ibaHD-Server-API-Read

gRPC-API-Schnittstelle zur Abfrage historischer Daten und Ereignisse durch eigene Anwendungen

Handbuch

Ausgabe 1.6

Messsysteme für Industrie und Energie

www.iba-ag.com

Hersteller

iba AG

Königswarterstraße 44

90762 Fürth

Deutschland

Kontakte

Zentrale	+49 911 97282-0
Support	+49 911 97282-14
Technik	+49 911 97282-13
E-Mail	iba@iba-ag.com
Web	www.iba-ag.com

Version	Datum	Revision	Autor	Version SW
1.6	09-2025	Redaktionelle Überarbeitung, Daten für nicht-äquidistante Digitalsignale, Alarms & Events, neue Parameter autoClosed und dataMissing, neues Attribut isData für Infofelder	nm	3.5.0

Inhalt

1 Über diese Dokumentation

Diese Dokumentation beschreibt die Verwendung der Programmierschnittstelle *ibaHD-Server-API-Read*.

Andere Dokumentation

Diese Dokumentation ist eine Ergänzung zur *ibaHD-Server*-Dokumentation. Informationen über alle weiteren Eigenschaften und Funktionen von *ibaHD-Server* finden Sie im *ibaHD-Server*-Handbuch, der Online-Hilfe oder im iba-Hilfeportal unter https://docs.iba-ag.com.

1.1 Zielgruppe und Vorwissen

Diese Dokumentation wendet sich an ausgebildete Fachkräfte, die mit dem Umgang mit elektrischen und elektronischen Baugruppen sowie der Kommunikations- und Messtechnik vertraut sind. Als Fachkraft gilt, wer auf Grund der fachlichen Ausbildung, Kenntnisse und Erfahrungen sowie Kenntnis der einschlägigen Bestimmungen die übertragenen Arbeiten beurteilen und mögliche Gefahren erkennen kann.

Diese Dokumentation richtet sich insbesondere an Personen, die mit der Erfassung und Speicherung von Messdaten befasst sind. Für den Umgang mit *ibaHD-Server-API-Read* sind die folgenden Grundkenntnisse erforderlich bzw. sinnvoll:

- Betriebssystem Windows
- Grundkenntnisse *ibaHD-Server*
- Grundkenntnisse in Programmiersprachen, insbesondere C# oder C++ oder Python oder einer anderen Sprache, die von der gRPC-Gruppe angeboten wird

1.2 Schreibweisen

In dieser Dokumentation werden folgende Schreibweisen verwendet:

Aktion	Schreibweise
Menübefehle	Menü *Funktionsplan*
Aufruf von Menübefehlen	*Schritt 1 – Schritt 2 – Schritt 3 – Schritt x* Beispiel: Wählen Sie Menü *Funktionsplan – Hinzufügen – Neuer Funktionsblock*
Tastaturtasten	<Tastenname> Beispiel: <Alt>; <F1>
Tastaturtasten gleichzeitig drücken	<Tastenname> + <Tastenname> Beispiel: <Alt> + <Strg>
Grafische Tasten (Buttons)	<Tastenname> Beispiel: <OK>; <Abbrechen>
Dateinamen, Pfade	`Dateiname`, `Pfad` Beispiel: `Test.docx`

1.3 Verwendete Symbole

Wenn in dieser Dokumentation Sicherheitshinweise oder andere Hinweise verwendet werden, dann bedeuten diese:

Gefahr!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die unmittelbare Gefahr des Todes oder der schweren Körperverletzung!

- Beachten Sie die angegebenen Maßnahmen.

Warnung!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr des Todes oder schwerer Körperverletzung!

- Beachten Sie die angegebenen Maßnahmen.

Vorsicht!

Wenn Sie diesen Sicherheitshinweis nicht beachten, dann droht die mögliche Gefahr der Körperverletzung oder des Sachschadens!

- Beachten Sie die angegebenen Maßnahmen.

Hinweis

Hinweis, wenn es etwas Besonderes zu beachten gibt, wie z. B. Ausnahmen von der Regel usw.

Tipp

Tipp oder Beispiel als hilfreicher Hinweis oder Griff in die Trickkiste, um sich die Arbeit ein wenig zu erleichtern.

Andere Dokumentation

Verweis auf ergänzende Dokumentation oder weiterführende Literatur.

2 Über ibaHD-Server-API-Read

Mit der Schnittstelle *ibaHD-Server-API-Read* können Sie über Anwendungen von Drittanbietern historische Daten und Ereignisse von *ibaHD-Server* abfragen.

Die API basiert auf dem Open-Source-Framework gRPC. Dieses RPC-Framework (Remote Procedure Call) verwendet das HTTP/2-Protokoll und nutzt Protocol Buffers zur Serialisierung.

Andere Dokumentation

Für weiterführende Informationen zu gRPC siehe https://grpc.io/.

Auf der Website finden Sie verschiedene Tools und Anleitungen für die Erstellung von gRPC-Client-Code in verschiedenen Programmiersprachen.

Mit Version 1 (v1) der API können Sie Informationen über die verfügbaren Datenspeicher, die Signalstruktur sowie Kanal- und Ereignisdaten von *ibaHD-Server* abrufen.

Im Folgenden wird für *ibaHD-Server-API-Read* die Kurzbezeichnung *ibaHD-API* verwendet.

Lizenzierung

ibaHD-Server-API-Read ist eine lizenzpflichtige Schnittstelle für *ibaHD-Server*. Fügen Sie die Lizenz zu dem Lizenzcontainer hinzu, der die Basislizenz von *ibaHD-Server* enthält.

Weitere Informationen zur Lizenzierung finden Sie in der *ibaHD-Server*-Dokumentation, oder wenden Sie sich an den iba-Support.

2.1 gRPC

gRPC nutzt Protocol Buffer als IDL (Interface Definition Language), um die Struktur der API und die Nachrichtenformate festzulegen, siehe https://grpc.io/docs/guides/concepts/.

Sie können auf Basis der Schnittstellendefinition Client-Code für verschiedene Programmiersprachen generieren. Eine vollständige Liste der derzeit unterstützten Sprachen finden Sie hier: https://grpc.io/docs/languages/

Verwenden Sie den generierten Client-Code, um eine Verbindung mit der Server-Seite der API herzustellen, die in *ibaHD-Server* integriert ist.

2.2 Nutzung von ibaHD-API

Um die API programmatisch zu nutzen, müssen Sie zunächst den *ibaHD-API*-Client-Code für die gewünschte Programmiersprache generieren.

Hierfür benötigen Sie Folgendes:

- Die Schnittstellendefinitionsdatei `ibaHD-API.proto`.

  Die Proto-Datei wird mit der *ibaHD-Server*-Installation ausgeliefert. Sie finden die Datei im Installationsverzeichnis von *ibaHD-Server* unter folgendem Pfad:

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

- Das gRPC-Tooling, das für die Erstellung des *ibaHD-API*-Clients erforderlich ist.

  Für jede unterstützte Sprache gibt es auf der gRPC-Website eine detaillierte Anleitung, wie Sie den Client-Code generieren können https://grpc.io/docs/tutorials/

Nachdem Sie den *ibaHD-API*-Client-Code generiert haben, können Sie ihn mit der Zielanwendung verknüpfen. Je nach verwendeter Sprache kann es erforderlich sein, zusätzliche gRPC- und ProtoBuf-Basispakete in die Anwendung einzubinden. Einzelheiten hierzu finden Sie in den programmiersprachenspezifischen Leitfäden.

Hinweis

Die gRPC-Toolchain zur Generierung des Client-Codes ist als Open-Source-Software frei verfügbar und kein geistiges Eigentum der iba AG. Die Werkzeuge können ohne vorherige Ankündigung geändert werden.

iba AG empfiehlt eine Sicherungskopie der Toolchain zur Code-Generierung zu erstellen, die zur Generierung des Client-Codes für *ibaHD-API* verwendet wurde.

2.3 Testen der API

Es gibt mehrere GUI-Anwendungen, mit denen gRPC-Schnittstellendefinitionsdateien (.proto) geladen und Anfragen ausgeführt werden können, z. B. BloomRPC, siehe https://github.com/uw-labs/bloomrpc.

So können Sie die API testen, ohne den Client-Code generieren und integrieren zu müssen.

Eine Auswahl ähnlicher Tools finden Sie hier:
https://github.com/grpc-ecosystem/awesome-grpc#tools-gui

Für die Konvertierung von Unix-Zeitstempeln in den verschiedenen Requests und Responses finden Sie im Internet verschiedene Tools. Ein Beispiel ist "Current Millis" für die Umrechnung von Millisekunden, siehe https://currentmillis.com.

Hinweis

Alle diese Tools sind als Open-Source-Software frei verfügbar und kein geistiges Eigentum der iba AG. Sie können ohne vorherige Ankündigung geändert werden. Funktion und Verfügbarkeit werden nicht von der iba AG garantiert.

2.4 Sicherheit beim Datentransport

Die Sicherheit beim Datentransport über das Netz wird über TLS-Zertifikate sichergestellt. Da TLS eine zwingende Voraussetzung für die Funktion von *ibaHD-API* ist, müssen Sie Zertifikate konfigurieren, bevor Sie die API aktivieren können. Clients müssen auf das TLS-Zertifikat des Servers verweisen, um eine Verbindung mit dem API-Endpunkt herstellen zu können.

Informationen zur Erstellung und Konfiguration von Zertifikaten finden Sie im Kapitel ↗ *Konfiguration*, Seite 12.

2.5 Authentifizierung

Wenn die Benutzerverwaltung aktiviert ist, erfolgt die Authentifizierung auf Basis der Benutzer. Um einen Benutzer zu authentifizieren, müssen Sie für den Benutzer einen API-Schlüssel generieren, siehe Kapitel ↗ *Konfiguration*, Seite 12.

Den API-Schlüssel müssen Sie dann bei jeder Anfrage als Header-Feld (im gRPC-Kontext auch Metadatum genannt) mit dem Namen "ibahdapi-apikey" angeben.

2.6 Design-Prinzipien

Das Ziel der *ibaHD-API* ist es, einen generischen Zugriff auf die in *ibaHD-Server* gespeicherten Daten zu ermöglichen. Dieser generische Ansatz ermöglicht die Unterstützung eines breiten Spektrums von Anwendungsfällen.

Kleinere Versionsänderungen der API sind immer kompatibel zueinander und sollten keine Neuerstellung des Client-Codes erfordern, es sei denn, Sie wollen neue API-Funktionen verwenden.

Größere Versionsänderungen enthalten Änderungen in der API, die eine Erneuerung des Client-Codes zwingend erforderlich machen und möglicherweise Änderungen an der Implementierung erfordern.

2.7 Speicherverwaltung

Alle Daten-APIs verfügen über Optionen zur Begrenzung oder Spezifikation der Datenmenge, die in einer einzigen Response Message gesendet werden soll. Es gibt zwar Standardwerte, aber diese Grenzen sind möglicherweise nicht für alle Anwendungsfälle optimal und sollten an die Zielumgebung angepasst werden. Der Speicherbedarf für eine einzelne Response Message hängt von einer Vielzahl von Faktoren ab, wie z. B. der ursprünglichen Abtastrate, dem angeforderten Zeitbereich, der angeforderten Anzahl von Kanälen und im Falle von Ereigniskanälen der Länge der Ereignismeldungen. Daher kann der Speicherbedarf nicht vor der Ausführung bestimmt werden.

iba AG empfiehlt, den Speicherverbrauch für Datenanfragen zu evaluieren und zu testen, sowie zusätzliche Beschränkungen für Endanwender in der Client-Anwendung zu konfigurieren. Indem Sie z. B, den maximalen Zeitbereich oder die maximale Anzahl der Abtastwerte oder Signale begrenzen, erreichen Sie eine erhöhte Betriebssicherheit und optimieren die Anwendung für Ihren Bedarf.

Wenn Sie Begrenzungen deaktivieren oder über die Möglichkeiten der Server-Hardware hinaus erhöhen, kann das den Speicher im System erschöpfen und möglicherweise zu Datenverlusten führen, wenn aktive schreibende Systeme auf der *ibaHD-Server*-Instanz vorhanden sind.

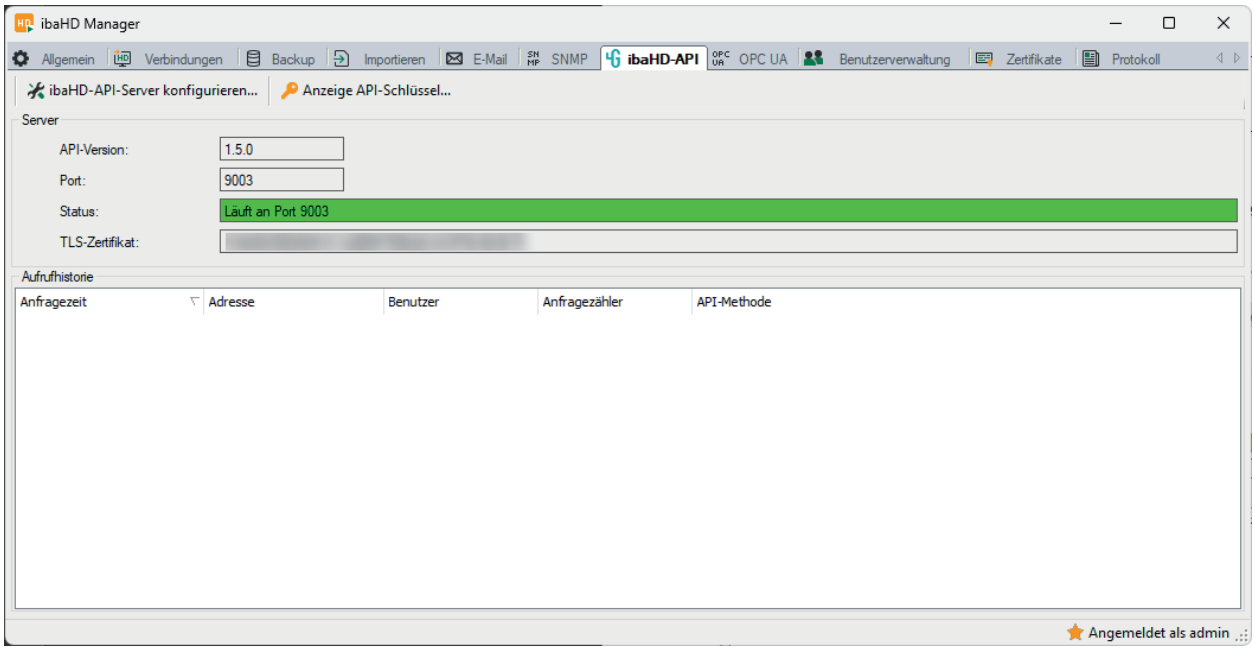
3 ibaHD-API in ibaHD Manager

Die Konfiguration der *ibaHD-API* erfolgt im *ibaHD Manager*.

3.1 Allgemeine Informationen

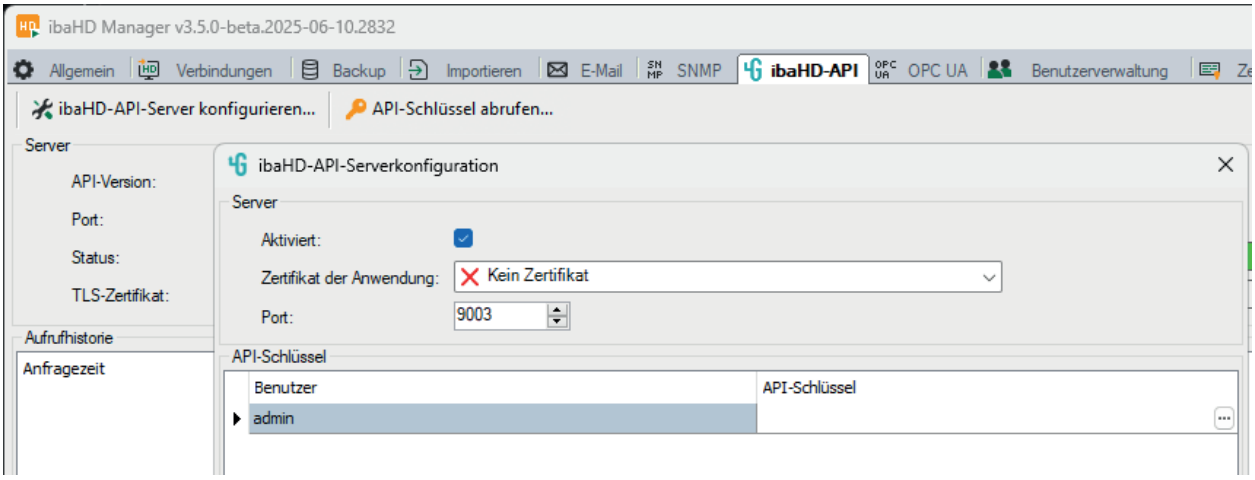
Die aktuelle *ibaHD-API*-Konfiguration wird im Register *ibaHD-API* angezeigt:

API-Version	Zeigt die aktuelle API-Version an
Port	Portnummer für die *ibaHD-API*-Kommunikation
Status	Aktueller Status der API
TLS-Zertifikat	Fingerabdruck des aktuell genutzten TLS-Zertifikats
Aufrufhistorie	Liste der verbundenen gRPC-Clients seit Start des *ibaHD-Server*-Dienstes



3.2 Konfiguration

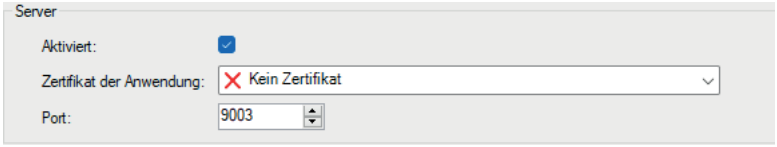
Um die Schnittstelle zu konfigurieren, klicken Sie auf den Button <ibaHD-API-Server konfigurieren...>. Der Dialog *ibaHD-API-Serverkonfiguration* öffnet sich.



Der Dialog ist in die zwei Konfigurationsbereiche *Server* und *API-Schlüssel* unterteilt.

3.2.1 Server

Im Konfigurationsbereich *Server* nehmen Sie folgende Einstellungen für die Schnittstelle *ibaHD-API* vor.



Aktiviert
ibaHD-API-Funktion aktivieren oder deaktivieren

Zertifikat der Anwendung
Wählen Sie aus den folgenden Optionen:

- Zertifikat aus der Liste der verfügbaren Zertifikate wählen
- Neues Zertifikat erzeugen, siehe ↗ *Neues Zertifikat erzeugen*, Seite 14.
- Zertifikate verwalten, siehe ↗ *Zertifikate verwalten*, Seite 13.

Port
Geben Sie einen anderen Port an, wenn der Standardport 9003 bereits verwendet wird oder ein anderer Port für die *ibaHD-API*-Kommunikation verwendet werden soll.

3.2.1.1 Zertifikate verwalten

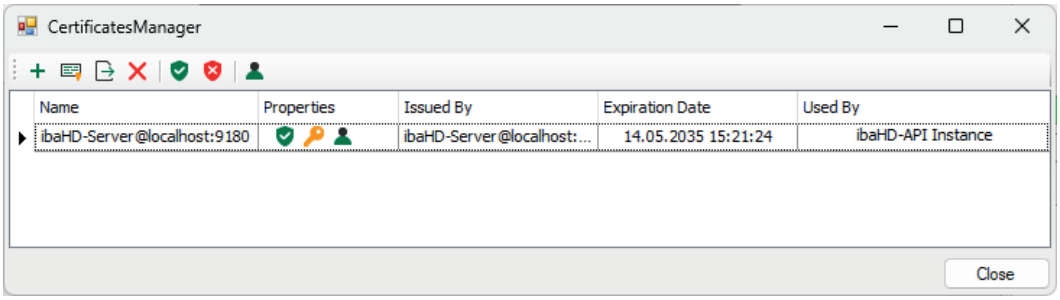
Für eine sichere Kommunikation verwendet *ibaHD-API* X.509-Zertifikate. Für die Kommunikation über TLS ist ein Zertifikat erforderlich. TLS-Zertifikate können vom Server bereitgestellt werden und müssen an den Client übertragen werden. Eine *gRPC-API*-Kommunikation kann nur dann stattfinden, wenn jeder Kommunikationspartner dem Zertifikat vertraut. Sie können auch Zertifikate registrieren und sie als "nicht vertrauenswürdig" kennzeichnen. Die Kommunikation mit einem Partner, der ein "nicht vertrauenswürdiges" Zertifikat besitzt, wird abgewiesen.

Um Zertifikate zu verwalten, wählen Sie im Feld *Zertifikat der Anwendung* die Option *Zertifikate verwalten* wählen. Ein Dialogfenster öffnet sich und zeigt tabellarisch die verfügbaren Zertifikate. Hier können Sie Zertifikate hinzufügen, erstellen und löschen.

In der Symbolleiste der Tabelle finden Sie eine Reihe von Buttons mit folgenden Funktionen:

Button	Funktion
+	Mit diesem Button öffnen Sie einen Dialog, mit dem Sie eine vorhandene Zertifikatdatei laden können. Es werden verschiedene Dateiformate unterstützt (.der, .cer, .crt, .cert, .pem, .pfx, .p12). Falls Sie ein Zertifikat mit einer unbekannten Dateiendung haben, erweitern Sie den Dateifilter auf "*.*" und versuchen Sie die Datei trotzdem zu öffnen. In den meisten Fällen funktioniert es. Das vorhandene Zertifikat muss einen privaten Schlüssel enthalten.
	Mit diesem Button öffnen Sie einen Dialog, mit dem Sie ein neues Zertifikat erzeugen können.
	Mit diesem Button können Sie ein Zertifikat in eine Datei exportieren, um diese dann für Windows oder eine andere Applikation, z. B. auf einem OPC UA-Client, zu registrieren. Auch hier werden mehrere Dateiformate unterstützt.
	Mit diesem Button entfernen Sie das markierte Zertifikat aus der Tabelle.
	Mit diesem Button kennzeichnen Sie das markierte Zertifikat als "vertrauenswürdig".
	Mit diesem Button kennzeichnen Sie das markierte Zertifikat als "nicht vertrauenswürdig". Das Zertifikat bleibt aber trotzdem in der Tabelle des Zertifikatspeichers.
	Mit diesem Button legen Sie fest, ob ein Zertifikat auch zur Benutzerauthentifizierung für OPC UA verwendet werden kann. Nicht relevant für *ibaHD-API*.

Beispiel

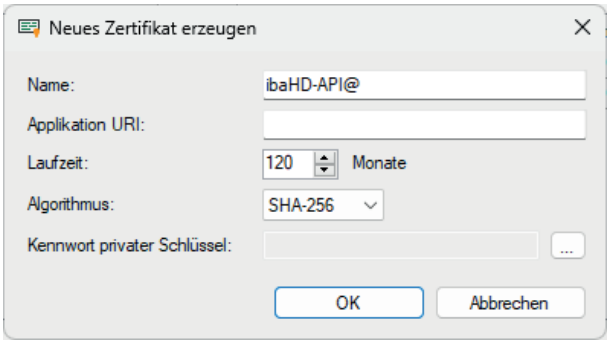


3.2.1.2 Neues Zertifikat erzeugen

Gehen Sie wie folgt vor, um ein neues Zertifikat zu erzeugen.

1. Klicken Sie auf den Button .

→ Der folgende Dialog öffnet sich.



2. Tragen Sie einen beliebigen Namen für das Zertifikat ein oder verwenden Sie den Standardnamen.
3. Tragen Sie bei Bedarf einen Application URI ein.

   Der URI (Uniform Resource Identifier) ist ein global eindeutige Identifikation der Applikation. Wenn Sie dieses Feld nicht ausfüllen, dann wird, sofern vom OPC UA-Client ein Application URI geprüft wird, ein Standard-URI erzeugt, der sich aus Maschinenname und Applikationsname zusammensetzt: `urn:machinename:applicationName`
4. Stellen Sie die gewünschte Laufzeit für die Gültigkeit Ihres Zertifikats ein.
5. Wählen Sie den gewünschten Hash-Algorithmus für die Verschlüsselung aus.

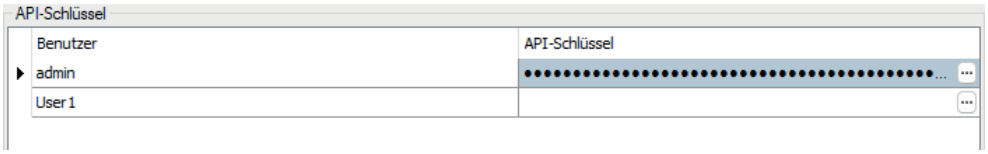
   Zur Auswahl stehen die Algorithmen SHA-256, SHA-384 und SHA-512. Achten Sie darauf, dass die anderen Kommunikationspartner den gewählten Algorithmus auch unterstützen.
6. Legen Sie ein Kennwort für den privaten Schlüssel fest.

   Solange kein Kennwort eingegeben wurde, bleibt der Button <OK> inaktiv. Um das Kennwort zu vergeben, klicken Sie auf den Button <...>, tragen das Kennwort zweimal ein und bestätigen mit <OK>. Für das Kennwort gibt es keine besonderen Anforderungen. Bewahren Sie das Kennwort an einem sicheren Ort auf, damit das selbst erstellte Zertifikat exportiert und für Windows oder andere Applikationen verwendet werden kann.
7. Schließen Sie den Dialog mit <OK>.

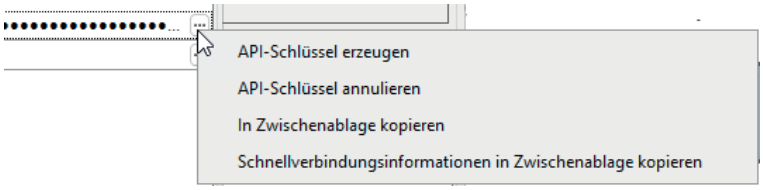
Sie können nun das soeben erstellte Zertifikat als API-Kommunikationszertifikat verwenden.

3.2.2 API-Schlüssel

Der Konfigurationsbereich API-Schlüssel zeigt eine Liste aller in der Benutzerverwaltung von *ibaHD-Server* konfigurierten Benutzer. Sie können für jeden Benutzer einen API-Schlüssel generieren. Dieser ist nur erforderlich, wenn die Benutzerverwaltung aktiviert ist.



Wenn Sie auf den Button  klicken, öffnet sich ein Kontextmenü zur Verwaltung des API-Schlüssels und der Verbindungsinformationen des Benutzers:



API-Schlüssel erzeugen
Neuen API-Schlüssel erzeugen, wenn noch kein Schlüssel erstellt wurde oder der vorhandene Schlüssel nicht mehr verwendet werden soll. Der bisherige API-Schlüssel wird durch den neu generierten Schlüssel ersetzt.

API-Schlüssel löschen
API-Schlüssel des Benutzers löschen.

In Zwischenablage kopieren
Kopiert den API-Schlüssel des Benutzers in die Zwischenablage, z. B. um den API-Schlüssel des Benutzers an die Client-Anwendung zu übertragen.

Schnellverbindungsinformationen in die Zwischenablage kopieren
Alle benutzerbezogenen Verbindungsinformationen in die Zwischenablage kopieren.

So können Sie die Informationen wie Host, Zertifikat, API-Schlüssel und Benutzer als Text an die Client-Anwendung übertragen. Diese Funktion können Sie auch zur Übertragung der Verbindungsdaten an *ibaDaVIS* v2.8.0 oder höher verwenden.

3.3 API-Schlüssel abrufen

Um den API-Schlüssel des aktuell angemeldeten Benutzers abzurufen, klicken Sie auf den Button <API-Schlüssel abrufen...> am oberen Rand des Registers *ibaHD-API* im *ibaHD Manager*.

Folgender Dialog öffnet sich (Beispiel für den Benutzer *admin*):

Um den API-Schlüssel als Text in die Zwischenablage zu kopieren, klicken Sie auf den Button  .

Um die Schnellverbindungsinformationen in die Zwischenablage zu kopieren, klicken Sie auf den Button  .

4 ibaHD-API-Funktionalität

Detaillierte Informationen zur API finden Sie in der Datei `ibaHD-API.proto`. Die Proto-Datei ist eine Textdatei und kann mit jedem Texteditor geöffnet werden. Sie enthält die vollständige Struktur der API sowie zusätzliche Kommentare zu Formaten der Messages.

Sie finden die Datei `ibaHD-API.proto` im Installationspfad von *ibaHD-Server* im folgenden Verzeichnis: `C:\Program Files\iba\ibaHD-Server\ibaHD-API`

Die nachfolgenden Ausführungen geben einen Überblick über die Möglichkeiten der API.

Wenn die Authentifizierung nicht korrekt ist, weist der Server die Anfrage ab und antwortet mit einer Fehlermeldung.

4.1 GetHdStores()

Dieser Aufruf gibt eine Liste der HD-Ablagen des Servers zurück. Unterstützt werden zeitbasierte und ereignisbasierte HD-Ablagen sowie Zeitabschnittsablagen. Längenbasierte HD-Ablagen sind nicht enthalten.

Message "GetHdStoreRequest"

Struktur

```
GetHdStoresRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	include_diagnostic_stores	bool	Fügt Diagnose-Ablagen hinzu, die standardmäßig ausgeblendet sind.

Message "GetHdStoreResponse"

Listet alle verfügbaren zeit- und ereignisbasierten HD-Ablagen, sowie Zeitabschnittsablagen.

Struktur

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

Parameter

Pro Ablage wird eine Message **HdStore** mit folgenden Parametern zurückgegeben.

Pos.	Parameter	Datentyp	Bedeutung
1	hd_store_name	string	Name der HD-Ablage, der auch im ersten Teil einer vollständig qualifizierten Kanal-ID verwendet wird.

Pos.	Parameter	Datentyp	Bedeutung
2	hd_store_type	enum (HdStoreType)	Der Ablagentyp legt fest, welche Art von Daten (Zeitreihen oder Ereignisse) aus der Ablage abgerufen werden können, siehe ↗ *Enum "HdStoreType"*, Seite 17.
3	enabled	bool	Status der Ablage Daten können nur von Ablagen mit dem Parameter *enabled* = TRUE abgerufen werden.
4	active	bool	Gibt zurück, ob aktuell Daten in die Ablage geschrieben werden.
5	is_backup	bool	Gibt zurück, ob es sich bei der Ablage um ein angehängtes Backup handelt.
6	hd_store_parent	string	Name der übergeordneten Ablage, zu der diese Ablage gehört. Wird z. B. für Zeitabschnittsablagen oder Diagnoseablagen verwendet.

Enum "HdStoreType"

Folgende Werte stehen für den Parameter **HdStoreType** zur Verfügung:

Wert		Bedeutung
0	HD_STORE_TYPE_UNSPECIFIED	nicht spezifiziert
1	HD_STORE_TYPE_TIME	zeitbasierte HD-Ablage
2	HD_STORE_TYPE_EVENT	ereignisbasierte HD-Ablage
3	HD_STORE_TYPE_TIME_PERIOD	Zeitabschnittsablage
4	HD_STORE_TYPE_ALARM_EVENT	Alarms & Events-Ablage

4.2 GetHdStoreSchema()

Gibt das Schema der Daten für die angegebene HD-Ablage zurück. Es ist möglich, das Schema nach Modulen oder logischen Gruppen zu sortieren. Schemata ereignisbasierter Ablagen werden immer in einer logischen Gruppenstruktur (Ordnern) zurückgegeben.

Message "GetHdStoreSchemaRequest"

Struktur

```
GetHdStoreSchemaRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	hd_store_name	string	Name der Ablage, wie er in der Response von GetHdStores() enthalten ist, siehe ↗ *GetHdStores()*, Seite 16.

Pos.	Parameter	Datentyp	Bedeutung
2	sort_by	enum (SortByType)	Legt fest, ob die Kanäle nach Modulen oder logischen Gruppen sortiert zurückgegeben werden sollen, siehe ↗ *Enum "SortByType"*, Seite 20.
3	info_fields	bool	Wenn True, dann werden optionale Infofelder für Zeit- oder Ereigniskanäle angefordert.

Message "GetHdStoreSchemaResponse"

Struktur

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

Parameter

Enthält, abhängig von der gewählten Einstellung im Parameter **sort_by**, Module oder die Wurzel der logischen Gruppen. Pro Modul bzw. Gruppe wird eine Message **ChannelGroup** mit folgenden Parametern zurückgegeben.

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Moduls oder der logischen Gruppe
2	group_type	enum (GroupType)	Visualisierungsart der Gruppe oder des Moduls, siehe ↗ *Enum "GroupType"*, Seite 20.
3	channels	-	Gibt für Kanäle aus zeitbasierten Ablagen je Kanal die Message **ChannelDefinition** zurück, siehe ↗ *Message "ChannelDefinition"*, Seite 19. Für andere Ablagentypen leer.
4	events	-	Gibt für Kanäle aus ereignisbasierten Ablagen je Kanal die Message **EventDefinition** zurück, siehe ↗ *Message "EventDefinition"*, Seite 19. Für andere Ablagentypen leer.
5	channel_groups	-	Gibt für Gruppen, die innerhalb dieser Gruppe verschachtelt sind, je eine Message **ChannelGroup** zurück (Baumhierarchie).

Pos.	Parameter	Datentyp	Bedeutung
6	group_number	int32	ID der Gruppe Wird verwendet, wenn *group_type* = GROUP_TYPE_MODULE, GROUP_TYPE_GROUP oder GROUP_TYPE_FOLDER

Message "ChannelDefinition"

Die Message **ChannelDefinition** enthält folgende Parameter.

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Vollständig qualifizierte ID im Format `<HD store name>\<channel id>` Beispiel: `store_1\[0:0]`
2	name	string	Name des Kanals
3	active	bool	Angabe, ob der Kanal aktuell aufgezeichnet wird
4	data_type	enum (DataType)	Datentyp, der gesendet wird, wenn Daten von diesem Kanal angefordert werden, siehe ↗ *Enum "DataType"*, Seite 21.
5	unit	string	Benutzerdefinierte Einheit für den Kanal
6	comment_1	string	Benutzerdefinierter Kommentar
7	comment_2	string	Benutzerdefinierter Kommentar
8	info_fields	-	Optionale Liste der Infofelder des Kanals, siehe ↗ *Message "InfoField"*, Seite 20. Standardmäßig leer, wenn nicht angefordert
9	channel_number	int32	Signalnummer (Teil der *channel_id*).

Message "EventDefinition"

Die Message **EventDefinition** enthält folgende Parameter.

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Vollständig qualifizierte ID im Format `<HD store name>\<event channel id>` Beispiel: `store_1\[0]`
2	name	string	Name des Ereignisses
3	active	bool	Angabe, ob der Kanal aktuell aufgezeichnet wird
4	event_type	enum (EventType)	Art des Ereignisses, siehe ↗ *Enum "EventType"*, Seite 21.
5	comment_1	string	Benutzerdefinierter Kommentar
6	comment_2	string	Benutzerdefinierter Kommentar

Pos.	Parameter	Datentyp	Bedeutung
7	priority	string	Benutzerdefinierte Priorität
8	numeric_fields	-	Definition der numerischen Felder, die bei der Aufzeichnung des Ereignisses erfasst werden sollen, siehe ↗ *Message "FieldDefinition"*, Seite 20.
9	text_fields	-	Definition der Textfelder, die bei der Aufzeichnung des Ereignisses erfasst werden sollen, siehe ↗ *Message "FieldDefinition"*, Seite 20.
10	info_fields	-	Optionale Liste der Infofelder des Ereignisses, siehe ↗ *Message "InfoField"*, Seite 20. Standardmäßig leer, wenn nicht angefordert

Message "InfoField"

Die Message **InfoField** enthält folgende Parameter.

Pos.	Parameter	Datentyp	Bedeutung
1	key	string	Schlüssel des Infofelds
2	values	string	Liste der Infofeldwerte

Message "FieldDefinition"

Die Message **FieldDefinition** enthält folgende Parameter.

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes

Enum "SortByType"

Folgende Werte stehen für **SortByType** zur Verfügung.

Wert		Bedeutung
0	SORT_BY_UNSPECIFIED	nicht spezifiziert
1	SORT_BY_MODULE	Sortiert nach Modulen
2	SORT_BY_LOGICAL_GROUP	Sortiert nach logischen Gruppen

Enum "GroupType"

Visualisierungsart der Gruppe oder des Moduls. Folgende Werte stehen für **GroupType** zur Verfügung.

Wert		Bedeutung
0	GROUP_TYPE_UNSPECIFIED	nicht spezifiziert
1	GROUP_TYPE_HDSTORE_TIME	Zeitbasierte HD-Ablage
2	GROUP_TYPE_HDSTORE_EVENT	Ereignisbasierte HD-Ablage
3	GROUP_TYPE_HDSTORE_ALARM_EVENT	Alarms & Events-Ablage

Wert		Bedeutung
4	GROUP_TYPE_DIAGNOSTIC	Diagnoseablage
10	GROUP_TYPE_ANNOTATIONS	nicht genutzt
11	GROUP_TYPE_ANNOTATION	Modul, das Vermerke enthält
12	GROUP_TYPE_MODULE	Module
13	GROUP_TYPE_VECTOR2D	Vektoren
14	GROUP_TYPE_FOLDER	Modulordner
15	GROUP_TYPE_LOCATION	Messort einer längenbasierten HD-Ablage
16	GROUP_TYPE_EVENTFLOAT	nicht genutzt
17	GROUP_TYPE_EVENTTEXT	nicht genutzt
18	GROUP_TYPE_GROUP	Signalgruppe
19	GROUP_TYPE_BACKUP	nicht genutzt
22	GROUP_TYPE_BACKUPSTART	nicht genutzt
23	GROUP_TYPE_BACKUPSTOP	nicht genutzt
24	GROUP_TYPE_DATFILE	nicht genutzt
25	GROUP_TYPE_INFOHEADER	nicht genutzt
26	GROUP_TYPE_INFOFIELD	nicht genutzt
27	GROUP_TYPE_ASTERISK	nicht genutzt
28	GROUP_TYPE_DISABLED	nicht genutzt

Enum "DataType"

Folgende Werte stehen für **DataType** zur Verfügung.

Wert		Bedeutung
0	DATA_TYPE_UNSPECIFIED	nicht spezifiziert
1	DATA_TYPE_FLOAT_VALUES	Wert vom Typ "float"
2	DATA_TYPE_DOUBLE_VALUES	Wert vom Typ "double"
3	DATA_TYPE_STRING_VALUES	Wert vom Typ "string"
4	DATA_TYPE_DIGITAL_VALUES	Digitaler Wert
5	DATA_TYPE_NE_FLOAT_VALUES	Wert vom Typ "float", der nicht-äquidistant gespeichert wurde
6	DATA_TYPE_NE_DOUBLE_VALUES	Digitaler Wert, det nicht-äquidistant gespeichert wurde
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Anzahl der Wechsel eines digitalen nicht-äquidistanten Werts

Enum "EventType"

Folgende Werte stehen für den Parameter **EventType** zur Verfügung.

Wert		Bedeutung
0	EVENT_TYPE_UNSPECIFIED	nicht spezifiziert
1	EVENT_TYPE_GENERAL_EVENT	Allgemeines Ereignis

Wert		Bedeutung
2	EVENT_TYPE_ANNOTATION	Ereignis vom Typ "Anmerkung"
3	EVENT_TYPE_ALARM	Ereignis vom Typ "Alarm"

4.3 GetRawChannelData()

Dieser API-Aufruf ermöglicht den Zugriff auf Rohdaten und unveränderte Kanaldaten. Mögliche Anwendungsfälle sind:

- Maschinelles Lernen
- Bereitstellung von Daten für andere Systeme
- Berechnungen, für die unaggregierte Daten benötigt werden

Der Server antwortet immer mit einem Nachrichtenstrom. Mit dem Feld *max_sample_count_per_message* können Sie angeben, wie viele Messwerte in einer einzelnen Message gesendet werden sollen. Wenn mehr Datenpunkte im abgefragten Zeitbereich vorhanden sind, werden die Daten auf mehrere Messages verteilt (Chunking).

Der Zeitstempel der ersten Response Message muss nicht genau mit der Startzeit der Abfrage übereinstimmen. Die Antworttelegramme liefern die Zeitstempel zurück, die im *ibaHD-Server* für die angeforderten Signalwerte gespeichert sind.

Wenn mehrere Kanäle angefordert werden, enthält jede Response Message nur Daten für einen einzigen Kanal. Die Reihenfolge der Daten für einen Kanal innerhalb einer Message ist garantiert zeitlich aufeinanderfolgend. Die Empfangs- und Sendereihenfolge der Messages für Daten der angefragten Kanäle ist nicht garantiert.

Die folgende Grafik zeigt den Antwortstrom, wenn mehrere Kanäle angefordert werden.

Der Response ist abgeschlossen, sobald alle Nachrichten im Strom gesendet wurden, siehe https://grpc.io/docs/guides/concepts/#server-streaming-rpc.

Message "GetRawChannelDataRequest"

Struktur

```
GetRawChannelDataRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	time_range_from	int64	Anfangszeit des angeforderten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden.
2	time_range_to	int64	Endzeit des angeforderten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden.

Pos.	Parameter	Datentyp	Bedeutung
3	channel_ids	string	Kanal-IDs im gleichen Format, wie sie in der Response-Liste von *GetHdStoreSchema()* zurückgegeben wurden, siehe ↗ *GetHdStoreSchema()*, Seite 17.
4	max_sample_count_per_message	int64	Maximale Anzahl der Messwerte pro Message Ohne Angabe wird die Anzahl automatisch vom Server bestimmt.
5	add_extra_sample_out_of_time_range	bool	Wenn True, sind zusätzlich zum abgefragten Bereich die nächsten Werte vor und nach dem abgefragten Zeitbereich Teil der Abfrage. Diese Erweiterung des Datenraums ermöglicht eine bessere Darstellung der Daten in den Anzeigen.

Message "GetRawChannelDataResponse"

Liste aller gespeicherten Kanalrohdaten im definierten Zeitbereich mit exaktem Zeitbereich, aufgeteilt auf die verschiedenen Kanalformate.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Kanal-ID im Format `<HD store name>\<channel id>` Beispiel: `store_1\[0:0]`
3	start_timestamp	int64	Unix-Zeitstempel in Mikrosekunden
4	step	int64	Schritt in Mikrosekunden für folgende Werte: DATA_TYPE_FLOAT_VALUES DATA_TYPE_DOUBLE_VALUES 0 für folgende Werte: DATA_TYPE_STRING_VALUES DATA_TYPE_DIGITAL_VALUES
5	data_type	enum (DataType)	Datentyp der zurückgegebenen Werte, siehe ↗ *Enum "DataType"*, Seite 25.

Pos.	Parameter	Datentyp	Bedeutung
6	float_values	float	Äquidistante Werte für DATA_TYPE_FLOAT_VALUES-Kanäle Zeitstempel für Werte können berechnet werden mit: start_timestamp + step * array index
7	double_values	double	Äquidistante Werte für DATA_TYPE_DOUBLE_VALUES-Kanäle Zeitstempel für Werte können berechnet werden mit: start_timestamp + step * array index
8	string_values	-	Nicht-äquidistante Werte für DATA_TYPE_STRING_VALUES-Kanäle Zeitstempel und Werte werden nach Array-Index abgeglichen, siehe ↗ *Message "StringValues"*, Seite 25.
9	digital_values	-	Nicht-äquidistante steigende/fallende Flanken oder Lücken bei DATA_TYPE_DIGITAL_VALUES-Kanälen Der Wert ist bis zum nächsten Flankenwechsel gültig. Zeitstempel und Werte werden nach Array-Index abgeglichen, siehe ↗ *Message "DigitalValues"*, Seite 25.
10	ne_float_values	-	Nicht-äquidistante Werte für DATA_TYPE_NE_FLOAT_VALUES-Kanäle Der Wert ist bis zum nächsten Flankenwechsel gültig. Zeitstempel und Werte werden nach Array-Index abgeglichen, siehe ↗ *Message "NeFloatValues"*, Seite 25.
11	ne_double_values	-	Nicht-äquidistante Werte für DATA_TYPE_NE_DOUBLE_VALUES-Kanäle. Der Wert ist bis zum nächsten Flankenwechsel gültig. Zeitstempel und Werte werden nach Array-Index abgeglichen, siehe ↗ *Message "NeDoubleValues"*, Seite 25.

Message "StringValues"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	value	Nicht-äquidistante String-Werte

Message "DigitalValues"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	float	Nicht-äquidistante Flankenwechsel (0 oder 1) oder Lücken (NaN) des digitalen Kanals
3	edgeCount	float	Nicht-äquidistante Flankenzählung des digitalen Kanals. Gibt an, wie oft sich der Signalwert im aggregierten Zeitintervall geändert hat.
4	min	float	Nicht-äquidistanter Minimalwert des digitalen Kanals
5	max	float	Nicht-äquidistanter Maximalwert des digitalen Kanals

Message "NeFloatValues"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	float	Nicht-äquidistante Werte des Float-Kanals

Message "NeDoubleValues"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	double	Nicht-äquidistante Werte des Double-Kanals

Enum "DataType"

Folgende Werte stehen für **DataType** zur Verfügung.

Wert		Bedeutung
0	DATA_TYPE_UNSPECIFIED	nicht spezifiziert
1	DATA_TYPE_FLOAT_VALUES	Wert vom Typ "float"
2	DATA_TYPE_DOUBLE_VALUES	Wert vom Typ "double"
3	DATA_TYPE_STRING_VALUES	Wert vom Typ "string"
4	DATA_TYPE_DIGITAL_VALUES	Digitaler Wert

Wert		Bedeutung
5	DATA_TYPE_NE_FLOAT_VALUES	Wert vom Typ "float", der nicht-äquidistant gespeichert wurde
6	DATA_TYPE_NE_DOUBLE_VALUES	Digitaler Wert, det nicht-äquidistant gespeichert wurde
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Anzahl der Wechsel eines digitalen nicht-äquidistanten Werts

4.4 GetAggregatedChannelData()

Bietet eine Möglichkeit zur Abfrage aggregierter Kanaldaten für zeitbasierte Kanäle. Sie können die Anzahl der Messwerte festlegen, die für einen bestimmten Zeitbereich gesendet werden sollen, sowie für äquidistante Signale die Minimal-, Maximal- und Durchschnittsaggregationen und für nicht-äquidistante Signale den Flankenzählwert und den anfänglichen Rohwert auswählen.

Standardmäßig nutzt die API die Architektur von *ibaHD-Server*, bei der jeder Kanal in mehreren Aggregationsstufen gespeichert wird, um einen schnellen und effizienten Abruf der Daten über längere Zeiträume zu ermöglichen. Mögliche Anwendungsfälle sind Client-Visualisierungen (z. B. eine Monatsübersicht), bei denen die Bildschirmgröße die Menge der gleichzeitig darstellbaren Daten begrenzt, sowie die Bereitstellung voraggregierter Daten für nachfolgende Aggregationen, bei denen Min/Max/Durchschnittsdaten als Quelle anwendbar sind.

Der Aggregationsalgorithmus kann von dem Standardalgorithmus "Min/Max/Avg downsampling" auf den alternativen Algorithmus "Linear interpolation downsampling" umgestellt werden. Dieser Algorithmus versucht, Rohdatenpunkte in äquidistanten Abständen (geforderter Zeitbereich / geforderter Messwert) auszuwählen und wendet eine lineare Interpolation zwischen benachbarten Messwerten an, wenn die gespeicherten Messwerte nicht genau mit den Zeitstempeln der Zielwerte übereinstimmen.

Hinweis

Der Algorithmus arbeitet je nach abgefragtem Zeitbereich auf der nächstgelegenen Aggregationsstufe oder den Rohdaten. Es kann zu Unschärfen in der Ermittlung des Max-, Min- und Mittelwerts kommen.

Im Fall der Verwendung von aggregierten Daten werden in den Randbereichen mitunter Daten mit in die Ermittlung der Werte einbezogen, die nicht Teil des abgefragten Zeitraums sind. Um die Aufwände zur Ermittlung der exakten Daten in einem für Nutzer vertretbaren Rahmen zu halten, werden die Daten in den Randbereichen unter Zuhilfenahme einer Gewichtungsfunktion ermittelt.

Bei Verwendung von Rohdaten kann *ibaHD-Server* die Aggregationsstufen nicht zur Beschleunigung der Verarbeitung nutzen und ist darum ressourcenintensiver als der Standardalgorithmus "Min/Max/Avg downsampling".

Message "GetAggregatedChannelDataRequest"

Struktur

```
GetAggregatedChannelDataRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	time_range_from	int64	Anfangszeit des angeforderten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden
2	time_range_to	int64	Endzeit des angeforderten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden
3	channel_ids	string	Kanal-ID im gleichen Format, wie sie in der Response-Liste von *GetHdStoreSchema()* zurückgegeben wurde, siehe ↗ *GetHdStoreSchema()*, Seite 17.
4	sample_count	int64	Anzahl der Messwerte, die zurückgegeben werden. Der Wert muss > 0 sein.
5	min_aggregation	bool	Auswahl der aggregierten Werte, die mit der Antwort gesendet werden sollen
6	max_aggregation	bool	
7	avg_aggregation	bool	
8	aggregation_ algorithm_type	enum (AggregationAlgorithmType)	Algorithmus für das Downsampling der Daten, Standardwert MIN_MAX_AVG_ DOWNSAMPLING, wenn keine Angabe gemacht wird
9	add_extra_sample_ out_of_time_range	bool	Wenn True, wird zusätzlich wird vor und nach dem gewünschten Zeitbereich ein zusätzlicher Messwert angefordert. Bei digitalen Kanälen werden keine zusätzlichen Werte hinzugefügt, sondern der erste Wert liegt vor time_range_from. Diese Option ist nur für den Algorithmus MIN_MAX_AVG_DOWNSAMPLING verfügbar.

Message "GetAggregatedChannelDataResponse"

Liste der Zeitstempel für die aggregierten Messwerte, aufgeteilt in die verschiedenen Kanalformate.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	timestamps	int64	Zeitstempel für die Kanäle DATA_TYPE_FLOAT_VALUES und DATA_TYPE_DOUBLE_VALUES als Unix-Zeitstempel in Mikrosekunden
2	aggregated_channels	-	Kanäle, die Werte für die ausgewählten Aggregationen enthalten (min, max, avg) Pro Kanal wird eine Message AggregatedChannel zurückgegeben, siehe ↗ *Message "AggregatedChannel"*, Seite 28.

Message "AggregatedChannel"

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Kanal-ID im Format `<HD store name>\<channel id>` Beispiel: `store_1\[0:0]`
2	data_type	enum (DataType)	Datentyp der zurückgegebenen Werte, siehe ↗ *Enum "DataType"*, Seite 30.
3	float_values	-	Werte für DATA_TYPE_FLOAT_VALUES-Kanäle, ansonsten leer, siehe ↗ *Message "FloatValues"*, Seite 29. Zeitstempel und Werte werden anhand des Array-Index abgeglichen.
4	double_values	-	Werte für DATA_TYPE_DOUBLE_VALUES-Kanäle, ansonsten leer, siehe ↗ *Message "DoubleValues"*, Seite 29. Zeitstempel und Werte werden anhand des Array-Index abgeglichen.
5	string_values	-	Nicht-äquidistante Werte für DATA_TYPE_STRING_VALUES-Kanäle, sonst leer, siehe ↗ *Message "StringValues"*, Seite 29. Zeitstempel und Werte werden nach Array-Index abgeglichen.
6	digital_values	-	Nicht-äquidistante steigende/fallende Flanken oder Lücken für DATA_TYPE_DIGITAL_VALUES-Kanäle, sonst leer, siehe ↗ *Message "DigitalValues"*, Seite 29. Die Werte sind bis zum nächsten Flankenwechsel gültig, Zeitstempel und Werte werden durch den Array-Index aufeinander abgestimmt.

Message "FloatValues"

Pos.	Parameter	Datentyp	Bedeutung
1	min_values	float	Optional: Mindestwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
2	max_values	float	Optional: Maximalwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
3	avg_values	float	Optional: Durchschnittswerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert

Message "DoubleValues"

Pos.	Parameter	Datentyp	Bedeutung
1	min_values	double	Optional: Mindestwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
2	max_values	double	Optional: Maximalwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
3	avg_values	double	Optional: Durchschnittswerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert

Message "StringValues"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	value	Nicht-äquidistante String-Werte

Message "DigitalValues"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	float	Nicht-äquidistante Flankenwechsel (0 oder 1) oder Lücken (NaN) des digitalen Kanals
3	edgeCount	float	Nicht-äquidistante Flankenzählung des digitalen Kanals. Gibt an, wie oft sich der Signalwert im aggregierten Zeitintervall geändert hat.

Pos.	Parameter	Datentyp	Bedeutung
4	min	float	Nicht-äquidistanter Minimalwert des digitalen Kanals
5	max	float	Nicht-äquidistanter Maximalwert des digitalen Kanals

Enum "AggregationAlgorithmType"

Wert		Bedeutung
0	AGGR_ALGO_TYPE_UNSPECIFIED	nicht spezifiziert
1	AGGR_ALGO_TYPE_MIN_MAX_AVG_ DOWNSAMPLING	Min/Max/Avg Downsampling
2	AGGR_ALGO_TYPE_LINEAR_ INTERPOLATION_DOWNSAMPLING	Lineare Interpolation

Enum "DataType"

Folgende Werte stehen für **DataType** zur Verfügung.

Wert		Bedeutung
0	DATA_TYPE_UNSPECIFIED	nicht spezifiziert
1	DATA_TYPE_FLOAT_VALUES	Wert vom Typ "float"
2	DATA_TYPE_DOUBLE_VALUES	Wert vom Typ "double"
3	DATA_TYPE_STRING_VALUES	Wert vom Typ "string"
4	DATA_TYPE_DIGITAL_VALUES	Digitaler Wert
5	DATA_TYPE_NE_FLOAT_VALUES	Wert vom Typ "float", der nicht-äquidistant gespeichert wurde
6	DATA_TYPE_NE_DOUBLE_VALUES	Digitaler Wert, det nicht-äquidistant gespeichert wurde
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Anzahl der Wechsel eines digitalen nicht-äquidistanten Werts

4.5 GetEventData()

Liefert alle Ereignisse für die bereitgestellten Ereigniskanäle und den angegebenen Zeitbereich zurück. Zusätzliche Ereignisfelder, die zusammen mit dem Ereignis gespeichert werden, können optional in der Anfrage ausgewählt werden.

Message "GetEventDataRequest"

Struktur

```
GetEventDataRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	time_range_from	int64	Anfangszeit des angeforderten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden
2	time_range_to	int64	Endzeit des angeforderten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden
3	limit_per_channel	int64	Begrenzung der Anzahl der Ereignisse, die pro Kanal zurückgegeben werden, Standard (Wert = 0) 1000 -1 bedeutet keine Begrenzung
4	order_by	enum (OrderByType)	Chronologische Reihenfolge, in der die Ereignisse zurückgegeben werden, siehe ↗ *Enum "OrderByType" Enum "EventTriggerType"*, Seite 33. Standardmäßig ORDER_BY_TYPE_ DESCENDING, wenn nicht angegeben
5	channel_ids	string	Kanal-IDs im gleichen Format, wie sie in der Response-Liste von *GetHdStoreSchema()* zurückgegeben werden, siehe ↗ *GetHdStoreSchema()*, Seite 17.
6	acknowledgements	bool	Abfrage optionaler Details über die Quittierung von Ereignissen durch den Benutzer
7	numeric_fields	string	Abfrage von optionalen numerischen Feldern in der Response Message für jedes Ereignis, das die angegebenen Felder enthält
8	text_fields	string	Abfrage von optionalen Textfeldern in der Response Message für jedes Ereignis, das die angegebenen Felder enthält

Message "GetEventDataResponse"

Listet alle Ereignisse im definierten Zeitbereich. Pro Ereignis wird eine Message **Event** zurückgegeben mit den folgenden Parametern.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Vollständig qualifizierte ID im Format `<HD store name>\<event channel id>` Beispiel: `store_1\[0]`
2	timestamp	int64	Unix-Zeitstempel in Mikrosekunden, der das Ende des Ereignisses markiert
3	duration	int64	Dauer des Ereignisses in Mikrosekunden bis zum Zeitstempel
4	message	string	Konfigurierte Nachricht oder vom Benutzer eingegebene Anmerkung zum Ereignis
5	trigger	enum (EventTrigger-Type)	Legt fest, ob das Ereignis ein eingehendes oder ein ausgehendes Ereignis ist, siehe ↗ *Enum "OrderByType" Enum "EventTriggerType"*, Seite 33
6	event_ acknowledgement	-	Optionale Angaben zur Bestätigung von Ereignissen durch den Benutzer, siehe ↗ *Message "EventAcknowledgement"*, Seite 32. Standardwert null, wenn nicht angefordert
7	numeric_fields	-	Optionale numerische Feldwerte des Ereignisses, siehe ↗ *Message "NumericField"*, Seite 33. Standardmäßig leer, wenn nicht angefordert
8	text_fields	-	Optionale Textfeldwerte des Ereignisses, siehe ↗ *Message "TextField"*, Seite 33. Standardmäßig leer, wenn nicht angefordert

Message "EventAcknowledgement"

Pos.	Parameter	Datentyp	Bedeutung
1	acknowledged	enum (AckType)	Status der Quittierung
2	acknowledge_com-ment	string	Benutzerdefinierter Kommentar der Quittierung
3	acknowledge_ timestamp	int64	Unix-Zeitstempel in Mikrosekunden, wenn acknowledged = ACK_TYPE_ACKNOWLEDGED, sonst 0
4	acknowledge_os_user	string	OS-Benutzer, der angemeldet war, als das Ereignis quittiert wurde
5	acknowledge_iba_hd-server_user	string	*ibaHD-Server*-Benutzer, der angemeldet war, als das Ereignis quittiert wurde
6	acknowledge_iba_ pda_user_user	string	*ibaPDA*-Benutzer, der angemeldet war, als das Ereignis quittiert wurde

Message "NumericField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "TextField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Enum "OrderByType"

Folgende Werte stehen für **OrderByType** zur Verfügung.

Wert		Bedeutung
0	ORDER_BY_TYPE_UNSPECIFIED	nicht spezifiziert
1	ORDER_BY_TYPE_DESCENDING	Absteigende Reihenfolge
2	ORDER_BY_TYPE_ASCENDING	Aufsteigende Reihenfolge

Enum "EventTriggerType"

Wert		Bedeutung
0	EVENT_TRIGGER_TYPE_UNSPECIFIED	nicht spezifiziert
1	EVENT_TRIGGER_TYPE_INCOMING	Eingehendes Ereignis
2	EVENT_TRIGGER_TYPE_OUTGOING	Ausgehendes Ereignis

Enum "AckType"

Wert		Bedeutung
0	ACK_TYPE_UNSPECIFIED	nicht spezifiziert
1	ACK_TYPE_ACKNOWLEDGED	Status "quittiert"
2	ACK_TYPE_NOT_ACKNOWLEDGED	Status "nicht quittiert"
3	ACK_TYPE_PENDING	Status "ausstehend"

4.6 GetLastRecordedChannelValue()

Gibt den letzten aufgezeichneten Rohdatenwert für alle angeforderten Kanäle innerhalb des angegebenen Zeitbereichs zurück. Insbesondere bei digitalen Kanälen oder Textkanälen, die sich nur selten ändern, können die zurückgegebenen Werte auf Zeitstempel in der Vergangenheit verweisen. Dies bedeutet, dass sich der Wert des Kanals innerhalb des angegebenen Zeitraums nicht geändert hat und weiterhin gültig ist.

Message "GetLastRecordedChannelValueRequest"

Struktur

```
GetLastRecordedChannelValueRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	time_range_from	int64	Unix-Zeitstempel in Mikrosekunden, der begrenzt, wie weit in der Vergangenheit nach dem letzten aufgezeichneten Wert gesucht werden soll 0 für maximalen Bereich lassen
2	time_range_to	int64	Unix-Zeitstempel in Mikrosekunden, gibt den letzten aufgezeichneten Wert vor diesem Zeitstempel zurück Um den letzten jemals aufgezeichneten Wert abzurufen, setzen Sie *time_range_to* auf ein Datum in der Zukunft oder einen int64 Max-Wert.
3	channel_ids	string	Liste der Kanal-IDs im Format `<HD store name>\<channel id>` Beispiel: `store_1\[0:0]`

Message "GetLastRecordedChannelValueResponse"

Gibt einen einzelnen Rohdatenpunkt bzw. die Message **ChannelValue** für jeden angeforderten Kanal zurück. Wenn für einen bestimmten Kanal kein Messwert gefunden wurde, enthält die Response keine Daten für diesen Kanal. Die Message enthält die folgenden Parameter.

Struktur

```
GetLastRecordedChannelValueResponse{
   ChannelValue{
      FloatValue{…}
      DoubleValue{…}
      StringValue{…}
      DigitalValue{…}
   }
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Kanal-ID im Format `<HD store name>\<channel id>` Beispiel: `store_1\[0:0]`
2	timestamp	int64	Unix-Zeitstempel in Mikrosekunden
3	data_type	enum (DataType)	Datentyp des zurückgegebenen Werts, siehe ↗ *Enum "DataType"*, Seite 36.
4	float_value	float	Siehe ↗ *Message "FloatValue"*, Seite 35.
5	double_value	double	Siehe ↗ *Message "DoubleValue"*, Seite 35.
6	string_value	string	Siehe ↗ *Message "StringValue"*, Seite 36.
7	digital_value	float	Siehe ↗ *Message "DigitalValue"*, Seite 36.

Message "FloatValue"

Pos.	Parameter	Datentyp	Bedeutung
1	min_values	float	Optional: Mindestwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
2	max_values	float	Optional: Maximalwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
3	avg_values	float	Optional: Durchschnittswerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert

Message "DoubleValue"

Pos.	Parameter	Datentyp	Bedeutung
1	min_values	double	Optional: Mindestwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
2	max_values	double	Optional: Maximalwerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert
3	avg_values	double	Optional: Durchschnittswerte eines aggregierten Kanals Standardmäßig leer, wenn nicht angefordert

Message "StringValue"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	value	Nicht-äquidistante String-Werte

Message "DigitalValue"

Pos.	Parameter	Datentyp	Bedeutung
1	timestamp	int64	Nicht-äquidistante Unix-Zeitstempel in Mikrosekunden
2	value	float	Nicht-äquidistante Flankenwechsel (0 oder 1) oder Lücken (NaN) des digitalen Kanals
3	edgeCount	float	Nicht-äquidistante Flankenzählung des digitalen Kanals. Gibt an, wie oft sich der Signalwert im aggregierten Zeitintervall geändert hat.
4	min	float	Nicht-äquidistanter Minimalwert des digitalen Kanals
5	max	float	Nicht-äquidistanter Maximalwert des digitalen Kanals

Enum "DataType"

Folgende Werte stehen für **DataType** zur Verfügung.

Wert		Bedeutung
0	DATA_TYPE_UNSPECIFIED	nicht spezifiziert
1	DATA_TYPE_FLOAT_VALUES	Wert vom Typ "float"
2	DATA_TYPE_DOUBLE_VALUES	Wert vom Typ "double"
3	DATA_TYPE_STRING_VALUES	Wert vom Typ "string"
4	DATA_TYPE_DIGITAL_VALUES	Digitaler Wert
5	DATA_TYPE_NE_FLOAT_VALUES	Wert vom Typ "float", der nicht-äquidistant gespeichert wurde
6	DATA_TYPE_NE_DOUBLE_VALUES	Digitaler Wert, det nicht-äquidistant gespeichert wurde
7	DATA_TYPE_DIGITAL_EDGE_VALUES	Anzahl der Wechsel eines digitalen nicht-äquidistanten Werts

4.7 GetLastEventOccurence()

Gibt das letzte Auftreten aller angeforderten Ereignisse innerhalb des angegebenen Zeitbereichs zurück.

Message "GetLastEventOccurrenceRequest"

Struktur

```
GetLastEventOccurranceRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	time_range_from	int64	Unix-Zeitstempel in Mikrosekunden, der angibt, wie weit in der Vergangenheit nach dem letzten Auftreten des Ereignisses gesucht werden soll Für den maximalen Bereich 0 lassen.
2	time_range_to	int64	Unix-Zeitstempel in Mikrosekunden, gibt das letzte Ereignis zurück, das vor diesem Zeitstempel eingetreten ist Um das letzte Ereignis abzurufen, das jemals aufgezeichnet wurde, setzen Sie "time_range_to" auf ein Datum in der Zukunft oder einen int64 Max-Wert.
3	channel_ids	string	Liste der Ereigniskanal-IDs im Format `<HD store name>\<event channel id>` Beispiel `store_1\[0]`
4	acknowledgements	bool	Abfrage optionaler Details über die Quittierung von Ereignissen durch den Benutzer
5	numeric_fields	string	Abfrage von optionalen numerischen Feldern in der Response Message für jedes Ereignis, das die angegebenen Felder enthält
6	text_fields	string	Abfrage optionaler Textfeldern in der in der Response Message für jedes Ereignis, das die angegebenen Felder enthält

Message "GetLastEventOccurrenceResponse"

Gibt ein einzelnes Ereignis bzw. eine Message **Event** für jeden angeforderten Ereigniskanal mit den folgenden Parametern zurück.

Wenn für einen bestimmten Kanal kein Ereignis gefunden wurde, enthält die Response keine Daten für diesen Ereigniskanal.

Struktur

```
GetLastEventOccurrenceResponse{
   EventAcklowledgement{…}
   NumericField{…}
   TextField{…}
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	channel_id	string	Vollständig qualifizierte ID im Format `<HD store name>\<event channel id>` Beispiel: `store_1\[0]`
2	timestamp	int64	Unix-Zeitstempel in Mikrosekunden, der das Ende des Ereignisses markiert
3	duration	int64	Dauer des Ereignisses in Mikrosekunden bis zum Zeitstempel
4	message	string	Konfigurierte Nachricht oder vom Benutzer eingegebene Anmerkung zum Ereignis
5	trigger	enum (EventTrigger-Type)	Legt fest, ob das Ereignis ein eingehendes oder ein ausgehendes Ereignis ist, siehe ↗ *Enum "EventTriggerType"*, Seite 39
6	event_ acknowledgement	-	Optionale Angaben zur Quittierung von Ereignissen durch den Benutzer, siehe ↗ *Message "EventAcknowledgement"*, Seite 32. Standardwert null, wenn nicht angefordert
7	numeric_fields	-	Optionale numerische Feldwerte des Ereignisses, siehe ↗ *Message "NumericField"*, Seite 33. Standardmäßig leer, wenn nicht angefordert
8	text_fields	-	Optionale Textfeldwerte des Ereignisses, siehe ↗ *Message "TextField"*, Seite 33. Standardmäßig leer, wenn nicht angefordert

Message "EventAcknowledgement"

Pos.	Parameter	Datentyp	Bedeutung
1	acknowledged	enum (AckType)	Status der Quittierung
2	acknowledge_ comment	string	Benutzerdefinierter Kommentar der Quittierung
3	acknowledge_ timestamp	int64	Unix-Zeitstempel in Mikrosekunden, wenn acknowledged = ACK_TYPE_ACKNOWLEDGED, sonst 0

Pos.	Parameter	Datentyp	Bedeutung
4	acknowledge_os_user	string	OS-Benutzer, der angemeldet war, als das Ereignis quittiert wurde
5	acknowledge_iba_hdserver_user	string	*ibaHD-Server*-Benutzer, der angemeldet war, als das Ereignis quittiert wurde
6	acknowledge_iba_pda_user_user	string	*ibaPDA*-Benutzer, der angemeldet war, als das Ereignis quittiert wurde

Message "NumericField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "TextField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Enum "EventTriggerType"

Wert		Bedeutung
1	EVENT_TRIGGER_TYPE_UNSPECIFIED	nicht spezifiziert
2	EVENT_TRIGGER_TYPE_INCOMING	eingehendes Ereignis
3	EVENT_TRIGGER_TYPE_OUTGOING	ausgehendes Ereignis

4.8 GetHdTimePeriodStoreSchema()

Gibt die Liste aller verfügbaren Infofelder und ihrer Eigenschaften als Schema für die gewählte Zeitabschnittsablage zurück.

Message "GetHdTimePeriodStoreSchemaRequest"

Struktur

```
GetHdTimePeriodStoreSchemaRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	hd_store_name	string	Eindeutiger Name der zeitbasierten HD-Ablage, die die übergeordnete Ablage für die Zeitabschnittsablage ist
2	time_period_store_name	string	Eindeutiger Name der Zeitabschnittsablage, deren Infofelder und Schema zurückgegeben werden sollen
3	include_standard_fields	bool	Option, um die Standard-Infofelder in die Message zu integrieren. Benutzerdefinierte Infofelder werden immer zurückgegeben.
4	locale	string	Für die spätere Verwendung: Option, den Anzeigenamen der Felder in einer gewünschten Sprache zurückzugeben. Wenn leer, wird die Systemsprache von ibaHD-Server zurückgegeben.

Message "GetHdTimePeriodStoreSchemaResponse"

Liste der verfügbaren Infofelder in der Zeitabschnittsablage. Je Infofeld wird eine Message **InfoFieldDefinition** mit den folgenden Parametern zurückgegeben.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	info_field_data_type	enum (InfoFieldDataType)	Typ des Infofelds, siehe ↗ *Enum "InfoFieldDataType"*, Seite 40.
2	field_name	string	Name des Infofelds
3	display_name	string	Lokalisierter Anzeigename des Infofelds

Enum "InfoFieldDataType"

Folgende Werte stehen für **InfoFieldDataType** zur Verfügung.

Wert		Bedeutung
0	IF_DATA_TYPE_UNSPECIFIED	nicht spezifiziert
1	IF_DATA_TYPE_FLOAT_VALUES	Wert vom Typ "float"
2	IF_DATA_TYPE_DOUBLE_VALUES	Wert vom Typ "double"
3	IF_DATA_TYPE_STRING_VALUES	Wert vom Typ "string"
4	IF_DATA_TYPE_BOOL_VALUES	Wert vom Typ "bool"
5	IF_DATA_TYPE_INT16_VALUES	int16-Wert

Wert		Bedeutung
6	IF_DATA_TYPE_INT32_VALUES	int32-Wert
7	IF_DATA_TYPE_INT64_VALUES	int64-Wert

4.9 GetHdTimePeriodData()

Gibt eine Streaming Response der verfügbaren Zeitabschnitte in einem bestimmten Zeitbereich und einer Zeitabschnittsablage zurück. Die zurückgegebenen Infofelder in der Response können gefiltert werden.

Message "GetHdTimePeriodDataRequest"

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	hd_store_name	string	Eindeutiger Name der zeitbasierten HD-Ablage, die die übergeordnete Ablage für die Zeitabschnittsablage ist.
2	time_period_store_name	string	Eindeutiger Name des Zeitabschnitts, der von Interesse ist.
3	time_range_from	int64	Startzeit des Request-Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden.
4	time_range_to	int64	Endzeit des Request-Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden.
5	query_mode	-	Der Modus legt fest, ob Start- und Endzeit im Abfragezeitbereich enthalten sind oder nicht, siehe ↗ *Message "QueryMode"*, Seite 43. Standardmäßig sind alle Werte falsch und es wird keine Sortierreihenfolge angewendet.
6	filter	-	Filter für angeforderte Infofelder, siehe ↗ *Message "ColumnFilter"*, Seite 44. Standardmäßig wird kein Filter angewendet.
7	max_sample_count_per_message	int64	Angabe, wie viele Zeitabschnitte in einer einzigen Message gesendet werden sollen, bevor eine neue Message gestartet wird (Chunk-Streaming).

Pos.	Parameter	Datentyp	Bedeutung
8	limit	int64	Angabe der maximalen Anzahl von Zeitabschnitten, die insgesamt gesendet werden können. Über das Feld *order_by* können Sie bestimmen, welche Zeitabschnitte übersprungen werden, falls mehr Zeitabschnitte verfügbar sind.

Message "GetHdTimePeriodDataResponse"

Gibt die Liste der Zeitabschnittseinträge mit Start- und Endzeit im Unix-Zeitstempelformat in Mikrosekunden zurück. Für jeden Zeitabschnitt wird eine Message **TimePeriodData** mit folgenden Parametern zurückgegeben.

Alle Standardwerte der Infofelder werden standardmäßig hinzugefügt. Die nach *Info_field_names* gefilterten benutzerdefinierten Infofelder werden hinzugefügt und nach Variablentyp geordnet.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	id	int64	Eindeutige ID des Zeitabschnitts
2	start_time	int64	Unix-Zeitstempel für die Startzeit des Zeitabschnitts in Mikrosekunden.
3	end_time	int64	Unix-Zeitstempel für die Endzeit des Zeitabschnitts in Mikrosekunden. Für Zeitabschnitte ohne gültige Endzeit wird eine "0" zurückgegeben.
4	name	string	Name des Zeitabschnittseintrags
5	start_trigger	double	Absolute Zeit in Sekunden im Verhältnis zur Startzeit
6	stop_trigger	double	Absolute Zeit in Sekunden, zu dem die Startzeit im Verhältnis zur Startzeit eintrat. Der Wert ist 0,0, wenn kein Stopptrigger aufgetreten ist.

Pos.	Parameter	Datentyp	Bedeutung
7	comment	string	Kommentar
8	metadata_id	int32	ID der für diesen Zeitabschnitt verwendeten Metadatengruppe
9	double_fields	-	Infofelder vom Typ "double", siehe ↗ *Message "NumericField"*, Seite 44.
10	int32_fields	-	Infofelder vom Typ "int32", siehe ↗ *Message "Int32Field"*, Seite 44.
11	int64_fields	-	Infofelder vom Typ "int64", siehe ↗ *Message "Int64Field"*, Seite 44.
12	text_fields	-	Textinfofelder, siehe ↗ *Message "TextField"*, Seite 44.
13	digital_fields	-	Infofelder vom Typ "bool", siehe ↗ *Message "DigitalField"*, Seite 45.
14	autoClosed	bool	True, wenn der Zeitabschnitt die maximale Zeitabschnittsdauer vor dem Stopp-Trigger erreicht, und automatisch geschlossen wurde
15	dataMissing	bool	True, wenn Daten innerhalb des Zeitabschnitts fehlen, z. B. wenn die Datenaufzeichnung unterbrochen wurde

Message "QueryMode"

Pos.	Parameter	Datentyp	Bedeutung
1	is_start_time_in_time_range	bool	Legt fest, ob die Startzeit der Zeitabschnitte in die Abfrage einbezogen werden soll oder ob die Startzeit vor dem abgefragten Zeitbereich liegt
2	is_end_time_in_time_range	bool	Legt fest, ob der Endzeitpunkt der Zeitabschnitte in die Abfrage einbezogen werden soll oder ob der Endzeitpunkt später als das Ende des abgefragten Zeitbereichs liegen kann
3	include_open	bool	Angabe, ob nicht abgeschlossene Zeitabschnitte in das Ergebnis einbezogen werden oder nur abgeschlossene Zeiträume mit Start- und Endzeit verwendet werden
4	column_filter_active	bool	Spaltenfilter aktivieren oder deaktivieren
5	order_by	enum (OrderByType)	Ergebnisse nach der Startzeit in aufsteigender oder absteigender Reihenfolge ordnen

Message "ColumnFilter"

Pos.	Parameter	Datentyp	Bedeutung
1	info_field_names	string	Es werden nur die Infofelder abgefragt, die als Filter angegeben sind. Die vollständige Liste der verfügbaren Feldnamen können Sie mit *GetHdTimePeriodStoreSchema()* anfordern, siehe ↗ *GetHdTimePeriodStoreSchema()*, Seite 39.

Message "NumericField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "Int32Field"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	int32	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "Int64Field"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	int64	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "TextField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "DigitalField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	bool	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Enum "OrderByType"

Folgende Werte stehen für **OrderByType** zur Verfügung.

Wert		Bedeutung
0	ORDER_BY_TYPE_UNSPECIFIED	nicht spezifiziert
1	ORDER_BY_TYPE_DESCENDING	Absteigende Reihenfolge
2	ORDER_BY_TYPE_ASCENDING	Aufsteigende Reihenfolge

4.10 GetHdTimePeriodMetaData()

Liefert die Metadaten des Infofelds, wie z. B. die Einheit oder den Anzeigenamen und den Kommentar für eine ausgewählte Zeitabschnittsablage zurück.

Message "GetHdTimePeriodMetaDataRequest"

Struktur

```
GetHdTimePeriodMetaDataRequest{
}
```

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	hd_store_name	string	Eindeutiger Name der zeitbasierten HD-Ablage, die die übergeordnete Ablage für die Zeitabschnittsablage ist
2	time_period_store_name	string	Eindeutiger Name der Zeitabschnittsablage, deren Infofelder zurückgegeben werden sollen.
3	meta_ids	int32	Die IDs, für die die Metadaten zurückgegeben werden sollen Die zugehörige Metadaten-ID eines Zeitabschnitts ist Teil der Response *GetHdTimePeriodData()*, siehe ↗ *GetHdTimePeriodData()*, Seite 41.

Pos.	Parameter	Datentyp	Bedeutung
4	locale	string	Für die spätere Verwendung: Option, den Anzeigenamen der Felder in einer gewünschten Sprache zurückzugeben Wenn leer, wird die Systemsprache von *ibaHD-Server* zurückgegeben.

Message "GetHdTimePeriodMetaDataResponse"

Liste der verfügbaren Metadatendefinitionen von benutzerdefinierten Infofeldern zur ausgewählten Meta-ID innerhalb der angeforderten Zeitabschnittsablage. Pro Metadatendefinition wird eine Message **MetadataField** mit den folgenden Parametern zurückgegeben.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	metadata_id	int32	ID der Metadatengruppe
2	info_field_name	string	Datenbankname des Infofelds
3	info_field_display_name	string	Anzeigename des Infofelds
4	comment_1	string	Kommentar
5	comment_2	sting	Kommentar
6	info_field_unit	string	Einheit des Infofelds
7	info_field_data_type	enum (InfoFieldDataType)	Infofeldtyp, siehe ↗ *Enum "InfoFieldDataType"*, Seite 46

Enum "InfoFieldDataType"

Folgende Werte stehen für **InfoFieldDataType** zur Verfügung.

Wert		Bedeutung
0	IF_DATA_TYPE_UNSPECIFIED	nicht spezifiziert
1	IF_DATA_TYPE_FLOAT_VALUES	Wert vom Typ "float"
2	IF_DATA_TYPE_DOUBLE_VALUES	Wert vom Typ "double"
3	IF_DATA_TYPE_STRING_VALUES	Wert vom Typ "string"
4	IF_DATA_TYPE_BOOL_VALUES	Wert vom Typ "bool"
5	IF_DATA_TYPE_INT16_VALUES	int16-Wert
6	IF_DATA_TYPE_INT32_VALUES	int32-Wert
7	IF_DATA_TYPE_INT64_VALUES	int64-Wert

4.11 GetLastHdTimePeriodOccurrence()

Gibt das letzte Vorkommen eines Zeitabschnitts in einem definierten Zeitbereich für eine ausgewählte Zeitabschnittsablage zurück.

Message "GetLastHdTimePeriodOccurrenceRequest"

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	hd_store_name	string	Eindeutiger Name der zeitbasierten HD-Ablage, die die übergeordnete Ablage für die Zeitabschnittsablage ist.
2	time_period_ store_name	string	Name der Zeitabschnittsablage, deren Daten abgefragt werden.
3	time_range_from	int64	Startzeit des angefragten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden.
4	time_range_to	int64	Endzeit des angefragten Zeitbereichs im Unix-Zeitstempelformat in Mikrosekunden.
5	query_mode	-	Abfragemodus, um festzulegen, ob Start- und Endzeit im Abfragezeitbereich enthalten sind oder nicht, siehe ↗ *Message "QueryMode"*, Seite 49.
6	filter	-	Filter für angeforderte Infofelder, siehe ↗ *Message "ColumnFilter"*, Seite 49.

Message "GetLastHdTimePeriodOccurrenceResponse"

Gibt für den letzten Zeitabschnitt der angeforderten Zeitabschnittsablage die Message **TimePeriodData** mit Start- und Endzeitstempel im Unix-Zeitstempelformat in Mikrosekunden zurück.

Für Zeitabschnitte ohne gültige Endzeit wird eine "0" zurückgegeben.

Alle Standardwerte der Infofelder werden standardmäßig hinzugefügt. Die nach *Info_field_names* gefilterten benutzerdefinierten Infofelder werden hinzugefügt und nach Variablentyp geordnet.

Struktur

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

Parameter

Pos.	Parameter	Datentyp	Bedeutung
1	id	int64	Eindeutige ID des Zeitabschnitts
2	start_time	int64	Unix-Zeitstempel für die Startzeit des Zeitabschnitts in Mikrosekunden.
3	end_time	int64	Unix-Zeitstempel für die Endzeit des Zeitabschnitts in Mikrosekunden. Für Zeitabschnitte ohne gültige Endzeit wird eine "0" zurückgegeben.
4	name	string	Name des Zeitabschnittseintrags
5	start_trigger	double	Absolute Zeit in Sekunden im Verhältnis zur Startzeit
6	stop_trigger	double	Absolute Zeit in Sekunden, zu dem die Startzeit im Verhältnis zur Startzeit eintrat. Der Wert ist 0,0, wenn kein Stopptrigger aufgetreten ist.
7	comment	string	Kommentar
8	metadata_id	int32	ID der für diesen Zeitabschnitt verwendeten Metadatengruppe
9	double_fields	-	Infofelder vom Typ "double", siehe ↗ *Message "NumericField"*, Seite 44.
10	int32_fields	-	Infofelder vom Typ "int32", siehe ↗ *Message "Int32Field"*, Seite 44.
11	int64_fields	-	Infofelder vom Typ "int64", siehe ↗ *Message "Int64Field"*, Seite 44.
12	text_fields	-	Textinfofelder, siehe ↗ *Message "TextField"*, Seite 44.
13	digital_fields	-	Infofelder vom Typ "bool", siehe ↗ *Message "DigitalField"*, Seite 45.
14	autoClosed	bool	True, wenn der Zeitabschnitt die maximale Zeitabschnittsdauer vor dem Stopp-Trigger erreicht, und automatisch geschlossen wurde

Pos.	Parameter	Datentyp	Bedeutung
15	dataMissing	bool	True, wenn Daten innerhalb des Zeitabschnitts fehlen, z. B. wenn die Datenaufzeichnung unterbrochen wurde

Message "QueryMode"

Pos.	Parameter	Datentyp	Bedeutung
1	is_start_time_in_time_range	bool	Legt fest, ob die Startzeit der Zeitabschnitte in die Abfrage einbezogen werden soll oder ob die Startzeit vor dem abgefragten Zeitbereich liegt
2	is_end_time_in_time_range	bool	Legt fest, ob der Endzeitpunkt der Zeitabschnitte in die Abfrage einbezogen werden soll oder ob der Endzeitpunkt später als das Ende des abgefragten Zeitbereichs liegen kann
3	include_open	bool	Angabe, ob nicht abgeschlossene Zeitabschnitte in das Ergebnis einbezogen werden oder nur abgeschlossene Zeiträume mit Start- und Endzeit verwendet werden
4	column_filter_active	bool	Spaltenfilter aktivieren oder deaktivieren
5	order_by	enum (OrderByType)	Ergebnisse nach der Startzeit in aufsteigender oder absteigender Reihenfolge ordnen

Message "ColumnFilter"

Pos.	Parameter	Datentyp	Bedeutung
1	info_field_names	string	Es werden nur die Infofelder abgefragt, die als Filter angegeben sind. Die vollständige Liste der verfügbaren Feldnamen können Sie mit *GetHdTimePeriodStoreSchema()* anfordern, siehe ↗ *GetHdTimePeriodStoreSchema()*, Seite 39.

Message "NumericField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "Int32Field"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	int32	Wert des Feldes
2	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "Int64Field"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	int64	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "TextField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Feldes
2	value	double	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

Message "DigitalField"

Pos.	Parameter	Datentyp	Bedeutung
1	name	string	Name des Fldes
2	value	bool	Wert des Feldes
3	isData	bool	Ist TRUE, wenn der aktuelle Wert entweder wahr oder falsch ist, und ist FALSE, wenn der aktuelle Wert null ist.

5 Sample Clients

iba AG stellt Sample Clients (Beispiel-Clients) für die folgenden Programmiersprachen zur Verfügung:

- C#
- C++
- Python

  Sie können ein Python-Paket herunterladen und in Ihre Python-basierte Anwendung integrieren. Die Funktionen zur Abfrage von Daten aus *ibaHD-Server* adressieren die *ibaHD-Server-API-Read*-Schnittstelle. Die Verwendung des Python-Pakets verringert Ihren Implementierungsaufwand. Das Python-Paket wird von der iba AG gepflegt und aktualisiert.

Alle Sample Clients sind so konzipiert, dass sie eine Minimalkonfiguration für den Zugriff auf die *ibaHD-API* demonstrieren, sie decken nicht alle API-Funktionen ab.

Die Sample Clients sind auf GitHub verfügbar:

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

Disclaimer:
Dieser Code wird "so wie er ist" zur Verfügung gestellt und dient als Beispiel für die Verwendung der *ibaHD-API*. Die Integration der *ibaHD-API* in Fremdsysteme und die generelle Unterstützung von Programmiersprachen wird von iba AG nicht geleistet. Es gelten alle Bestimmungen der Lizenzvereinbarung von *ibaHD-Server*.

6 Support und Kontakt

Support

Tel.: +49 911 97282-14

E-Mail: support@iba-ag.com

Hinweis

Wenn Sie Support benötigen, dann geben Sie bitte bei Softwareprodukten die Nummer des Lizenzcontainers an. Bei Hardwareprodukten halten Sie bitte ggf. die Seriennummer des Geräts bereit.

Kontakt

Hausanschrift

iba AG
Königswarterstraße 44
90762 Fürth
Deutschland

Tel.: +49 911 97282-0

E-Mail: iba@iba-ag.com

Postanschrift

iba AG
Postfach 1828
90708 Fürth

Warenanlieferung, Retouren

iba AG
Gebhardtstraße 10
90762 Fürth

Regional und weltweit

Weitere Kontaktadressen unserer regionalen Niederlassungen oder Vertretungen finden Sie auf unserer Webseite:

www.iba-ag.com