings correspond to those of the MQTT timebased data store, see chapter ↗ *MQTT timebased data store*, page 10:

Locked, active, name of the data store, broker address, port, client ID, security options.

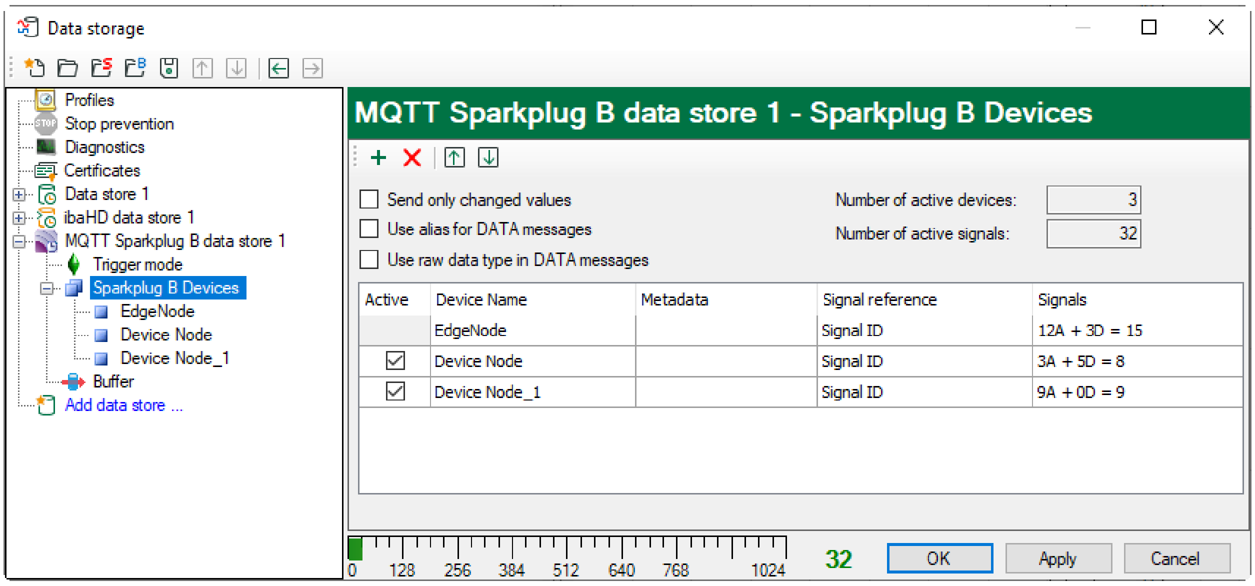
The following settings specifically apply to MQTT Sparkplug B data store:

Group identifier
The group identifier according to the Sparkplug B specification.

Edge node identifier
ibaPDA acts as an edge node, as defined in the Sparkplug B specification. Enter here the required edge node name.

The data path for the topic with the Sparkplug B message is defined as follows:
“spBv1.0/<Group identifier>/<message type>/<Edge node identifier>/

3.3.1 Configuring Sparkplug B devices



☐ Send only changed values
☐ Use alias for DATA messages
☐ Use raw data type in DATA messages

Number of active devices: 3
Number of active signals: 32

Active	Device Name	Metadata	Signal reference	Signals
	EdgeNode		Signal ID	12A + 3D = 15
✓	Device Node		Signal ID	3A + 5D = 8
✓	Device Node_1		Signal ID	9A + 0D = 9

In the *Sparkplug B devices* node, the signals transmitted in the DATA messages can be selected for the Edge Node and the other devices.

Buttons for the configuration of the devices:

+	Manually add a new device
✕	Remove selected device
↑	Move selected device up
↓	Move selected device down

Send only changed values
Each NDATA/DDATA message contains only the signals that have changed when this option is enabled.

Use alias for DATA messages
Instead of the signal reference, the alias number is used in NDATA/DDATA messages. The NBIRTH/DBIRTH messages contain the signal reference together with the alias.

Use raw data type in DATA messages.
Usually *ibaPDA* writes signal values using float or double value formatting. By enabling this option, the original value format is used (e.g. INT16 or DWORD). This can be useful for bitmapped values. But by using this option, the signal values will disregard any scale/offset option selected.

Meaning of the columns:

Active
Here you enable/disable writing a message.

Device Name
Enter an unambiguous name for the device here. The base name *EdgeNode* cannot be changed.

Metadata
You can choose metadata in the drop-down menu that is to be sent with the message.

Available for selection are: Unit, comment 1 + 2, timestamp, signal name, signal ID. Select the desired metadata with a check mark.

Signal reference
Select what to use as a signal reference from the drop-down menu. Available for selection are signal name, signal ID or signal comment 1 or 2.

Signals
In order to configure the signals that are to be written, select the edge node or the device in the structure tree or click on the <...> button in the *Signals* column.

In the following dialog, you assign the desired signals to the *devices* using the storage profiles. See chapter ➚ *Signal selection*, page 23.

3.4 Buffer

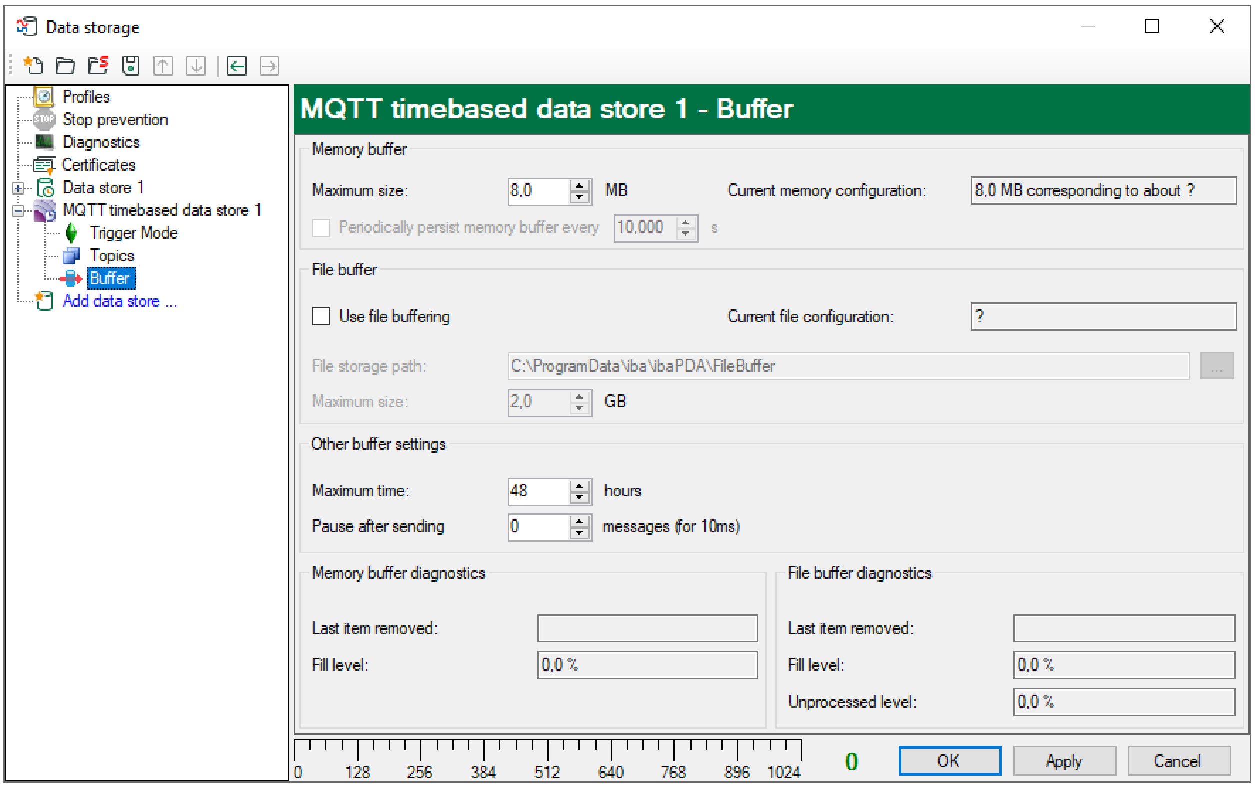
The data storage uses a memory buffer and additionally a file buffer that can be enabled optionally.

Data to be sent to the target system always passes through the internal *ibaPDA* memory buffer. If the connection to the target system exists, the data from the memory buffer is sent there 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.



MQTT timebased data store 1 - Buffer

Memory buffer

Maximum size: 8,0 MB

Current memory configuration: 8,0 MB corresponding to about ?

Periodically persist memory buffer every 10.000 s

File buffer

Use file buffering

Current file configuration: ?

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

Maximum size: 2,0 GB

Other buffer settings

Maximum time: 48 hours

Pause after sending 0 messages (for 10ms)

Memory buffer diagnostics

Last item removed:

Fill level: 0,0 %

File buffer diagnostics

Last item removed:

Fill level: 0,0 %

Unprocessed level: 0,0 %

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.

Pause after sending … messages (for 10 ms)
ibaPDA pauses sending the telegrams after the number of telegrams set here is sent to the broker without interruption. This should avoid possible problems of the broker with a continuous data stream. Regarding the number, take into account that each topic is treated as a separate telegram. Set the value to 0 to disable the feature.

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.

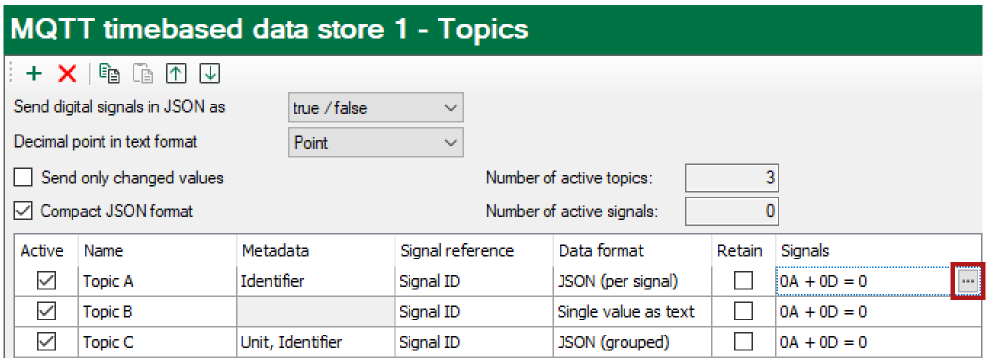
Unprocessed level
Items transferred to the target system are not deleted immediately in the file buffer. Only when a buffer file is completely processed, it is deleted. Therefore, it is possible that only a part of a buffer file contains data that has not yet been transferred. The fill level refers to 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 record signals, you have to assign the signals to a topic by using a storage profile of the type *Time*.

Signal selection for MQTT timebased data store

Either click in the *Topics* node, in the *Signals* column of the topic list on the <...> button to access the signal selection dialog.



MQTT timebased data store 1 - Topics

Send digital signals in JSON as: true / false

Decimal point in text format: Point

☐ Send only changed values

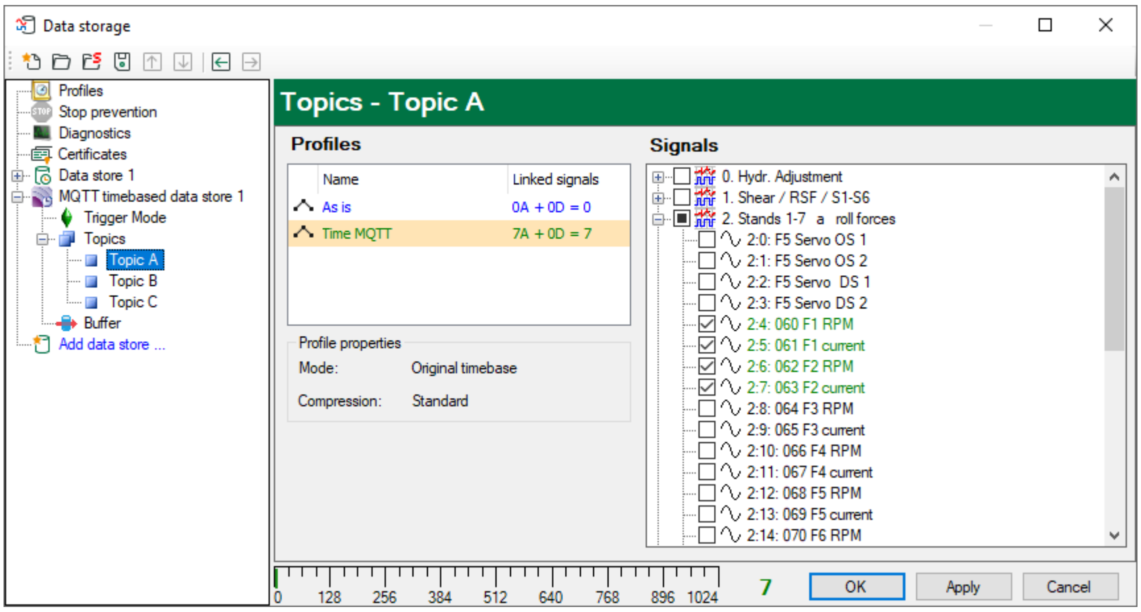
☑ Compact JSON format

Number of active topics: 3

Number of active signals: 0

Active	Name	Metadata	Signal reference	Data format	Retain	Signals
☑	Topic A	Identifier	Signal ID	JSON (per signal)	☐	0A + 0D = 0 ...
☑	Topic B		Signal ID	Single value as text	☐	0A + 0D = 0
☑	Topic C	Unit, Identifier	Signal ID	JSON (grouped)	☐	0A + 0D = 0

Or select a topic in the tree structure.



Data storage

Profiles
Stop prevention
Diagnostics
Certificates
Data store 1
MQTT timebased data store 1
Trigger Mode
Topics
Topic A
Topic B
Topic C
Buffer
Add data store ...

Topics - Topic A

Profiles

Name	Linked signals
As is	0A + 0D = 0
Time MQTT	7A + 0D = 7

Profile properties

Mode: Original timebase

Compression: Standard

Signals

☐ 0. Hydr. Adjustment
☐ 1. Shear / RSF / S1-S6
■ 2. Stands 1-7 a roll forces
☐ 2:0: F5 Servo OS 1
☐ 2:1: F5 Servo OS 2
☐ 2:2: F5 Servo DS 1
☐ 2:3: F5 Servo DS 2
☑ 2:4: 060 F1 RPM
☑ 2:5: 061 F1 current
☑ 2:6: 062 F2 RPM
☑ 2:7: 063 F2 current
☐ 2:8: 064 F3 RPM
☐ 2:9: 065 F3 current
☐ 2:10: 066 F4 RPM
☐ 2:11: 067 F4 current
☐ 2:12: 068 F5 RPM
☐ 2:13: 069 F5 current
☐ 2:14: 070 F6 RPM

Note

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

Select the topic to which you would like to assign certain signals and select a storage profile in the profile list. Set a check mark in the selection fields next to the signals which you would like to assign to this profile.

The *Profile properties* section displays some information about the configured timebase and filtering of the selected profile.

For the single value formats, only the first signal is used for writing if more than one signal is selected. All other signals are ignored.

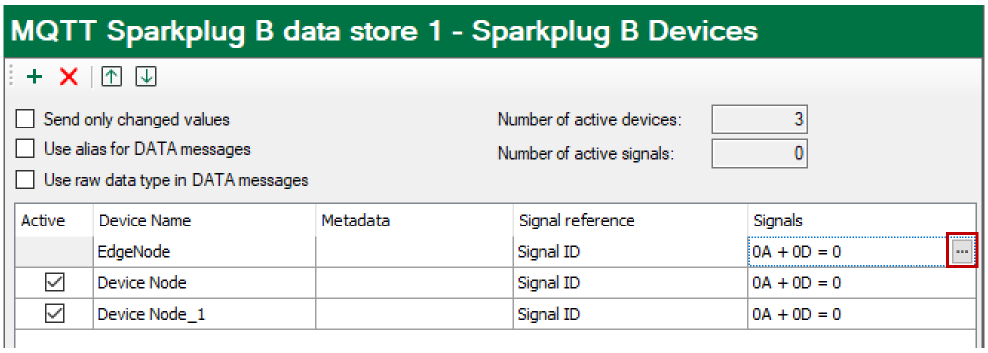
MQTT data stores are licensed per number of written signals. You can find the current number of selected signals in all MQTT data stores at the bottom of the dialog, similar to the number of configured signals in the I/O Manager. The length of the bar corresponds to the licensed number of signals.

In the example above, a maximum of 1024 signals can be written via MQTT data stores. Currently 7 signals are enabled.

Signal selection for MQTT Sparkplug B data store

The procedure for *MQTT Sparkplug B data store* is basically identical to the *MQTT timebased data store*. The difference is that the signals are selected in the Edge Node or Devices nodes.

Click in the *Sparkplug B devices* node, in the *Signals* column of the device list on the <...> button to access the signal selection dialog.



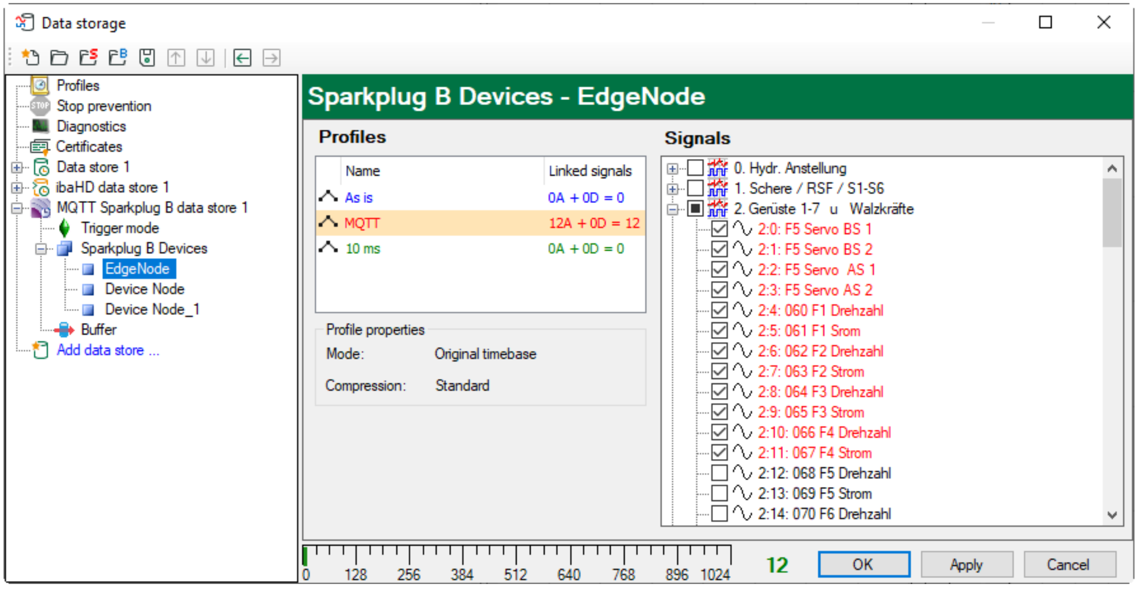
MQTT Sparkplug B data store 1 - Sparkplug B Devices

☐ Send only changed values
☐ Use alias for DATA messages
☐ Use raw data type in DATA messages

Number of active devices: 3
Number of active signals: 0

Active	Device Name	Metadata	Signal reference	Signals
	EdgeNode		Signal ID	0A + 0D = 0
✓	Device Node		Signal ID	0A + 0D = 0
✓	Device Node_1		Signal ID	0A + 0D = 0

Or select the Edge Node or a device in the tree structure.



Sparkplug B Devices - EdgeNode

Profiles

Name	Linked signals
As is	0A + 0D = 0
MQTT	12A + 0D = 12
10 ms	0A + 0D = 0

Profile properties
Mode: Original timebase
Compression: Standard

Signals

☐ 0. Hydr. Anstellung
☐ 1. Schere / RSF / S1-S6
■ 2. Gerüste 1-7 u Walzkräfte
☑ 2:0: F5 Servo BS 1
☑ 2:1: F5 Servo BS 2
☑ 2:2: F5 Servo AS 1
☑ 2:3: F5 Servo AS 2
☑ 2:4: 060 F1 Drehzahl
☑ 2:5: 061 F1 Srom
☑ 2:6: 062 F2 Drehzahl
☑ 2:7: 063 F2 Strom
☑ 2:8: 064 F3 Drehzahl
☑ 2:9: 065 F3 Strom
☑ 2:10: 066 F4 Drehzahl
☑ 2:11: 067 F4 Strom
☐ 2:12: 068 F5 Drehzahl
☐ 2:13: 069 F5 Strom
☐ 2:14: 070 F6 Drehzahl

0 128 256 384 512 640 768 896 1024 12

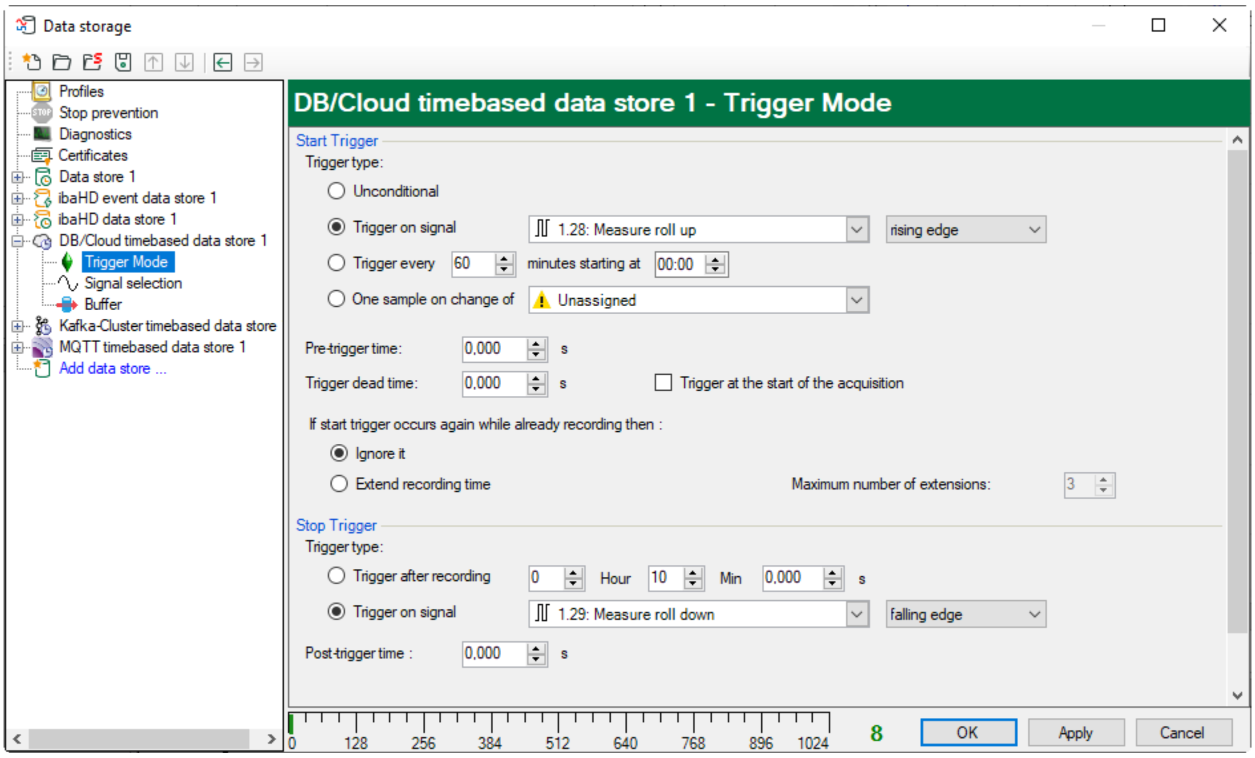
Select a storage profile of type *Time* as described above and select the signals in the signal tree.

5 Trigger mode

The description applies to the following types of data stores:

- DB/Cloud timebased
- Kafka cluster timebased
- MQTT timebased and MQTT Sparkplug B
- 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*.



Data storage

Profiles
Stop prevention
Diagnostics
Certificates
Data store 1
ibaHD event data store 1
ibaHD data store 1
DB/Cloud timebased data store 1
Trigger Mode
Signal selection
Buffer
Kafka-Cluster timebased data store
MQTT timebased data store 1
Add data store ...

DB/Cloud timebased data store 1 - Trigger Mode

Start Trigger
Trigger type:
Unconditional
[✓ Trigger on signal] 1.28: Measure roll up — rising edge
Trigger every 60 minutes starting at 00:00
One sample on change of: Unassigned

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

Stop Trigger
Trigger type:
Trigger after recording 0 Hour 10 Min 0,000 s
[✓ Trigger on signal] 1.29: Measure roll down — falling edge

Post-trigger time : 0,000 s

Start trigger

You initially choose whether you want to record continuously or initiated by a trigger.

Unconditional

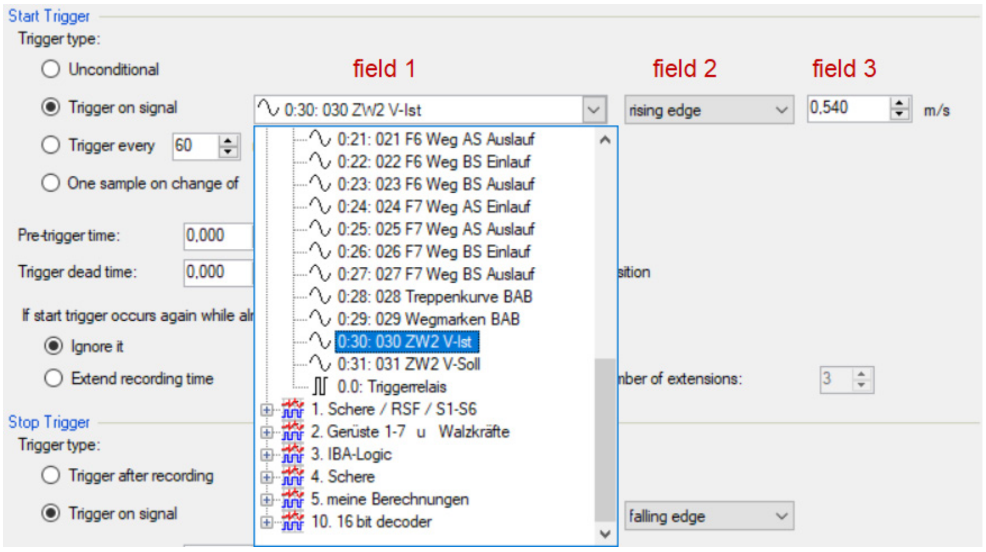
Select this option to record data continuously. The recording starts immediately at the start of the measurement or when clicking the <GO> button.

Trigger on signal

If you want to trigger the recording on a measured signal or a virtual signal, select the *Trigger on signal* option. 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

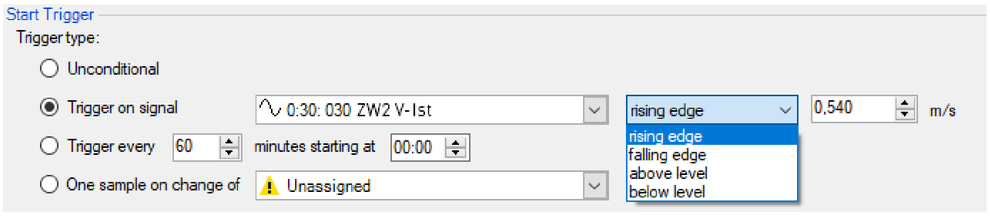
- Field 3: Drop-down list for selecting the trigger level value given in the specific physical unit (field 3 is only enabled in case of analog trigger signals)



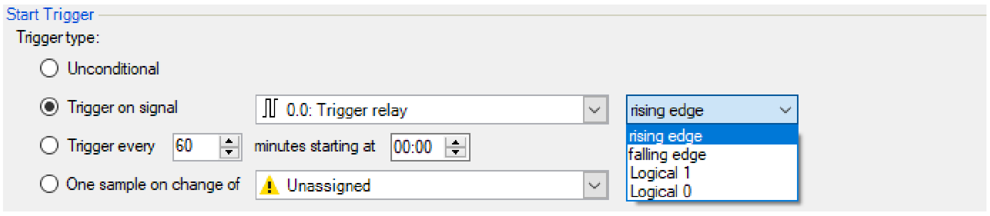
You can use analog signals and digital signals as triggers. Select the signal to be triggered via the signal tree in the selection list in field 1.

In fields 2 and 3, you can define the trigger event more precisely. These fields vary depending on whether the selected measuring channel is analog or digital.

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



- For digital signals, you can choose between level triggers 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, select 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 stops after one sample, until the next signal change is detected. A deadtime can be configured to deter-

mine a minimum amount of time between samples. Before the deadtime has elapsed, no new sample is 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, select the *Trigger on acquisition start* option. If you do not enable this option, the recording only starts when 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:
  If you select this option, the system ignores any new start trigger during a running recording for as long as the stop trigger occurs
- Extend recording time:
  If you select this option, the system 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" field. If the max. number of extensions is reached, all subsequent start triggers are 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.

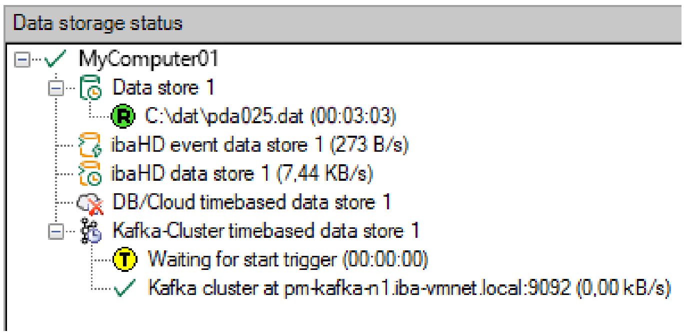
Trigger on signal
See explanation for start trigger above.

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.



Data storage status

MyComputer01
Data store 1
C:\dat\pda025.dat (00:03:03)
ibaHD event data store 1 (273 B/s)
ibaHD data store 1 (7,44 KB/s)
DB/Cloud timebased data store 1
Kafka-Cluster timebased data store 1
Waiting for start trigger (00:00:00)
Kafka cluster at pm-kafka-n1.iba-vmnet.local:9092 (0,00 kB/s)

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:

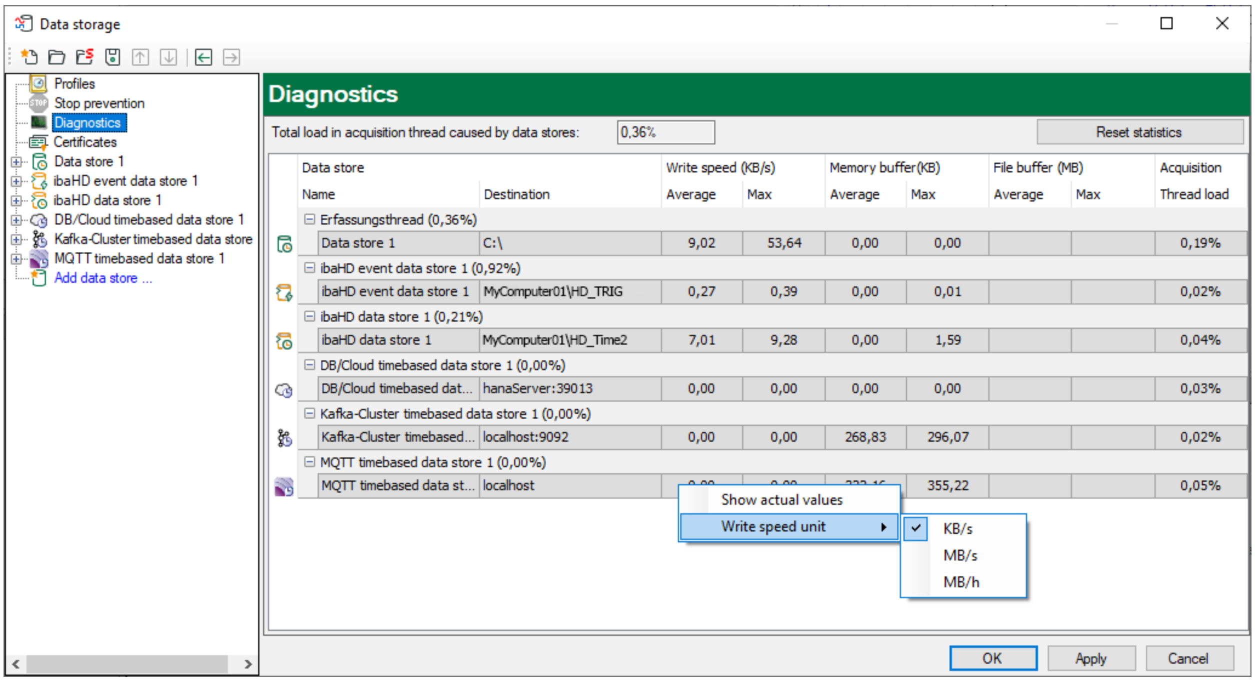
T	Wait for the start trigger (only for triggered recording)
R	Recording in progress
P	Post-trigger phase; stop trigger occurred, but acquisition continues until the post-trigger time is over

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.



Total load in acquisition thread caused by data stores: 0,36%

Data store		Write speed (KB/s)		Memory buffer(KB)		File buffer (MB)		Acquisition
Name	Destination	Average	Max	Average	Max	Average	Max	Thread load
Erfassungsthread (0,36%)								
Data store 1	C:\	9,02	53,64	0,00	0,00			0,19%
ibaHD event data store 1 (0,92%)								
ibaHD event data store 1	MyComputer01\HD_TRIG	0,27	0,39	0,00	0,01			0,02%
ibaHD data store 1 (0,21%)								
ibaHD data store 1	MyComputer01\HD_Time2	7,01	9,28	0,00	1,59			0,04%
DB/Cloud timebased data store 1 (0,00%)								
DB/Cloud timebased dat...	hanaServer:39013	0,00	0,00	0,00	0,00			0,03%
Kafka-Cluster timebased data store 1 (0,00%)								
Kafka-Cluster timebased...	localhost:9092	0,00	0,00	268,83	296,07			0,02%
MQTT timebased data store 1 (0,00%)								
MQTT timebased data st...	localhost	[illegible]	[illegible]	[illegible]	355,22			0,05%

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) the time utilization (100% means the thread is overloaded). The load average is displayed by default. But you can switch between the average and actual value using the context menu.

The *Destination* 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 is 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.

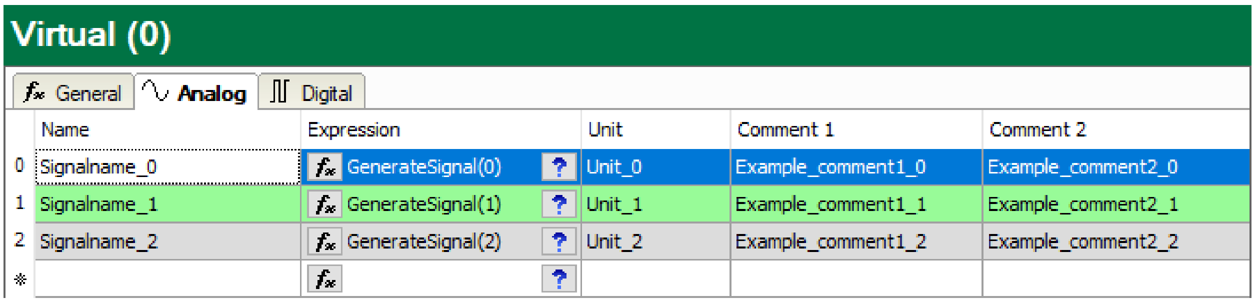
Additional information about diagnostics can be found in the *ibaPDA* manual, part 5.

7 Appendix

7.1 Description of the data formats

7.1.1 Data format JSON (grouped)

The following explains the setup of the data format "JSON (grouped)" using an example with three signals. The signals are defined as follows:



	Name	Expression	Unit	Comment 1	Comment 2
0	Signalname_0	GenerateSignal(0)	Unit_0	Example_comment1_0	Example_comment2_0
1	Signalname_1	GenerateSignal(1)	Unit_1	Example_comment1_1	Example_comment2_1
2	Signalname_2	GenerateSignal(2)	Unit_2	Example_comment1_2	Example_comment2_2
*					

JSON example for signal ID as a signal reference

```
{
  "[0:0]": Actual value,
  "[0:0].ID": "[0:0]",
  "[0:0].Name": "Signalname_0",
  "[0:0].Unit": "Unit_0",
  "[0:0].Comment1": "Example_comment1_0",
  "[0:0].Comment2": "Example_comment2_0",
  "[0:1]": Actual value,
  "[0:1].ID": "[0:1]",
  "[0:1].Name": "Signalname_1",
  "[0:1].Unit": "Unit_1",
  "[0:1].Comment1": "Example_comment1_1",
  "[0:1].Comment2": "Example_comment2_1",
  "[0:2]": Actual value,
  "[0:2].ID": "[0:2]",
  "[0:2].Name": "Signalname_2",
  "[0:2].Unit": "Unit_2",
  "[0:2].Comment1": "Example_comment1_2",
  "[0:2].Comment2": "Example_comment2_2",
  "Timestamp": "2020-01-21T13:10:53.0002189Z",
  "Identifier": "My identifier"
}
```

Red: optional signal-related metadata
Green: optional group-related metadata

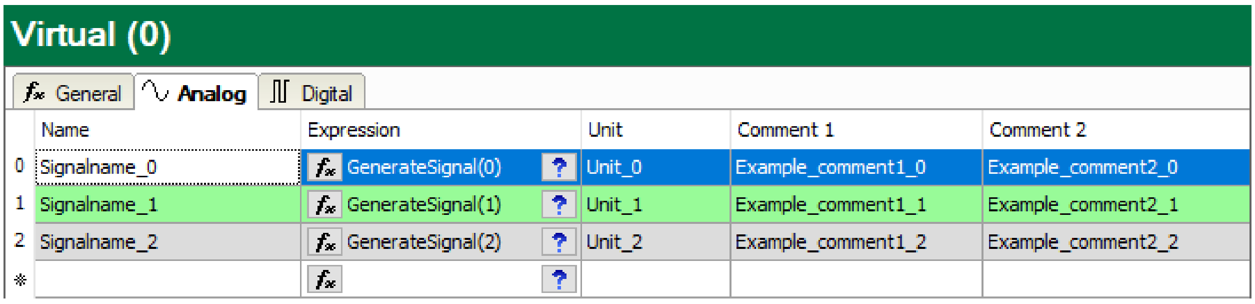
JSON example for signal name as a signal reference

```
{
  "Signalname_0": Actual value,
  "Signalname_0.ID": "[0:0]",
  "Signalname_0.Name": "Signalname_0",
  "Signalname_0.Unit": "Unit_0",
  "Signalname_0.Comment1": "Example_comment1_0",
  "Signalname_0.Comment2": "Example_comment2_0",
  "Signalname_1": Actual value,
  "Signalname_1.ID": "[0:1]",
  "Signalname_1.Name": "Signalname_1",
  "Signalname_1.Unit": "Unit_1",
  "Signalname_1.Comment1": "Example_comment1_1",
  "Signalname_1.Comment2": "Example_comment2_1",
  "Signalname_2": Actual value,
  "Signalname_2.ID": "[0:2]",
  "Signalname_2.Name": "Signalname_2",
  "Signalname_2.Unit": "Unit_2",
  "Signalname_2.Comment1": "Example_comment1_2",
  "Signalname_2.Comment2": "Example_comment2_2",
  "Timestamp": "2020-01-21T13:20:13.0009119Z",
  "Identifier": "My identifier"
}
```

Red: optional signal-related metadata
Green: optional group-related metadata

7.1.2 Data format JSON (per signal)

The following explains the setup of the data format “JSON (per signal)” using an example with three signals. The signals are defined as follows:



	Name	Expression	Unit	Comment 1	Comment 2
0	Signalname_0	GenerateSignal(0)	Unit_0	Example_comment1_0	Example_comment2_0
1	Signalname_1	GenerateSignal(1)	Unit_1	Example_comment1_1	Example_comment2_1
2	Signalname_2	GenerateSignal(2)	Unit_2	Example_comment1_2	Example_comment2_2
*					

JSON example for signal ID as a signal reference

```
{
  "Signal": "[0:0]",
  "Value": Actual value,
  "ID": "[0:0]",
  "Name": "Signalname_0",
  "Unit": "Unit_0",
  "Comment1": "Example_comment1_0",
  "Comment2": "Example_comment2_0",
  "Timestamp": "2020-01-21T13:26:50.8784074Z",
  "Identifier": "My identifier"
}

{
  "Signal": "[0:1]",
  "Value": Actual value,
  "ID": "[0:1]",
  "Name": "Signalname_1",
  "Unit": "Unit_1",
  "Comment1": "Example_comment1_1",
  "Comment2": "Example_comment2_1",
  "Timestamp": "2020-01-21T13:26:50.8784074Z",
  "Identifier": "My identifier"
}

{
  "Signal": "[0:2]",
  "Value": Actual value,
  "ID": "[0:2]",
  "Name": "Signalname_2",
  "Unit": "Unit_2",
  "Comment1": "Example_comment1_2",
  "Comment2": "Example_comment2_2",
  "Timestamp": "2020-01-21T13:26:50.8784074Z",
  "Identifier": "My identifier"
}
```

Red: optional signal-related metadata

JSON example for signal name as a signal reference

```
{
  "Signal": "Signalname_0",
  "Value": Actual value,
  "ID": "[0:0]",
  "Name": "Signalname_0",
  "Unit": "Unit_0",
  "Comment1": "Example_comment1_0",
  "Comment2": "Example_comment2_0",
  "Timestamp": "2020-01-21T13:36:37.5310016Z",
  "Identifier": "My identifier"
}

{
  "Signal": "Signalname_1",
  "Value": Actual value,
  "ID": "[0:1]",
  "Name": "Signalname_1",
  "Unit": "Unit_1",
  "Comment1": "Example_comment1_1",
  "Comment2": "Example_comment2_1",
  "Timestamp": "2020-01-21T13:36:37.5310016Z",
  "Identifier": "My identifier"
}

{
  "Signal": "Signalname_2",
  "Value": Actual value,
  "ID": "[0:2]",
  "Name": "Signalname_2",
  "Unit": "Unit_2",
  "Comment1": "Example_comment1_2",
  "Comment2": "Example_comment2_2",
  "Timestamp": "2020-01-21T13:36:37.5310016Z",
  "Identifier": "My identifier"
}
```

Red: optional signal-related metadata

8 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 number of the license container. For hardware products, please have the serial number of the device ready.

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Phone:	+49 911 97282-0
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.