ibaHD-Server-API-Read

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

Manual

Issue 1.3

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

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
1.3	11-2024	New field in GetHdTimePeriodDataRequest	GS	3.4.0

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

This documentation describes the usage of the programming interface *ibaHD-Server-API-Read*. It has to be considered as documentation being complementary to the *ibaHD-Server* manual. If information is required we refer to the *ibaHD-Server* manual.

The latest version of this manual can befound on our website www.iba-ag.com under download.

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 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 Introduction

ibaHD-Server-API-Read is a licensed API enabling 3rd party applications to query historical data and events from *ibaHD-Server*.

The API is built on the open source gRPC framework, which is a Remote Procedure Call (RPC) framework based on the HTTP2 protocol and Protocol Buffers serialization.

For further information see https://grpc.io/. This website provides different tools and guides for the generation of gRPC-client code for various programming languages.

Version 1 (v1) of the API allows the client to retrieve information about the available data stores, the signal structure as well as channel and event data from *ibaHD-Server*.

In the following, *ibaHD-Server-API-Read* is referred to with the short name *ibaHD-API*.

2.1 gRPC

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

The interface definition makes it possible to generate the client code for various programming languages. A complete list of currently supported languages can be found here: https://grpc.io/docs/languages/

Once the client code has been generated it can be used to connect to the server-side of the API, which is built into *ibaHD-Server*.

2.2 Programmatic usage of ibaHD-API

To use the API programmatically, first the *ibaHD-API* client code for the desired programming language must be generated. To do this, two parts are required:

- The *ibaHD-API.proto* interface definition file. It is supplied with the *ibaHD-Server* installation and can be found in the "ibaHD-API" folder inside the *ibaHD-Server* installation directory.

- The gRPC tooling required to generate the *ibaHD-API* client. For each supported language the gRPC website provides a detailed tutorial how the client code can be generated: https://grpc.io/docs/tutorials/

Once the *ibaHD-API* client code has been generated, it can be linked to the target application. Depending on the used language, it may be necessary to link additional gRPC and ProtoBuf base packages to the application. Details are provided in the programming language-specific guides.

Note

The gRPC toolchain for generating the client code is freely available as open source and not intellectual property of iba AG. The tools are subject to change without notice. It is recommended to make a backup of the code generation toolchain that was used to generate the client code for *ibaHD-API*.

2.3   Testing the API

There are multiple GUI applications available which allow loading gRPC interface definition files (.proto) and executing requests, for example BloomRPC https://github.com/uw-labs/bloomrpc, to test the API without generating and integrating the client code.

A selection of similar tools can be found here:
https://github.com/grpc-ecosystem/awesome-grpc#tools-gui

For the conversion of Unix time stamps in the different requests and responses, several tools can be found on the internet. One example is https://currentmillis.com which can be used for millisecond conversions.

Note

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

2.4   Transport security

Transport security over network is done via TLS certificates. As TLS is a mandatory requirement for *ibaHD-API* to function, certificates must be configured before the API can be enabled. Clients must reference the server TLS certificate to be able to connect to the API endpoint. See chapter ⬈ *Configuration*, page 10 for information about certificate generation and configuration.

2.5   Authentication

Authentication is done on a per-user basis if user management is enabled. To authenticate a user, an API key must be generated for the user (see chapter ⬈ *Configuration*, page 10). The API key must then be supplied with each request as a header field (also called metadata in gRPC context) named "ibahdapi-apikey".

2.6   Design principles

The goal of *ibaHD-API* is to provide generic access to the data stored in *ibaHD-Server*. This generic approach allows to support a wide range of customer use cases.

Minor version changes of the API are always compatible to each other and should not require to regenerate the client code, unless new API features should be used.

Major version changes signal breaking changes in the API that make the regeneration of the client code mandatory and may require implementation changes.

2.7   Licenses

ibaHD-Server-API-Read is an additional licensed interface of *ibaHD-Server*. The license must be added to the dongle that contains the base license of *ibaHD-Server*. For further information please contact your local iba support.
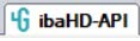
2.8   Memory management

All data APIs have options to either limit or specify the amount of data that should be sent in a single response message. While there are default values in place, these limits may not be optimal for all use cases and should be adjusted for the target environment. The memory requirement for a single response message depends on a multitude of factors, such as original sample rate, requested time range, requested number of channels and in case of event channels length of the event messages and therefore cannot be determined before execution.

It is strongly recommended to evaluate and test memory consumption for data requests as well as employing additional limitations for end users in the client application that is consuming *ibaHD-API* (e.g. limiting the maximum time range, or the number of samples or signals that can be requested).

Disabling or increasing limits beyond the server hardware capabilities can lead to memory exhaustion of the system and could potentially result in lost data if there are active writers on the *ibaHD-Server* instance.

3 ibaHD-API in ibaHD manager

3.1 General information

The current *ibaHD-API* configuration is displayed at the tab *ibaHD-API* ibaHD-API at the *ibaHD manager* GUI.

The following information is visualized:

API version	Version of the API
Port	Port number for the *ibaHD-API* communication
Status	Current status of the API
TLS certificate	Fingerprint of the TLS certificate which is currently in use
Active user API key	API key of current logged in user
Connected clients	List of connected gRPC clients since start of the *ibaHD-Server* service
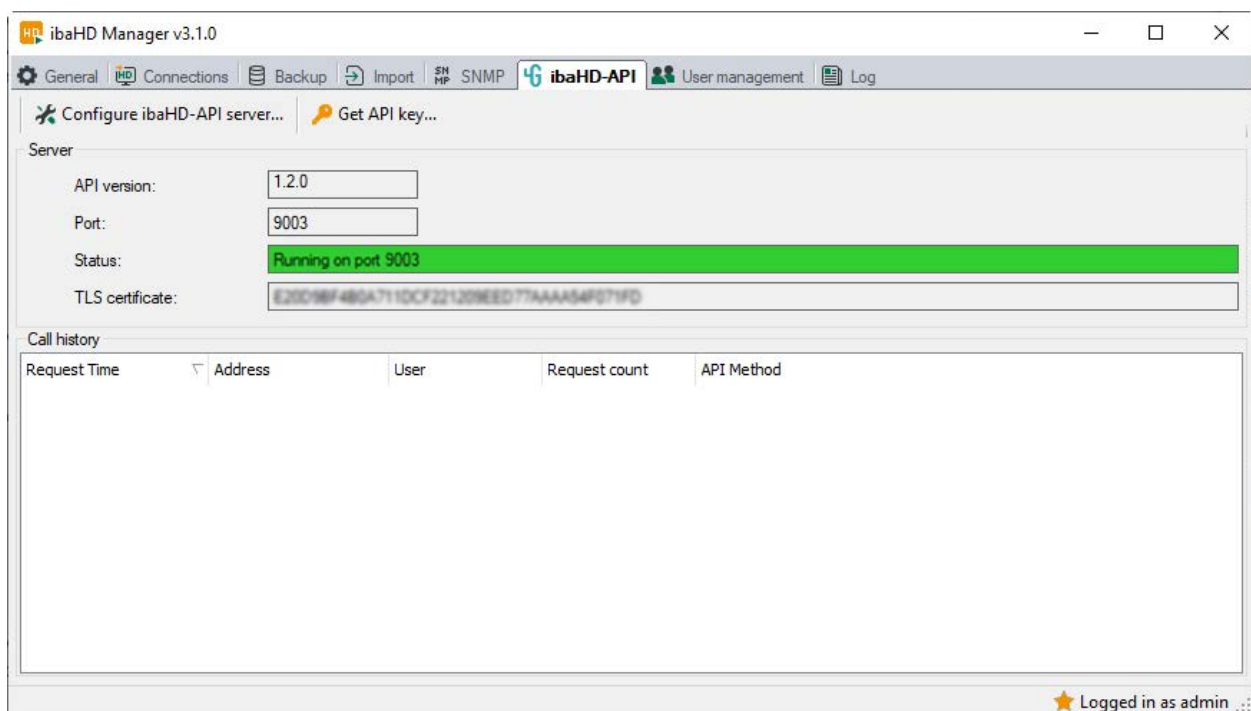
3.2 Configuration

By clicking the *Configure ibaHD-API* button Configure ibaHD-API server... at the top of the tab page the configuration dialog for the *ibaHD-API* settings is displayed.
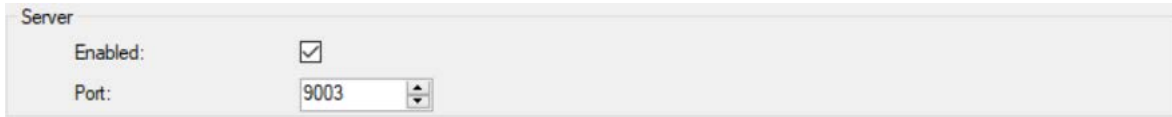
The dialog is split into three configuration areas.

3.2.1   Server

**Enable**
Enable or disable the *ibaHD-API* functionally

Port
Enter another port if the default port 9003 is in use or another port should be used for the *ibaHD-API* communication
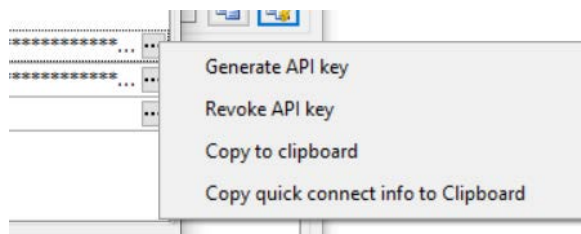

3.2.2   API keys

A list of all the users configured in the ibaHD user management is shown. For each user an API key can be generated in the *API Key* column. This API key is only necessary when user management is enabled.
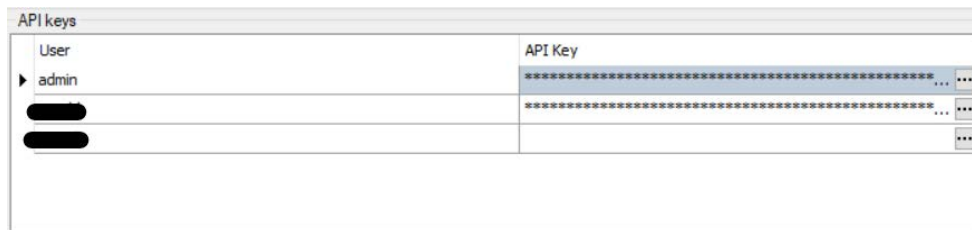
The button provides a context menu to manage the user's API key and connection information:

Generate API key
Generate an API key by clicking on the context menu item if no key was created before or if the existing key shouldn't be used anymore. The previous API key will be replaced by the newly generated one.

Revoke API key
Clear the user's API key.

Copy to clipboard
Click this context menu item to copy the user's API key to clipboard. Use this function to transfer the user's API key to your client application.

Copy quick connect info to clipboard
Click this context menu item to copy all user related connect information to the clipboard. The information such as host, certificate API key and user can now be transferred to the client application as text. It can be used to transfer the connection info to ibaDaVIS v2.8.0 or higher.

3.2.3   Certificates

For a safe way of communication, *ibaHD-API* uses X.509 certificates. A certificate is required to communicate through TLS. TLS certificates can be provided by the server and have to be transferred to the client. A *gRPC-API* communication can only take place in case each communication partner trusts the certificate. You can also register certificates and mark them as "not trusted". The communication with a partner having a "not trusted" certificate will be rejected.

The *Certificates* table displays a list of available certificates. You can manage the certificates here.

You find several buttons in the tool bar of the table. These buttons have the following functions.

Button	Function
+	With this button you open a dialog that allows you to load an existing certificate file. The existing certificate has to contain a private key.
📜	With this button you open a dialog that allows you to generate a new certificate.
→	With this button you can export a certificate to a file. You can then register this in an API-client.
✕	With this button you delete the marked certificate from the table.
✅	With this button you mark the selected certificate as "trusted".
❌	With this button you mark the selected certificate as "not trusted". However, the certificate remains in the table.
4G	With this button you select the certificate that should be used for the API communication.

The symbols in the *Properties* column have the following meaning:

Symbol	Meaning
✅	The certificate is classified as "trusted".
❌	The certificate is not trustworthy.
4G	This certificate is used for the API communication

Example
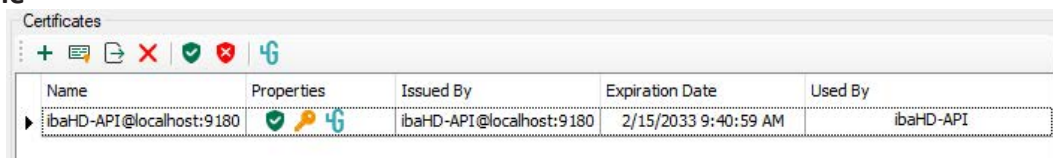
3.2.4   Generate a new certificate

1. Click on the button ⌨ and the following dialog will open
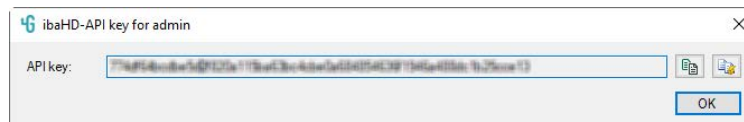
2. Enter a name of your choice for the certificate or keep the default.

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. This standard URI consists of the machine name and the name of
    the application:
    `urn:machinename:applicationName.`

4. Please, define the desired period for the validity (lifetime) of the certificate.
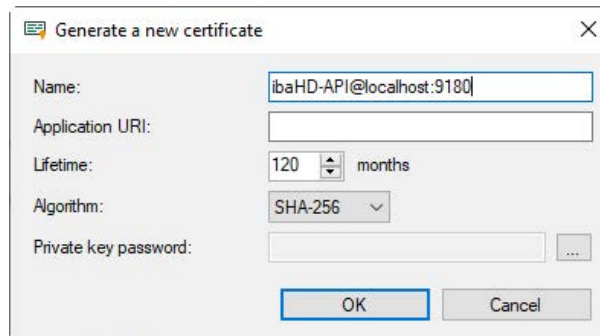
5. Select the desired hash algorithm for the encryption.
    Make sure that the other communication partners support the selected algorithm too.

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. Close the dialog with <OK>.

Now, you are asked, if you want to use the certificate you have just generated as API communi-
cation certificate.

3.3   Get API key

Click the *Get API key* button 🔑 Get API key... at the top of the tab page to open the following dialog.

In this dialog, you see the API key associated with the currently logged in user (in the example it
is the user *admin*).

By clicking the copy button 📋 it is possible to copy the current API key as text to the clipboard.
By clicking the quick connect button 📋 the quick connect information is copied to the clip-
board.

4   ibaHD-API Functionality

Full details of the API can be found in the *ibaHD-API.proto* file. The proto file is plain text and can be opened with any text editor. It contains the complete structure of the API as well as additional comments about message formats.

The location of the *ibaHD-API.proto* file is in the *ibaHD-Server* installation path in the directory *ibaHD-API*.

The following should give an overview over the APIs capabilities.

If the authentication is not correct, the server will reject the request and answer with an error message.

4.1   GetHdStores()

Allows the client to retrieve a list of HD stores of the server. Time-based, event-based HD stores and time period stores are supported, length-based HD stores are not included.

Involved Messages

GetHdStoreRequest
Optionally it is possible to request diagnostic stores

GetHdStoreResponse
A list of all available time- and event HD stores is returned

4.2   GetHdStoreSchema()

Returns the channel hierarchy for the supplied HD store. It is possible to sort the hierarchy by modules or by logical groups. Event store hierarchies are always returned in a logical group structure (folders).

Involved Messages

GetHdStoreSchemaRequest
The store name, like it is part of the response list from the *GetHdStores* response

GetHdStoreSchemaResponse
List of all available analog and digital channels in a time-based HD store, or all available event names in an event-based HD store; Available additional information in the response can be seen in the proto file

4.3   GetRawChannelData()

This API call provides access to raw and unmodified channel data. Possible use cases are machine learning, providing data to other systems or calculations that require unaggregated data.

The request consists of the time range and the channels for which the data should be retrieved. Additionally, the client can specify the number of samples per message, which otherwise would be auto-determined by the server.

The server always responds with a stream of messages. The *max_sample_count_per_message* fieldcan be used to specify how many samples should be sent in a single message before a new message is started (chunking). This allows the client to control the impact on memory consumption when dealing with large response sizes caused by querying data with high sampling rates.

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

When multiple channels are requested, each response message only contains data for a single channel. The order of data for a single channel is guaranteed. The order of messages for channels is not guaranteed. The server tries to balance the messages equally for all requested channels. It may not be fully possible if for example a channel with low and a channel with high sample rate are requested together.

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

The response completes once all messages in the stream have been sent (https://grpc.io/docs/guides/concepts/#server-streaming-rpc).

Involved Messages

GetRawChannelDataRequest
Start and end time of the requested time range in Unix timestamp format in microseconds.

Channel ids in the same format like they were returned in the response list of *GetHdStoreSchema*.

Number of samples per message.

It is possible to request an extra sample before (all channel types) and after (only float and double channels) that can be used for accurately drawing charts.

GetRawChannelDataResponse
List of all stored raw channel data in the defined time range with exact time range, split into the different channel formats

Further information can be seen in the proto file.

4.4 GetAggregatedChannelData()

GetAggregatedChannelData() provides a way to query aggregated channel data for time-based channels. It allows specifying the number of samples that should be sent for a given time range, as well as the selection of min, max and average aggregations.

By default, the API makes use of *ibaHD-Server's* architecture, where each channel is stored in multiple aggregation levels to allow fast and efficient retrieval of data for 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 at once, as well as providing pre-aggregated data for subsequent aggregations where min/max/average data as source is applicable.

The aggregation algorithm can be changed from the default "Min/Max/Avg downsampling" algorithm to the alternative "Linear interpolation downsampling" algorithm. This algorithm tries to select raw data points in an equidistant manner (requested time range / requested samples) and applies linear interpolation between adjacent samples if the stored samples do not exactly match the target sample timestamps.

Note

As this algorithm operates on raw data and therefore *ibaHD-Server* cannot use the aggregation levels to speed up processing, it is more resource intensive than the default "Min/Max/Avg downsampling" algorithm.

Involved Messages

GetAggregatedChannelDataRequest
Start and end time of the requested time range in Unix timestamp format in microseconds.

Channel id in the same format like it was returned in the response list of *GetHdStoreSchema.*

Number of samples which will be returned.

Option to select the aggregated values which will be sent.

The downsampling algorithm can be selected, by default the "Min/Max/Avg downsampling" algorithm is used.

It is possible to request an extra sample before (all channel types) and after (only float and double channels) that can be used for accurately drawing charts.

Further information about possible options can be seen in the proto file.

GetAggregatedChannelDataResponse
List of time stamps for the aggregated samples split into the different channel formats.

4.5   GetEventData()

Provides all event occurrences for the supplied event channels and the given time range. Additional event fields that are stored together with the event can be optionally selected in the request.

Involved Messages

GetEventDataRequest
Start time and end time of the requested time range in Unix timestamp format in microseconds.

Channel ids in the same format like they were returned in the response list of *GetHdStoreSchema.*

Further information about possible options can be seen in the proto file.

GetEventDataResponse

List of all event occurrences in the defined time range with exact time stamp, message text and so on. Further included information depends on request message and can be seen in the proto file.

4.6   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.

Involved Messages

GetLastRecordedChannelValueRequest

Allows specifying the requested channels. The time range can be used to narrow down the search, e.g., to query the last recorded sample within a week. To retrieve the last value that was ever recorded, "time_range_to" can be set to a date in the future or int64 max value.

GetLastRecordedChannelValueResponse

Returns a single raw data point for each requested channel. If no sample was found for a given channel, the response will not contain any data for this channel.

4.7   GetLastEventOccurence()

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

Involved Messages

GetLastEventOccurrenceRequest

Allows specifying the requested event channels. The time range can be used to narrow down the search, e.g., to query the last event occurrence within a week. To retrieve the last event occurrence that was ever recorded, "time_range_to" can be set to a date in the future or int64 max value.

GetLastEventOccurrenceResponse

Returns a single event for each requested event channel. If no event was found for a given event channel, the response will not contain any data for this event channel.

4.8   GetHdTimePeriodStoreSchema()

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

Involved Messages

GetHdTimePeriodStoreSchemaRequest
hd_store_name: The store name of the time-based HD store which is the parent store for the time period store.

time_period_store_name: The name of the time period store, whose info fields and schema should be returned.

include_standard_fields: The custom info fields will be returned by default. With this option the standard info fields will be included in the response message.

locale: For later usage, the option to return the display name of the fields in a requested language.

GetHdTimePeriodStoreSchemaResponse
List of available info fields in time period store, containing the field type, the field name and the display name.

4.9　　GetHdTimePeriodData()

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

Involved Messages

GetHdTimePeriodDataRequest
hd_store_name: The store name of the time-based HD store which is the parent store for the time period store.

time_period_store_name: The name of the time period store of interest.

time_range_from: Start time of the request time range in Unix timestamps format in microseconds.

time_range_to: End time of the request time range in Unix timestamps format in microseconds.

query_mode:

is_start_time_in_time_range: This parameter determines whether the start time of the time periods should be included in the query or if the start time is before the query time range.

is_end_time_in_time_range: This parameter determines whether the end time of the time periods should be included in the query or if the stop time can be later than the end of the query time range.

Include_open:Include time periods that have not been closed in the result or restrict to completed time periods including start and end time.

Column_filter_active: Enable or disable the column filter.

order_by: Order the results with start time in ascending or descending order

Filter:

Info_field_names: Only info fields which are given as filter will be queried. The full list of available field names can be requested using *GetHdTimePeriodStoreSchema.*

Max_sample_count_per_message: This field can be used to specify how many time periods should be sent in a single message before a new message is started (chunk streaming).

limit: This field can be used to specify the maximum number of time periods that can be sent in total. The *order_by* field is used to determine which time periods will be skipped in case more time periods are available

GetHdTimePeriodDataResponse
Returns the list of time period entries with start and end time in Unix timestamps format in microseconds. For time periods with no valid end time a '0' will be returned. All standard info field values will be added by default. The 'Info_field_names' filtered customer info fields are added and ordered by variable type.

4.10 GetHdTimePeriodMetaData()

Returns the info field meta data such as unit or display name and comment for a selected time period store.

Involved Messages

GetHdTimePeriodMetaDataRequest
hd_store_name: The store name of the time-based HD store which is the parent store for the time period store.

time_period_store_name: The name of the time period store, whose info fields should be returned.

Meta_ids: The ids for which the metadata should be returned. The related metadata id of a time period is part of the *GetHdTimePeriodData* response.

locale: For later usage, the option to return the display name of the fields in a requested language.

GetHdTimePeriodMetaDataResponse
List of available meta data definitions of customer info fields for the selected meta id in the requested time period store.

4.11 GetLastHdTimePeriodOccurrence()

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

Involved Messages

GetLastHdTimePeriodOccurrenceRequest
hd_store_name: The store name of the time-based HD store which is the parent store for the time period store.

time_period_store_name: The name of the time period store, whose data is queried.

time_range_from: Start time of the request time range in Unix timestamps format in microseconds.

time_range_to: End time of the request time range in Unix timestamps format in microseconds.

query_mode:

is_start_time_in_time_range: This parameter determines whether the start time of the time periods should be included in the query or if the start time is before the query time range.

is_end_time_in_time_range: This parameter determines whether the end time of the time periods should be included in the query or if the stop time can be later than the end of the query time range.

Include_open:Include time periods that have not been closed in the result or restrict to completed time periods including start and end time.

Column_filter_active: Enable or disable the column filter.

order_by: Order the results with start time in ascending or descending order

Filter:

Info_field_names: If the column filter is active, only the customer info fields in this list will be requested. The available field names can be found in the response list of *GetHdTimePeriodStoreSchema.*

Max_sample_count_per_message: This field can be used to specify how many time periods should be sent in a single message before a new message is started (chunk streaming).

GetLastHdTimePeriodOccurrenceResponse
Returns the last time period in the requested time period store with start and end timestamp in Unix timestamps format in microseconds. For time periods with no valid end time a '0' will be returned. All standard info field values will be added by default. The 'Info_field_names' filtered customer info fields are added and ordered by variable type.

5   Sample Clients

iba AG provides sample clients for the following programming languages:

- C#

- C++

- Python

All sample clients are designed to demonstrate a minimal setup for accessing ibaHD-API, they do not cover all API functionality.

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 usage of ibaHD-API. The integration of ibaHD-API in external systems and general support for programming languages is not provided by iba AG. All terms of ibaHD-Server's license agreement apply.

6   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 licen-se 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