



ibaPDA-Data-Store-Kafka

Data streaming to Apache Kafka cluster

Manual
Issue 1.4

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

Unless explicitly stated to the contrary, it is not permitted to pass on or copy this document, nor to make use of its contents or disclose its contents. Infringements are liable for compensation.

© iba AG 2025, All rights reserved.

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	Author	Version SW
1.4	03-2025	Changes for Event hub	st, mm	8.10.0

Windows® is a brand and registered trademark of Microsoft Corporation. Other product and company names mentioned in this manual can be labels or registered trademarks of the corresponding owners.

Contents

1	About this documentation	4
1.1	Target group and previous knowledge	4
1.2	Notations	5
1.3	Used symbols.....	6
2	Introduction	7
2.1	System requirements.....	8
3	Data store configuration.....	9
3.1	Adding a data store.....	9
3.2	Kafka cluster data store	10
3.2.1	Type Kafka.....	11
3.2.2	Type Event Hub.....	16
3.3	Buffer	17
3.4	Configuration of topics	20
4	Signal selection	23
5	Trigger mode.....	25
6	Diagnostics.....	29
6.1	Data storage status	29
6.2	Diagnostics of data stores.....	30
7	Appendix	31
7.1	Description of the data formats.....	31
7.1.1	Data format JSON (grouped).....	31
7.1.2	Data format JSON (per signal)	33
7.1.3	Data format AVRO (per signal).....	35
8	Support and contact.....	37

1 About this documentation

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

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

This documentation in particular addresses persons, who are concerned with the configuration, test, commissioning or maintenance of the supported database, cloud or cluster storage technology. For the handling of *ibaPDA-Data-Store-Kafka* 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 <i>Logic diagram</i>
Calling the menu command	<i>Step 1 – Step 2 – Step 3 – Step x</i> Example: Select the menu <i>Logic diagram – Add – New function block</i> .
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	<i>Filename, Path</i> Example: <i>Test.docx</i>

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

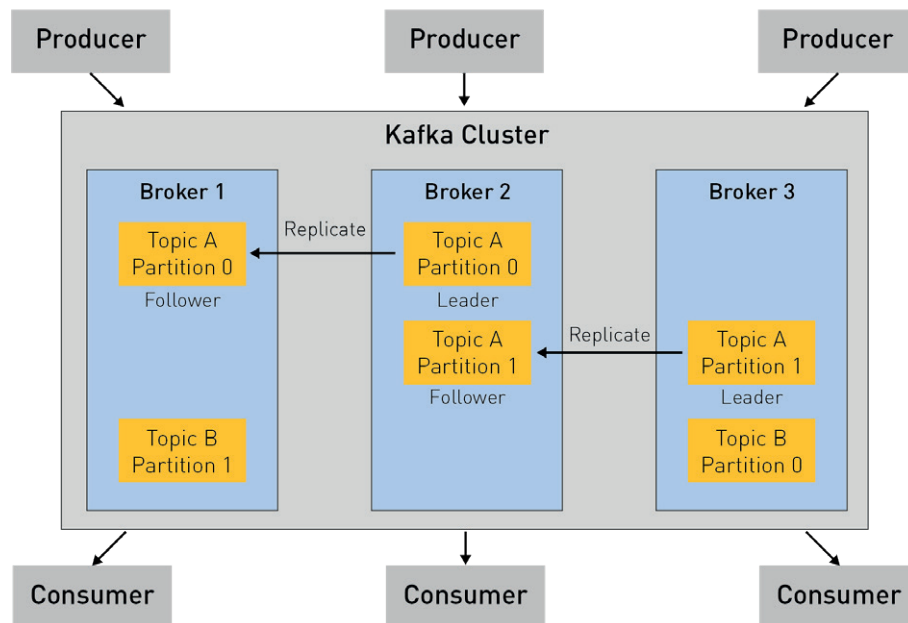
Different types of data stores are available in *ibaPDA* for different purposes and methods of data storage. Depending on the licenses registered in the license container, different types of data stores are available for configuration in the dialog.

This documentation describes the "Kafka cluster timebased data store" type of recording. This type of data store writes timebased data into a Kafka cluster.

The architecture of Apache Kafka consists of a cluster computer network. So-called 'brokers' in this computer network store received messages with a time stamp in so-called 'topics.'

Topics can be split into several partitions in the cluster and replicated. Applications that write messages in a Kafka cluster are called producers. The messages are sent to the leader. Among themselves the brokers take care of the replication with the followers. Consumers are applications that read messages from the Kafka cluster.

ibaPDA has the role of a producer in a Kafka environment: The data store sends messages to the cluster.



Additional information on Apache Kafka can be found at

- <https://kafka.apache.org/>
- <https://www.confluent.io/what-is-apache-kafka>

Chapter ↗ *Signal selection*, page 23 describes the selection of the signals that are to be recorded in the topics.

The data can be continuously recorded or recorded by trigger, see chapter ↗ *Trigger mode*, page 25.

2.1 System requirements

The following system requirements are necessary when using data storage in an Apache Kafka cluster:

- *ibaPDA* v8.0.0 or higher
- License for *ibaPDA-Data-Store-Kafka*

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


Order no.	Product name	Description
30.670160	ibaPDA-Data-Store-Kafka-16	Additional license for data streaming into a Kafka cluster, max. 16 signals
30.670161	ibaPDA-Data-Store-Kafka-64	Additional license for data streaming into a Kafka cluster, max. 64 signals
30.670162	ibaPDA-Data-Store-Kafka-256	Additional license for data streaming into a Kafka cluster, max. 256 signals
30.670163	ibaPDA-Data-Store-Kafka-1024	Additional license for data streaming into a Kafka cluster, max. 1024 signals
30.670171	upgrade-ibaPDA-Data-Store-Kafka-16 to 64	License for extension from 16 to 64 signals
30.670172	upgrade-ibaPDA-Data-Store-Kafka-64 to 256	License for extension from 64 to 256 signals
30.670173	upgrade-ibaPDA-Data-Store-Kafka-256 to 1024	License for extension from 256 to 1024 signals

Table 1: Available licenses for the Kafka cluster data storage

3 Data store configuration

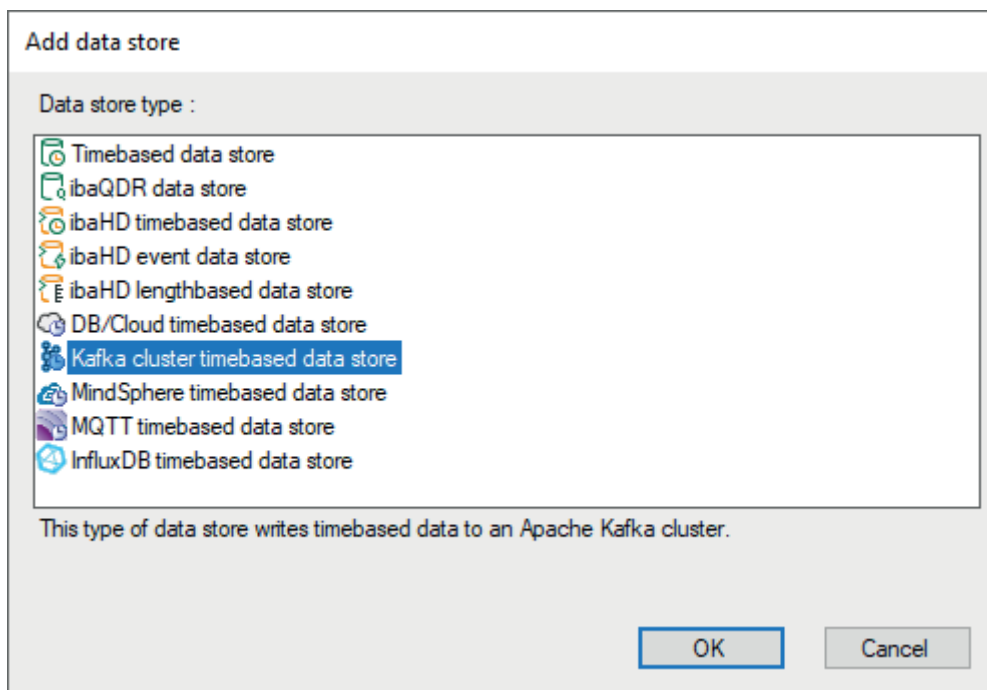
In the following, you will learn how to configure a data store.

3.1 Adding 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.

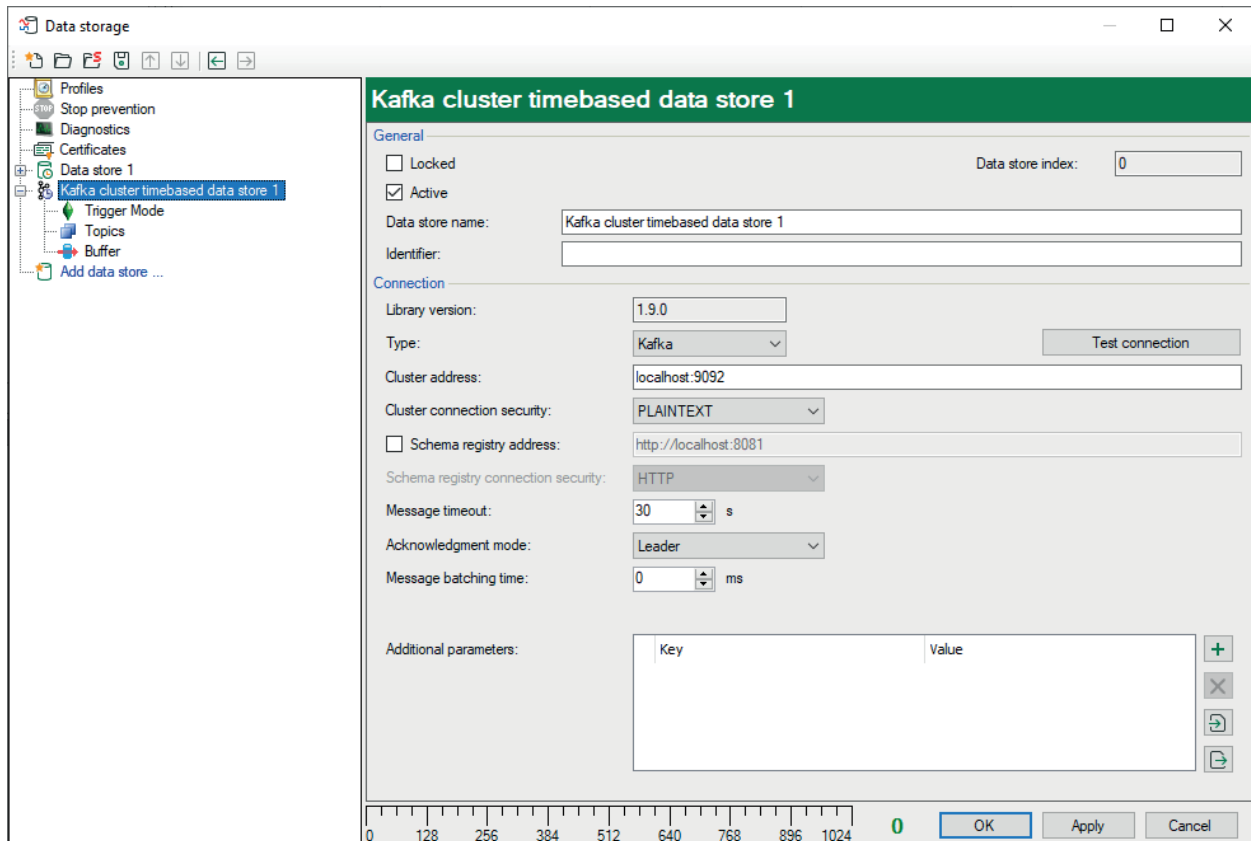
In order to add a new data store, click on the blue link *Add data store* in the tree structure. You can also right-click on the data store node in the tree structure and choose *Add data store* from the context menu.

Select *Kafka cluster timebased data store* for the recording of timebased data in an Apache Kafka cluster.



3.2 Kafka cluster data store

This chapter describes the configuration of the *Kafka cluster timebased data store*. The topics are configured in the *Topics* subnode, see chapter [↗ Configuration of topics](#), page 20.



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 Kafka cluster data stores. You need to reference this index e.g. in the virtual function *DataStoreInfoKafka()* for generating diagnostic data for a specific Kafka cluster data store.

Data store name

You can enter a name for the data store here.

Identifier

The identifier is a text-based value that can be included in the datasets written in the cluster. For the subsequent processing of data, this value may be useful for distinguishing between several *ibaPDA* systems that write into the same cluster.

Connection

Library Version

Display of the library version contained in *ibaPDA* (only for information). *ibaPDA* uses the open-source library *librdkafka*.

Type

There are two types to choose from:

- **Kafka:** Generic Kafka cluster, see chapter [Type Kafka](#), page 11
- **Event Hub:** for the connection with a Microsoft Azure event hub with enabled Kafka support, see chapter [Type Event Hub](#), page 16

3.2.1 Type Kafka

The following settings are required for the **Kafka** type:

Cluster address

Enter the host name and port of one of the brokers in the Kafka cluster here. In case you want to address multiple brokers at the same time, separate the single entries by commas (e.g. 'hostname1:9092, hostname2:9092, hostname3:9092').

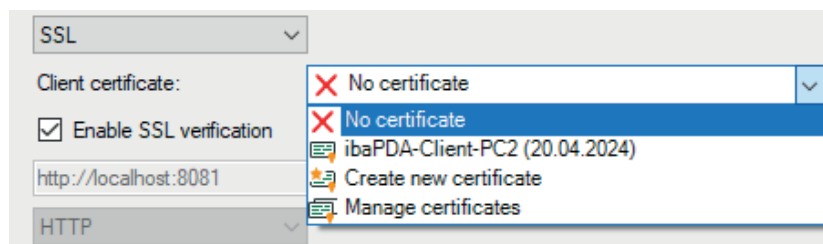
Cluster connection security

Different encryption and authentication methods are supported. Depending on the selected option, further entries are required.

■ PLAINTEXT

Data is exchanged in plain text, no further entries

■ SSL

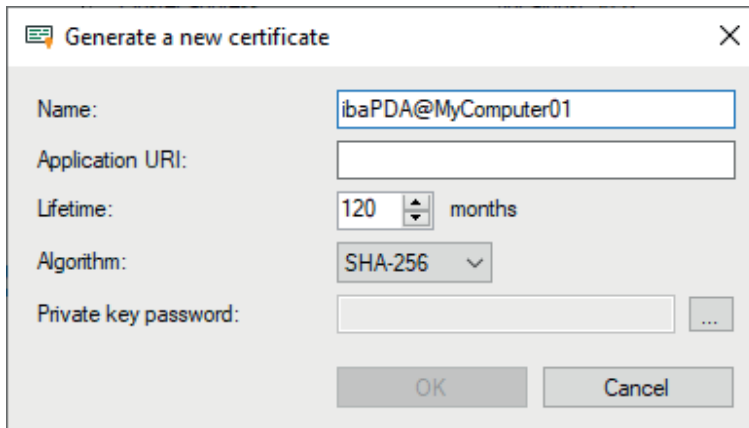


The SSL communication requires certificates. You can store and manage certificates in the central certificate store in *ibaPDA*. Detailed information about the certificate store can be found in the *ibaPDA* manual.

The certificates that can be used are available for selection in the *Client certificate* 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.



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.

Enable SSL verification

If you enable the SSL verification, the certificate of the Kafka server is verified automatically. Among the certificates in *ibaPDA*, the issuer certificate with which the Kafka server certificate was signed must be available.

Note



Using chained SSL certificates

In *ibaPDA*, you can also import chained certificates. When importing (e.g. PEM file or PFX file), the chained certificates are splitted into single certificates. After the import of a chained certificate, there are therefore several individual entries under the certificates.

If the Kafka server certificate was signed using a chain of issuer certificates and you want to use the SSL verification at the same time, the configuration of the keystore in the Kafka server is important: If the keystore also contains the complete chain of issuer certificates in addition to the Kafka server certificate, then only the certificate of the root CA must be available in *ibaPDA*. If the keystore contains only the Kafka server certificate, then in *ibaPDA* the complete chain of issuer certificates must be available, so that the verification is possible.

■ SASL/PLAINTEXT

First of all, select the SASL authentication mechanism:

- PLAIN (all data is exchanged in plain text)
- SCRAM-SHA-256
- SCRAM-SHA-512

Enter the username and the password in the corresponding fields for logging in to the Kafka cluster.

■ SASL/SSL

As with *SASL/PLAINTEXT*, select the SASL mechanism and enter the username and password. The certificates that can be used are available for selection in the *Client certificate* drop-down menu.

For the activation and use of the SSL verification, the instructions from the section on cluster connection security with SSL.

Schema registry address

Enable this option if a schema registry is used in the Kafka cluster. Enter the host name and port of the schema registry. Depending on the connection security used, the address must start with "http://" or "https://". By default, this option is disabled.

Schema registry connection security

ibaPDA supports different methods for connection security. Depending on the selected option, further entries are required.

■ HTTP

No further entry required

■ HTTPS



Schema registry connection security: HTTPS ▼

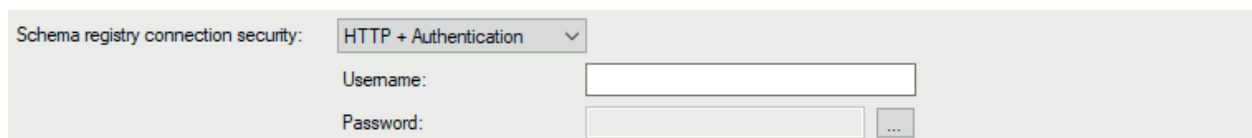
Client certificate: ✗ No certificate ▼

☒ Enable SSL verification

The HTTPS communication requires certificates. The certificates that can be used are available for selection in the *Client certificate* drop-down menu.

The selection of a certificate is done as described above in the description for cluster connection security with *SSL*.

■ HTTP + Authentication



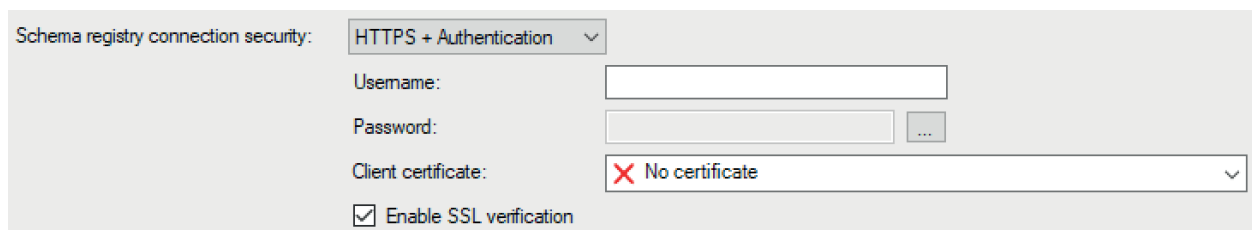
Schema registry connection security: HTTP + Authentication ▼

Username:

Password: ...

This method requires the input of username and password.

■ HTTPS + Authentication



Schema registry connection security: HTTPS + Authentication ▼

Username:

Password: ...

Client certificate: ✗ No certificate ▼

☒ Enable SSL verification

Like HTTPS, additionally the input of username and password is required.

Message timeout

Time in seconds that *ibaPDA* waits for a response from the Kafka cluster. The value corresponds to the parameter *message.timeout.ms* in the library *librdkafka*.

Acknowledgment mode

- *None*: Data messages are sent as a continuous stream, but without confirmation of whether the Leader received them or not. This is the fastest mode, but it is not guaranteed that the broker has acquired the data.
- *Leader*: *ibaPDA* waits until the topic leader confirms the acquisition of the sent data. If the leader malfunctions after sending the confirmation and before the followers have replicated the dataset, the data is lost.
- *All*: *ibaPDA* does not send new data until the topic leader and the followers have confirmed the respective recording of the data.

The value corresponds to the parameter *acks* in the library *librdkafka*.

Message Batching Time

Time in milliseconds that *ibaPDA* waits to send messages. If the value is 0, messages are sent as quickly as possible to the Kafka cluster. If the value is set to 100 ms, for example, packets with buffered messages are sent every 100 ms. This increases the latency time, but reduces the processing effort both on the *ibaPDA* side and on the cluster side.





The value corresponds to the parameter *linger.ms* in the library *librdkafka*.

Additional parameters





You can configure additional parameters of the *librdkafka* library.

Additional parameters:

Key	Value
<input type="text"/>	<input type="text"/>

Parameters that have already been created are listed in the list with the name and value. Next to the list, there are buttons with the following features:

	Add parameters Enter the name and a value in the fields <i>Parameter</i> and <i>Value</i> .
	Delete selected parameters
	Import parameters You can import parameters as CSV files by selecting the CSV file in the file browser.
	Export parameters You can export to a CSV file. Enter a file name and select a folder.

Note



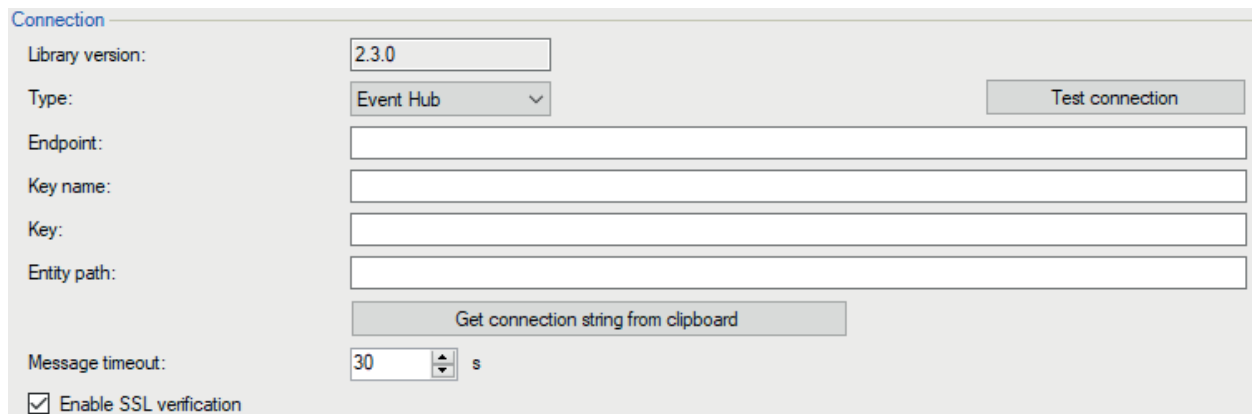
Configured parameters in the list always have priority and overrule other settings.

The documentation of the parameters available in the *librdkafka* library can be found at:

<https://github.com/edenhill/librdkafka/blob/master/CONFIGURATION.md>

3.2.2 Type Event Hub

The following settings are required for the **Event Hub** type:



The screenshot shows a configuration window titled "Connection". It contains the following fields and controls:

- Library version:** A text box containing "2.3.0".
- Type:** A dropdown menu currently showing "Event Hub". To its right is a "Test connection" button.
- Endpoint:** An empty text box.
- Key name:** An empty text box.
- Key:** An empty text box.
- Entity path:** An empty text box.
- Below the Entity path field is a button labeled "Get connection string from clipboard".
- Message timeout:** A spinner box set to "30" with a unit "s" (seconds).
- At the bottom left is a checkbox labeled "Enable SSL verification" which is checked.

Endpoint, Key name, Key, Entity path

The corresponding parts of a connection string from the properties of the Event Hub in the Microsoft Azure Portal are inserted into these fields. You can use the button below to fill in the fields automatically.

<Get connection string from clipboard>

If you have copied a connection string from the properties of the Event Hub in the Microsoft Azure Portal to the clipboard, you can paste it using this button. The fields above it are then filled in automatically.

Connection string

You can find the connection string in the properties of the Event Hub in the Microsoft Azure portal.

Message timeout

Time in seconds that *ibapda* waits for a response from the Event Hub.

<Test connection>

Use the <Test connection> button to test the connection to the Event Hub.

Enable SSL verification

When the SSL verification is enabled, a CA certificate is required. Usually, the "DigiCert Global Root G2" certificate is to be selected. This is the current CA root certificate. When installing *ibapda*, this certificate is automatically stored in the certificate store.

3.3 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:

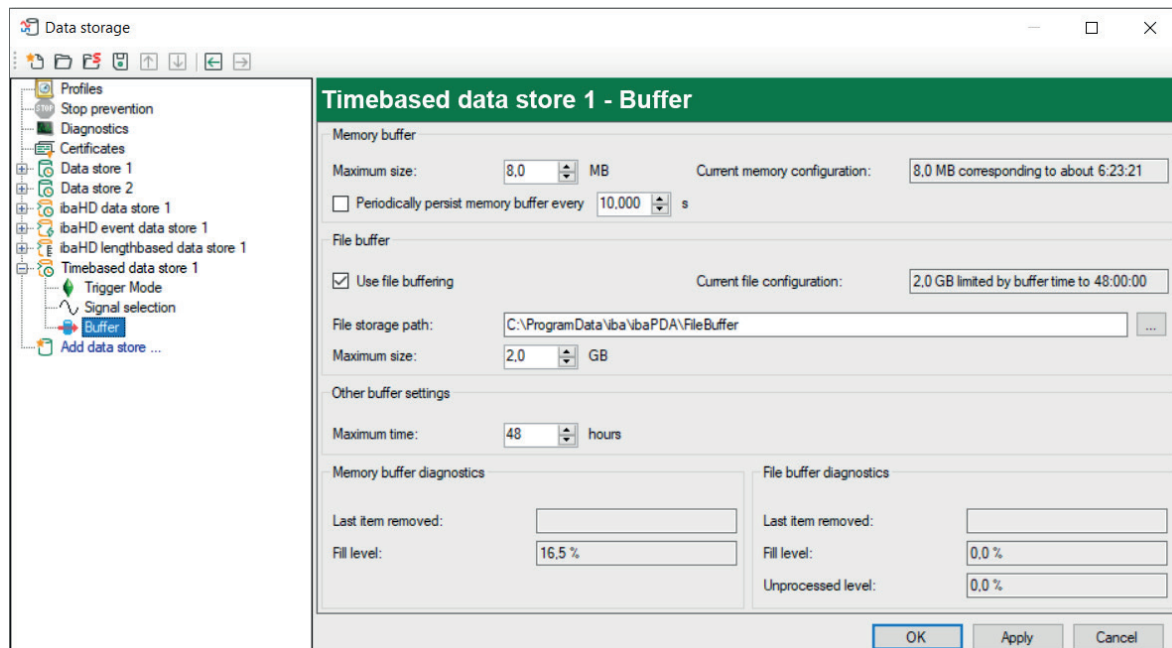
- ibaHD timebased/eventbased/lengthbased
- DB/Cloud timebased
- Kafka cluster timebased
- MQTT timebased
- MindSphere timebased
- InfluxDB timebased

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 is 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 is 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 timebased data store.



Memory buffer

The memory buffer is always enabled. You cannot deactivate it because data to be transmitted always passes through the buffer before being forwarded to the target system.

Maximum size

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

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

Periodically persist memory buffer every ... s

You can only enable this option if file buffering is enabled. If the option is enabled, the entire memory buffer is periodically swapped to a buffer file.

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

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

Current memory configuration

Display of the approximate time period that can be temporarily stored in the memory buffer with the configured settings. 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.

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

Maximum size

You can configure the maximum total size of the buffer files of 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 is not 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 diagnostics/File buffer diagnostics

Last item removed

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

Fill level

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

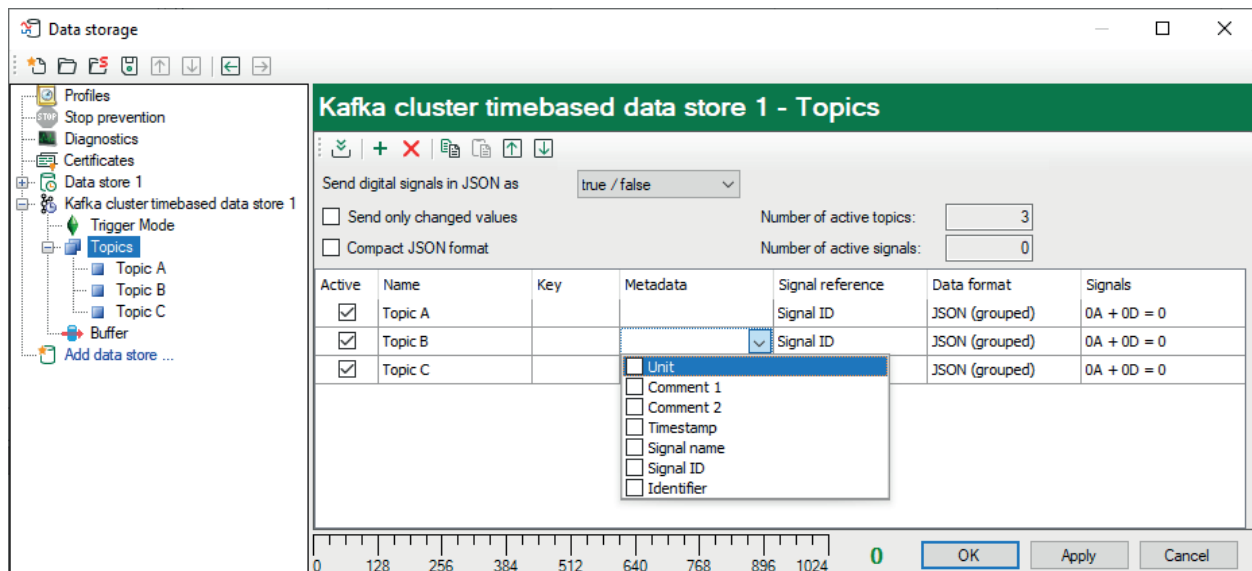
Unprocessed level

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

3.4 Configuration of topics

Several topics are usually defined in the Kafka cluster, which are created by the cluster administrator. It is not possible to create or delete topics in the cluster from *ibaPDA*.

If you highlight the *Topics* node, the list of topics is shown, that are to be written by the data store.



Buttons for the configuration of the topics:

	Download the topic list from the Kafka cluster (a connection to the cluster is required)
	Manually add a new topic
	Remove selected topic
	Copy selected 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.

Meaning of the columns

Active

Here you enable/disable writing to a topic.

Name

Enter an unambiguous name for the topic here.

Key

The *key frame* property is optional. You can define this property with a series of placeholders:

- *\$identifier*: Identifier, defined in the data storage configuration, see chapter [Kafka cluster data store](#), page 10
- *\$signalid*: Signal ID
- *\$signalname*: Signal name
- *\$unit*: Signal unit
- *\$comment1*: Signal comment 1
- *\$comment2*: Signal comment 2

When using the grouped data format *JSON (grouped)*, signal-related placeholders are replaced by empty text.

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.

Note



The metadata is included in every message. Because all metadata is constant during a measurement, except for the timestamp, this leads to a lot of redundant data written in the cluster. It is therefore important to consider which metadata is actually required.

The timestamp is implicitly already included in every message. Adding the *timestamp* to the metadata of a message therefore also leads to redundant information. From the data analysis perspective, however, it may be useful to enable the *timestamp* in the metadata.

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

The following data formats are supported:

- *JSON (grouped)*: all signal data for a certain time stamp is combined in a message
- *JSON (per signal)*: one message is created per signal and time stamp
- *AVRO (per signal)*: one message is created per signal and time stamp

You can find examples of the description of the data formats in chapter [➤ Description of the data formats](#), page 31.

Signals

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

Kafka cluster time based data store 1 - Topics

Send digital signals in JSON as

true / false

Number of active topics:

3

☐ Send only changed values
☐ Compact JSON format

Number of active signals:

0

Active	Name	Key	Metadata	Signal reference	Data format	Signals
<input checked="" type="checkbox"/>	Topic A			Signal ID	JSON (grouped)	0A + 0D = 0 ...
<input checked="" type="checkbox"/>	Topic B			Signal ID	JSON (grouped)	0A + 0D = 0
<input checked="" type="checkbox"/>	Topic C			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.

4 Signal selection

To record signals, you have to assign the signals to a topic by using a storage profile of the type *Time*.

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

Kafka cluster time based data store 1 - Topics

Send digital signals in JSON as:

☐ Send only changed values Number of active topics:

☐ Compact JSON format Number of active signals:

Active	Name	Key	Metadata	Signal reference	Data format	Signals
<input checked="" type="checkbox"/>	Topic A			Signal ID	JSON (grouped)	0A + 0D = 0 ...
<input checked="" type="checkbox"/>	Topic B			Signal ID	JSON (grouped)	0A + 0D = 0
<input checked="" type="checkbox"/>	Topic C			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
 - Kafka cluster timebased data store 1
 - Trigger Mode
 - Topics
 - Topic A
 - Topic B
 - Topic C
 - Buffer
 - Add data store ...

Topics - Topic A

Name	Linked signals
As is	0A + 0D = 0
Time Kafka	36A + 1D = 37

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
- ☐ 2:15: 071 F6 current
- ☐ 2:16: 072 F7 RPM

0 128 256 384 512 640 768 896 1024 **37**

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.

Kafka cluster data stores are licensed per number of written signals. You can find the current number of selected signals in all Kafka cluster 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.

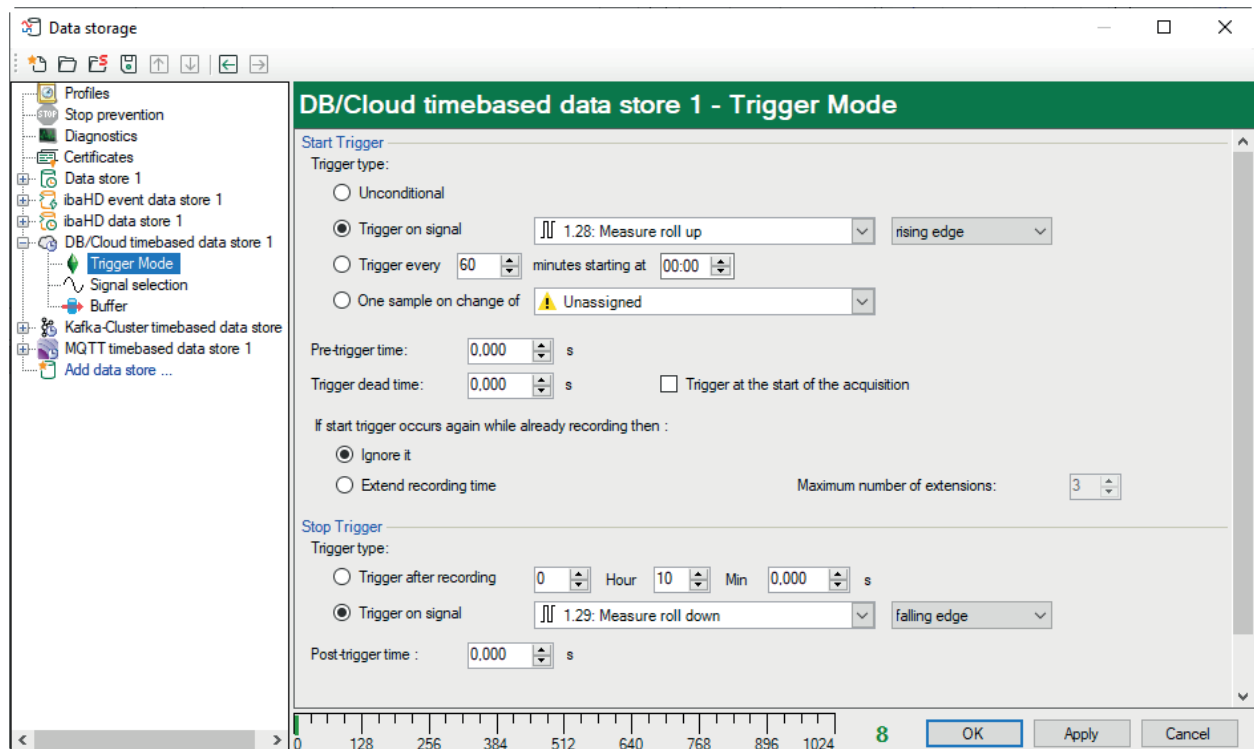
In the example above, a maximum of 1024 signals can be written via Kafka data stores. Currently 37 signals are enabled.

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*.



Start trigger

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

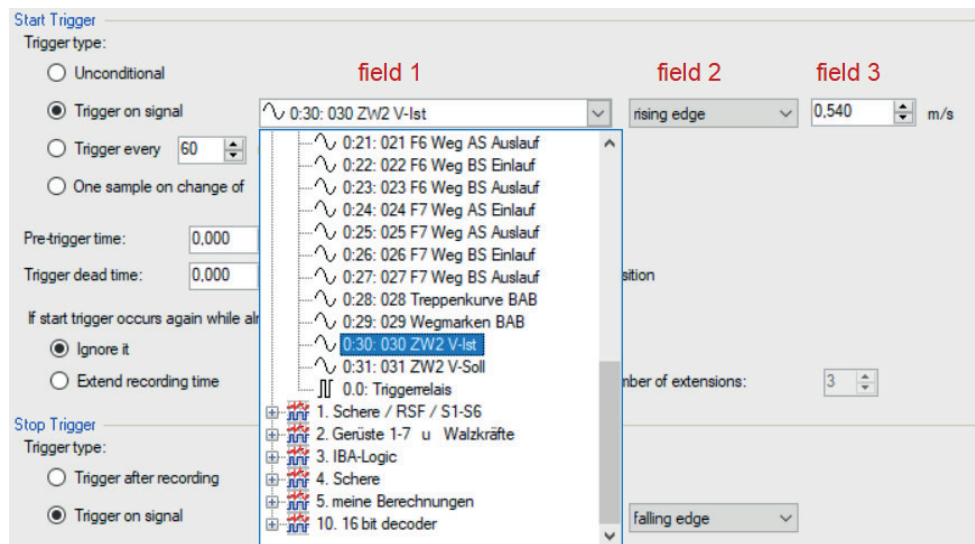
Unconditional

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 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.

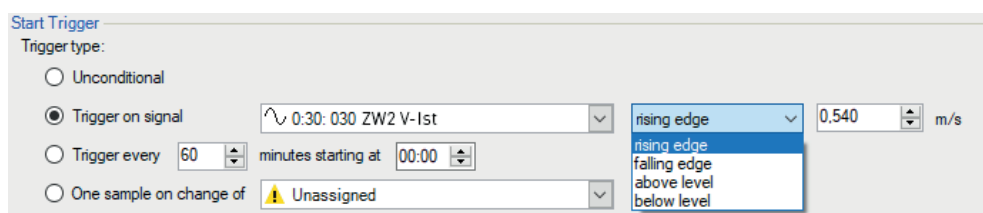
- Field 1: Drop-down list for signal selection (available analog and digital signals)
- Field 2: Drop-down list for selecting edges or levels
- Field 3: Drop-down list for selecting the trigger level value given in the specific physical unit (field 3 is only enabled in case of analog trigger signals)



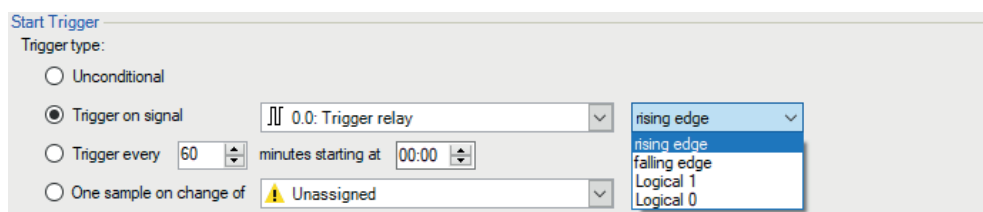
Both analog and digital signals can serve as triggers. Select the signal to be triggered via the signal tree in the selection list in field 1.

In fields 2 and 3 you can define the trigger event more specifically. These fields vary depending on whether the selected trigger signal is analog or digital.

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



- 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 stops 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 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.

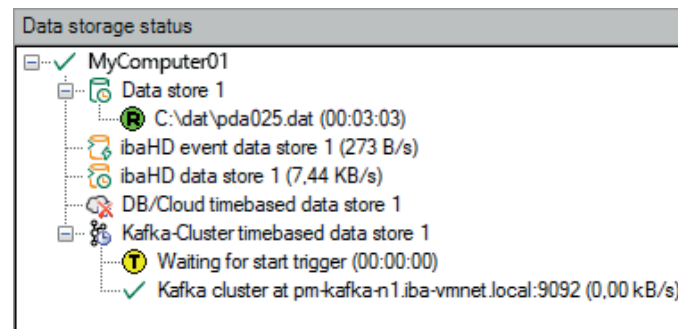
Post trigger time

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.



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 screenshot shows the 'Data storage' window with the 'Diagnostics' tab selected. The left sidebar shows a tree view of data stores. The main area displays a table of performance metrics. A context menu is open over the table, showing options to 'Show actual values' and 'Write speed unit' (with a dropdown menu showing 'KB/s', 'MB/s', and 'MB/h').

Data store		Write speed (KB/s)		Memory buffer (KB)		File buffer (MB)		Acquisition Thread load
Name	Destination	Average	Max	Average	Max	Average	Max	
Total load in acquisition thread caused by data stores: 0,36%								
Reset statistics								
<input type="checkbox"/> Erfassungsthread (0,36%)								
Data store 1	C:\	9,02	53,64	0,00	0,00			0,19%
<input type="checkbox"/> 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%
<input type="checkbox"/> ibaHD data store 1 (0,21%)								
ibaHD data store 1	MyComputer01\HD_Time2	7,01	9,28	0,00	1,59			0,04%
<input type="checkbox"/> DB/Cloud timebased data store 1 (0,00%)								
DB/Cloud timebased dat...	hanaServer:39013	0,00	0,00	0,00	0,00			0,03%
<input type="checkbox"/> Kafka-Cluster timebased data store 1 (0,00%)								
Kafka-Cluster timebased...	localhost:9092	0,00	0,00	268,83	296,07			0,02%
<input type="checkbox"/> MQTT timebased data store 1 (0,00%)								
MQTT timebased data st...	localhost	0,00	0,00	333,16	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 lengthbased 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:

Virtual (0)					
<div> General Analog Digital </div>					
	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:

Virtual (0)					
<div> <div>fx General</div> <div>Analog</div> <div>Digital</div> </div>					
	Name	Expression	Unit	Comment 1	Comment 2
0	Signalname_0	fx GenerateSignal(0) ?	Unit_0	Example_comment1_0	Example_comment2_0
1	Signalname_1	fx GenerateSignal(1) ?	Unit_1	Example_comment1_1	Example_comment2_1
2	Signalname_2	fx GenerateSignal(2) ?	Unit_2	Example_comment1_2	Example_comment2_2
*		fx ?			

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

7.1.3 Data format AVRO (per signal)

Compared to JSON, the data is saved in a readable format. AVRO uses a binary coding, which reduces the band width and the required disk space.

ibaPDA uses the following schema for the serialization of the signal data:

```
{
  "namespace:" "de.iba,"
  "type:" "record,"
  "name:" "PdaRecord,"
  "fields:" [
    {"name:" "Signal," "type:" "string"},
    {"name:" "ID," "type:" ["null," "string"]},
    {"name:" "Name," "type:" ["null," "string"]},
    {"name:" "Unit," "type:" ["null," "string"]},
    {"name:" "Comment1," "type:" ["null," "string"]},
    {"name:" "Comment2," "type:" ["null," "string"]},
    {"name:" "Timestamp," "type:" [
      "null,"
      {"type:" "long," "logicalType:" "timestamp-micros"}
    ]},
    {"name:" "Identifier," "type:" ["null," "string"]},
    {"name:" "ValueType," "type:" {
      "type:" "enum,"
      "name:" "ValueTypeEnum,"
      "symbols:" ["BOOLEAN," "BYTES," "DOUBLE," "FLOAT," "INT," "LONG," "STRING"]
    }},
    {"name:" "BooleanValue," "type:" ["null," "boolean"]},
    {"name:" "BytesValue," "type:" ["null," "bytes"]},
    {"name:" "DoubleValue," "type:" ["null," "double"]},
    {"name:" "FloatValue," "type:" ["null," "float"]},
    {"name:" "IntValue," "type:" ["null," "int"]},
    {"name:" "LongValue," "type:" ["null," "long"]},
    {"name:" "StringValue," "type:" ["null," "string"]}
  ]
}
```

ibaPDA also supports a connection to a confluent schema registry. However, this is not necessarily required so that the field can be left empty. If a schema registry is configured, *ibaPDA* registers the schema used for the coding in the schema registry. The returned ID is then appended to each dataset as follows:

Byte offset	Description
0	0x00 (Confluent AVRO Marker)
1	Schema ID (Big endian)
5	Signal data

The address and port of the schema registry can be entered in the schema registry address field.

Use the <Test connection> button to test the connection to the schema registry.

If no schema registry is used, each dataset is coded as follows:

Byte offset	Description
0	0xC3 0x01 (Single-object encoding marker)
2	CRC-64-AVRO fingerprint of encoding schema
10	Signal data

Note



You can find detailed information about AVRO here:

<https://avro.apache.org/docs/>

8 Support and contact

Support

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

Note



If you need support for software products, please state the number of the license container. For hardware products, please have the serial number of the device ready.

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

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

Mailing address

iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address

iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

Regional and Worldwide

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

www.iba-ag.com