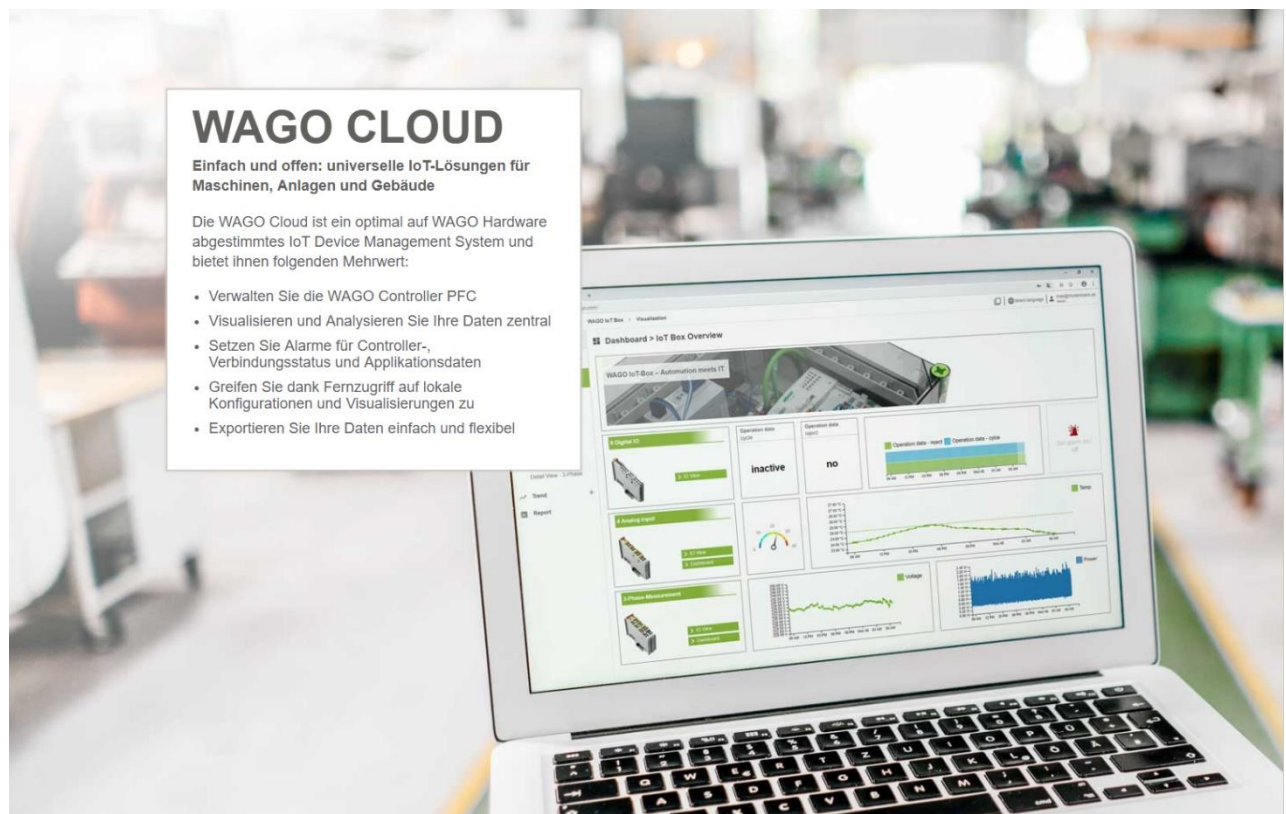


## WAGO Cloud - REST-API Documentation



Version: 2.1

2022-12-06

File name: WAGO Cloud - REST-API Documentation.docx

# WAGO Cloud - REST-API Documentation

## Table of contents

<b>1 Overview .....</b>	<b>4</b>
<b>2 Supported Components by REST-API.....</b>	<b>5</b>
2.1 Overview .....	5
2.2 Definition of Terms .....	5
2.3 Supported Operations by REST-API .....	6
<b>3 REST-API Usage via Swagger .....</b>	<b>13</b>
3.1 Overview .....	13
3.2 REST-API Key .....	14
3.2.1 Create an API Key .....	14
3.3 Authorization via Swagger .....	14
3.4 REST-API Operations.....	15
<b>4 REST-API Scenarios via HTTP requests .....</b>	<b>17</b>
4.1 Authorization via REST-API.....	17
4.2 Get Subscription and Workspace Information .....	18
4.3 Create a Device .....	19
4.4 Get all Commands Supported for a Device .....	19
4.5 Send a Command to the Device .....	21
4.6 Get the Workspace Structure .....	23
4.7 Get Telemetry Data .....	23
<b>5 Public Event Channel .....</b>	<b>25</b>
5.1 Supported Public Event Types .....	25
5.2 Subscribe for a Public Event - POST .....	27
5.3 Sequence of Events: Synchronous Handshake .....	29
5.4 List all Public Event Subscriptions - GET .....	30
5.5 Unsubscribe from a Public Event - DELETE .....	31
5.6 Receive Scheduled CSV Export Events .....	31
5.7 Receive Alarm Events .....	32
5.8 Receive Device State Events .....	32
5.9 Receive Telemetry Data Events .....	34
<b>6 Asynchronous Commands .....</b>	<b>35</b>
6.1 How to use it .....	35
6.2 Device Command Response States .....	38
6.3 Change Timeout Time .....	39
6.4 Retry Pattern for WAGO Cloud Device Command Service .....	39
<b>7 Query the tag information for multiple devices via a batch api .....</b>	<b>40</b>
7.1 General: .....	40
7.2 Request body: .....	40
7.3 Responses: .....	40
7.4 Additional Notes: .....	41



# WAGO Cloud - REST-API Documentation

## 1 Overview

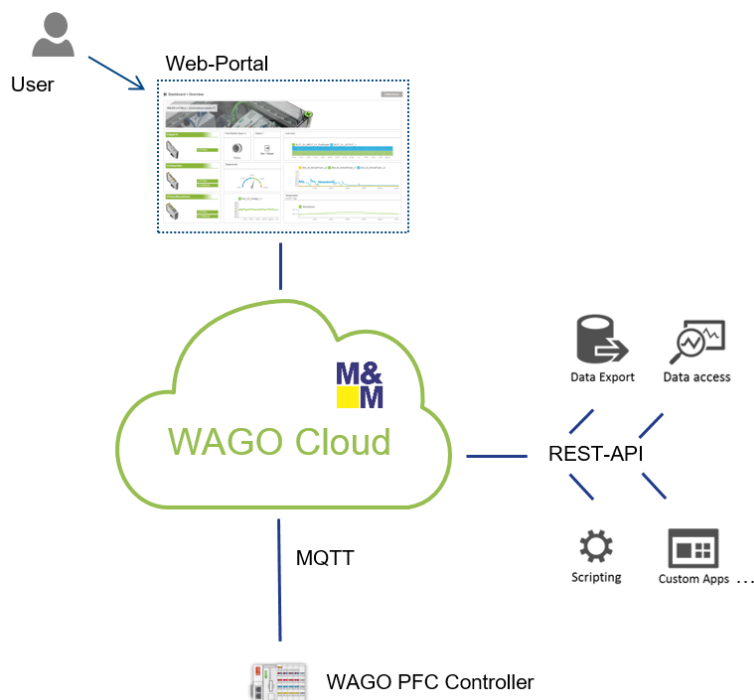
Besides managing the devices in the WAGO Cloud Web-Portal, it is also possible to manage the WAGO Cloud and the data via the REST-API. The REST-API provides the following use cases and functions:

Use cases of the REST-API:

- Scripting  
e.g. automatic device registration
- Custom applications
- Data access on historical data

REST-API Functions:

- Device management and access to device data
- Subscription and workspace management
- Alarm configuration and access to alarm data



This document describes the architecture from the new version of the REST-API and gives some basic example how to register a device and receive data from the device. If you are working with an older version of the REST API v1 (deprecated) or v2 (deprecated), please use the REST-API documentation from 2018-10-01, Version 2.2.

# WAGO Cloud - REST-API Documentation

## 2 Supported Components by REST-API

### 2.1 Overview

The following picture shows the components, which are supported by the REST-API. Each term used in the picture is described in the table followed by the picture.

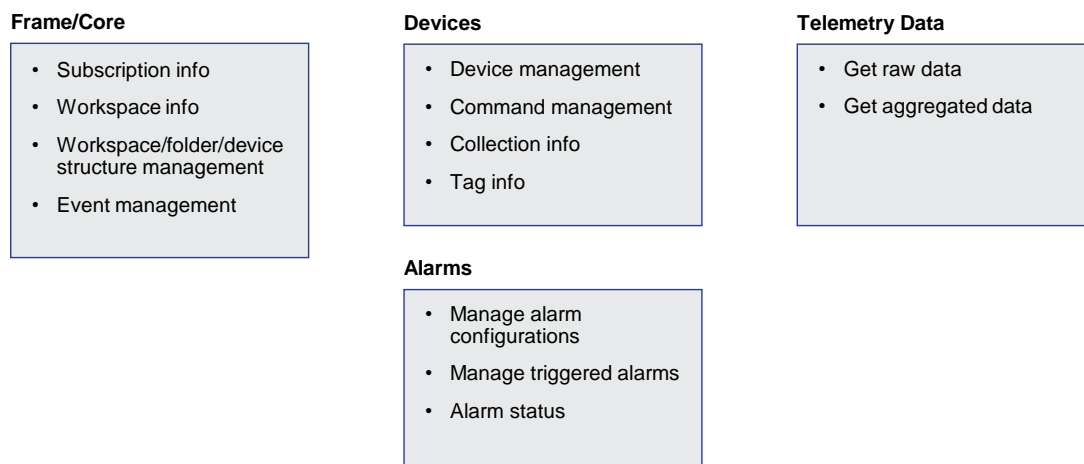


Figure 1: Supported components

### 2.2 Definition of Terms

Term	Definition
<b>Frame/Core</b>	<p>The general functionality in the WAGO Cloud is clustered in the frame/core component, e.g. subscription and workspace management.</p> <p>The REST-API allows to get subscription and workspace information. The management of the workspace structure is also possible. It is also possible to create a subscription that informs about an event by sending callbacks to a specified client.</p>
<b>Device</b>	<p>Devices can be created in the WAGO Cloud. Devices, together with folders, are contained in the workspace structure. Data can be sent from the physical device. The sent data use collections and tags for the transmission of device data. Commands can be executed on the device.</p> <p>The REST-API allows to get device info and create or delete devices. Commands can be read and triggered on the device. Collections and tags can be also created, updated and deleted. Configuration of device is also possible. Developer can use API to configure device instead of sending a tag configuration message.</p>

## WAGO Cloud - REST-API Documentation

Term	Definition
<b>Telemetry data</b>	<p>The physical device can send telemetry data to the cloud. The telemetry data can be visualized in the WAGO Cloud, e.g. in a dashboard or in a trend.</p> <p>The REST-API allows to export the raw data from the device and the aggregated data from the device. You can use the data in any other 3<sup>rd</sup> party tool.</p>
<b>Alarms</b>	<p>The WAGO Cloud contains an alarm management app. Alarm configuration can be created with several alarm rules. While sending telemetry data from devices, the alarm configurations will be checked and if a rule is met an alarm in the WAGO Cloud will be created.</p> <p>The REST-API supports the management of the alarm configurations and the management of triggered alarms.</p>

The access to the WAGO Cloud components is always restricted by the WAGO Cloud user management. An API key (primary or secondary) is necessary Access to WAGO Cloud is always via a user.

### 2.3 Supported Operations by REST-API

App / Category	Method / Operation	Description
<b>Core 1.0 - Subscriptions</b>	GET /api/core/subscriptions	Gets all subscription information the user does have access to.
	GET /api/core/subscriptions/{id}	Gets the subscription information for the specified id.
	GET /api/core/subscriptions/{id}/workspaces	Gets a list of all workspaces for the subscription with the given id.
	PUT /api/core/subscriptions/{id}/apps	Register or update the registration of an existing application.
	GET /api/core/subscriptions/{id}/apps	Gets the list of all registered apps that were registered for the subscription with the given id.
	DELETE /api/core/subscriptions/{id}/apps/{name}	Deregisters the app with the given name from the subscription with the given id.

## WAGO Cloud - REST-API Documentation

App / Category	Method / Operation	Description
Core 1.0 - Workspaces	GET /api/core/workspaces/{id}/structure	Gets the workspace structure for the workspace with the given id.
	POST /api/core/workspaces/{id}/structure	Creates a folder in the workspace structure.
	PUT /api/core/workspaces/{id}/structure	Updates a node in the workspace structure.
	GET /api/core/workspaces/{id}/structure/search	Searches for objects (devices/folder) within the workspace with the given id.
	GET /api/core/workspaces/{id}/structure/{nodeId}	Searches for the structure node with the given node id in the workspace with the given workspace id.
	GET /api/core/workspaces/{id}/structure/{nodeId}/additional-properties	Gets the additional properties for a specific node folder or device, collection or tag.
	PUT /api/core/workspaces/{id}/structure/{nodeId}/additional-properties	Creates or updates the additional properties for a specific node folder or device, collection or tag.
	GET /api/core/workspaces/{id}/structure/{nodeId}/aggregated-tags	Gets the configured structural tag aggregates for a folder, device or workspace
	DELETE /api/core/workspaces/{folderId}/folder	Deletes the specified folder and its sub-folders.
	DELETE /api/core/workspaces/{id}/structure/{nodeId}/additional-properties/{propertyName}	Deletes the additional properties for a specific node folder or device, collection or tag with the given property name.
	GET /api/core/workspaces/{id}/feature-permissions	Gets the feature permissions the calling user does have within the workspace with the given identifier.
Core 1.0 - Events	POST /api/core/eventsubscriptions	Creates a subscription that informs about an event by

## WAGO Cloud - REST-API Documentation

App / Category	Method / Operation	Description
		sending callbacks to a specified client.
	GET /api/core/eventsubscriptions	Lists all currently active event subscriptions.
	DELETE /api/core/eventsubscriptions/{id}	Deletes a previously created event subscription.
<b>Device 1.0 – Devices</b>	POST/api/deviceapp/batch	Execute batch requests related to the Devices. It support multiple API requests set in the body and request can depends on others in the batch list.
	GET /api/deviceapp/devices	Gets all devices for the given workspace the user does have access to.
	POST /api/deviceapp/devices	Creates a device in the workspace with the given id.
	GET /api/deviceapp/devices/{id}	Returns the device with the given id.
	PUT /api/deviceapp/devices/{id}	Updates properties of a device with the given id.
	DELETE /api/deviceapp/devices/{id}	Deletes the device with given id.
	GET /api/deviceapp/devices/{id}/commands	Gets all commands supported by the device with the given id.
	POST /api/deviceapp/devices/{id}/commands/{command}	Sends the specified command request to the device with the given id.
	GET /api/deviceapp/devices/{deviceId}/commandResponse/{commandResponseId}	Gets the specified command response asynchronously with the given command response id
	POST /api/deviceapp/devices/{deviceId}/commandsAsync/{command}	Sends the specified command request asynchronously to the device with the given parameters.



## WAGO Cloud - REST-API Documentation

App / Category	Method / Operation	Description
	GET /api/deviceapp/devices/{id} /collections	Gets all collections for the device with the given id.
	PUT /api/deviceapp/devices/{id} /collections	Creates/Updates the collections for the device with the given id.
	GET /api/deviceapp/devices/{id} /collections/{key}/tags	Gets the tags for the specified collection for the device with the given id.
	PUT /api/deviceapp/devices/{id} /collections/{key}/tags	Creates/Updates the tags for the device collection.
<b>Device 2.0 – Devices</b>	POST/api/deviceapp/devices/{id} /commands/{command}	Sends the specified command request to the device with the given id.
	GET/api/deviceapp/devices/{deviceId} /commandResponse /{commandResponseId}	Gets the specified command response asynchronously with the given command response id
	POST/api/deviceapp/devices /{deviceId}/commandsAsync /{command}	Sends the specified command request asynchronously to the device with the given parameters.
<b>Telemetry 1.0 Telemetry Data</b>	POST /api/telemetry/telemetrydata/raw	Gets the raw data values for all given tag descriptions.
	POST /api/telemetry/telemetrydata /aggregates/datapoints	Gets the aggregated data values for the given tag descriptions. The result will be packed into the given number of even sized datapoints.
	POST /api/telemetry/telemetrydata /aggregates/timeinterval	Gets the aggregated data values for the given tag descriptions. The result will be aggregated into time buckets of the given time interval.
	POST /api/telemetry/telemetrydata /aggregates/timewindow	Get aggregates with windowing.

## WAGO Cloud - REST-API Documentation

App / Category	Method / Operation	Description
	POST /api/telemetry/telemetrydata/latest	Gets for given tags the latest telemetry data values.
	POST /api/telemetry/telemetrydata/at	Gets the data values for the given tags at the specified point in time.
	POST /api/telemetry/telemetrydata/structuralaggregates/latest	Gets structural aggregation based on the last values.
	POST /api/telemetry/telemetrydata/structuralaggregates/timewindow	Get structural aggregates with windowing
<b>Alarm 1.0 - Alarm Configurations</b>	GET /api/alarmapp/alarmconfigurations	Gets the existing alarm configurations for the given workspace or device respectively.
	POST /api/alarmapp/alarmconfigurations/plcStatusBased	Create a plc status based alarm configuration.
	PUT /api/alarmapp/alarmconfigurations/plcStatusBased/{id}	Updates a plc status based alarm configuration.
	POST /api/alarmapp/alarmconfigurations/valueBased	Create a value based alarm configuration.
	PUT /api/alarmapp/alarmconfigurations/valueBased/{id}	Updates a value based alarm configuration.
	POST /api/alarmapp/alarmconfigurations/connectionStatusBased	Create a connection status based alarm configuration.
	PUT /api/alarmapp/alarmconfigurations/connectionStatusBased/{id}	Updates a connection status based alarm configuration.
	DELETE /api/alarmapp/alarmconfigurations/{id}	Deletes the alarm configuration with the given id.
<b>Alarm 1.0 – Alarms</b>	GET /api/alarmapp/alarms	Gets a list of all raised alarms for the given workspace or device respectively.

## WAGO Cloud - REST-API Documentation

App / Category	Method / Operation	Description
	DELETE /api/alarmapp/alarms	Deletes all the alarms for the given workspace or device respectively.
	DELETE /api/alarmapp/alarms/{id}	Deletes the alarm with the given id.
	PATCH /api/alarmapp/alarms/{id}	Change the status of the alarm with the given id.
<b>Alarm 2.0 - Alarm Configurations</b>	GET /api/alarmapp/alarmconfigurations	Gets the configured alarm configurations for either the workspace or the device respectively.
	POST /api/alarmapp/alarmconfigurations/valueBased	Creates a value based alarm configuration.
	PUT /api/alarmapp/alarmconfigurations/valueBased/{id}	Updates the alarm configuration identified with the given id with the given configuration data.
	POST /api/alarmapp/alarmconfigurations/timeIntervalBased	Creates a time interval based alarm configuration.
	PUT /api/alarmapp/alarmconfigurations/timeIntervalBased/{id}	Updates the alarm configuration identified with the given id with the given configuration data.
	POST /api/alarmapp/alarmconfigurations/timeoutBased	Creates an alarm configuration of the type timeout based within the workspace with the given identifier.
	PUT /api/alarmapp/alarmconfigurations/timeoutBased/{id}	Updates the alarm configuration with the given identifier.
	POST /api/alarmapp/alarmconfigurations/alarmFlagsBased	Creates an alarm flags alarm configuration for the workspace.
	PUT /api/alarmapp/alarmconfigurations/alarmFlagsBased/{id}	Updates the alarm flags alarm configuration for the workspace.

## WAGO Cloud - REST-API Documentation

---

App / Category	Method / Operation	Description
<b>Alarm 2.0 - Alarm Status</b>	GET /api/alarmapp/alarmstatus	Gets the list of alarm statuses created for the given subscription
	PUT /api/alarmapp/alarmstatus	Creates/Updates alarm statuses for the given subscription

# WAGO Cloud - REST-API Documentation

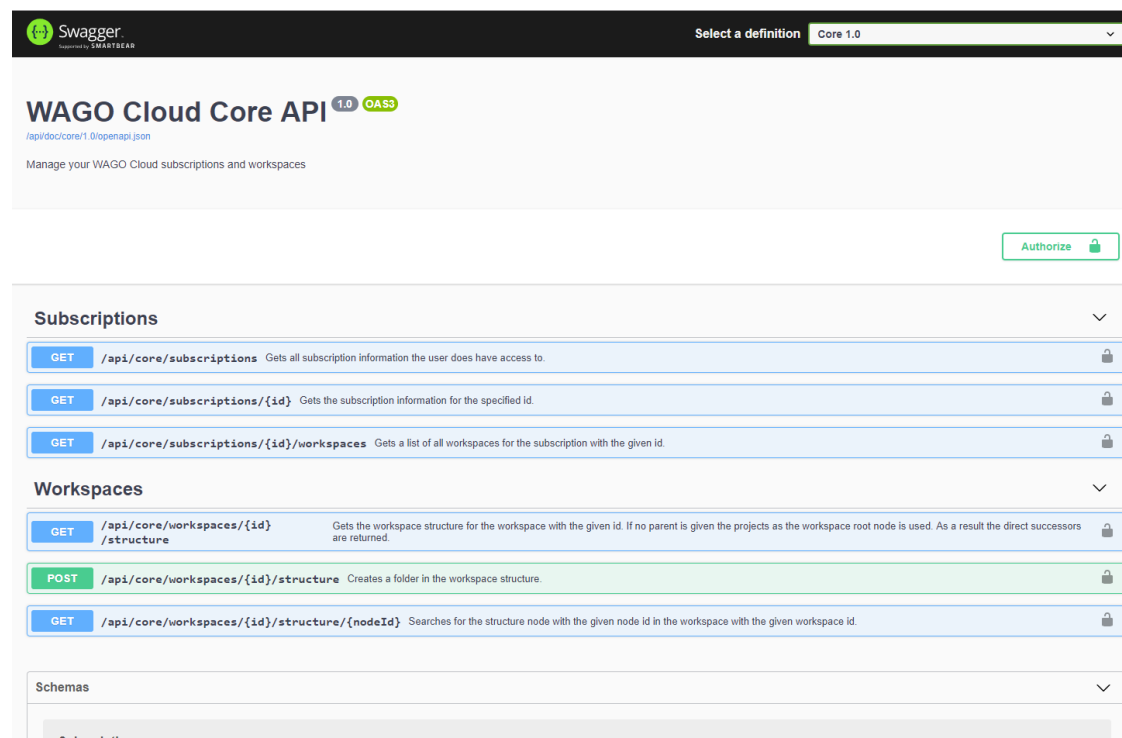
## 3 REST-API Usage via Swagger

### 3.1 Overview

Swagger is a tool to manage REST-APIs. The WAGO Cloud REST-API operations can be used via the Swagger UI, the detailed documentation of the REST-API is also part of the Swagger UI. You can open the WAGO Cloud REST-API via the following link:

<https://cloud.wago.com/api/doc/index.html#/>

The Swagger UI Timeout is set to 60 Minutes. After that a new login is required.



The new REST-API was introduced with WAGO Cloud Release 2.2. The following components are supported:

#### Release 2.2

- Core (Frame)
- Devices
- Telemetry
- Alarm

The REST-API operations before release 2.2 should no longer be used and they are marked as deprecated:

- V1 (deprecated) - WAGO Cloud API
- V2 (deprecated) - WAGO Cloud API

Below sections describe the usage of a REST-API key, the authorization and the usage of REST-API operations for the WAGO Cloud.

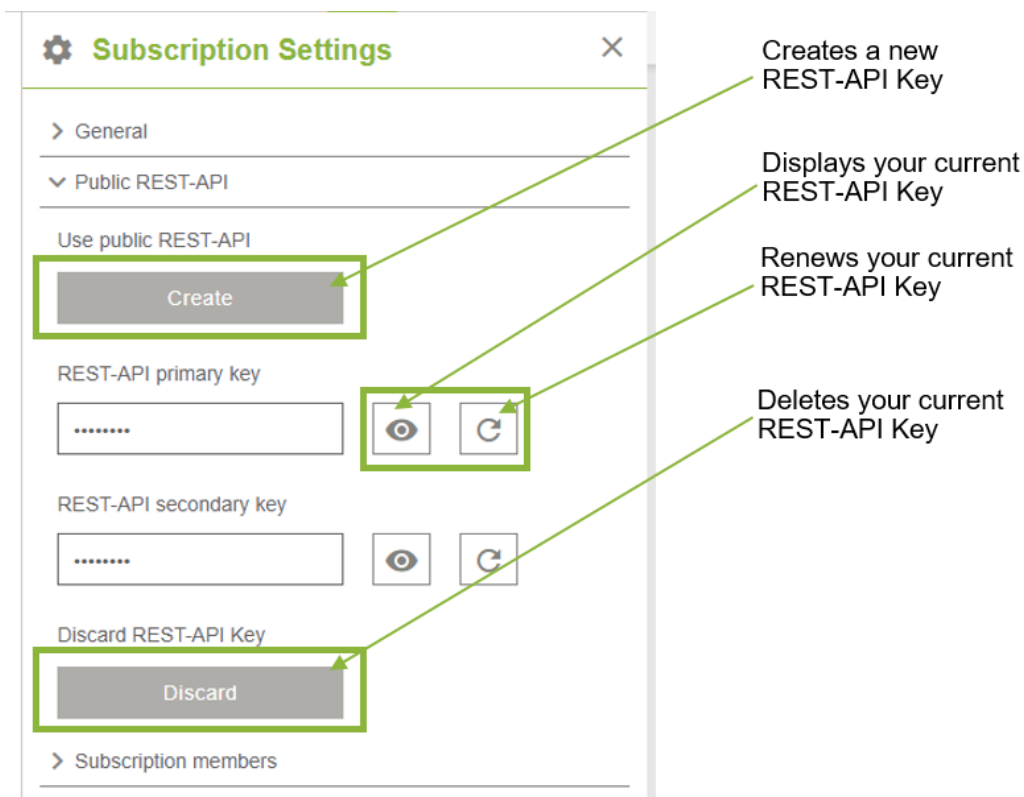
## WAGO Cloud - REST-API Documentation

### 3.2 REST-API Key

A new REST-API key is required to be able to use the new version of the REST-API. This key uniquely assigns the REST-API to a subscription. The following steps explain shortly how the key is generated and used.

#### 3.2.1 Create an API Key

The first thing you need to do is to login to the WAGO Cloud and open the subscription settings. Open the tab "REST-API" and click on "Create". A new key is generated for this subscription. Make sure that you are Subscription admin, because only then you can create, delete or renew a key. As a normal user you can only see the created key. The secondary key can be used if the primary key is currently not available.



The screenshot shows the "Subscription Settings" dialog box with the "Public REST-API" tab selected. The "Use public REST-API" section contains a "Create" button, which is highlighted with a green box and an arrow pointing to the text "Creates a new REST-API Key". Below this, the "REST-API primary key" is displayed as a masked string (.....). To the right of the primary key are two icons: an eye icon (highlighted with a green box and an arrow pointing to "Displays your current REST-API Key") and a refresh icon (highlighted with a green box and an arrow pointing to "Renews your current REST-API Key"). Below the primary key, the "REST-API secondary key" is also displayed as a masked string (.....). At the bottom of the "Use public REST-API" section, there is a "Discard REST-API Key" section with a "Discard" button, highlighted with a green box and an arrow pointing to the text "Deletes your current REST-API Key". The "Subscription members" section is visible at the bottom of the dialog.

### 3.3 Authorization via Swagger

Open the REST-API [Swagger UI](#) and click on "Authorize". Select the checkbox of the Scope and click again on "Authorize". You will be redirected to a Microsoft page where you will need to login with your WAGO Cloud credentials.

**NOTE:** Do not change the "client\_id", see below screenshot.

# WAGO Cloud - REST-API Documentation

Available authorizations

X

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.

API requires the following scopes. Select which ones you want to grant to Swagger UI.

### OAuth2 (OAuth2, implicit)

Application: FrontendSettings-Deployment-Name

Authorization URL: [https://te.cpim.windows.net/14bb95ba-3fd1-4abe-9e6e-e85e8e6f8e9e/B2C\\_1\\_wctsignuppolicy/oauth2/v2.0/authorize](https://te.cpim.windows.net/14bb95ba-3fd1-4abe-9e6e-e85e8e6f8e9e/B2C_1_wctsignuppolicy/oauth2/v2.0/authorize)

Flow: implicit

client\_id:

96df7cf2-3d1f-4787-9817-05e

Scopes:

☒

<https://wctb2c.onmicrosoft.com/2ca3876d-485d-4022-a5f4-18a7d3c8a864/read>

Default

Authorize

Close

After you have successfully signed up you have to copy the REST-API key from the WAGO Cloud into the appropriate field and click again on *"Authorize"*.

**ApiKeyAuth (apiKey)**

Name: api-key

In: header

Value:

Insert your REST-API Key here

[Authorize](#) [Close](#)

You can now work with the WAGO Cloud REST-API. This is a paid feature. For detailed information see the [WAGO Online Calculator](#).

### 3.4 REST-API Operations

To execute the WAGO Cloud REST-API operations, you have to choose the appropriate definition, e.g. Core, Device, Telemetry or Alarm and then select the operation.

Subscriptions			
GET	/api/core/subscriptions	Gets all subscription information the user does have access to.	
GET	/api/core/subscriptions/{id}	Gets the subscription information for the specified id.	
GET	/api/core/subscriptions/{id}/workspaces	Gets a list of all workspaces for the subscription with the given id.	
Workspaces			
GET	/api/core/workspaces/{id}/structure	Gets the workspace structure for the workspace with the given id. If no parent is given the projects as the workspace root node is used. As a result the direct successors are returned.	
POST	/api/core/workspaces/{id}/structure	Creates a folder in the workspace structure.	
GET	/api/core/workspaces/{id}/structure/{nodeId}	Searches for the structure node with the given node id in the workspace with the given workspace id.	

## WAGO Cloud - REST-API Documentation

If you select the operation, the parameters and the responses will be displayed. The mandatory parameters are marked with “**\*required**”. For some operations, different response codes are possible.

After the successful authentication and the click on “Try it out”, the request body is displayed and can be changed. The operation will be performed, if you click on the “Execute” button. The result will be displayed under “Responses”.

Subscriptions

GET /api/core/subscriptions

Gets all subscription information the user does have access to.

Try it out

Parameters

Name	Description
api-version <b>*required</b> string (query)	Available values : 1.0

Responses

Code	Description	Links
200	<div>Returns a list with all found subscriptions. The list is empty when no subscriptions were found.</div> <div> <div>application/json</div> <div>Controls accept header.</div> <div>Example Value   Schema</div> <pre>{   "id": "string",   "name": "string",   "description": "string" }</pre> </div>	No links

Below the operations, the schemas are displayed. The schemas describe the possible data types with their attributes, e.g. subscription or device. Enumerations can also be shown in the schema definition.

Schemas

Subscription

```
{
  id: string($guid)
  name: string
  description: string
  nullable: true
}
```



## WAGO Cloud - REST-API Documentation

### 4 REST-API Scenarios via HTTP requests

The following chapters contain step-by-step manuals of how to get started with the WAGO Cloud REST-API.

These chapters will show you:

- How to authorize via REST-API
- How to get subscription and workspace information
- How to create a device in your workspace
- How to get all commands supported by your device
- How to send a command to your device
- How to get a workspace structure
- How to get telemetry data

#### 4.1 Authorization via REST-API

To get an authorization token you can send a HTTP POST request to <https://cloud.wago.com/api/token> with the content type “application/x-www-form-urlencoded” and the following parameters in the request body:

- grant\_type=password
- username=<your-username>
- password=<your-password>

Example with curl:

```
curl --location --request POST 'https://cloud.wago.com/api/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'username=***' \
--data-urlencode 'password=***'
```

You will receive a response in the following format:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbG...",
  "name": "<your-username>",
  "token_type": "bearer",
  "expires_in": 3600
}
```

The token expires after 1 hour and you have to retrieve a new one afterwards.

**Note:** We recommend caching the token and using it for its full duration. If you request a new authorization token for each API request, throttling may occur.

Using the token in API requests:

For all requests to the WAGO Cloud REST API, you need to add an Authorization Header. The value of the header is constructed with the token\_type and access\_token values, separated by a whitespace, e.g.:

## WAGO Cloud - REST-API Documentation

```
Authorization: bearer eyJ0eXAiOiJKV1QiLCJhbG...
```

### 4.2 Get Subscription and Workspace Information

For executing all APIs, the precondition would be you have already get authorized, which mentioned in [chapter 3.3](#).

To get the information for a subscription, you need to know the subscription Id, which is created after you registered your account. To get this information use the HTTP GET /api/core/subscriptions/{id} method.

The required parameter is subscription Id, which is a GUID string:

```
[ "<mySubscriptionId>" ]
```

If the request is successful, you will get the HTTP status code 200. The response body would be like this:

```
{
  "id": "ecfb9777-d46d-4ccf-bffa-3ff5961d38bf",
  "name": "Sample subscription",
  "description": " Sample subscription"
}
```

## WAGO Cloud - REST-API Documentation

### 4.3 Create a Device

To create a device, you need to invoke the HTTP POST `/api/deviceapp/devices` method. For the parameter, you need give the workspace Id:

```
[ "<workspaceId>" ]
```

The request body should contains the folder Id and device name you defined:

```
{
  "folder": "<yourFolderId>",
  "name": "<yourDeviceName>"
}
```

If the request is successful, you will get the HTTP status code 201. The response body would be like this:

```
{
  "name": "Device_X",
  "key": "30etXnPPIkjjOlzrdoRpkqxXQew7Em6qZ+zDQBn3yL8="
}
```

### 4.4 Get all Commands Supported for a Device

To get all commands supported for a device use the HTTP Get `/api/deviceapp/devices/{id}/commands` method. For the parameter you need to provide the device id.

```
[ "<deviceId>" ]
```

If the request is successful, you will get the HTTP status code 200. The response body would be like this:

```
[
  {
    "id": 502,
    "name": "StartDataSubmission",
    "commandRequest": [],
    "commandResponse": [
      {
        "name": "Started",
        "type": "string"
      }
    ]
  },
  {
    "id": 501,
    "name": "StopDataSubmission",
    "commandRequest": [],
    "commandResponse": [
      {
        "name": "Stopped",
        "type": "string"
      }
    ]
  },
  {
    "id": 256,
    "name": "ChangeSamplingRate",
    "commandRequest": [

```

## WAGO Cloud - REST-API Documentation

```

    "name": "SampleIntervalMilliseconds",
    "type": "dint"
  },
  {
    "name": "CollectionId",
    "type": "dint"
  }
],
"commandResponse": [
  {
    "name": "Result",
    "type": "string"
  },
  {
    "name": "CollectionId",
    "type": "string"
  },
  {
    "name": "SampleIntervalMilliseconds",
    "type": "string"
  }
]
},
{
  "id": 257,
  "name": "ChangePublishingRate",
  "commandRequest": [
    {
      "name": "PublishIntervalMilliseconds",
      "type": "dint"
    },
    {
      "name": "CollectionId",
      "type": "dint"
    }
  ],
  "commandResponse": [
    {
      "name": "Result",
      "type": "string"
    },
    {
      "name": "CollectionId",
      "type": "string"
    },
    {
      "name": "PublishIntervalMilliseconds",
      "type": "string"
    }
  ]
},
{
  "id": 500,
  "name": "DeleteController",
  "commandRequest": [],
  "commandResponse": []
},
{
  "id": 504,
  "name": "GetNodeInfo",
  "commandRequest": [],
  "commandResponse": [
    {
      "name": "TransferTriggered",
      "type": "string"
    }
  ]
}

```

## WAGO Cloud - REST-API Documentation

```
}
}
},
{
  "id": 503,
  "name": "SetHeartbeatInterval",
  "commandRequest": [
    {
      "name": "IntervallInSeconds",
      "type": "string"
    }
  ],
  "commandResponse": [
    {
      "name": "IntervallInSeconds",
      "type": "string"
    }
  ]
},
{
  "id": 509,
  "name": "StartRemoteAccessMediator",
  "commandRequest": [
    {
      "name": "URL",
      "type": "string"
    }
  ],
  "commandResponse": [
    {
      "name": "Started",
      "type": "bool"
    }
  ]
},
{
  "id": 510,
  "name": "StopRemoteAccessMediator",
  "commandRequest": [],
  "commandResponse": [
    {
      "name": "Stopped",
      "type": "bool"
    }
  ]
}
}
```

### 4.5 Send a Command to the Device

To send a command to the device, you need to invoke the HTTP PUT `/api/deviceapp/devices/{id}/commands/{command}` method.

For the parameters, you need to provide the device id and the key of the command to send. The system supports 9 general default commands described in below table:

Command Name	Command ID
ChangeSamplingRate	256
ChangePublishingRate	257
DeleteController	500

## WAGO Cloud - REST-API Documentation

StopDataSubmission	501
StartDataSubmission	502
SetHeartbeatInterval	503
GetNodeInfo	504
StartRemoteAccessMediator	509
StopRemoteAccessMediator	510

There are difference between commands for the command request and command response. If you are not familiar with the command request part you can go to [4.7](#) to get the overview of all basic commands.

For instance, if you want to stop the data submission for a running device, you need to provide:

- device ID, e.g. 84f7b993-d5dc-499f-9d5d-...
- command ID, e.g. 501
- command request in the request body: "[ ]"

If the request is successful, you will get the HTTP status code 200. The response body would be like this:

```
{
  "parameters": [
    {
      "name": "Stopped",
      "value": "true"
    }
  ]
}
```

Since the WAGO Cloud release 3.4 the device app 2.0 is available. This new version includes additional return codes. If a device is offline and a command is sent to this device via the rest API, the user also receives an immediate response. The following response codes were introduced:

- "408 - Request Timeout": Indicates that the server did not receive a complete request from the client within the server's allotted timeout period.
- "409 - Conflict": The request could not be completed due to a conflict with the current state of the resource.
- "429 - Too Many Requests": The user has sent too many requests in a given amount of time.
- "500 - Internal Server Error": The server encountered an unexpected condition which prevented it from fulfilling the request.
- "504 Gateway Timeout": The server is acting as a gateway and cannot get a response in time for a request.
- The current response codes 200, 202, 400, 404 still remain.

Other return codes may also have changed in the 2.0 version. The following operations are available in the Swagger UI device app 2.0:

## WAGO Cloud - REST-API Documentation

- POST/api/deviceapp/devices/{id}/commands/{command}
- GET/api/deviceapp/devices/{deviceId}/commandResponse/{commandResponseId}
- POST/api/deviceapp/devices/{deviceId}/commandsAsync/{command}

### 4.6 Get the Workspace Structure

To get the workspace structure use the HTTP Get /api/core/workspaces/{id}/structure method. The workspace id has to be passed as a parameter:

```
[ "<workspaceId>" ]
```

You can add an optional parameter "parent Id" if you want to get the node structure where to start the search from:

```
[ "<parentId>" ]
```

If the request is successful, you will get the HTTP status code 200. The response body would be like this:

```
{
  "rootNode": {
    "id": "4c70f62d-f8aa-4da4-a589-469d8fbfed63",
    "parentId": "00000000-0000-0000-0000-000000000000",
    "nodeId": "4c70f62d-f8aa-4da4-a589-469d8fbfed63",
    "name": "API TEST",
    "type": "Project"
  },
  "nodes": [
    {
      "id": "bbd2d91d-8248-4b27-a61c-a37a72d65562",
      "parentId": "4c70f62d-f8aa-4da4-a589-469d8fbfed63",
      "nodeId": "bbd2d91d-8248-4b27-a61c-a37a72d65562",
      "name": "F1",
      "type": "Folder"
    },
    {
      "id": "73d1f164-db90-4899-b25e-7f839ed644e1",
      "parentId": "4c70f62d-f8aa-4da4-a589-469d8fbfed63",
      "nodeId": "73d1f164-db90-4899-b25e-7f839ed644e1",
      "name": "D1 with no parent id",
      "type": "Folder"
    },
    {
      "id": "4a641f6d-fa5d-4682-be20-47edd21661e8",
      "parentId": "4c70f62d-f8aa-4da4-a589-469d8fbfed63",
      "nodeId": "4a641f6d-fa5d-4682-be20-47edd21661e8",
      "name": "f4",
      "type": "Folder"
    }
  ]
}
```

### 4.7 Get Telemetry Data

To get the device raw data values for a given tag description use the HTTP POST /api/telemetry/telemetrydata/raw. For the parameter you need to provide the start time and end time.

## WAGO Cloud - REST-API Documentation

In the request body scheme is:

```
[
  {
    "deviceId": "string",
    "collectionKey": "string",
    "tagKey": "string"
  }
]
```

If the request is successful, you will get the HTTP status code 200. The response body would be like this:

```
{
  "tags": [
    {
      "deviceId": "fb309847-e6cf-431d-a174-17905ba57002",
      "collectionKey": "0",
      "tagKey": "CTDint"
    }
  ],
  "values": [
    [
      "2019-07-09T07:59:55.5867939Z",
      -59
    ],
    [
      "2019-07-09T07:59:56.1864314Z",
      -61
    ],
    [
      "2019-07-09T07:59:56.7867947Z",
      -62
    ],
    [
      "2019-07-09T07:59:57.3869341Z",
      -64
    ],
    [
      "2019-07-09T07:59:57.986446Z",
      -66
    ],
    [
      "2019-07-09T07:59:58.5865269Z",
      -68
    ],
    [
      "2019-07-09T07:59:59.186103Z",
      -69
    ],
    [
      "2019-07-09T07:59:59.7863017Z",
      -71
    ]
  ]
}
```



## WAGO Cloud - REST-API Documentation

---

### 5 Public Event Channel

WAGO Cloud allows you to create your own event subscriptions and react to different WAGO Cloud events. The word “subscription” has two different meanings in the following. In order to prevent confusion, we use “event subscription” (subscribe to an event) and “WAGO Cloud subscription” (subscription for a user in the WAGO Cloud, which contains workspaces).

The public event channel feature provides three REST-API methods:

- Subscribing to an event (POST)
- List all event subscriptions (GET)
- Unsubscribe an event subscription (DELETE)

Public events can be triggered in the WAGO Cloud by different activities and components. After the registration to a specific event type, WAGO Cloud will create a public event, if the internal event occurs. The specific content of the received event message is stored in JSON format. The event is managed in WAGO Cloud using “WebHook” as the communication method. This allows the user to be notified of events, e.g. for scheduled CSV exports, and to further use the data in the event message in a 3<sup>rd</sup> party system.

#### 5.1 Supported Public Event Types

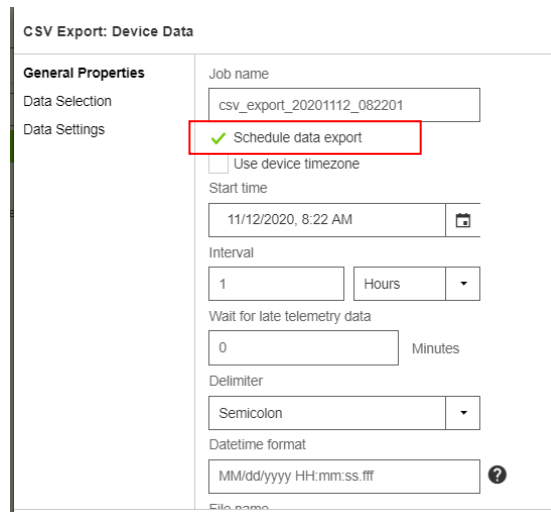
Currently, scheduled csv export, alarm, device status and telemetry events are supported are supported.

**Scheduled csv data exports** can be created via the device app or via the trend element in the visualization app. Prerequisite for that is to enable the “Scheduled data export” for the WAGO Cloud subscription. One more prerequisite for triggering a public event is that the device sends data. A scheduled CSV export can be created using the "Schedule data export" checkbox in the device configuration or in the trend element within the visualization app.

The scheduled CSV export events can be subscribed on

- workspace level

## WAGO Cloud - REST-API Documentation



CSV Export: Device Data

**General Properties**

Job name

Data Selection

csv\_export\_20201112\_082201

**Data Settings**

☒ Schedule data export

☐ Use device timezone

Start time

11/12/2020, 8:22 AM

Interval

1 Hours

Wait for late telemetry data

0 Minutes

Delimiter

Semicolon

Datetime format

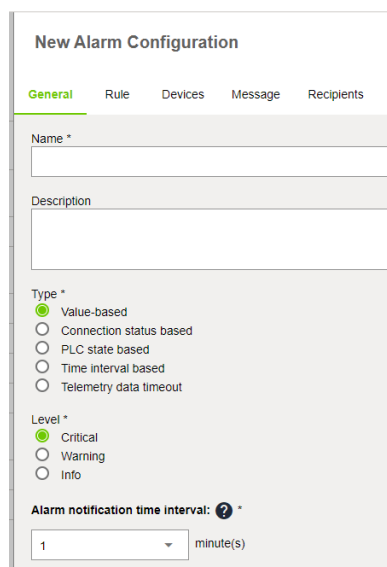
MM/dd/yyyy HH:mm:ss.fff

Figure 2: Enable cyclic data export for CSV export events

**Alarms** can be received via creating an alarm configuration in the alarm app, if the alarm rule meets the current situation. If an alarm event is registered, all alarm types can be received, e.g. Value-based, connection status based, PLC state based, time interval based and telemetry timeout based.

The alarm events can be subscribed on

- device level
- workspace level
- subscription level



New Alarm Configuration

**General** Rule Devices Message Recipients

Name \*

Description

Type \*

☒ Value-based

☐ Connection status based

☐ PLC state based

☐ Time interval based

☐ Telemetry data timeout

Level \*

☒ Critical

☐ Warning

☐ Info

Alarm notification time interval: 1 minute(s)

Figure 3: Create an alarm configuration

**Device status changes** can be triggered by the device itself or by the customer. Currently, the connection status (connected/disconnected) and the device configuration changes can

# WAGO Cloud - REST-API Documentation

be received as events. The following figure illustrates, where the device status can be displayed and device configuration can be changed.

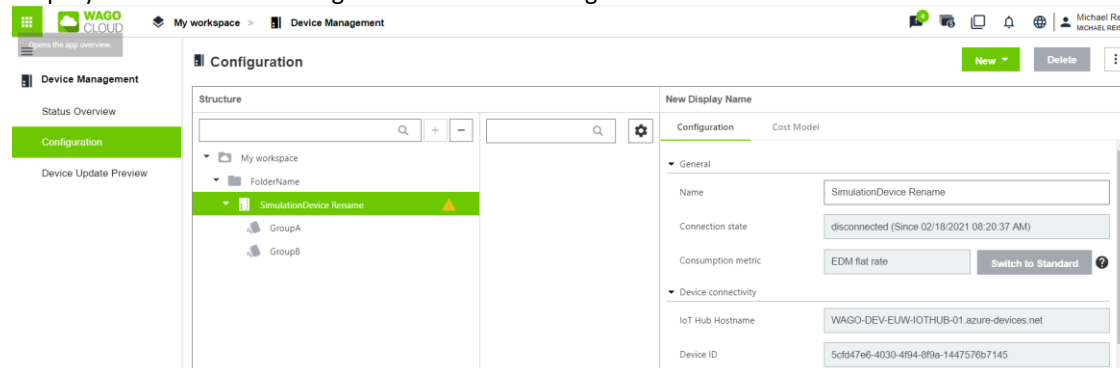


Figure 4: Device status & device configuration

**Telemetry data changes** can be triggered by the device. All tag changes can be received as an event to further work with the updated telemetry data. The following figure illustrates, where the last tag values can be displayed.

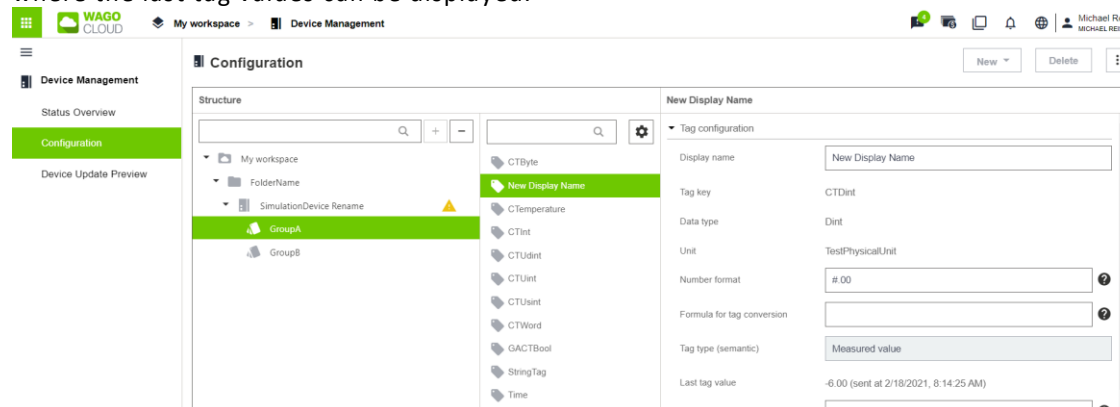


Figure 5: Telemetry data, e.g. "Last tag value"

## 5.2 Subscribe for a Public Event - POST

The required information for the authentication can be obtained via several REST-API methods. For a general process description of how to use the REST-API calls via the Swagger interface, the previous chapters can be used.

The *Subscription ID* of a WAGO Cloud Subscription can be retrieved using the GET /api/core/subscriptions method. In the following sample code, the *Subscription ID* is highlighted green.

```
{
  "id": "ebf5acb1-1639-443f-a8eb-c2eb3edbb339",
  "name": "BOC Subscription",
  "description": ""
}
```

The *Workspace ID* of a WAGO Cloud subscription can be retrieved using the GET /api/core/subscriptions/{id}/workspaces method. In the following section, the *Workspace ID* is highlighted red.

```
{
```

## WAGO Cloud - REST-API Documentation

```
"id": "26333ed1-69c4-4d6d-9369-d3372480ab88",
"name": "My workspace"
}
```

This information can be used in the request body of the method POST `/api/core/eventsubscriptions`.

```
{
  "SubscriptionId": "ebf5acb1-1639-443f-a8eb-c2eb3edbb339",
  "NodeId": "26333ed1-69c4-4d6d-9369-d3372480ab88",
  "EventType": "CsvExportEvents",
  "Configuration": {
    "endpointType": "WebHook",
    "subscriberUri": "[your callback client URI]"
  }
}
```

The *SubscriptionId* is the ID of the WAGO Cloud subscription, whereas the *Node ID* is defined via the *Workspace ID* for the WAGO Cloud Subscription. The *Node ID* represents the ID that defines the scope of the event-causing devices. Depending on the type of event, this can denote different levels, such as a workspace, a single device or an entire WAGO Cloud subscription.

The *EventType* is the type of the specific event. In this example the event type *CsvExportEvents* is specified. The following event types are supported from WAGO Cloud Release 3.0 on):

- CsvExportEvents
- AlarmEvents
- DeviceEvents
- TelemetryDataEvents

In the configuration of the event the *endpointType* can be set. At the moment only WebHook is used here, which offers the possibility to inform other applications about a certain event and to provide it promptly with information.

The *subscriberUri* defines the destination address the event is sent to. If an Azure Function has been created for events, the URI for the callback client is constructed with the following parts: `[base URL] + /api/ + [FunctionName]`

The *base URL* can be found for example in the Azure Portal in an Azure Function. The *FunctionName* can be obtained in the source code of the Azure Function in the attribute *FunctionName*. Based on the information, the *subscriberURI* should look like this:

```
{
  "SubscriptionId": "ebf5acb1-1639-443f-a8eb-c2eb3edbb339",
  "NodeId": "26333ed1-69c4-4d6d-9369-d3372480ab88",
  "EventType": "CsvExportEvents",
  "Configuration": {
    "endpointType": "WebHook",
    "subscriberUri": "https://wagocloud-eventchannelcb.azure.net/api/HttpTriggerCallback"
  }
}
```

## WAGO Cloud - REST-API Documentation

```
}

```

At the same time, the Azure Function in the portal allows you to see which calls were made via the Azure Function.

The usage of an Azure Function is only one of the many possibilities of an event receiver. Any web service that can receive an HTTP POST and respond to the validation code sent by WAGO Cloud with a suitable response can be used here.

### 5.3 Sequence of Events: Synchronous Handshake

Events are received via WebHook. The following diagram shows the handshake sequence while subscribing to events via WAGO Cloud REST-API.

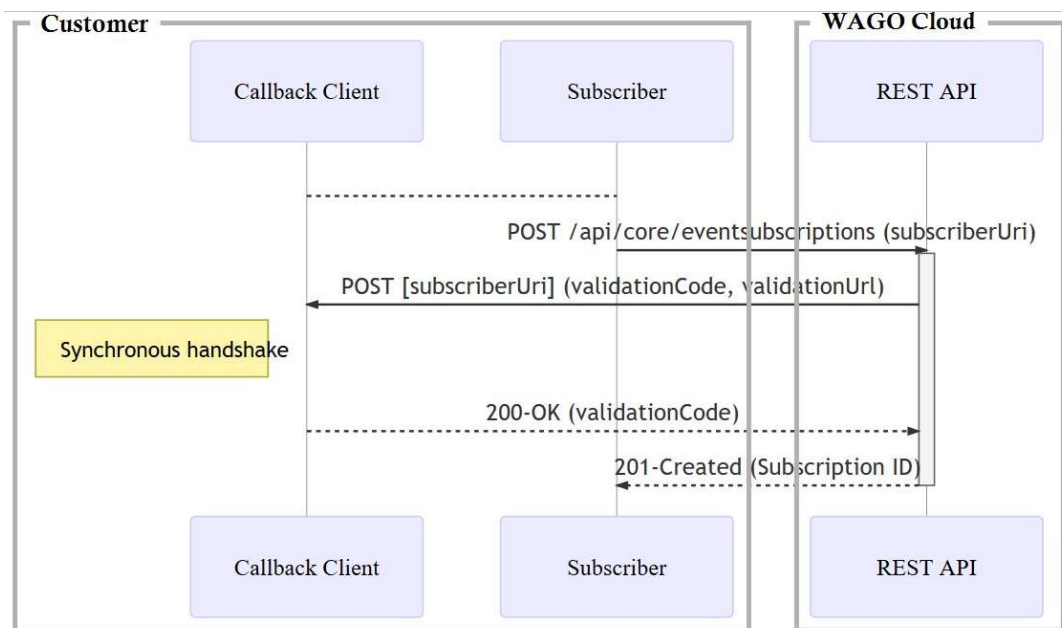


Figure 6: How the event subscription handshake works

By providing a web server, which also serves as a callback client, event subscriptions can be received. The server must be able to receive the actual events, consisting of HTTP POSTs with JSON-body. In addition, the callback client must be running at the time the WAGO Cloud REST-API is called, as WAGO Cloud is actively trying to reach it during the event subscription request.

As soon as a new event is ready, a POST request is sent to the configured endpoint including the event in the request body. The type of event is a `Microsoft.EventGrid.SubscriptionValidationEvent`:

```

{
  "id": "45c11881-5b58-4d90-8601-1460fea12e64",
  "topic": "/subscriptions/700327a3-18e1-470b-9f06-bc3bd2e61c99/resourceGroups/wagocloudtest/providers/microsoft.eventgrid/topics/wago-test-eun-egri-csvexport",
  "subject": "",
  "data": {
    "validationCode": "01C525AF-AA91-4BB6-BF63-FF64426ACA75",
  }
}

```

## WAGO Cloud - REST-API Documentation

```

    "validationUrl": "https://rp-northeurope.eventgrid.azure.net:553/events/subscriptions/csvexportfinished-12ca6a2b-50d9-42b9-9480-c23ecce2aede/validate?id=01C525AF-AA91-4BB6-BF63-FF64426ACA75&t=2020-10-12T09:34:07.3031219Z&apiVersion=2020-06-01&[Hidden Credential]",
  },
  "eventType": "Microsoft.EventGrid.SubscriptionValidationEvent",
  "eventTime": "2020-10-12T09:34:07.3031219Z",
  "metadataVersion": "1",
  "dataVersion": "1.0"
}
]

```

Using the HTTP POST mentioned above, a *validationCode* event will be sent to the callback client to identify an actual recipient behind the URL. For successful verification, the callback client must return the received *validationCode* to WAGO Cloud as part of its response. As status code the client must return HTTP RESPONSE 200 - OK and specify *validationCode* as the only JSON element *validationResponse* in the body:

```

{
  "validationResponse": "01C525AF-AA91-4BB6-BF63-FF64426ACA75"
}

```

In addition, the *validationUrl* is also sent with an URL for manual verification of the event. The event subscription has been created and receiving events is enabled.

The event message contains the *dataVersion*, which is the REST-API version used for the event subscription with the POST method. Moreover, it contains the *metadataVersion*, which is the version of the event schema in the WAGO Cloud. The content of the *data* tag is not part of the event schema.

**Hint:** If the client is unavailable, the retry strategy of the WAGO Cloud supports repeated event delivery attempts. Those attempts are made based on performance after 10 seconds up to once per hour up to 24 hours.

In the case of handshake, there is only one attempt to reach the client. If this fails, no event subscription is created.

### 5.4 List all Public Event Subscriptions - GET

Depending on the use case, several event subscriptions can be created. To see how many active events exist for a specific WAGO Cloud subscription, GET/api/core/events/subscription?subscriptionId={id} can be used. This API method lists all running events using the *Subscription ID* (WAGO Cloud subscription ID) identified above.

```

[
  {
    "id": "ebf5acb1-1639-443f-a8eb-c2eb3edbb339",
    "eventType": "CsvExportEvents",
    "filterNodeType": "Workspace",
    "filterNodeId": "0120ea86-e839-443d-a7ea-66db4lef8899",
    "version": "1.0",
    "endpointConfiguration": {

```

## WAGO Cloud - REST-API Documentation

```
[
  {
    "endpointType": "WebHook",
    "subscriberUri": "https://wagocloud-eventchannelcb.azure.net/api/HttpTriggerCallback",
    "resourceId": null
  }
]
```

Once again, you can see the details of what was entered when the event subscription was created. In the code example above, a CSV export event with corresponding Workspace ID (*filterNodeId*) is sent to the *subscriberUri* for event handling.

### 5.5 Unsubscribe from a Public Event - DELETE

For deleting existing events, the ID of the event is required. Simply call the public REST-API DELETE `/api/core/eventsubscriptions/{id}` using the event subscription ID.

If the ID is unknown, the method GET `/api/core/eventsubscriptions` can be used to identify which events are explicitly CSV export events. Then the required ID can be viewed (see code example above). Afterwards the DELETE method can be executed and the event registration is deleted.

### 5.6 Receive Scheduled CSV Export Events

Event subscriptions can be created using the code example above and the POST `/api/core/eventsubscriptions` API method with the event type "CsvExportEvents". The CSV export events can be subscribed on workspace level (*nodeID* = *workspaceID*).

A prerequisite for triggering an event is that the device sends data, otherwise no event can be triggered.

The JSON body for the event message after a triggered event looks like this:

```
[
  {
    "id": "81209303-37a0-4487-b832-98dc40f4aa26",
    "subject": "CsvDataExport",
    "data": {
      "CsvLink": "https://wctstorage.blob.core.windows.net/dataexport/job/bedb0667-630e-4911-94e2-b6877bdfed92/cbec9a3a-6303-4426-82c2-6ff24abe79b3/20201012/913efd47-995c-40cd-8566-fd8883b13dc8/f1ccf3ec-8a7a-41d9-ad0b-4d6adcf0d81/dx_My%20workspace_2020101208390000-20201012093234000.csv?sv=2018-03-28&sr=b&sig=v87X82XVAukLeInda3fdFhfuyaBFOjka%2BZjynkS3Oco%3D&se=2020-10-19T09%3A39%3A52Z&sp=rwdll",
      "DeviceId": "913efd47-995c-40cd-8566-fd8883b13dc8",
      "DeviceName": "My sample device",
      "IntervalEndTime": "2020-10-12T09:39:00Z",
      "IntervalStartTime": "2020-10-12T08:39:00Z",
      "JobId": "f1ccf3ec-8a7a-41d9-ad0b-4d6adcf0d81",
      "SubscriptionId": "bedb0667-630e-4911-94e2-b6877bdfed92",
      "WorkspaceId": "cbec9a3a-6303-4426-82c2-6ff24abe79b3"
    },
    "eventType": "CsvExportFinished",
    "dataVersion": "1.0",
    "metadataVersion": "1"
  }
]
```

## WAGO Cloud - REST-API Documentation

```
"eventTime": "2020-10-12T09:39:53.2313174Z",
"topic": "/subscriptions/700327a3-18e1-470b-9f06-bc3bd2e61c99/resourceGroups/wagocloudtest/providers/Microsoft.EventGrid/topics/WAGO-TEST-EUN-EGRI-CsvExport"
}
]
```

### 5.7 Receive Alarm Events

Event subscriptions can be created using the code example above and the POST `/api/core/eventsubscriptions` API method with the event type "AlarmEvents". The alarm events can be subscribed on device level (`nodeID = deviceID`), on workspace level (`nodeID = workspaceID`) or on subscription level (no `nodeID` in the request body of the POST method).

The JSON body for the event message, e.g. connection state based event, after a triggered event looks like this:

```
[
{
  "id": "81209303-37a0-4487-b832-98dc40f4aa26",
  "subject": "Alarms",
  "data": {
    "DeviceId": "758b1b79-97a5-4508-a76a-6d757a4f243c",
    "AlarmMessage": "<p>Connection status based - disconnected (message)</p><p>Connection state: Disconnected</p><p>Device: SimulationDevice</p>",
    "SubscriptionId": "0dc39e2a-30f1-4c71-92b5-d152aecc456c",
    "WorkspaceId": "df098db9-6c9e-4fef-80f7-5268c3fd2c16",
    "AlarmType": "ConnectionStatusBased",
    "AlarmLevel": "Critical",
    "TagValues": []
  },
  "eventType": "Alarm",
  "dataVersion": "1.0",
  "metadataVersion": "1",
  "eventTime": "2020-12-07T10:50:44.2781234Z",
  "topic": "/subscriptions/700327a3-18e1-470b-9f06-bc3bd2e61c99/resourceGroups/wagocloudtest/providers/Microsoft.EventGrid/topics/WAGO-TEST-EUN-EGRI-Alarms"
}
]
```

### 5.8 Receive Device State Events

Event subscriptions can be created using the code example above and the POST `/api/core/eventsubscriptions` API method with the event type "DeviceEvents". The device events can be subscribed on device level (`nodeID = deviceID`), on workspace level (`nodeID = workspaceID`) or on subscription level (no `nodeID` in the request body of the POST method).

The JSON body for the event message, e.g. connection state changed event, after a triggered event looks like this:

```
[
```



## WAGO Cloud - REST-API Documentation

```
{
  "id": "81209303-37a0-4487-b832-98dc40f4aa26",
  "subject": "Device",
  "data": {
    "DeviceId": "758b1b79-97a5-4508-a76a-6d757a4f243c",
    "SubscriptionId": "0dc39e2a-30f1-4c71-92b5-d152aecc456c",
    "WorkspaceId": "df098db9-6c9e-4fef-80f7-5268c3fd2c16",
    "EventType": "DeviceConnectionStateChanged",
    "EventData": {
      "ConnectionState": "Disconnected"
    },
    "Timestamp": "2021-02-07T10:50:44.2781234Z"
  },
  "eventType": " ConnectionChanged",
  "dataVersion": "1.0",
  "metadataVersion": "1",
  "eventTime": "2021-02-07T10:50:44.2781234Z",
  "topic": "/subscriptions/700327a3-18e1-470b-9f06-bc3bd2e61c99/resourceGroups/wagocloudtest/providers/Microsoft.EventGrid/topics/WAGO-TEST-EUN-EGRI-DeviceState"
}
```

The JSON body for the event message, e.g. configuration changed event, after a triggered event looks like this:

```
[
  {
    "id": "81209303-37a0-4487-b832-98dc40f4aa26",
    "subject": "Device",
    "data": {
      "DeviceId": "758b1b79-97a5-4508-a76a-6d757a4f243c",
      "SubscriptionId": "0dc39e2a-30f1-4c71-92b5-d152aecc456c",
      "WorkspaceId": "df098db9-6c9e-4fef-80f7-5268c3fd2c16",
      "EventType": "DeviceConnectionStateChanged",
      "EventData": {
        "ConnectionState": "Disconnected"
      },
      "Timestamp": "2021-02-07T10:50:44.2781234Z"
    },
    "eventType": " ConnectionChanged",
    "dataVersion": "1.0",
    "metadataVersion": "1",
    "eventTime": "2021-02-07T10:50:44.2781234Z",
    "topic": "/subscriptions/700327a3-18e1-470b-9f06-bc3bd2e61c99/resourceGroups/wagocloudtest/providers/Microsoft.EventGrid/topics/WAGO-TEST-EUN-EGRI-DeviceState"
  }
]
```

## WAGO Cloud - REST-API Documentation

```
}
]
```

### 5.9 Receive Telemetry Data Events

Event subscriptions can be created using the code example above and the POST `/api/core/eventsubscriptions` API method with the event type "TelemetryDataEvents". The telemetry data events can be subscribed on device level (`nodeID = deviceID`) only.

The JSON body for the event message after a triggered event looks like this:

```
[
{
  "id": "81209303-37a0-4487-b832-98dc40f4aa26",
  "subject": "Telemetry",
  "data": {
    "DeviceId": "758b1b79-97a5-4508-a76a-6d757a4f243c",
    "SubscriptionId": "0dc39e2a-30f1-4c71-92b5-d152aecc456c",
    "Telemetry": [
      {
        "Timestamp": "2021-02-07T10:50:44.2781234Z",
        "Key": "KeyName1",
        "Value": 1,
        "BrowsePath": "/CollectionKey/KeyName1",
        "OriginalValue": 1
      },
      {
        "Timestamp": "2021-02-07T10:50:44.2781234Z",
        "Key": "KeyName2",
        "Value": 2,
        "BrowsePath": "/CollectionKey/KeyName2",
        "OriginalValue": 2
      }
    ],
    "eventType": "Telemetry",
    "dataVersion": "1.0",
    "metadataVersion": "1",
    "eventTime": "2021-02-07T10:50:44.2781234Z",
    "topic": "/subscriptions/700327a3-18e1-470b-9f06-bc3bd2e61c99/resourceGroups/wagocloudtest/providers/Microsoft.EventGrid/topics/WAGO-TEST-EUN-EGRI-DeviceState"
  }
]
```

The collections are contained in the "BrowsePath". Because it is possible to define a formula for tag conversion in the device app, the "original value" is introduced here, to get the original tag value before the tag conversion via the formula.

# WAGO Cloud - REST-API Documentation

## 6 Asynchronous Commands

In the past, command timeout problems occurred frequently within the WAGO Cloud. This command timeouts can have various reasons such as connection problems, problems with the data agent, cloud service or IoT hub. With the WAGO Cloud Release 3.3, we have added the possibility to send asynchronous commands to the device without waiting for a response or until the execution of the device command is completed. Therefore, the following new APIs have been added:

- *GET/api/deviceapp/devices/{deviceId}/commandResponse/{commandResponseId}*
- *POST/api/deviceapp/devices/{deviceId}/commandsAsync/{command}*

The API `GET/api/deviceapp/devices/{deviceId}/commandResponse/{commandResponseId}` will retrieve the specified command response with the specified command response ID. With the second API `POST/api/deviceapp/devices/{deviceId}/commandsAsync/{command}`, the client can check the command response with device ID and command response ID. If the specified command response has already been reached in the cloud, it will be returned. If it was not found, it responds with "not found", if it timed out, it responds with "command time out".

## 6.1 How to use it

1. Execute the REST API `/api/deviceapp/devices/{id}/commands` to query device commands.

[illegible]

## WAGO Cloud - REST-API Documentation

- If successful, the device commands are displayed (including command ID and request/response format).

200

Response body

```

{
  "id": 257,
  "name": "ChangePublishingRate",
  "commandRequest": [
    {
      "name": "PublishIntervalMilliseconds",
      "type": "dint"
    }
  ],
  "name": "CollectionId",
  "type": "dint"
},
{
  "name": "Result",
  "type": "string"
},
{
  "name": "CollectionId",
  "type": "string"
},
{
  "name": "PublishIntervalMilliseconds",
  "type": "string"
}

```

Response headers

```

cache-control: private
content-encoding: gzip
content-type: application/json; charset=utf-8
date: Fri, 24 Sep 2021 08:27:59 GMT
request-context: appId=cid-v1:4e7b83ec-1e85-46da-b3a7-61325702a994
transfer-encoding: chunked
vary: Accept-Encoding
x-servicefabricrequestid: 1384073d-daad-4927-9579-173c9c9b1788

```

Download

- Execute the REST API `/api/deviceapp/devices/{deviceId}/commandsAsync/{command}` by entering the device and command ID.

**POST** `/api/deviceapp/devices/{deviceId}/commandsAsync/{command}` Sends the specified command request asynchronously to the device with the given parameters.

Parameters Cancel

Name	Description
<b>deviceId</b> * required string (path)	The id of the device. <input type="text" value="ad7d040e-839e-4300-9359-860c8fedca8c"/>
<b>command</b> * required integer (path)	The key of the command to send. <input type="text" value="257"/>
<b>api-version</b> * required string (query)	<input type="text" value="1.0"/>

**Request body** required application/json

The parameters to send with the command.

Edit Value | Schema

```

[
  {
    "name": "PublishIntervalMilliseconds",
    "value": "16000"
  },
  {
    "name": "CollectionId",
    "type": "0"
  }
]

```

# WAGO Cloud - REST-API Documentation

## 4. You will receive the command response ID

Server response

Code	Details
202	<p>Response body</p> <pre>{   "deviceId": "ad7d040e-839e-4300-9359-860c8fedca8c",   "commandResponseId": "Cmdad7d040e839e43009359860c8fedca8c1357304084",   "commandId": 257,   "deviceName": "701",   "state": "Accepted",   "expiration": "2021-09-24T07:42:10.0833639Z",   "acceptTime": "2021-09-24T07:41:40.0989544Z",   "submitTime": "2001-01-01T00:00:00",   "responseEnqueueTime": "2001-01-01T00:00:00",   "responseReceiveTime": "2001-01-01T00:00:00" }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * access-control-expose-headers: Request-Context,X-ServiceFabricRequestId,Content-Length,Date,Server cache-control: private content-length: 383 content-type: application/json; charset=utf-8 date: Fri, 24 Sep 2021 07:41:39 GMT request-context: appId=cid-v1:4c7083ec-1e95-46da-b3a7-61325702a994 x-servicefabricrequestid: 9d0c23f7-0490-4f66-b879-ab8690a7775a</pre>

Responses

Code	Description	Links
200	Is returned if the command was send successful and the asynchronous operation is completed	No links
202	Is returned if the request by the client was accepted or submitted	No links
400	Is returned if the parameter or the command validation caused an error	No links
404	Is returned if the command for the device wasn't found or you are not authorized to access it.	No links

## 5. Execute the REST API

/api/deviceapp/devices/{deviceId}/commandResponse/{commandResponseId} by using the command response ID

**GET** /api/deviceapp/devices/{deviceId}/commandResponse/{commandResponseId} Gets the specified command response asynchronously with the given command response id

Parameters

Name	Description
<b>deviceId</b> * required string (path)	The id of the device.
<b>commandResponseId</b> * required string (path)	The response id of the pending command, it is the key for checking the state of the current command
<b>api-version</b> * required string (query)	1.0

Execute Clear

Responses

## WAGO Cloud - REST-API Documentation

6. You will receive the status of the execution of the device command

server response

Code	Details
200	<p>Response body</p> <pre>{   "deviceId": "ad7d040e-839e-4300-9359-860c8fedca8c",   "commandResponseId": "Cmdad7d040e839e43009359860c8fedca8c1546530743",   "commandId": 252,   "deviceName": "D1",   "state": "Completed",   "response": {     "parameters": [       {         "name": "Result",         "value": "TRK"       },       {         "name": "CollectionId",         "value": "1"       },       {         "name": "PublishIntervalMilliseconds",         "value": "16000"       }     ]   },   "expiration": "2021-09-24T08:46:11.851513Z",   "acceptTime": "2021-09-24T08:46:01.8574817Z",   "submitTime": "2021-09-24T08:46:01.8831493Z",   "responseEnqueueTime": "2021-09-24T08:46:04.378Z",   "responseEffectiveTime": "2021-09-24T08:46:04.4026225Z" }</pre> <p>Response headers</p> <pre>cache-control: private content-encoding: gzip content-type: application/json; charset=utf-8 date: Fri, 24 Sep 2021 08:46:22 GMT request-context: appId=cid-v1:4c7b83ec-1e85-46da-b3a7-61325702a994 transfer-encoding: chunked vary: Accept-Encoding x-servicefabricrequestid: f1fd15bd-070b-4a9d-827b-991037e4b61f</pre>

### 6.2 Device Command Response States

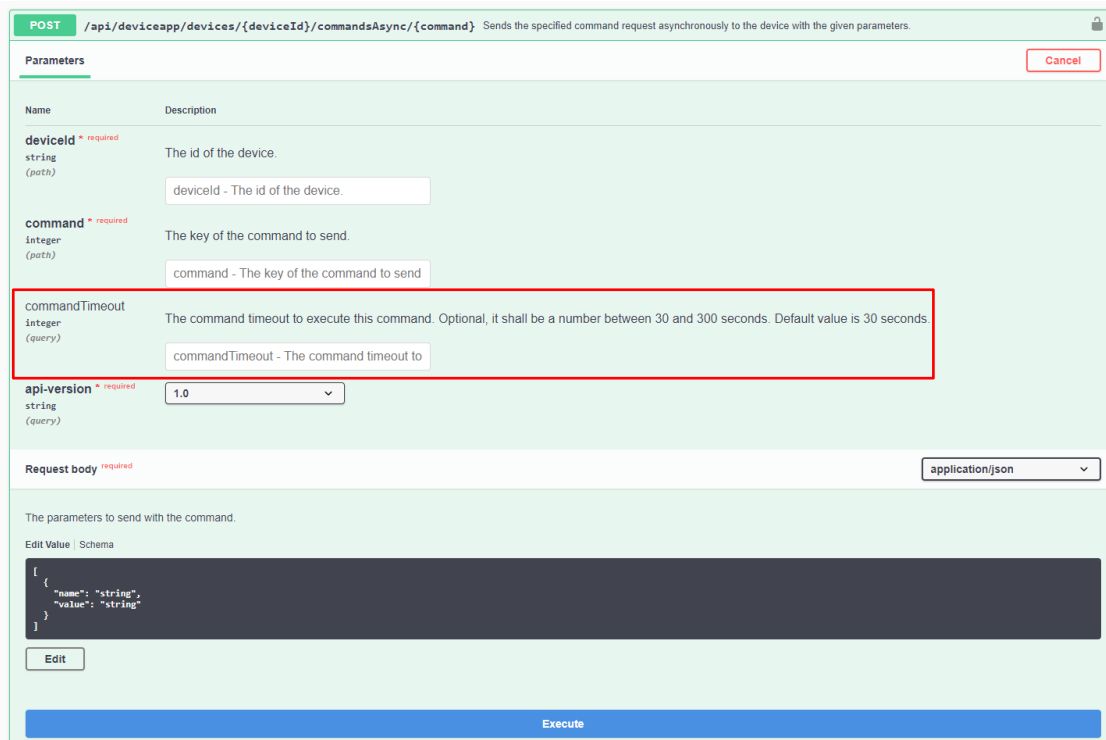
The following states can be returned:

State	Description
Accepted	WAGO Cloud receives valid device command requests
Throttled	The time between two device command requests exceeds 15s
Reject	Device rejects command or device is offline, the reason is displayed in the response
NotFound	Device or device command response ID does not exist
Submitted	Send device command to device
Failed	The execution of the device command has failed, the reason is displayed in the response
Timeout	Device command execution timeout
Completed	Execution of the device command was successful

# WAGO Cloud - REST-API Documentation

## 6.3 Change Timeout Time

The REST-API method `POST/api/deviceapp/devices/{deviceId}/commandsAsync/{command}` has an optional parameter to set the timeout time of a command. The default value is 30 seconds and can be increased up to 300 seconds.



**POST** `/api/deviceapp/devices/{deviceId}/commandsAsync/{command}` Sends the specified command request asynchronously to the device with the given parameters.

**Parameters**

Name	Description
<b>deviceId</b> * required string (path)	The id of the device. deviceId - The id of the device.
<b>command</b> * required integer (path)	The key of the command to send. command - The key of the command to send
<b>commandTimeout</b> integer (query)	The command timeout to execute this command. Optional, it shall be a number between 30 and 300 seconds. Default value is 30 seconds. commandTimeout - The command timeout to
<b>api-version</b> * required string (query)	1.0

**Request body** \* required  
application/json

The parameters to send with the command.

Edit Value | Schema

```
{
  "name": "string",
  "value": "string"
}
```

Edit

Execute

## 6.4 Retry Pattern for WAGO Cloud Device Command Service

In the cloud, transient errors are not uncommon, and an application should be designed to handle them elegantly and transparently. This reduces the impact that errors can have on the business tasks that the application performs.

Following Microsoft's remarks <https://docs.microsoft.com/en-us/azure/architecture/patterns/retry#solution> it is recommended when using the WAGO Cloud Public REST API, to implement the pattern "Retry after delay" with a delay of 2 - 5 seconds and a maximum of 5 retries.

# WAGO Cloud - REST-API Documentation

## 7 Query the tag information for multiple devices via a batch api

### 7.1 General:

As a user of the WAGO Cloud Rest-api, I would like to query collections and tags for several devices at once via the Rest-api, so that I can process the query and the result very efficiently in my own application without much network traffic. Previously, users had to use the following two Rest-api methods to get current values for multiple devices:

- GET /api/deviceapp/devices/{id}/collections
- GET /api/telemetry/telemetrydata/latest

The api method must be called very often, because the structure of the device sometimes changes. Therefore, for each request of the latest values, the GET/api/deviceapp/devices/{id}/collections method must also be called. The above scenario is used for many devices which can lead to performance problems on the user side.

Since the WAGO Cloud release 3.7, there is now a new REST API method POST/api/deviceapp/batch to request the collections for a list of devices, including tag information. This has the advantage of significantly reducing the number of Rest-api calls.

### 7.2 Request body:

Example Value | Schema

```
[
  {
    "id": "string",
    "url": "string",
    "method": "string",
    "headers": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "dependsOn": [
      "string"
    ]
  }
]
```

### 7.3 Responses:

Code	Description
200	<p>Is returned when execution succeeds.</p> <p>application/json</p> <p>Controls Accept header.</p>
400	<p>Is returned when the parameter validation caused an error.</p>
404	<p>Is returned when you are not authorized to access it.</p>



## WAGO Cloud - REST-API Documentation

---

### 7.4 Additional Notes:

- The number of possible queries is limited to 50
- The batch api cannot call batch api
- Api management is configured with a low limit of batch api calls
- The batch API is only available for paid subscriptions