

ibaHD-Server-API-Read

gRPC-API interface to query historical data and events by own applications

Manual

Issue 1.10

Measurement Systems for Industry and Energy

www.iba-ag.com

Manufacturer

iba AG

Gebhardtstrasse 10-20

90762 Fürth

Germany

Main office: +49 911 97282-0

Email: info@iba-ag.com

Contacts

You can find the contact information for our regional offices or representatives on our website: www.iba-ag.com/en/contact

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 2026, 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 website www.iba-ag.com and can be found in the iba help center docs.iba-ag.com.

Version	Date	Revision	Author	Version SW
1.10	11-2026	GetHdInformation(), GetLiveTimeData(), GetLiveEventData(), GetLiveTimePeriodData()	nm	3.7.0

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 5

1.1 Target group and previous knowledge 5

1.2 Notations 6

1.3 Used symbols 7

2 About ibaHD-Server-API-Read 8

2.1 Licenses 8

2.2 Information on gRPC 9

2.3 Using ibaHD-API 9

2.4 Testing the API 10

2.5 Data transport security 10

2.6 Authentication 10

2.7 Design principles 10

2.8 Memory management 11

3 ibaHD-API in ibaHD Manager 12

3.1 General information 12

3.2 Configuration 13

3.2.1 Server 13

3.2.1.1 Manage certificates 14

3.2.1.2 Generating a new certificate 15

3.2.2 API keys 16

3.3 Getting the API key 17

4 ibaHD-API functionality 18

4.1 GetHdInformation() 18

4.2 GetHdStores() 19

4.3 GetHdStoreSchema() 20

4.4 GetRawChannelData() 25

4.5 GetAggregatedChannelData() 30

4.6 GetLiveTimeData() 35

4.7 GetEventData() 41

4.8 GetLiveEventData() 44

4.9 GetLastRecordedChannelValue() 45

4.10 GetLastEventOccurrence() 48
4.11 GetHdTimePeriodStoreSchema() 50
4.12 GetHdTimePeriodData() 52
4.13 GetHdTimePeriodMetaData() 56
4.14 GetLastHdTimePeriodOccurrence() 57
4.15 GetLiveTimePeriodData() 61
5 Sample clients 62
6 Support and contact 63

1 About this documentation

This documentation describes the use of the programming interface *ibaHD-Server-API-Read*.

Other documentation



This documentation is a supplement to the *ibaHD-Server* documentation. Information about all other features and functions of *ibaHD-Server* can be found in the *ibaHD-Server* manual, the online help or in the iba Help center at https://docs.iba-ag.com.

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 trained specialist personnel if they are capable of assessing the work assigned to them and identifying possible risks based on their specialist training, knowledge, experience and awareness of the applicable regulations.

This documentation in particular addresses persons, who are concerned with capturing and storage of measuring data. For the handling of *ibaHD-Server-API-Read* the following basic knowledge is required and/or useful:

- Windows operating system
- Basic knowledge *ibaHD-Server*
- Basic knowledge programming languages, especially C# or C++ or Python or another language which is provided by the gRPC group

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 About ibaHD-Server-API-Read

With the *ibaHD-Server-API-Read* interface, you can query historical data and events from *ibaHD-Server* via third-party applications. This allows one or more external clients to access HD data.

The API is based on the open source framework gRPC. This RPC framework (Remote Procedure Call) uses the HTTP/2 protocol and uses protocol buffers for serialization.

Other documentation

For further information on gRPC see https://grpc.io/.

On the website you will find various tools and instructions for creating gRPC client code in different programming languages.

With version 1 (v1) of the API, you can retrieve information about the available data stores, the signal structure and channel and event data from *ibaHD-Server*.

In the following, the abbreviation *ibaHD-API* is used for *ibaHD-Server-API-Read*

2.1 Licenses

ibaHD-Server-API-Read is an interface requiring a license for *ibaHD-Server*. Add the license to the license container that contains the basic license of *ibaHD-Server*.

You need the following licenses to use the ibaHD-API:

Order no.	Product name	Description
30.800001	ibaHD-Server-API-Read	gRPC-API interface to query saved signals and events from existing HD stores
30.800005	ibaHD-Server-Client	License extension to include one additional HD client for streaming time-based, event-based, and time period data. One ibaHD-Server client license is required per data stream.

Note

HD data streaming is available via the *ibaHD-Server-API-Read* interface starting with *ibaHD-Server* v3.7.0 and ibaHD-API v1.10.

Other documentation

For more information on licensing *ibaHD-Server,* see the *ibaHD-Server* documentation in the Downloads section of the iba website or on the iba help portal at https://docs.iba-ag.com.

2.2 Information on gRPC

gRPC uses Protocol Buffer as IDL (Interface Definition Language) to define the structure of the API and the message formats, see https://grpc.io/docs/guides/concepts/.

You can generate client code for various programming languages on the basis of the interface definition. A complete list of currently supported languages can be found here: https://grpc.io/docs/languages/

Use the generated client code to connect to the server side of the API that is integrated in *ibaHD-Server*.

2.3 Using ibaHD-API

To use the API programmatically, you must first generate the *ibaHD-API* client code for the desired programming language.

You will need the following for this:

- The interface definition file `ibaHD-API.proto`.

  The proto file is delivered with the *ibaHD-Server* installation. You will find the file in the installation directory of *ibaHD-Server* under the following path:

  `C:\Program Files\iba\ibaHD-Server\ibaHD-API\ibaHD-API.proto`

- The gRPC tooling required to generate the ibaHD-API client.

  For each supported language, there are detailed instructions on the gRPC website on how to generate the client code https://grpc.io/docs/tutorials/

Once you have generated the *ibaHD-API* client code, you can link it to the target application. Depending on the language used, it may be necessary to integrate additional gRPC and ProtoBuf basic packages into the application. Details on this can be found in the programming language-specific guides.

Note



The gRPC toolchain for generating the client code is freely available as open source software and is not the intellectual property of iba AG. The tools are subject to change without prior notice.

iba AG recommends creating a backup copy of the toolchain for code generation that was used to generate the client code for *ibaHD-API*.

2.4 Testing the API

There are several GUI applications that can be used to load gRPC interface definition files (*.proto) and execute requests, e.g. Postman, see https://www.postman.com/.

This allows you to test the API without having to generate and integrate the client code.

You can find a selection of similar tools here:
https://github.com/grpc-ecosystem/awesome-grpc#tools-gui

You can find various tools on the Internet for converting Unix timestamps in the various requests and responses. An example is "Current Millis" for the conversion of milliseconds, see https://currentmillis.com.

Note



All these tools are freely available as open source software and are not the intellectual property of iba AG. They are subject to change without prior notice. Function and availability are not guaranteed by iba AG.

2.5 Data transport security

TLS certificates are used to ensure the security of data transport over the network. Since TLS is a mandatory requirement for the function of *ibaHD-API*, you must configure certificates before you can activate the API. Clients must reference the server's TLS certificate in order to connect to the API endpoint.

Information on generating and configuring certificates can be found in the chapter ↗ *Configuration, page 13*.

2.6 Authentication

If user management is activated, authentication is based on the user. To authenticate a user, you must generate an API key for the user, see chapter ↗ *Configuration, page 13*.

You must then specify the API key for each request as a header field (also called metadata in the gRPC context) with the name "ibahdapi-apikey".

2.7 Design principles

The aim of *ibaHD-API* is to enable generic access to the data stored in *ibaHD-Server*. This generic approach makes it possible to support a wide range of use cases.

Minor version changes of the API are always compatible with each other and should not require a regeneration of the client code, unless you want to use new API functions.

Major version changes include changes to the API that make it necessary to update the client code and may require changes to the implementation.

2.8 Memory management

All data APIs have options for limiting or specifying the amount of data to be sent in a single response message. Although there are default values, these limits may not be optimal for all use cases and should be adapted to the target environment. The memory requirement for a single response message depends on a number of factors, such as the original sampling rate, the requested time range, the requested number of channels and, in the case of event channels, the length of the event messages. Therefore, the memory requirement cannot be determined before execution.

iba AG recommends evaluating and testing the memory consumption for data requests and configuring additional restrictions for end users in the client application. By limiting the maximum time range or the maximum number of sampled values or signals, for example, you achieve increased operational reliability and optimize the application for your requirements.

If you deactivate limitations or increase them beyond the capabilities of the server hardware, this can exhaust the memory in the system and possibly lead to data loss if there are active writing systems on the *ibaHD-Server* instance.

3 ibaHD-API in ibaHD Manager

The *ibaHD-API* is configured in the *ibaHD Manager*.

3.1 General information

The current *ibaHD-API* configuration is displayed in the *ibaHD-API* tab:

API version Displays the current API version

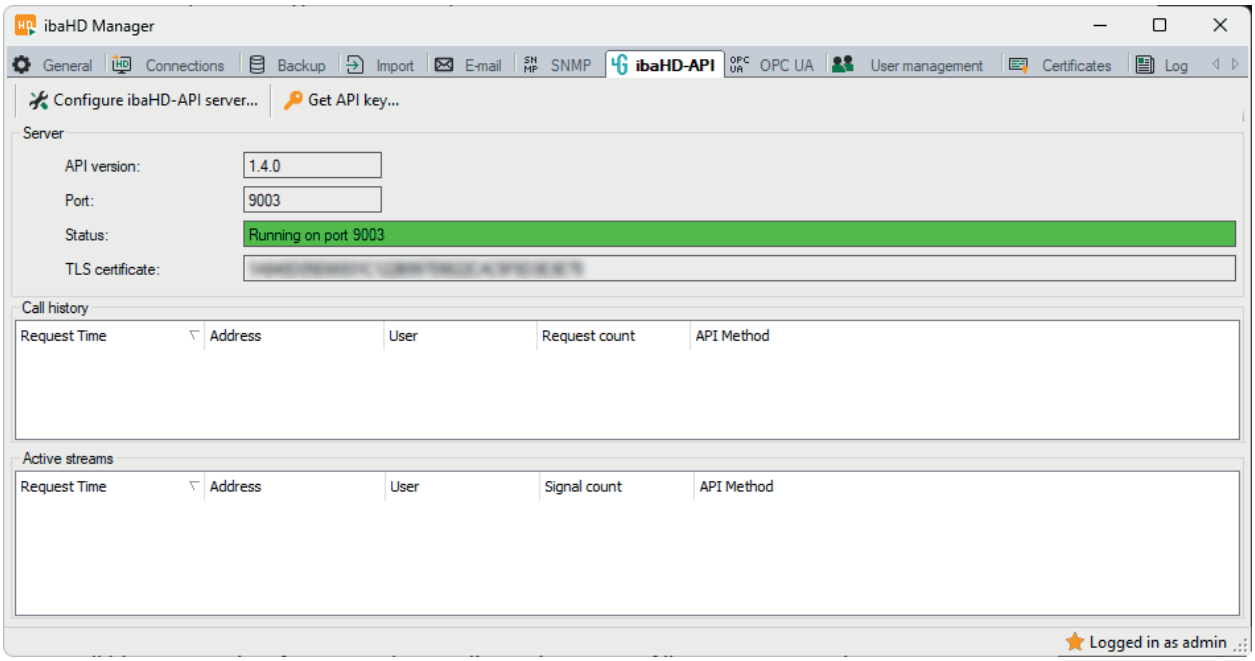
Port Port number for *ibaHD-API* communication

Status Current API status

TLS certificate Fingerprint of the currently used TLS certificate

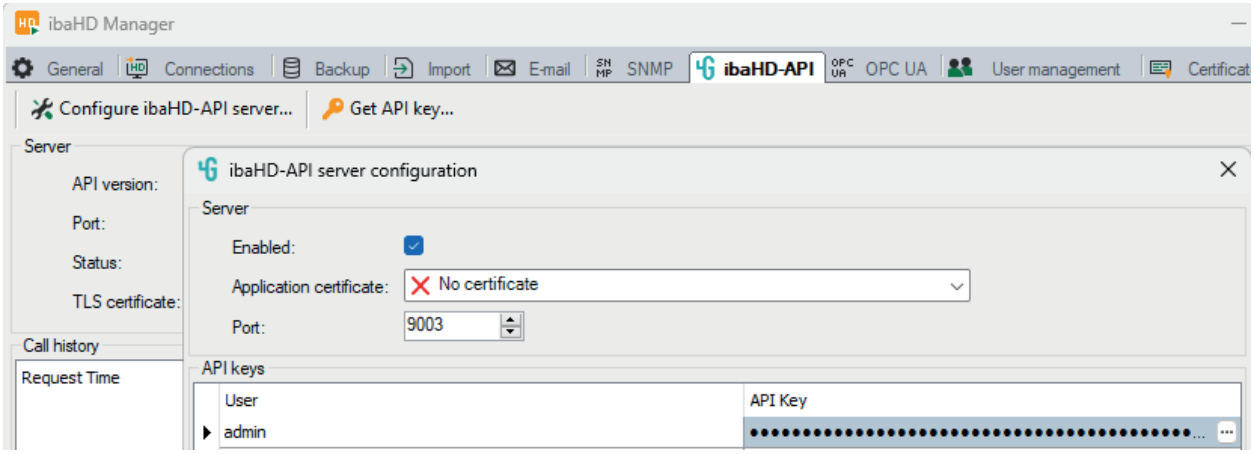
Call history List of connected gRPC clients since start of *ibaHD-Server* service

Active streams List of all clients currently streaming



3.2 Configuration

To configure the interface, click on the <Configure ibaHD-API server...> button. The *ibaHD-API server configuration* dialog opens.



The dialog is divided into the two configuration areas *Server* and *API keys*.

3.2.1 Server

In the *Server* configuration area, make the following settings for the *ibaHD-API* interface.



Enabled
Activate or deactivate the *ibaHD-API* function

Application certificate
Choose from the following options:

- Select certificate from the list of available certificates
- Generate a new certificate, see ↗ *Generating a new certificate, page 15*.
- Manage certificates, see ↗ *Manage certificates, page 14*.

Note

When you generate a certificate in *ibaHD-Server*, the "Is CA" attribute in the certificate is set to "True" by default. However, some TLS implementations do not accept this setting, which can lead to problems when using the certificates.

iba recommends using your own certificates that comply with the standards and requirements of your local IT policies.

Port
Enter a different port if the default port 9003 is already in use or a different port is to be used for *ibaHD-API* communication.

3.2.1.1 Manage certificates

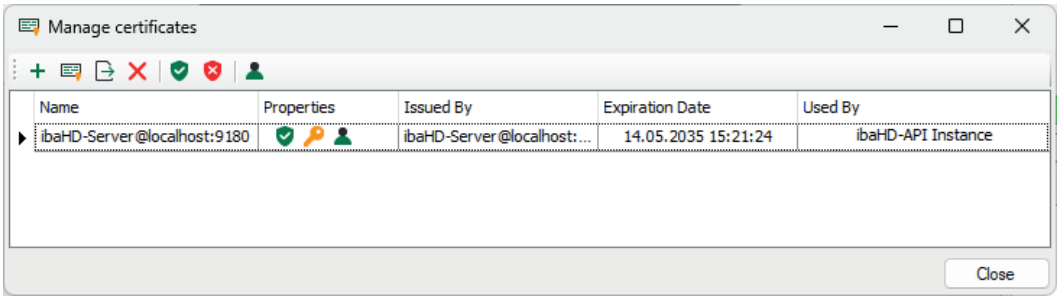
For secure communication, *ibaHD-API* uses X.509 certificates. A certificate is required for communication via TLS. TLS certificates can be provided by the server and must be transferred to the client. A *gRPC-API* communication can only take place if each communication partner trusts the certificate. You can also register certificates and mark them as "not trusted". Communication with a partner who has a "not trusted" certificate is rejected.

To manage certificates, select the *Manage certificates* option in the *Application certificate* field. A dialog opens and shows the available certificates in tabular form. Here you can add, generate and remove certificates.

In the toolbar of the table you will find a series of buttons with the following functions:

Button	Function
+	This button opens a dialog box, which you can use to load an existing certificate file. Various file formats are supported (.der, .cer, .crt, .cert, .pem, .pfx, .p12). If you have a certificate with an unknown file extension, expand the file filter to "*.*" and try to open the file anyway. This works in most cases. The existing certificate must contain a private key.
	Thus button opens a dialog box, which you can use to create a new certificate.
	You can use this button to export a certificate to a file in order to register it for Windows or another application, e.g. on an OPC UA client. Several file formats are also supported here.
✕	This button is used to delete the selected certificate from the table.
ⓘ	This button is used to open a detailed view of the selected certificate.
	This button is used to flag the selected certificate as "trusted"
	This button is used to flag the selected certificate as “not trusted”. However, the certificate remains in the certificate store table.
	Use this button to specify whether a certificate can also be used for user authentication for OPC UA. Not relevant for *ibaHD-API*.

Example



3.2.1.2 Generating a new certificate

Proceed as follows to generate a new certificate.

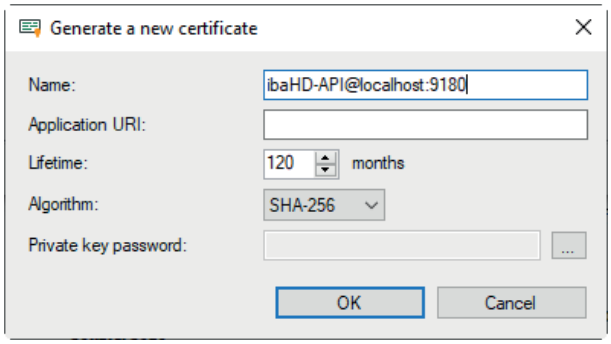
Note

When you generate a certificate in *ibaHD-Server*, the "Is CA" attribute in the certificate is set to "True" by default. However, some TLS implementations do not accept this setting, which can lead to problems when using the certificates.

iba recommends using your own certificates that comply with the standards and requirements of your local IT policies.

1. Click the button.

→ The following dialog opens.



2. Enter a name of your choice for the certificate or use the default name.
3. If required, enter an Application URI.

   The URI (Uniform Resource Identifier) is a global unique identifier for the application. If you do not fill in this field, a standard URI will be generated, provided, that the OPC UA client verifies an Application URI which is made up of the machine name and application name: `urn:machinename:applicationName`
4. Define the desired validity period (lifetime) for your certificate.
5. Select the desired hash algorithm for the encryption.

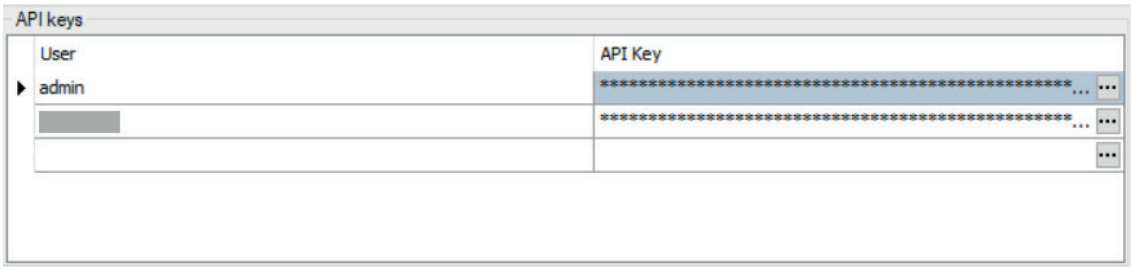
   You have the choice between the algorithms SHA-256, SHA-384 and SHA-512. Make sure that the other communication partners support the selected algorithm too.
6. Define a password for the private key.

   If no password has been entered, the <OK> button remains inactive. To assign the password, click the <...> button and enter the password twice and confirm with <OK>. There are no special requirements for the password. Keep the password in a safe place so that the self-generated certificate can be exported and used for Windows or other applications.
7. Exit the dialog with <OK>.

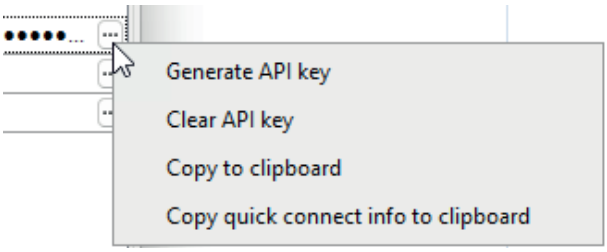
You can now use the certificate you have just created as an API communication certificate.

3.2.2 API keys

The API key configuration area shows a list of all users configured in the *ibaHD-Server* user administration. You can generate an API key for each user. This is only required if user management is activated.



If you click on the ··· button, a context menu opens to manage the API key and the user's connection information:



Generate API key
Generate new API key if no key has been created yet or the existing key should no longer be used. The previous API key is replaced by the newly generated key.

Clear API key
Delete the user's API key.

Copy to clipboard
Copies the user's API key to the clipboard, e.g. to transfer the user's API key to the client application.

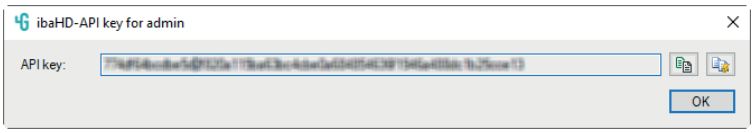
Copy quick connect info to clipboard
Copy all user-related connection information to the clipboard.

This allows you to transfer information such as host, certificate, API key and user as text to the client application. You can also use this function to transfer the connection data to *ibaDaVIS* v2.8.0 or higher.

3.3 Getting the API key

To retrieve the API key of the currently logged in user, click on the <Get API key...> button at the top of the *ibaHD-API* tab in *ibaHD Manager*.

The following dialog opens (example for the user *admin*):



To copy the API key as text to the clipboard, click on the button.

To copy the quick connect information to the clipboard, click on the button.

4 ibaHD-API functionality

Detailed information on the API can also be found in the `ibaHD-API.proto` file. The proto file is a text file and can be opened with any text editor. It contains the complete structure of the API as well as additional comments on message formats.

You will find the file `ibaHD-API.proto` in the installation path of *ibaHD-Server* in the following directory: `C:\Program Files\iba\ibaHD-Server\ibaHD-API`

The following explanations provide an overview of the possibilities of the API.

If the authentication is not correct, the server rejects the request and responds with an error message.

4.1 GetHdInformation()

This call returns a list of information about the HD server.

Message "GetHdInformationRequest"

Structure

```
GetHdInformationRequest{
}
```

The call has no parameters.

Message "GetHdInformationResponse"

Lists all available information about the HD server.

Structure

```
GetHdInformationResponse{
   LicenseInformation{…}
}
```

Parameter

The following parameters are returned.

Pos.	Parameter	Data type	Meaning
1	title	string	Name of the HD server
2	hd_version	string	Version number of the connected ibaHD-Server
3	api_version	string	API version of the connected ibaHD-Server
4	current_hd_time	int64	The current time of the connected *ibaHD-Server* as a Unix timestamp (UTC) in microseconds.
5	license	-	For license information regarding the connected ibaHD-Server, see ↗ *Message "LicenseInformation" Enum"HdStoreType", page 19*.

Message "LicenseInformation"

The Message **LicenseInformation** contains the following parameters:

Pos.	Parameter	Data type	Meaning
1	valid	bool	Indication of whether the license is valid
2	serial_number	string	Dongle serial number
3	hardware_id	string	Dongle hardware ID
4	dongle_type	string	Dongle type, e.g., "WIBU CmActLicense v3.0" or "MARX SmarxOS v3.0"
5	customer	string	License customer Information
6	time_limit	int32	Remaining license time limit in days
7	demo_time_limit	int32	Remaining license time limit of the demo license in days
8	required_eup_date	int64	Required EUP date for this ibaHD-Server version as a Unix timestamp in microseconds
9	eup_date	int64	EUP date of the installed dongle as a Unix timestamp in microseconds

4.2 GetHdStores()

This call returns a list of the server's HD stores. Time-based and event-based HD stores and time period stores are supported. Length-based HD stores are not included.

Message "GetHdStoresRequest"

Structure

```
GetHdStoresRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	include_diagnostic_stores	bool	Adds diagnostic stores that are hidden by default.

Message "GetHdStoresResponse"

Lists all available time and event-based HD stores, as well as time period stores.

Structure

```
GetHdStoresResponse{
   HdStore{…}
   HdStore{…}
   …
}
```

Parameter

One **HdStore** message with the following parameters is returned for each store.

Pos.	Parameter	Data type	Meaning
1	hd_store_name	string	Name of the HD store, which is also used in the first part of a fully qualified channel ID.
2	hd_store_type	enum (HdStoreType)	The store type determines what kind of data (time series or events) can be retrieved from the store, see ↗ *Message "LicenseInformation" Enum"HdStoreType", page 19*.
3	enabled	bool	State of the store Data can only be retrieved from stores with the parameter *enabled* = TRUE.
4	active	bool	Returns whether data is currently being written to the store.
5	is_backup	bool	Returns whether the file is an attached backup.
6	hd_store_parent	string	Name of the parent store to which this store belongs. Is used, for example, for time period stores or diagnostic stores.

Enum"HdStoreType"

The following values are available for the **HdStoreType** parameter:

Value		Meaning
0	HD_STORE_TYPE_UNSPECIFIED	Not specified
1	HD_STORE_TYPE_TIME	Time-based HD store
2	HD_STORE_TYPE_EVENT	Event-based HD store
3	HD_STORE_TYPE_TIME_PERIOD	Time period store
4	HD_STORE_TYPE_ALARM_EVENT	Alarms & Events store

4.3 GetHdStoreSchema()

Returns the schema of the data for the specified HD store. It is possible to sort the schema according to modules or logical groups. Schemas of event-based stores are always returned in a logical group structure (folders).

Message "GetHdStoreSchemaRequest"

Structure

```
GetHdStoreSchemaRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	hd_store_name	string	Name of the store as contained in the response of GetHdStores(), see ↗ *GetHdStores(), page 19*.
2	sort_by	enum (SortByType)	Determines whether the channels are to be returned sorted by modules or logical groups, see ↗ *Enum"SortByType", page 24*.
3	info_fields	bool	If True, optional info fields for time or event channels are requested.
4	include_timebase_ levels	bool	If True, the list of available time bases is returned for each signal. The position in the list corresponds to the ID of the aggregation level, see ↗ *Message "ChannelDefinition", page 22*. The list may change after a new signal configuration is set, for example, due to changes in the time base profile or the signal time base.

Message "GetHdStoreSchemaRequest"

Structure

```
GetHdStoreSchemaResponse{
   ChannelGroup{
      ChannelDefinition{…}
      EventDefinition {…}
      ChannelGroup{…}
   }
}
```

Parameter

Contains modules or the root of the logical groups, depending on the setting selected in the **sort_by** parameter. A **ChannelGroup** message with the following parameters is returned for each module or group.

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the module or logical group
2	group_type	enum (GroupType)	Visualization type of the group or module, see ↗ *Enum"GroupType", page 24*.
3	channels	-	Returns the message **ChannelDefinition** for channels from time-based stores per channel, see ↗ *Message "ChannelDefinition", page 22*. Empty for other store types.

Pos.	Parameter	Data type	Meaning
4	events	-	Returns the message **EventDefinition** for channels from event-based stores per channel, see ↗ *Message "EventDefinition", page 23*. Empty for other store types.
5	channel_groups	-	Returns a Message **ChannelGroup** for each group that is nested within this group (tree hierarchy).
6	group_number	int32	ID of the group Is used, if *group_type* = GROUP_TYPE_MODULE, GROUP_TYPE_GROUP or GROUP_TYPE_FOLDER

Message "ChannelDefinition"

The Message **ChannelDefinition** contains the following parameters.

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Fully qualified ID in the format `<HD store name>\<channel id>` Example: `store_1\[0:0]`
2	name	string	Name of the channel
3	active	bool	Indication of whether the channel is currently being recorded
4	data_type	enum (DataType)	Data type that is sent when data is requested from this channel, see ↗ *Enum "DataType", page 24*.
5	unit	string	User-defined unit for the channel
6	comment_1	string	User-defined comment
7	comment_2	string	User-defined comment
8	info_fields	-	Optional list of the channel's info fields, see ↗ *Message "InfoField", page 23*. Empty by default if not requested
9	channel_number	int32	Signal number (part of *channel_id*).
10	timebase_level	repeated int64	Available signal aggregation levels in microseconds. The index of an entry is the aggregation level that can be specified in the `aggregation_level` parameter of GetLiveTimeData(). This is returned only if `include_timebase_levels` = True is set in the request.

Message "EventDefinition"

The Message **EventDefinition** contains the following parameters.

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Fully qualified ID in the format `<HD store name>\<event channel id>` Example: `store_1\[0]`
2	name	string	Name of the event
3	active	bool	Indication of whether the channel is currently being recorded
4	event_type	enum (EventType)	Type of event, see ↗ *Enum"EventType", page 25*.
5	comment_1	string	User-defined comment
6	comment_2	string	User-defined comment
7	priority	string	User-defined priority
8	numeric_fields	-	Definition of the numerical fields that are to be captured when the event is recorded, see ↗ *Message "FieldDefinition", page 23*.
9	text_fields	-	Definition of the text fields that are to be captured when the event is recorded, see ↗ *Message "FieldDefinition", page 23*.
10	info_fields	-	Optional list of the event's info fields, see ↗ *Message "InfoField", page 23*. Empty by default if not requested

Message "InfoField"

The Message **InfoField** contains the following parameters.

Pos.	Parameter	Data type	Meaning
1	key	string	Key of the info field
2	values	string	List of the info field values

Message "FieldDefinition"

The Message **FieldField** contains the following parameters.

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field

Enum"SortByType"

The following values are available for the **SortByType**.

Value		Meaning
0	SORT_BY_UNSPECIFIED	Not specified
1	SORT_BY_MODULE	Sorted by modules
2	SORT_BY_LOGICAL_GROUP	Sorted by logical groups

Enum"GroupType"

Visualization type of the group or module. The following values are available for the **GroupType**.

Value		Meaning
0	GROUP_TYPE_UNSPECIFIED	Not specified
1	GROUP_TYPE_HDSTORE_TIME	Time-based store
2	GROUP_TYPE_HDSTORE_EVENT	Event-based store
3	GROUP_TYPE_HDSTORE_ALARM_EVENT	Alarms & Events store
4	GROUP_TYPE_DIAGNOSTIC	Diagnostic store
10	GROUP_TYPE_ANNOTATIONS	Not in use
11	GROUP_TYPE_ANNOTATIONS	Module that contains annotations
12	GROUP_TYPE_MODULE	Modules
13	GROUP_TYPE_VECTOR2D	Vectors
14	GROUP_TYPE_FOLDER	Module folder
15	GROUP_TYPE_LOCATION	Measurement location of a lengthbased HD store
16	GROUP_TYPE_EVENTFLOAT	Not in use
17	GROUP_TYPE_EVENTTEXT	Not in use
18	GROUP_TYPE_GROUP	Signal group
19	GROUP_TYPE_BACKUP	Not in use
22	GROUP_TYPE_BACKUPSTART	Not in use
23	GROUP_TYPE_BACKUPSTOP	Not in use
24	GROUP_TYPE_DATFILE	Not in use
25	GROUP_TYPE_INFOHEADER	Not in use
26	GROUP_TYPE_INFOFIELD	Not in use
27	GROUP_TYPE_ASTERISK	Not in use
28	GROUP_TYPE_DISABLED	Not in use

Enum "DataType"

The following values are available for **DataType**.

Value		Meaning
0	DATA_TYPE_UNSPECIFIED	Not specified
1	DATA_TYPE_FLOAT_VALUES	Value of type "float"

Value		Meaning
2	DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	DATA_TYPE_STRING_VALUES	Value of type "string"
4	DATA_TYPE_DIGITAL_VALUES	Digital value
5	DATA_TYPE_NE_FLOAT_VALUES	Value of type “float” that was stored non-equidistantly
6	DATA_TYPE_NE_DOUBLE_VALUES	Digital value that was stored non-equidistantly
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Number of changes of a digital non-equidistant value

Enum"EventType"

The following values are available for the **EventType** parameter.

Value		Meaning
0	EVENT_TYPE_UNSPECIFIED	Not specified
1	EVENT_TYPE_GENERAL_EVENT	General event
2	EVENT_TYPE_ANNOTATION	Event of type "annotation"
3	EVENT_TYPE_ALARM	Event of type "alerm"

4.4 GetRawChannelData()

This API call enables access to raw data and unmodified channel data. Possible applications are:

- Machine learning
- Providing data for other systems
- Calculations that require unaggregated data

The server always responds with a message stream. You can use the *max_sample_count_per_message* field to specify how many measured values should be sent in a single message. If there are more data points in the requested time range, the data is distributed across several messages (chunking).

The time stamp of the first response message does not have to match the start time of the query exactly. The response telegrams return the time stamps that are stored in the *ibaHD-Server* for the requested signal values.

If several channels are requested, each response message only contains data for a single channel. The sequence of data for a channel within a message is guaranteed to be in temporal order. The reception and transmission sequence of the messages for data of the requested channels is not guaranteed.

The following graphic shows the response stream when several channels are requested.



The response is completed as soon as all messages in the stream have been sent, see https://grpc.io/docs/guides/concepts/#server-streaming-rpc.

Message "GetRawChannelDataRequest"

Structure

```
GetRawChannelDataRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	time_range_from	int64	Start time of the requested time range in Unix timestamp format in microseconds.
2	time_range_to	int64	End time of the requested time range in Unix timestamp format in microseconds.
3	channel_ids	string	Channel IDs in the same format as they were returned in the response list of *GetHdStoreSchema()*, see ↗ *GetHdStoreSchema(), page 20*.
4	max_sample_count_per_message	int64	Maximum number of measured values per message If not specified (value = 0), the following default values apply: 10,000 samples for channels of types DATA_TYPE_FLOAT_VALUES, DATA_TYPE_DOUBLE_VALUES, and DATA_TYPE_DIGITAL_VALUES 1000 samples for channels of type DATA_TYPE_STRING_VALUES
5	add_extra_sample_out_of_time_range	bool	If True, the next values before and after the queried time range are part of the query in addition to the queried range. This expansion of the data space enables a better presentation of the data in the displays. Please read the following notes for *ibaHD-Server* up to version 3.6.

Important notes for ibaHD-Server up to version 3.6
If you request additional values (add_extra_sample_out_of_time_range = TRUE), it is possible that not all additional measurement values will be returned. This behavior depends on whether the specified times correspond to a measured value or to the measurement grid, as well as on the data type being queried. The following table shows the returned values and dependencies:

Data type	Start time falls on measured value/ measurement grid	Start time and end time do not fall on measured value or measurement grid
Float	No additional values before start time or after end time	Additional value before the start time and after the end time
String	Value before the start time is returned; no additional value after end time	Value before the start time is returned; no additional value after end time
Digital	no additional values before start time or after end time	Value before the start time is returned; no additional value after end time
BitDecoder	no additional values before start time or after end time	Value before the start time is returned; no additional value after end time

Message "GetRawChannelDataResponse"

List of all stored raw channel data in the defined time range with exact time range, divided into the various channel formats.

Structure

```
GetRawChannelDataResponse{
   StringValues{…}
   DigitalValues{…}
   NeFloatValues{…}
   NeDoubleValues{…}
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Channel ID in the format `<HD store name>\<channel id>` Example: `store_1\[0:0]`
3	start_timestamp	int64	Unix timestamp in microseconds
4	step	int64	Step in microseconds for the following values: DATA_TYPE_FLOAT_VALUES DATA_TYPE_DOUBLE_VALUES 0 for the following values: DATA_TYPE_STRING_VALUES DATA_TYPE_DIGITAL_VALUES
5	data_type	enum (DataType)	Data type of the returned values, see ↗ *Enum "DataType", page 29*.

Pos.	Parameter	Data type	Meaning
6	float_values	float	Equidistant values for DATA_TYPE_FLOAT_VALUES channels Time stamps for values can be calculated with: start_timestamp + step * array index
7	double_values	double	Equidistant values for DATA_TYPE_DOUBLE_VALUES channels Time stamps for values can be calculated with: start_timestamp + step * array index
8	string_values	-	Non-equidistant values for DATA_TYPE_STRING_VALUES channels Timestamps and values are compared by array index, see ↗ *Message "StringValues", page 28*.
9	digital_values	-	Non-equidistant rising/falling edges or gaps in DATA_TYPE_DIGITAL_VALUES channels The value is valid until the next edge change. Timestamps and values are compared by array index, see ↗ *Message "DigitalValues", page 29*.
10	ne_float_values	-	Non-equidistant values for DATA_TYPE_NE_FLOAT_VALUES channels The value is valid until the next edge change. Timestamps and values are compared by array index, see ↗ *Message "NeFloatValues", page 29*.
11	ne_double_values	-	Non-equidistant values for DATA_TYPE_NE_DOUBLE_VALUES channels. The value is valid until the next edge change. Timestamps and values are compared by array index, see ↗ *Message "NeDoubleValues", page 29*.

Message "StringValues"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamp in microseconds
2	value	value	Non-equidistant string values

Message "DigitalValues"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamps in micro-seconds
2	value	float	Non-equidistant edge changes (0 or 1) or gaps (NaN) in the digital channel
3	edgeCount	float	Non-equidistant edge counting of the digital channel. Indicates how often the signal value has changed in the aggregated time interval.
4	min	float	Non-equidistant minimum value of the digital channel
5	max	float	Non-equidistant maximum value of the digital channel

Message "NeFloatValues"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix timestamps in microsec-onds
2	value	float	Non-equidistant values of the float channel

Message "NeDoubleValues"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamps in micro-seconds
2	value	double	Non-equidistant values of the double channel

Enum "DataType"

The following values are available for **DataType**.

Value		Meaning
0	DATA_TYPE_UNSPECIFIED	Not specified
1	DATA_TYPE_FLOAT_VALUES	Value of type "float"
2	DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	DATA_TYPE_STRING_VALUES	Value of type "string"
4	DATA_TYPE_DIGITAL_VALUES	Digital value
5	DATA_TYPE_NE_FLOAT_VALUES	Value of type “float” that was stored non-equidistantly
6	DATA_TYPE_NE_DOUBLE_VALUES	Digital value that was stored non-equidis-tantly
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Number of changes of a digital non-equi-distant value

4.5 GetAggregatedChannelData()

Provides a way to query aggregated channel data for time-based channels. You can specify the number of measured values to be sent for a specific time range and select the minimum, maximum and average aggregations for equidistant signals and the edge count value and the initial raw value for non-equidistant signals.

By default, the API uses the architecture of *ibaHD-Server*, where each channel is stored in multiple aggregation levels to enable fast and efficient retrieval of data over longer periods of time. Possible use cases are client visualizations (e.g. a monthly overview) where the screen size limits the amount of data that can be displayed simultaneously, as well as the provision of pre-aggregated data for subsequent aggregations where min/max/average data can be used as a source.

The aggregation algorithm can be changed from the standard algorithm "Min/Max/Avg downsampling" to the alternative algorithm "Linear interpolation downsampling". This algorithm attempts to select raw data points at equidistant intervals (required time range / required measured value) and applies linear interpolation between neighboring measured values if the stored measured values do not exactly match the time stamps of the target values.

Note



Depending on the time range queried, the algorithm works on the next aggregation level or the raw data. The maximum, minimum and average values may not be accurate.

If aggregated data is used, data that is not part of the period queried is sometimes included in the calculation of the values in the marginal areas. In order to keep the effort required to determine the exact data within reasonable limits for users, the data in the marginal areas is determined with the aid of a weighting function.

When using raw data, *ibaHD-Server* cannot use the aggregation levels to speed up processing and is therefore more resource-intensive than the standard "Min/Max/Avg downsampling" algorithm.

Note



The HD server does not perform "oversampling". No more samples are transmitted than are available within the selected time range.

Message "GetAggregatedChannelDataRequest"

Structure

```
GetAggregatedChannelDataRequest{
}
```

Parameters

Pos.	Parameters	Data type	Meaning
1	time_range_from	int64	Start time of the requested time range in Unix time stamp format in microseconds
2	time_range_to	int64	End time of the requested time range in Unix time stamp format in microseconds
3	channel_ids	string	Channel ID in the same format as it was returned in the response list from *GetHdStoreSchema()*, see ↗ *GetHdStoreSchema(), page 20*.
4	sample_count	int64	Number of measured values that are returned. The value must be > 0.
5	min_aggregation	bool	Selection of the aggregated values to be sent with the response
6	max_aggregation	bool	
7	avg_aggregation	bool	
8	aggregation_ algorithm_type	enum (Aggregation AlgorithmType)	Algorithm for downsampling the data, Default value AGGR_ALGO_TYPE_MIN_MAX_ AVG_DOWNSAMPLING if no specification is made
9	add_extra_sample_ out_of_time_range	bool	If True, an additional measured value is requested before and after the desired time range. No additional values are added for digital channels, but the first value is before time_ range_from. This option is only available for the AGGR_ ALGO_TYPE_MIN_MAX_AVG_ DOWNSAMPLING algorithm. Please read the following notes for *ibaHD-Server* up to version 3.6.

Important notes for ibaHD-Server up to version 3.6

If you request additional values (add_extra_sample_out_of_time_range = TRUE), it is possible that not all additional measurement values will be returned. This behavior depends on whether the specified times correspond to a measured value or to the measurement grid, as well as on the data type being queried. The following table shows the returned values and dependencies:

Data type	Start time falls on measured value/ measurement grid	Start time and end time do not fall on measured value or measure-ment grid
Float	An additional value is returned be-fore the start time, but it is invalid; no additional value is returned after the end time.	Additional value before the start time and after the end time
String	Value before the start time is re-turned; no additional value after end time	Value before the start time is re-turned; no additional value after end time
Digital	no additional values before start time or after end time	no additional values before start time or after end time
BitDecoder	no additional values before start time or after end time	no additional values before start time or after end time

Message "GetAggregatedChannelDataResponse"

List of time stamps for the aggregated measured values, divided into the various channel for-mats.

Structure

```
GetAggregatedChannelDataResponse{
   AggregatedChannel{
      FloatValues{…}
      DoubleValues{…}
      StringValues{…}
      DigitalValues{…}
   }
}
```

Parameters

Pos.	Parameters	Data type	Meaning
1	timestamps	int64	Timestamp for the channels DATA_TYPE_ FLOAT_VALUES and DATA_TYPE_DOUBLE_ VALUES as Unix timestamp in microseconds
2	aggregated_channels	-	Channels containing values for the selected aggregations (min, max, avg) One message AggregatedChannel is returned per channel, see ↗ *Message "Aggregated-Channel", page 33*.

Message "AggregatedChannel"

Pos.	Parameters	Data type	Meaning
1	channel_id	string	Channel ID in the format `<HD store name>\<channel id>` Example: `store_1\[0:0]`
2	data_type	enum (DataType)	Data type of the returned values, see ↗ *Enum "DataType", page 35*.
3	float_values	-	Values for DATA_TYPE_FLOAT_VALUES channels, otherwise empty, see ↗ *Message "FloatValues", page 33*. Time stamps and values are compared using the array index.
4	double_values	-	Values for DATA_TYPE_DOUBLE_VALUES channels, otherwise empty, see ↗ *Message "DoubleValues", page 34*. Time stamps and values are compared using the array index.
5	string_values	-	Non-equidistant values for DATA_TYPE_STRING_VALUES channels, otherwise empty, see ↗ *Message "StringValues", page 34*. Time stamps and values are compared by array index.
6	digital_values	-	Non-equidistant rising/falling edges or gaps for DATA_TYPE_DIGITAL_VALUES channels, otherwise empty, see ↗ *Message "DigitalValues", page 34*. The values are valid until the next edge change, time stamp and values are matched by the array index.

Message "FloatValues"

Pos.	Parameter	Data type	Meaning
1	min_values	float	Optional: Minimum values of an aggregated channel Empty by default if not requested
2	max_values	float	Optional: Maximum values of an aggregated channel Empty by default if not requested
3	avg_values	float	Optional: Average values of an aggregated channel Empty by default if not requested

Message "DoubleValues"

Pos.	Parameter	Data type	Meaning
1	min_values	double	Optional: Minimum values of an aggregated channel Empty by default if not requested
2	max_values	double	Optional: Maximum values of an aggregated channel Empty by default if not requested
3	avg_values	double	Optional: Average values of an aggregated channel Empty by default if not requested

Message "StringValues"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamp in microseconds
2	value	value	Non-equidistant string values

Message "DigitalValues"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamps in microseconds
2	value	float	Non-equidistant edge changes (0 or 1) or gaps (NaN) in the digital channel
3	edgeCount	float	Non-equidistant edge counting of the digital channel. Indicates how often the signal value has changed in the aggregated time interval.
4	min	float	Non-equidistant minimum value of the digital channel
5	max	float	Non-equidistant maximum value of the digital channel

Enum"AggregationAlgorithmType"

Value		Meaning
0	AGGR_ALGO_TYPE_UNSPECIFIED	Not specified
1	AGGR_ALGO_TYPE_MIN_MAX_AVG_DOWNSAMPLING	Min/Max/Avg Downsampling
2	AGGR_ALGO_TYPE_LINEAR_INTERPOLATION_DOWNSAMPLING	Linear interpolation

Enum "DataType"

The following values are available for **DataType**.

Value		Meaning
0	DATA_TYPE_UNSPECIFIED	Not specified
1	DATA_TYPE_FLOAT_VALUES	Value of type "float"
2	DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	DATA_TYPE_STRING_VALUES	Value of type "string"
4	DATA_TYPE_DIGITAL_VALUES	Digital value
5	DATA_TYPE_NE_FLOAT_VALUES	Value of type “float” that was stored non-equidistantly
6	DATA_TYPE_NE_DOUBLE_VALUES	Digital value that was stored non-equidistantly
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Number of changes of a digital non-equidistant value

4.6 GetLiveTimeData()

Returns the current live data for the requested channel as streaming data. The data for the requested channel is sent from the HD server to the requesting API client. Once the request is invoked, the data is sent until the the stream is closed from client side. The data can be requested based on a user-defined time base or based on the raw data or aggregation level.

For streaming HD data, please note the requirements in chapter ↗ *Licenses*, page 8.

Message "GetLiveTimeDataRequest"

Structure

```
GetLiveTimeDataRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	channel_ids	repeated string	List of event channel IDs in the format `<HD store name>\<event channel id>` Example: `store_1\[0:0]`
2	mode	enum (AggregationRequestMode)	Specifies how the aggregation level is selected, see ↗ *Enum "AggregationRequestMode", page 41*.
3	aggregation_level	int32	Specify the desired aggregation level 0 = Streaming of live raw data 1…N = Streaming of aggregated levels To record data using specific aggregation time bases, define a "Time, ibaHD" profile in *ibaPDA*.

Pos.	Parameter	Data type	Meaning
4	preferred_timebase	int64	Target time base in microseconds *ibaHD-Server* will provide data at a time base that matches this value or is faster, with the level closest to this value.

Message "GetTimeLevelChannelDataResponse()"

Structure

```
GetLiveTimeLevelChannelDataRequest{
   oneof data{
      RawData{
         RawBitValues{…}
      }
      AggregatedData{
         AggregatedBitecoderValues{
            AggregatedBitValues{…}
         }
      }
   }
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Channel ID in the format `<HD store name>\<channel id>` Example: `store_1\[0:0]`
2	start_timestamp	int64	Unix timestamp in microseconds
3	step	int64	Step size in microseconds for values of type DATA_TYPE_FLOAT_VALUES, DATA_TYPE_DOUBLE_VALUES, and DATA_TYPE_DIGITAL_VALUES 0 for DATA_TYPE_STRING_VALUES and non-equidistant values of level 0
4	data_type	enum (DataType)	Data type of the returned values, see ↗ *Enum "DataType", page 41*

Pos.	Parameter	Data type	Meaning
5	rawData	-	Returns the raw data, see ↗ *Message "RawData", page 37* The fields `rawData` and `aggregated_channels` form a "oneof": For each message, exactly one of the two fields is populated. `rawData` is sent when `aggregation_level` = 0.
6	aggregated_channels	-	Returns aggregated data, see ↗ *Message "AggregatedData", page 38* Sent instead of `rawData` when `aggregation_level` ≥ 1.
7	ne_timestamps	repeated int64	Non-equidistant Unix timestamps in microseconds for all channels stored in a non-equidistant manner, such as DATA_TYPE_NE_FLOAT_VALUES, DATA_TYPE_NE_DOUBLE_VALUES, DATA_TYPE_STRING_VALUES, and DATA_TYPE_DIGITAL_VALUES The value remains valid until the next edge change occurs. Timestamps and values are matched by array index.

Message "RawData"

Pos.	Parameter	Data type	Meaning
1	float_values	repeated float	Values for the DATA_TYPE_FLOAT_VALUES channels, otherwise empty Timestamps and values are matched by array index.
2	double_values	repeated double	Values for channels of type DATA_TYPE_DOUBLE_VALUES, otherwise, empty Timestamps and values are matched by array index.
3	string_values	repeated string	Non-equidistant values for channels of type DATA_TYPE_STRING_VALUES, otherwise, empty Timestamps and values are matched by array index.

Pos.	Parameter	Data type	Meaning
4	digital_values	repeated float	Non-equidistant rising/falling edges or gaps for channels of type DATA_TYPE_DIGITAL_VALUES, otherwise empty The value is valid until the next edge change. Timestamps and values are matched by array index.
5	bit_decoder_values	-	For values of bit decoder signals, see ↗ *Message "RawBitValues", page 39* Bit decoders store multiple digital signals as integers. This response contains the digital value for all requested bits.

Message "AggregatedData"

The fields `float_values`, `double_values`, `string_values`, `digital_values` and `bit_decoder_values` form a “oneof”: For each message, exactly one of the fields is filled in, depending on the channel's `data_type` .

Pos.	Parameter	Data type	Meaning
1	float_values	-	Values for channels of type DATA_TYPE_FLOAT_VALUES, otherwise, empty. See ↗ *Message "FloatValues", page 40* Timestamps and values are matched by array index.
2	double_values	-	Values for channels of type DATA_TYPE_DOUBLE_VALUES, otherwise empty. See ↗ *Message "DoubleValues", page 40* Timestamps and values are matched by array index.
3	string_values	-	Non-equidistant values for channels of type DATA_TYPE_STRING_VALUES, otherwise empty. See ↗ *Message "StringValuesNoStamps", page 40* Timestamps and values are matched by array index.
4	digital_values	-	Non-equidistant rising/falling edges or gaps for DATA_TYPE_DIGITAL_VALUES channels, otherwise empty. See ↗ *Message "DigitalValuesNoStamps", page 40* The value is valid until the next edge change. Timestamps and values are matched by array index.

Pos.	Parameter	Data type	Meaning
5	bit_decoder_values	-	For values of bit decoder signals, see ↗ *Message "AggregatedBitDecoderValues", page 39* Bit decoders store multiple digital signals as integers. This response contains the digital value for all requested bits.

Message "RawBitValues"

Pos.	Parameter	Data type	Meaning
1	bitIndex	int32	Index of the bit within the bit decoder signal
2	bitValue	repeated float	Digital values (0 or 1) or missing values (NaN) for the bit. Timestamps and values are matched by array index.

Message "AggregatedBitDecoderValues"

Container for the aggregated values of a bit decoder signal. One Message **AggregatedBitValues** is returned for each requested bit.

Pos.	Parameter	Data type	Meaning
1	values	repeated AggregatedBitValues	Values per bit, see ↗ *Message "AggregatedBitValues", page 39*.

Message "AggregatedBitValues"

Optional values for an aggregated channel. Empty by default if not requested.

Pos.	Parameter	Data type	Meaning
1	bitIndex	int32	Index of the bit within the bit decoder signal
2	value	repeated float	Non-equidistant edge changes (0 or 1) or gaps (NaN) in the digital channel
3	edgeCount	repeated float	Number of non-equidistant edges in the digital channel This value indicates how many times the signal value has changed within the aggregated time interval.
4	min	repeated float	Non-equidistant minimum value of the digital channel
5	max	repeated float	Non-equidistant maximum value of the digital channel

Message "FloatValues"

Pos.	Parameter	Data type	Meaning
1	min_values	float	Optional: Minimum values of an aggregated channel Empty by default if not requested
2	max_values	float	Optional: Maximum values of an aggregated channel Empty by default if not requested
3	avg_values	float	Optional: Average values of an aggregated channel Empty by default if not requested

Message "DoubleValues"

Pos.	Parameter	Data type	Meaning
1	min_values	double	Optional: Minimum values of an aggregated channel Empty by default if not requested
2	max_values	double	Optional: Maximum values of an aggregated channel Empty by default if not requested
3	avg_values	double	Optional: Average values of an aggregated channel Empty by default if not requested

Message "StringValuesNoStamps"

Pos.	Parameter	Data type	Meaning
1	values	repeated string	Optional values for an aggregated channel; empty by default unless requested.

Message "DigitalValuesNoStamps"

Pos.	Parameter	Data type	Meaning
1	value	repeated float	Non-equidistant edge changes (0 or 1) or gaps (NaN) in the digital channel
2	edgeCount	repeated float	Number of non-equidistant edges in the digital channel Indicates how many times the signal value has changed within the aggregated time interval.
3	min	repeated float	Non-equidistant minimum value of the digital channel

Pos.	Parameter	Data type	Meaning
4	max	repeated float	Non-equidistant maximum value of the digital channel

Enum "AggregationRequestMode"

The following values are available for the **AggregationRequestMode**.

Value		Meaning
0	LEVEL	Aggregation based on a defined aggregation level
1	PREFERRED_TIMEBASE	Aggregation based on a time frame *ibaHD-Server* selects the most appropriate aggregation level. For information on the available aggregation levels, set the `include_timebase_level` parameter to True for *GetHdStoreSchema, see* ↗ *GetHdStoreSchema(), page 20*

Enum "DataType"

The following values are available for **DataType**.

Value		Meaning
0	DATA_TYPE_UNSPECIFIED	Not specified
1	DATA_TYPE_FLOAT_VALUES	Value of type "float"
2	DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	DATA_TYPE_STRING_VALUES	Value of type "string"
4	DATA_TYPE_DIGITAL_VALUES	Digital value
5	DATA_TYPE_NE_FLOAT_VALUES	Value of type “float” that was stored non-equidistantly
6	DATA_TYPE_NE_DOUBLE_VALUES	Digital value that was stored non-equidistantly
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Number of changes of a digital non-equidistant value

4.7 GetEventData()

Returns all events for the provided event channels and the specified time range. Additional event fields, which are saved together with the event, can optionally be selected in the request.

Message "GetEventDataRequest"

Structure

```
GetEventDataRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	time_range_from	int64	Start time of the requested time range in Unix time stamp format in microseconds
2	time_range_to	int64	End time of the requested time range in Unix time stamp format in microseconds
3	limit_per_channel	int64	Limits the number of events that are returned per channel, Standard (value = 0) 1000 -1 means no limitation
4	order_by	enum (OrderByType)	Chronological order in which the events are returned, see ↗ *Enum "OrderByType" Enum "EventTriggerType", page 44*. ORDER_BY_TYPE_DESCENDING by default, if not specified
5	channel_ids	string	Channel IDs in the same format as they are returned in the response list of *GetHdStoreSchema()*, see ↗ *GetHdStoreSchema(), page 20*.
6	acknowledgements	bool	Query optional details about the acknowledgement of events by the user
7	numeric_fields	string	Query optional numeric fields in the response message for each event that contains the specified fields
8	text_fields	string	Query of optional text fields in the response message for each event that contains the specified fields

Message "GetEventDataResponse"

Lists all events within the specified time range in the `events` field. A message **Event** is returned for each event with the following parameters.

Structure

```
GetEventDataResponse{
   Event{
      EventAcknowledgement{…}
      NumericField{…}
      TextField{…}
   }
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Fully qualified ID in the format `<HD store name>\<event channel id>` Example: `store_1\[0]`
2	timestamp	int64	Unix timestamp in microseconds that marks the end of the event
3	duration	int64	Duration of the event in microseconds until the time stamp
4	message	string	Configured message or user-entered comment on the event
5	trigger	enum (EventTriggerType)	Determines whether the event is an incoming or outgoing event, see ↗ *Enum "OrderByType" Enum "EventTriggerType", page 44*
6	event_acknowledgement	-	Optional information for the confirmation of events by the user, see ↗ *Message "EventAcknowledgement", page 43*. Default value zero if not requested
7	numeric_fields	-	Optional numerical field values of the event, see ↗ *Message "NumericField", page 44*. Empty by default if not requested
8	text_fields	-	Optional text field values of the event, see ↗ *Message "TextField", page 44*. Empty by default if not requested

Message "EventAcknowledgement"

Pos.	Parameter	Data type	Meaning
1	acknowledged	enum (AckType)	State of the acknowledgment
2	acknowledge_comment	string	User-defined acknowledgment comment
3	acknowledge_timestamp	int64	Unix time stamp in microseconds, if acknowledged = ACK_TYPE_ACKNOWLEDGED, otherwise 0
4	acknowledge_os_user	string	OS user who was logged in when the event was acknowledged
5	acknowledge_iba_hd-server_user	string	*ibaHD-Server* user who was logged in when the event was acknowledged
6	acknowledge_iba_pda_user_user	string	*ibaPDA* user who was logged in when the event was acknowledged

Message "NumericField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "TextField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Enum "OrderByType"

The following values are available for **OrderByType**.

Value		Meaning
0	ORDER_BY_TYPE_UNSPECIFIED	Not specified
1	ORDER_BY_TYPE_DESCENDING	Descending order
2	ORDER_BY_TYPE_ASCENDING	Ascending order

Enum "EventTriggerType"

Value		Meaning
0	EVENT_TRIGGER_TYPE_UNSPECIFIED	Not specified
1	EVENT_TRIGGER_TYPE_INCOMING	Incoming event
2	EVENT_TRIGGER_TYPE_OUTGOING	Outgoing event

Enum "AckType"

Value		Meaning
0	ACK_TYPE_UNSPECIFIED	Not specified
1	ACK_TYPE_ACKNOWLEDGED	Status "acknowledged"
2	ACK_TYPE_NOT_ACKNOWLEDGED	Status "not acknowledged"
3	ACK_TYPE_PENDING	Status "pending"

4.8 GetLiveEventData()

Returns the events currently being received on the requested event channels as streaming data. Once the request has been triggered, the data is sent until the client closes the stream.

For streaming HD data, please note the requirements in chapter ↗ *Licenses*, page 8.

Message "GetLiveDataRequest"

Structure

```
GetLiveDataRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	channel_ids	repeated string	List of event channel IDs in the format `<HD store name>\<event channel id>` Example: `store_1\[0]`
2	include_offline	bool	If TRUE, post-processed data, for example, from an *ibaDatCoordinator* job, is also included. This results in a mix of live and historical data. If FALSE, only live data is streamed in chronological order.

Message "GetEventDataResponse"

The server responds with a stream of GetEventDataResponse() messages. For information on the structure of the message and the "Event" message it contains, see ↗ *GetEventData(), page 41*.

4.9 GetLastRecordedChannelValue()

Returns the last recorded raw data value for all requested channels within the specified time range. Especially for digital or text channels that change infrequently, the returned samples may point to timestamps in the past. This means that the value of the channel has not changed within the specified time range and is still valid.

Message "GetLastRecordedChannelValueRequest"

Pos.	Parameter	Data type	Meaning
1	time_range_from	int64	Unix time stamp in microseconds, which limits how far in the past the last recorded value should be searched for Leave 0 for maximum range
2	time_range_to	int64	Unix time stamp in microseconds, returns the last recorded value before this time stamp To retrieve the last value ever recorded, set time_range_to to a date in the future or an int64 max value.

Pos.	Parameter	Data type	Meaning
3	channel_ids	string	List of channel IDs in the format `<HD store name>\<channel id>` Example: `store_1\[0:0]`

Message "GetLastRecordedChannelValueResponse"

Returns a single raw data point or the **ChannelValue** message for each requested channel. If no measured value was found for a specific channel, the response does not contain any data for this channel. The message contains the following parameters.

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Channel It in the format `<HD store name>\<channel id>` Example: `store_1\[0:0]`
2	timestamp	int64	Unix time stamp in microseconds
3	data_type	enum (DataType)	Data type of the returned value, see ↗ *Enum "DataType", page 47*.
4	float_value	float	See ↗ *Message "FloatValue", page 46*.
5	double_value	double	See ↗ *Message "DoubleValue", page 46*.
6	string_value	string	See ↗ *Message "StringValue", page 47*.
7	digital_value	float	See ↗ *Message "DigitalValue", page 47*.

Message "FloatValue"

Pos.	Parameter	Data type	Meaning
1	min_values	float	Optional: Minimum values of an aggregated channel Empty by default if not requested
2	max_values	float	Optional: Maximum values of an aggregated channel Empty by default if not requested
3	avg_values	float	Optional: Average values of an aggregated channel Empty by default if not requested

Message "DoubleValue"

Pos.	Parameter	Data type	Meaning
1	min_values	double	Optional: Minimum values of an aggregated channel Empty by default if not requested

Pos.	Parameter	Data type	Meaning
2	max_values	double	Optional: Maximum values of an aggregated channel Empty by default if not requested
3	avg_values	double	Optional: Average values of an aggregated channel Empty by default if not requested

Message "StringValue"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamp in microseconds
2	value	value	Non-equidistant string values

Message "DigitalValue"

Pos.	Parameter	Data type	Meaning
1	timestamp	int64	Non-equidistant Unix time stamps in microseconds
2	value	float	Non-equidistant edge changes (0 or 1) or gaps (NaN) in the digital channel
3	edgeCount	float	Non-equidistant edge counting of the digital channel. Indicates how often the signal value has changed in the aggregated time interval.
4	min	float	Non-equidistant minimum value of the digital channel
5	max	float	Non-equidistant maximum value of the digital channel

Enum "DataType"

The following values are available for **DataType**.

Value		Meaning
0	DATA_TYPE_UNSPECIFIED	Not specified
1	DATA_TYPE_FLOAT_VALUES	Value of type "float"
2	DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	DATA_TYPE_STRING_VALUES	Value of type "string"
4	DATA_TYPE_DIGITAL_VALUES	Digital value
5	DATA_TYPE_NE_FLOAT_VALUES	Value of type “float” that was stored non-equidistantly

Value		Meaning
6	DATA_TYPE_NE_DOUBLE_VALUES	Digital value that was stored non-equidistantly
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Number of changes of a digital non-equidistant value

4.10 GetLastEventOccurrence()

Returns the last occurrence for all requested events within the specified time range.

Message "GetLastEventOccurrenceRequest"

Pos.	Parameter	Data type	Meaning
1	time_range_from	int64	Unix time stamp in microseconds that specifies how far in the past to search for the last occurrence of the event Leave 0 for the maximum range.
2	time_range_to	int64	Unix time stamp in microseconds, returns the last event that occurred before this time stamp To retrieve the last event that was ever recorded, set "time_range_to" to a date in the future or an int64 max value.
3	channel_ids	string	List of event channel IDs in the format `<HD store name>\<event channel id>` Example: `store_1\[0]`
4	acknowledgements	bool	Query optional details about the acknowledgment of events by the user
5	numeric_fields	string	Query optional numeric fields in the response message for each event that contains the specified fields
6	text_fields	string	Query optional text fields in the response message for each event that contains the specified fields

Message "GetLastEventOccurrenceResponse"

Returns a single event or message **Event** for each requested event channel with the following parameters.

If no event was found for a specific channel, the response does not contain any data for this event channel.

Pos.	Parameter	Data type	Meaning
1	channel_id	string	Fully qualified ID in the format `<HD store name>\<event channel id>` Example: `store_1\[0]`
2	timestamp	int64	Unix time stamp in microseconds that marks the end of the event
3	duration	int64	Duration of the event in microseconds until the time stamp
4	message	string	Configured message or user-entered comment on the event
5	trigger	enum (EventTrigger Type)	Determines whether the event is an incoming or outgoing event, see ↗ *Enum "EventTriggerType", page 50*.
6	event_acknowledgement	-	Optional information for the acknowledgment of events by the user, see ↗ *Message "EventAcknowledgement", page 49*. Default value zero if not requested
7	numeric_fields	-	Optional numerical field values of the event, see ↗ *Message "NumericField", page 50*. Empty by default if not requested
8	text_fields	-	Optional text field values of the event, see ↗ *Message "TextField", page 50*. Empty by default if not requested

Message "EventAcknowledgement"

Pos.	Parameter	Data type	Meaning
1	acknowledged	enum (AckType)	State of the acknowledgment
2	acknowledge_comment	string	User-defined acknowledgment comment
3	acknowledge_timestamp	int64	Unix time stamp in microseconds, if acknowledged = ACK_TYPE_ACKNOWLEDGED, otherwise 0
4	acknowledge_os_user	string	OS user who was logged in when the event was acknowledged
5	acknowledge_iba_hd-server_user	string	*ibaHD-Server* user who was logged in when the event was acknowledged
6	acknowledge_iba_pda_user_user	string	*ibaPDA* user who was logged in when the event was acknowledged

Message "NumericField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "TextField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Enum "EventTriggerType"

Value		Meaning
1	EVENT_TRIGGER_TYPE_UNSPECIFIED	Not specified
2	EVENT_TRIGGER_TYPE_INCOMING	Incoming event
3	EVENT_TRIGGER_TYPE_OUTGOING	Outgoing event

4.11 GetHdTimePeriodStoreSchema()

Returns the list of all available info fields and their properties as a schema for the selected time period store.

Message "GetHdTimePeriodStoreSchemaRequest"

Structure

```
GetHdTimePeriodStoreSchemaRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	hd_store_name	string	Unique name of the time-based HD store, which is the parent store for the time period store
2	time_period_store_name	string	Unique name of the time period store whose info fields and schema are to be returned
3	include_standard_fields	bool	Option to integrate the standard info fields into the message. User-defined info fields are always returned.

Pos.	Parameter	Data type	Meaning
4	locale	string	For later use: Option to return the display name of the fields in a desired language. If empty, the system language of ibaHD-Server is returned.
5	include_post_processing_fields	bool	If True, the info fields from post-processing are also included in the response.

Message "GetHdTimePeriodStoreSchemaResponse"

List of available info fields in the time period store. A message **InfoFieldDefinition** with the following parameters is returned for each info field.

Structure

```
GetHdTimePeriodStoreSchemaResponse{
   InfoFieldDefinition{…}
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	info_field_data_type	enum (InfoFieldDataType)	Type of the info field, see ↗ *Enum "InfoFieldDataType", page 51*.
2	field_name	string	Name of the info field
3	display_name	string	Localized display name of the info field

Enum"InfoFieldDataType"

The following values are available for **InfoFieldDataType**.

Value		Meaning
0	IF_DATA_TYPE_UNSPECIFIED	Not specified
1	IF_DATA_TYPE_FLOAT_VALUES	Value of type "float"
2	IF_DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	IF_DATA_TYPE_STRING_VALUES	Value of type "string"
4	IF_DATA_TYPE_BOOL_VALUES	Value of type "bool"
5	IF_DATA_TYPE_INT16_VALUES	int16 value
6	IF_DATA_TYPE_INT32_VALUES	int32 value
7	IF_DATA_TYPE_INT64_VALUES	int64 value

4.12 GetHdTimePeriodData()

Returns a streaming response of the available time periods in a specific time range and a time period store. The info fields returned in the response can be filtered.

Message "GetHdTimePeriodDataRequest"

Structure

```
GetHdTimePeriodDataRequest{
   QueryMode{…}
   ColumnFilter{…}
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	hd_store_name	string	Unique name of the time-based HD store, which is the parent store for the time period store.
2	time_period_store_name	string	Unique name of the time period of interest.
3	time_range_from	int64	Start time of the requested time range in Unix time stamp format in microseconds.
4	time_range_to	int64	End time of the requested time range in Unix time stamp format in microseconds.
5	query_mode	-	The mode determines whether or not the start and end times are included in the query time range, see ↗ *Message "QueryMode", page 54*. By default, all values are incorrect and no sort order is applied.
6	filter	-	Filter for requested info fields, see ↗ *Message "ColumnFilter", page 54*. No filter is applied by default.
7	max_sample_count_per_message	int64	Specifies how many time periods are to be sent in a single message before a new message is started (chunk streaming).
8	limit	int64	Specifies the maximum number of time periods that can be sent in total. You can use the *order_by* field to determine which time periods are skipped if more time periods are available.

Message "GetHdTimePeriodDataResponse"

Returns the list of time period entries with start and end time in Unix time stamp format in microseconds. A **TimePeriodData** message with the following parameters is returned for each time period.

All default values of the info fields are added by default. The user-defined info fields filtered by *Info_field_names* are added and sorted by variable type.

Structure

```
GetHdTimePeriodDataResponse{
   TimePeriodData{
      NumericField{…}
      Int32Field{…}
      Int64Field{…}
      TextField{…}
      DigitalField{…}
   }
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	id	int64	Unique ID of the time period
2	start_time	int64	Unix time stamp for the start time of the time period in microseconds.
3	end_time	int64	Unix time stamp for the endtime of the time period in microseconds. A "0" is returned for time periods without a valid end time.
4	name	string	Name of the time period entry
5	start_trigger	double	Absolute time in seconds in relation to the start time
6	stop_trigger	double	Absolute time in seconds at which the start time occurred in relation to the start time. The value is 0.0 if no stop trigger has occurred.
7	comment	string	Comment
8	metadata_id	int32	ID of the metadata group used for this time period
9	double_fields	-	Info fields of type "double", see ↗ *Message "NumericField", page 54*.
10	int32_fields	-	Info fields of type "int32", see ↗ *Message "Int32Field", page 55*.
11	int64_fields	-	Info fields of type "int64", see ↗ *Message "Int64Field", page 55*.

Pos.	Parameter	Data type	Meaning
12	text_fields	-	Text info fields, see ↗ *Message "TextField", page 55*.
13	digital_fields	-	Info fields of type "bool", see ↗ *Message "DigitalField", page 55*.
14	autoClosed	bool	True if the time period reaches the maximum time period duration before the stop trigger and was closed automatically
15	dataMissing	bool	True if data is missing within the time period, e.g. if data recording was interrupted

Message "QueryMode"

Pos.	Parameter	Data type	Meaning
1	is_start_time_in_ time_range	bool	Determines whether the start time of the time periods should be included in the query or whether the start time is before the queried time range
2	is_end_time_in_ time_range	bool	Determines whether the end time of the time periods should be included in the query or whether the end time can be later than the end of the queried time range
3	include_open	bool	Indication of whether non-completed time periods are included in the result or whether only completed time periods with start and end times are used
4	column_filter_active	bool	Activate or deactivate column filter
5	order_by	enum (OrderByType)	Sort results by start time in ascending or descending order

Message "ColumnFilter"

Pos.	Parameter	Data type	Meaning
1	info_field_names	string	Only the info fields that are specified as filters are queried. You can request the complete list of available field names with *GetHdTimePeriodStoreSchema()*, see ↗ *GetHdTimePeriodStoreSchema(), page 50*.

Message "NumericField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field

Pos.	Parameter	Data type	Meaning
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "Int32Field"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	int32	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "Int64Field"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	int64	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "TextField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "DigitalField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	bool	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Enum "OrderByType"

The following values are available for **OrderByType**.

Value		Meaning
0	ORDER_BY_TYPE_UNSPECIFIED	Not specified
1	ORDER_BY_TYPE_DESCENDING	Descending order
2	ORDER_BY_TYPE_ASCENDING	Ascending order

4.13 GetHdTimePeriodMetaData()

Returns the metadata of the info field, such as the unit or the display name and the comment for a selected time period store.

Message "GetHdTimePeriodMetaDataRequest"

Structure

```
GetHdTimePeriodMetaDataRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	hd_store_name	string	Unique name of the time-based HD store, which is the parent store for the time period store
2	time_period_store_name	string	Unique name of the time period store whose info fields are to be returned.
3	meta_ids	int32	The IDs for which the metadata should be returned The associated metadata ID of a time period is part of the *GetHdTimePeriodData()* response, see ↗ *GetHdTimePeriodData(), page 52*.
4	locale	string	For later use: Option to return the display name of the fields in a desired language If empty, the system language of *ibaHD-Server* is returned.

Message "GetHdTimePeriodMetaDataResponse"

List of available metadata definitions of user-defined info fields for the selected meta ID within the requested time period store. A message **MetadataField** with the following parameters is returned for each metadata definition.

Structure

```
GetHdTimePeriodMetaDataResponse{
   MetadataField{…}
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	metadata_id	int32	ID of the metadata group
2	info_field_name	string	Database name of the info field
3	info_field_display_name	string	Display name of the info field

Pos.	Parameter	Data type	Meaning
4	comment_1	string	Comment
5	comment_2	string	Comment
6	info_field_unit	string	Unit of the info field
7	info_field_data_type	enum (InfoF-ieldDataType)	Info field type, see ↗ *Enum"InfoFieldDataType", page 57*

Enum"InfoFieldDataType"

The following values are available for **InfoFieldDataType**.

Value		Meaning
0	IF_DATA_TYPE_UNSPECIFIED	Not specified
1	IF_DATA_TYPE_FLOAT_VALUES	Value of type "float"
2	IF_DATA_TYPE_DOUBLE_VALUES	Value of type "double"
3	IF_DATA_TYPE_STRING_VALUES	Value of type "string"
4	IF_DATA_TYPE_BOOL_VALUES	Value of type "bool"
5	IF_DATA_TYPE_INT16_VALUES	int16 value
6	IF_DATA_TYPE_INT32_VALUES	int32 value
7	IF_DATA_TYPE_INT64_VALUES	int64 value

4.14 GetLastHdTimePeriodOccurrence()

Returns the last occurrence of a time period in a defined time range for a selected period store.

Message "GetLastHdTimePeriodOccurrenceRequest"

Structure

```
GetLastHdTimePeriodOccurrenceRequest{
   QueryMode{…}
   ColumnFilter{…}
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	hd_store_name	string	Unique name of the time-based HD store, which is the parent store for the time period store.
2	time_period_store_name	string	Name of the time period store whose data is queried.
3	time_range_from	int64	Start time of the requested time range in Unix time stamp format in microseconds.
4	time_range_to	int64	End time of the requested time range in Unix time stamp format in microseconds.

Pos.	Parameter	Data type	Meaning
5	query_mode	-	Query mode to determine whether the start and end time are included in the query time range or not, see ↗ *Message "QueryMode", page 59*.
6	filter	-	Filter for requested info fields, see ↗ *Message "ColumnFilter", page 60*.

Message "GetLastHdTimePeriodOccurrenceResponse"

Returns the **TimePeriodData** message with start and end time stamp in Unix time stamp format in microseconds for the last time period of the requested time period store.

A "0" is returned for time periods without a valid end time.

All default values of the info fields are added by default. The user-defined info fields filtered by *Info_field_names* are added and sorted by variable type.

Structure

```
GetLastHdTimePeriodOccurrenceResponse{
   TimePeriodData{
      NumericField{…}
      Int32Field{…}
      Int64Field{…}
      TextField{…}
      DigitalField{…}
   }
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	id	int64	Unique ID of the time period
2	start_time	int64	Unix time stamp for the start time of the time period in microseconds.
3	end_time	int64	Unix time stamp for the endtime of the time period in microseconds. A "0" is returned for time periods without a valid end time.
4	name	string	Name of the time period entry
5	start_trigger	double	Absolute time in seconds in relation to the start time
6	stop_trigger	double	Absolute time in seconds at which the start time occurred in relation to the start time. The value is 0.0 if no stop trigger has occurred.
7	comment	string	Comment

Pos.	Parameter	Data type	Meaning
8	metadata_id	int32	ID of the metadata group used for this time period
9	double_fields	-	Info fields of type "double", see ↗ *Message "NumericField", page 54*.
10	int32_fields	-	Info fields of type "int32", see ↗ *Message "Int32Field", page 55*.
11	int64_fields	-	Info fields of type "int64", see ↗ *Message "Int64Field", page 55*.
12	text_fields	-	Text info fields, see ↗ *Message "TextField", page 55*.
13	digital_fields	-	Info fields of type "bool", see ↗ *Message "DigitalField", page 55*.
14	autoClosed	bool	True if the time period reaches the maximum time period duration before the stop trigger and was closed automatically
15	dataMissing	bool	True if data is missing within the time period, e.g. if data recording was interrupted

Message "QueryMode"

Pos.	Parameter	Data type	Meaning
1	is_start_time_in_ time_range	bool	Determines whether the start time of the time periods should be included in the query or whether the start time is before the queried time range
2	is_end_time_in_ time_range	bool	Determines whether the end time of the time periods should be included in the query or whether the end time can be later than the end of the queried time range
3	include_open	bool	Indication of whether non-completed time periods are included in the result or whether only completed time periods with start and end times are used
4	column_filter_active	bool	Activate or deactivate column filter
5	order_by	enum (OrderByType)	Sort results by start time in ascending or descending order

Message "ColumnFilter"

Pos.	Parameter	Data type	Meaning
1	info_field_names	string	Only the info fields that are specified as filters are queried. You can request the complete list of available field names with *GetHdTimePeriodStoreSchema()*, see ↗ *GetHdTimePeriodStoreSchema(), page 50*.

Message "NumericField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "Int32Field"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	int32	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "Int64Field"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	int64	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "TextField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	double	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is null.

Message "DigitalField"

Pos.	Parameter	Data type	Meaning
1	name	string	Name of the field
2	value	bool	Value of the field
3	isData	bool	Is TRUE if the current value is either true or false, and is FALSE if the current value is zero.

4.15 GetLiveTimePeriodData()

Returns newly recorded time periods from the requested time period store as streaming data. Once the request has been triggered, data is sent until the client closes the stream. Aggregation levels and time bases do not apply to time period data.

For streaming HD data, please note the requirements in chapter ↗ *Licenses*, page 8.

Message "GetLiveDataRequest"

Structure

```
GetLiveDataRequest{
}
```

Parameter

Pos.	Parameter	Data type	Meaning
1	channel_ids	repeated string	List of time period stores in the format `<hd store name>\<time period store name>` Example: `store_1\timeperiod_1`
2	include_offline	bool	If TRUE, post-processed data, for example, from an ibaDatCoordinator job, is also included. This results in a mix of live and historical data. If FALSE, only live data is streamed in chronological order.

Message "GetHdTimePeriodDataResponse"

The server responds with a stream of GetHdTimePeriodDataResponse() messages. For information on the structure of the message and the "TimePeriodData" message it contains, see ↗ *GetHdTimePeriodData(), page 52*.

5 Sample clients

iba AG provides sample clients for the following programming languages:

- C#
- C++
- Python

  You can download a Python package and integrate it into your Python-based application. The functions for querying data from *ibaHD-Server* address the *ibaHD-Server-API-Read* interface. Using the Python package reduces your implementation effort. The Python package is maintained and updated by iba AG.

All sample clients are designed to demonstrate a minimum configuration for accessing the *ibaHD-API*, they do not cover all API functions.

The sample clients are available on GitHub:

https://github.com/iba-ag/ibaHD-API-Sample-Clients

Disclaimer:
This code is provided "as is" and serves as an example for the use of the *ibaHD-API*. The integration of the *ibaHD-API* into third-party systems and the general support of programming languages is not provided by iba AG. All terms of the license agreement of *ibaHD-Server* apply.

6 Support and contact

Support

Phone: +49 911 97282-14

Email: support@iba-ag.com

You can find the contact information for our regional offices or representatives on our website:

www.iba-ag.com/en/contact

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.

Product Security Incident Response Team (PSIRT)

Current advisories: www.iba-ag.com/en/security

Email: psirt@iba-ag.com

Contact

Headquarters
iba AG
Gebhardtstrasse 10–20
90762 Fuerth
Germany

Phone: +49 911 97282-0

Email: info@iba-ag.com

Mailing address
iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address
iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany