

Table of Contents

1	Introduction	2
2	API Overview	3
2.1	Permissions in AP Connect for certain API endpoints	3
2.2	Introduction	4
2.3	Preparations in AP Connect	4
2.4	Authentication via REST	4
2.4.1	Basic authentication	5
2.4.2	OAuth 2.0	5
2.5	Using Swagger	7
2.5.1	Overview	7
2.5.2	Authentication in the Swagger interface	8
2.5.3	Details of an endpoint	9
2.5.4	How to query API endpoints in Swagger	10
2.6	Subscription	12
2.6.1	Server requirements & setup	12
2.6.2	Creation and deletion of subscriptions	12
2.7	Behaviour using AP Connect Pharma Edition	13
2.8	Typical integration use cases	14
2.8.1	Create and execute a tasklist	14
2.8.2	Search for several measurements	15
2.8.3	Retrieve the results of one measurement	16
2.8.4	Subscribe to completed measurements, checks, or adjustments	16
2.8.5	Mark measurements as exported	17
2.9	Filtering, sorting, and paging	17
3	Appendix	19
3.1.1	API Versioning	19
3.2	Well-known tables	19
3.2.1	WellKnownSystemState	19
3.2.2	WellKnownAdjustmentStatus	19
3.2.3	WellKnownAdjustmentRecommendation	19
3.2.4	WellKnownAdjustmentDecision	19
3.2.5	WellKnownCheckStatus (Checkmetadata/status)	19
3.2.6	WellKnownCheckAssessment	20
3.2.7	WellKnownElectronicSignatureSigningLevel	20
3.2.8	WellKnownMeasurementStatus	20
3.2.9	WellKnownMeasurementAssessment	20
3.3	Reference Tables	21
3.3.1	Quantities and Units	21
3.3.2	WellKnownMeasuredValues	22
3.3.3	WellKnown Metadata	22
3.3.4	Variant	23

1 Introduction

The AP Connect REST Interface is a network-based communication interface that allows to query data like measurements, checks, adjustments etc.

The communication interface provides the solution for: *System Integrators* and *Customers* willing to query AP Connect. The current API version is **141** and **APC revision 120**.

This document is extended with

- Guidelines & best practices for common patterns to achieve consistency
 - Status codes and error representation
 - Sorting, filtering, pagination (filtering, sorting, and paging)
 - Expandable content
 - Long-running operations
 - Versioning (versioning strategy)
 - Caching
 - etc.
- A data model for representing common data types
 - Quantities and units (variant)
 - Precision, digits (variant)
 - Images (variant)
 - Translations (accept-language request header, include-languages request header)
 - etc.

2 API Overview

2.1 Permissions in AP Connect for certain API endpoints

The following permissions of the authorized user are required for the listed subsections and individual API endpoints.

Additionally to the individual permissions *Use remote access* is required for all restricted endpoints.

- Measurements/Checks/Adjustments
 - DELETE: /api/v1/checks/completed/{id}
 - Delete measurement data
 - GET: /api/v1/checks/completed/{id}
 - View measurement data
 - PATCH: /api/v1/checks/completed/{id}:
 - Action “Export”: Manual export permission
 - Action “Archive”: Manual export permission & Delete measurement data
 - GET: /api/v1/checks/completed
 - View measurement data
 - GET: /api/v1/checks/completed/attachment/{id}
 - View measurement data
 - GET: /api/v1/checks/completed/image/{id}
 - View measurement data
 - GET: /api/v1/checks/completed/latest
 - View measurement data
- Audit trail: (only available for AP Connect *Pharma Edition*)
 - GET: /api/v1/audittrail/entries
 - Returns a sequence of audit trail entries
 - GET: /api/v1/audittrail/entries/latest
 - Returns information about the most recent audit trail
- Tasklists
 - GET /api/v1/tasklists
 - Tasklists: View & Manage
 - POST/api/v1/tasklists
 - Tasklists: View & Manage
 - GET /api/v1/tasklists/{id}
 - Tasklists: View
 - POST /api/v1/tasklists/{id}
 - Tasklists: View & Manage
- Instruments
 - GET /api/v1/instruments
 - Instruments: View

AP Connect REST Interface Documentation

for AP Connect 4.0



2.2 Introduction

A complete endpoint URL looks like the following:

```
http(s)://<host>:<port>/api/v1
```

Examples:

```
http://10.111.111.12:8393/api/v1
```

```
http://PC_12345678:8393/api/v1
```

```
https://PC_12345678.anton-paar.com:8393/api/v1
```

The URL consists of:

Scheme	Either "http" or "https". Whether communication runs over HTTP or secured over HTTPS depends on the system configuration.
Host	The IPv4 address or the DNS lookup name.
Port	The TCP port 8393 is used.
api/v1	The root path for the first major API version.

Subsets

The functionality provided by the API is separated into different subsets.

api/v1	Root
api/v1/adjustments	All functionality related to adjustments.
api/v1/audittrail	All functionality related to the audit trail.
api/v1/checks	All functionality related to checks.
api/v1/measurements	All functionality related to measurements.
api/v1/subscriptions/measurements	Configuration of proactive notifications of new measurements.
api/v1/instruments	All functionality related to instruments
api/v1/tasklists	All functionality related to tasklists
api/v1/system	All functionality affecting the entire system.

2.3 Preparations in AP Connect

Not all endpoints need an authorized user, but especially if restricted endpoints should be queried the following steps in AP Connect have to be executed before.

1. Activate a REST license in AP Connect > ② > About AP Connect > Manage License.
(In the license configuration **REST API** needs to be set **active**.) (if not yet consult the software manual about instructions how to do that)
2. Check if a user with the activated permission of "Use remote access" is available
Use the default role "Remote Access" or create a new one.
3. Check or edit the permissions of the authorized user.
Mandatory: "Use remote access". Depending on the request find the required permissions here: 2.1
4. If Labs is activated: the authorized user's role needs to be assigned to the labs where data should be collected.

2.4 Authentication via REST

At the moment, *Basic (http)-authentication* and *OAuth 2.0* are supported. Authentication using OAuth 2.0 is recommended for security reasons. Details about the required user permissions can be found in chapter 2.1.

Note:

- If no credentials (username and password) are supplied, or if the provided credentials are wrong, then a *401 Unauthorized* status code is returned.
- If the credentials are correct, but the authenticated user does not have the necessary permission to perform the requested operation, then a *403 Forbidden* status code is returned.
- The username is not allowed to contain a colon ':' character.

2.4.1 Basic authentication

The credentials (username and password) supplied with a request must match with an active user of AP Connect who has sufficient permissions (refer to 2.1 Permissions in AP Connect for certain API endpoints).

Add an Authentication header with Basic to the requests:

- Credentials can be stored by the other software
 - E.g.: Swagger.ioCredentials for Basic authentication can be entered in a separate window of the user interface and are stored during the whole session.
- Credentials can be part of the URL of every request
[http://\[username\]:\[password\]@\[hostname\]:\[port\]/endpoint](http://[username]:[password]@[hostname]:[port]/endpoint)
default port: 8393
Not recommended due to visibility of sensible information. Authorization should be contained in the header and not in URLs.

2.4.2 OAuth 2.0

Authentication to access AP Connect's REST interface using OAuth 2.0 is available if central user management is active. Central user management technically relies on an Identity provider service and runs by default for AP Connect 4.0 installations and higher. If AP Connect was initially installed with a version earlier than 4.0 and updated afterwards, central user management must be activated manually. Follow the instructions in the AP Connect software manual to do this. To establish OAuth 2.0 connection is encrypted relying on enabled TLS communication of AP Connect, which is the case by default.

OAuth 2.0 authentication requires a registered client (client ID and client secret) on the Identity provider and a valid AP Connect user with sufficient permissions (refer to 2.1 Permissions in AP Connect for certain API endpoints). The client requests an access token from the Identity provider, which must be included in the Authorization: Bearer header for all API calls. The following steps describe how to create a client and authenticate against the AP Connect Server API, and manually requesting a token.

1. Create a client

To create a client for the Identity provider, make a *POST-request* to the Identity provider endpoint (default is [https://\[identity-provider-uri\]:5278/api/v1/clients](https://[identity-provider-uri]:5278/api/v1/clients)). When successful, the response return a *201 Created* and contain both the **id** and the **secret of the client**. These credentials should be stored by the application.

The **POST** request for creating a client should be formatted as shown in the example below.

POST to [https://\[identity-provider-uri\]:5278/api/v1/clients](https://[identity-provider-uri]:5278/api/v1/clients) (e.g. <https://localhost:5278/api/v1/clients>)

Request body

```
{
  "client_name": "[Remote access for LIMS]",
  "software_id": "urn:ap:auth:/software-type/my-software",
  "installation_id": "00000000-0000-0000-0000-000000000000",
  "software_version": null,
  "redirect_uris": [],
  "token_endpoint_auth_method": "client_secret_basic",
  "grant_types": [
    "authorization_code",
    "password"
  ]
}
```

```
],  
"response_types": [  
  "code"  
],  
"standard_token_exchange": true  
}
```

Response:

```
{  
  "id": "[id]",  
  "secret": "[secret]"  
}
```

2. Find token-endpoint

AP Connect's identity provider follows the *OpenIdConnect (OIDC)* standard, so the user can get the authority url at the info endpoint (<https://<identity-provider-uri>:5278/info>) and access the *oidcDiscoveryUri*, and pulling the token endpoint from the response. With that endpoint, create a token to be passed to the server on the bearer authorization header. Below is an example of the token endpoint in the *oidcDiscoveryUri* response.

<https://<identity-provider-uri>:5278/info> (e.g. <https://localhost:5278/info>)

Response:

```
{  
  "issuer": "http://localhost:5278/oidc/realms/leo",  
  "authorization_endpoint": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/auth",  
  "token_endpoint": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/token",  
  "introspection_endpoint": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/token/introspect",  
  "userinfo_endpoint": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/userinfo",  
  "end_session_endpoint": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/logout",  
  "frontchannel_logout_session_supported": true,  
  "frontchannel_logout_supported": true,  
  "jwks_uri": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/certs",  
  "check_session_iframe": "http://localhost:5278/oidc/realms/leo/protocol/openid-connect/login-status-iframe.html",  
  "grant_types_supported": [  
    "authorization_code",  
    "client_credentials",  
  ]  
}
```

3. Request token

With that token-endpoint, create a token to be passed to the server on the bearer authorization header. Below is an example of the token endpoint in the *oidcDiscoveryUri* response.

POST to <https://<identity-provider-uri>:5278/oidc/realms/leo/protocol/openid-connect/token>
(e.g. <https://localhost:5278/oidc/realms/leo/protocol/openid-connect/token>)

Request body:

```
{  
  "grant_type": "client_credentials",  
  "client_id": "[client id]",  
  "client_secret": "[client secret]"  
}
```

Response:

```
{  
  "access_token": "[access_token]",  
  "expires_in": 300,  
  "token_type": "Bearer",  
  "not-before-policy": 0,  
  "scope": "profile email"
```

} }

This token represents the authenticated user and client and must be included in all subsequent requests to the AP Connect REST API.

To use the token, it must be added to the HTTP request header using the **Bearer authentication scheme**. This is done by setting the Authorization header.

When calling an endpoint (e.g. retrieving a completed measurement), the request therefore consists of:

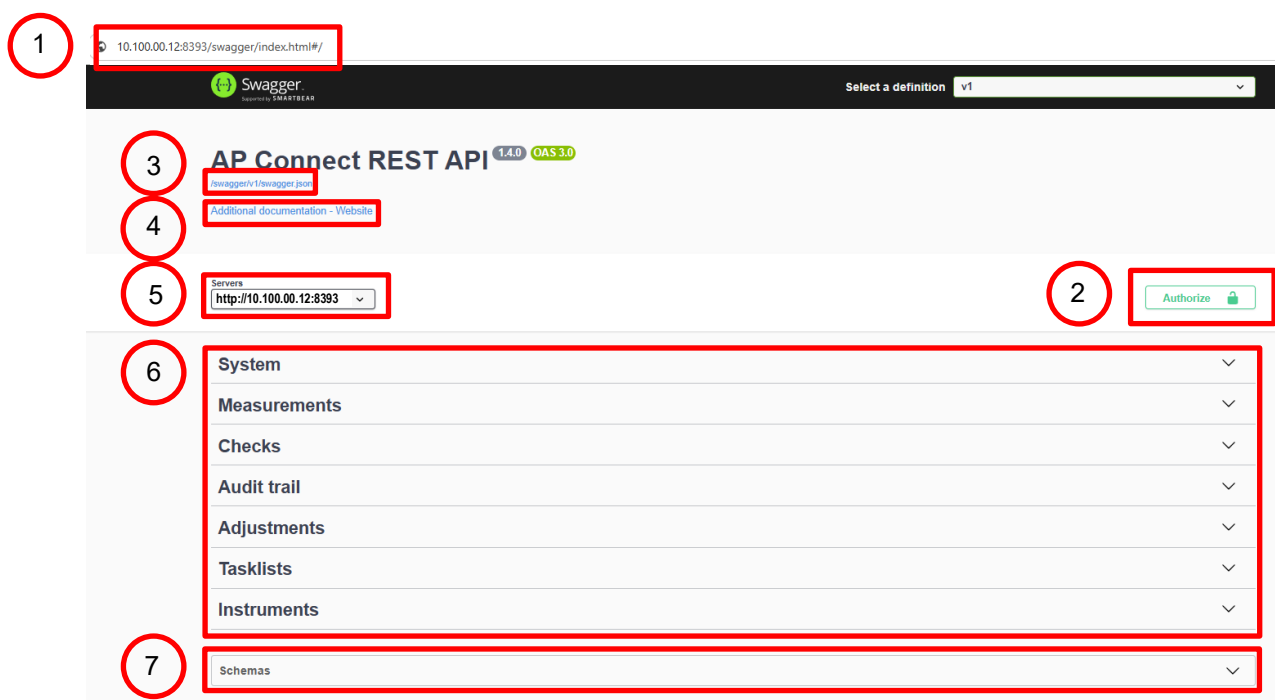
- the API URL (e.g. /api/v1/measurements/completed/{id})
- and the Authorization header containing the access token

AP Connect validates this token on each request. If the token is valid and contains sufficient permissions, the request is processed; otherwise, it is rejected (e.g. with HTTP 401 Unauthorized).

2.5 Using Swagger

An Open API specification version 3 is available where all API endpoints and used objects are shown, independent of the used version (Standard or Pharma Edition). A description about the interface follows.

2.5.1 Overview



1. URL to access the Swagger interface contains: [http://\[IP address/hostname\]:\[port\]/swagger/index.html#/](http://[IP address/hostname]:[port]/swagger/index.html#/) (https in case of OAuth 2.0)
The IP address or hostname refers to the location of the AP Connect server installation. In case AP Connect is used locally, as IP address hostname can be set: <http://localhost:8393/swagger/index.html#/>

Authorization of an AP Connect user
Refer to below to 0


2. Authentication in the Swagger interface to find more out about Basic-authentication and OAuth 2.0.
3. To open the Swagger documentation in form of a .json-file
4. Additional documentation that contain files
5. Currently used server, where AP Connect is installed

6. Overview of API endpoints
Can be expanded section wise.
7. Collection of objects
By expanding further their properties are shown and described. Properties marked with an red Asterisk (*) are mandatory fields.

2.5.2 Authentication in the Swagger interface

For demonstration and testing purposes, the same authentication flows can be executed directly in the Swagger.io interface without implementing a custom client or script. This allows users to validate configuration, credentials, and connectivity to the AP Connect API in a controlled and transparent way. Using Swagger.io is particularly useful during setup and troubleshooting, as it provides immediate feedback on authentication and request behavior.

2.5.2.1 Basic authentication in Swagger

In the Swagger interface (<http://localhost:8393/swagger/index.html#/>), click the  button in the upper right corner. Enter the AP Connect username and password for Basic Authentication and confirm to apply the credentials. The authentication is performed securely over HTTP, and the credentials are automatically included in subsequent API requests.

Basic (http, Basic)

Provide Basic Authentication

Username:

Password:

Authorize

Close

2.5.2.2 OAuth 2.0 in Swagger interface

To authenticate against AP Connect using OAuth 2.0, a client must first be created in the Identity Provider.

1. Open the Swagger interface of the Identity provider
[https://\[identity-provider-uri\]:5278/swagger/index.html#/2.%20Clients/Clients_RegisterClient](https://[identity-provider-uri]:5278/swagger/index.html#/2.%20Clients/Clients_RegisterClient)
(e.g. https://localhost:5278/swagger/index.html#/2.%20Clients/Clients_RegisterClient) and navigate to
2. *Clients* and choose the endpoint:

POST /api/v1/clients

Use the *Try it out* function with the provided example request to create a new client. The response will return a **client ID** and **client secret**, which must be stored.

```
{
  "client_name": "My Client Application",
  "software_id": "urn:ap:auth:/software-type/my-software",
  "installation_id": "00000000-0000-0000-0000-000000000000",
  "software_version": null,
  "redirect_uris": [],
  "token_endpoint_auth_method": "client_secret_basic",
  "grant_types": [
    "authorization_code",
    "password"
  ]
}
```

2. Open the Swagger interface of AP Connect and click *Authorize*.
In the OAuth2 section, enter:
 - the **username** and **password** of the user
 - the **client ID** and **client secret** obtained from the Identity Provider

AP Connect REST Interface Documentation

for AP Connect 4.0

- keep the client credentials location as *Request body*

Available authorizations x

OAuth2 (OAuth2, password)

Application: app_name

OAuth
Token URL: http://localhost:5278/oidc/realms/leo/protocol/openid-connect/token
Flow: password

username:

password:

Client credentials location:

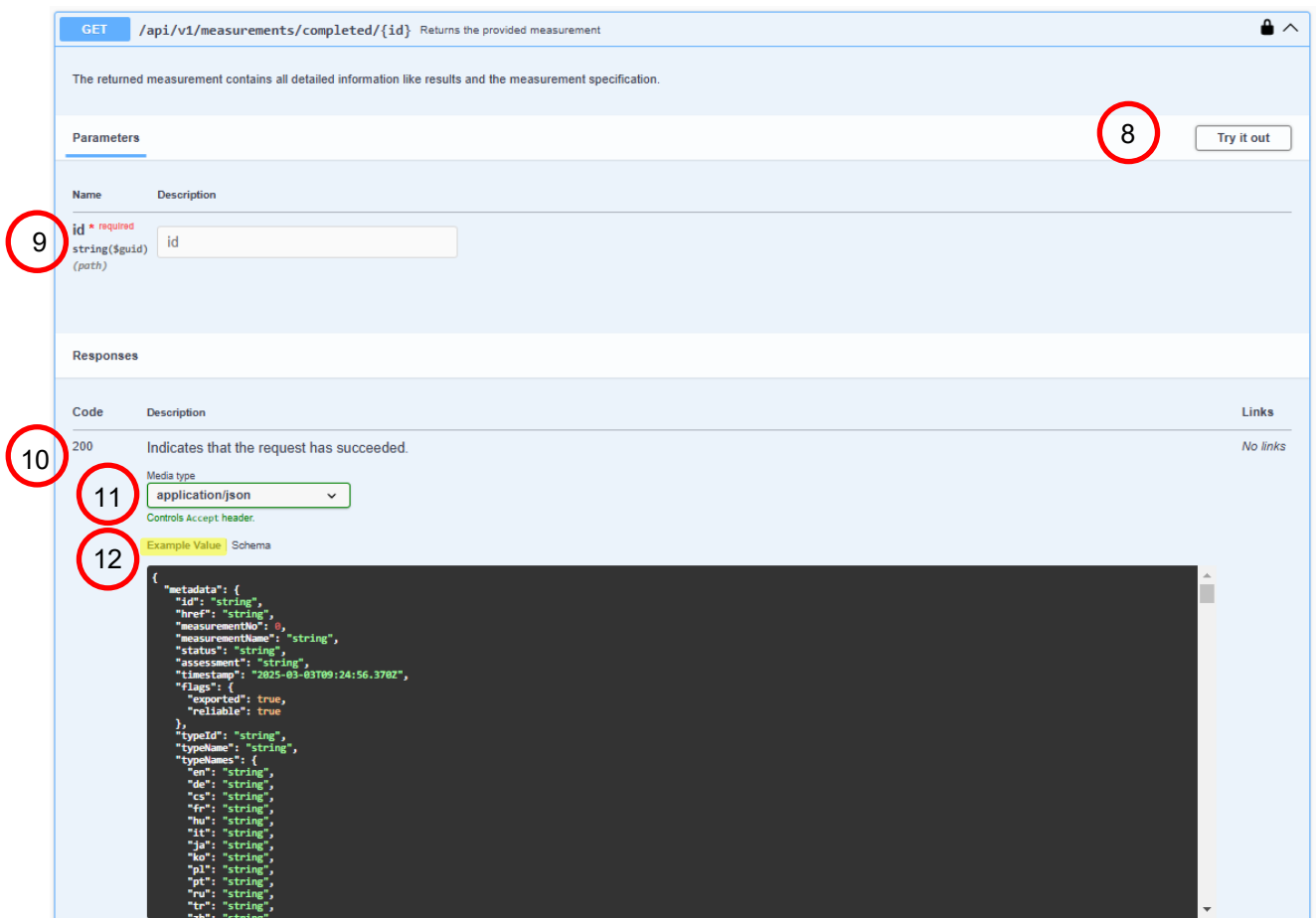
client_id:

client_secret:

3. Confirm with . In the background Swagger will request an access token from the Identity Provider and automatically include it in subsequent API requests.

For productive environments, credentials are not exchanged directly in this manner. Instead, a token-based approach is used where the client securely obtains and manages access tokens without repeatedly transmitting sensitive user and client credentials.

2.5.3 Details of an endpoint



GET /api/v1/measurements/completed/{id} Returns the provided measurement

The returned measurement contains all detailed information like results and the measurement specification.

Parameters

8

Name	Description
9 id * required string(\$guid) (path)	id

Responses

Code	Description	Links
10 200	Indicates that the request has succeeded.	No links

11

12

```
{
  "metadata": {
    "id": "string",
    "href": "string",
    "measurementNo": 0,
    "measurementName": "string",
    "status": "string",
    "assessment": "string",
    "timestamp": "2025-03-03T09:24:56.370Z",
    "flags": {
      "exported": true,
      "reliable": true
    }
  },
  "typeId": "string",
  "typeName": "string",
  "typeNames": {
    "en": "string",
    "de": "string",
    "cs": "string",
    "fr": "string",
    "hu": "string",
    "it": "string",
    "ja": "string",
    "ko": "string",
    "pl": "string",
    "pt": "string",
    "ru": "string",
    "tr": "string",
    "zh": "string"
  }
}
```

Code	Description	Links
200	Indicates that the request has succeeded.	No links

Media type

Controls Accept header.

Example Value **Schema**

Measurement {

- description: [API 100] Defines how a measurement is represented.
- metadata* {
 - description: [API 100] The metadata of the measurement like a timestamp, the completion status. Must not be null.
 - oneOf -> **MeasurementMetadata** { ... }
- specification {
 - > { ... }
 - nullable: true
- environment {
 - description: [API 100] The environment of the measurement describing the state of the instrument at the time of measurement. Can be null.
 - oneOf -> **MeasurementEnvironment** {
 - description: [API 100] Defines how a measurement environment is represented. A measurement environment contains information about the state of the instrument at the time when the measurement was performed.
 - systemInfo {
 - > { ... }
 - nullable: true
 - additional {
 - > [...]
- results {
 - > [...]

8. Button (*Try it out*) to start configuring the query of the endpoint
9. Parameters that can be set optional or mandatory (marked with * **required**), their data type and a description of them
10. List of possible response-codes with a description what they are indicating
11. Media/file type in which the response is provided
12. Example of a response body that is returned in case of a specific code
13. Schema of the objects that a response can contain. Mandatory objects are marked with *. Sections can be expanded and include descriptions of the contained data, data type and since which API version the object is available.

2.5.4 How to query API endpoints in Swagger

Find a general step-by-step instruction how the Swagger.io of AP Connect can be used to request data.

1. Open the URL to access the Swagger interface
 contains: [http://\[IP address/hostname\]:\[port\]/swagger/index.html#/](http://[IP address/hostname]:[port]/swagger/index.html#/)
 The IP address or hostname refers to the location of the AP Connect server installation. In case AP Connect is used locally, as IP address hostname can be set: <http://localhost:8393/swagger/index.html#/>
2. Expand the required subsection of API endpoints
3. Choose a specific endpoint by expanding
 (Endpoints with a lock need an authorized user to be logged in)

AP Connect REST Interface Documentation

for AP Connect 4.0

System			^
GET	/api/v1/system/apiVersion		▼
GET	/api/v1/system/capabilities		▼
GET	/api/v1/system/info		▼
GET	/api/v1/system/state		▼
GET	/api/v1/system/ping		▼
GET	/api/v1/system/ping/authenticated		▼
Measurements subscriptions			^
DELETE	/api/v1/subscriptions/measurements		▼
POST	/api/v1/subscriptions/measurements		▼
Measurements			^
DELETE	/api/v1/measurements/completed/{id}		▼
GET	/api/v1/measurements/completed/{id}		▼
PATCH	/api/v1/measurements/completed/{id}		▼
GET	/api/v1/measurements/completed		▼
POST	/api/v1/measurements/completed		▼
GET	/api/v1/measurements/completed/attachments/{id}		▼
GET	/api/v1/measurements/completed/images/{id}		▼
GET	/api/v1/measurements/completed/latest		▼

4. Authenticate to access endpoints.
(Detailed information about different options for authentication here in 2.5.2 Authentication in the Swagger interface)

By signing in an authorized user the symbol changes

from  to .

5. Click *Try out* to query the API endpoint or have an insight on the documentation

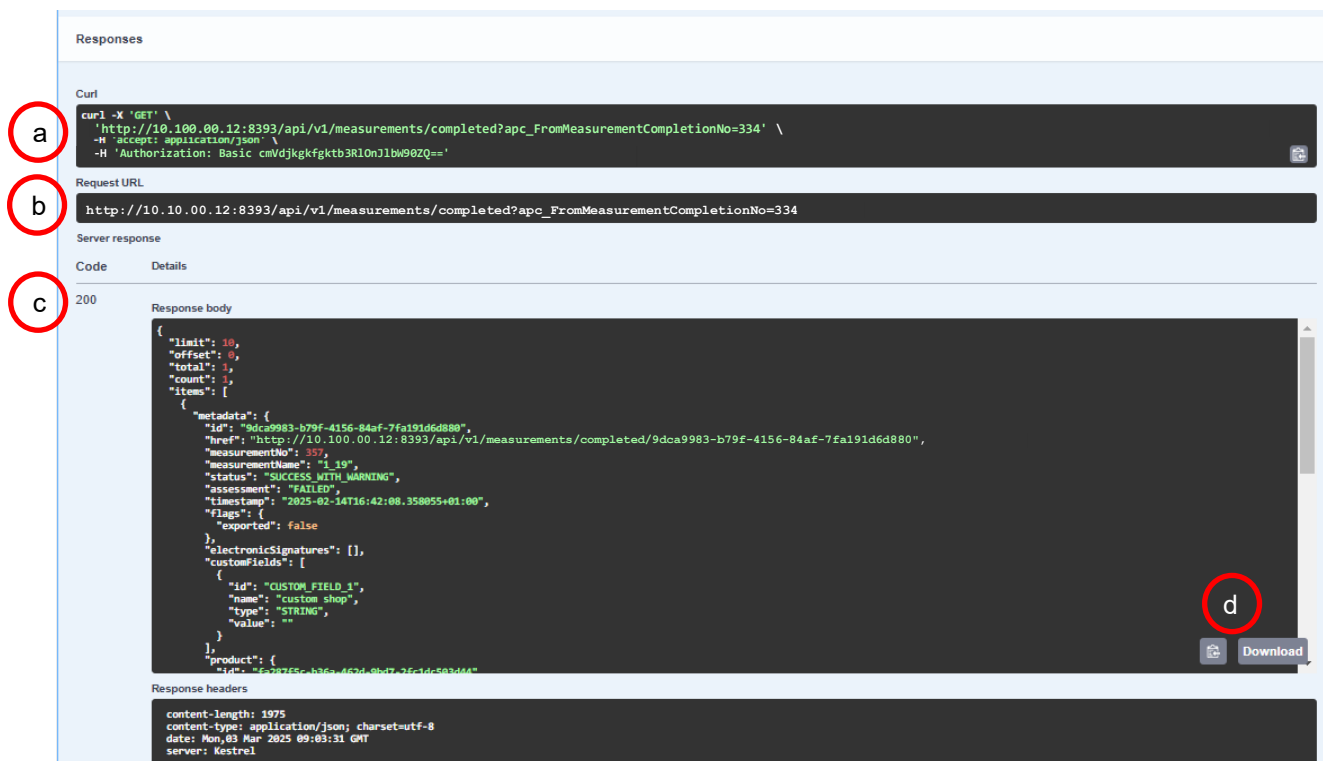
GET /api/v1/measurements/completed Returns a sequence of measurements

Each sequence item contains a reduced set of information about a single measurement, just the metadata, but no detailed information about the results or the measurement specification.
The request supports filtering and pagination. The sort order of the returned sequence is oldest to newest.

Parameters

Try it out

6. Enter information as described in the right format to specify the query
Required fields are marked with: *
7. Click *Execute* to query
8. Receive the response that gives back a code and contains a header and a body.
- Curl command and parameters used to execute the query
 - Request URL of the query
 - Server response: Code, Response body, Response header
Descriptions of all possible codes can be found after the response
 - To copy (left) or download (right) the response body



a

```
curl -X 'GET' \
  'http://10.100.00.12:8393/api/v1/measurements/completed?apc_FromMeasurementCompletionNo=334' \
  -H 'accept: application/json' \
  -H 'Authorization: Basic cmVkdjkgkfgktb3RlbnJlbW90ZQ=='
```

b

Request URL

http://10.100.00.12:8393/api/v1/measurements/completed?apc_FromMeasurementCompletionNo=334

c

Server response

Code Details

200

Response body

```
{
  "limit": 10,
  "offset": 0,
  "total": 1,
  "count": 1,
  "items": [
    {
      "metadata": {
        "id": "9dca9983-b79f-4156-84af-7fa191d6d880",
        "href": "http://10.100.00.12:8393/api/v1/measurements/completed/9dca9983-b79f-4156-84af-7fa191d6d880",
        "measurementNo": 357,
        "measurementName": "1_19",
        "status": "SUCCESS WITH WARNING",
        "assessment": "FAILED",
        "timestamp": "2025-02-14T16:42:08.358055+01:00",
        "flags": {
          "exported": false
        },
        "electronicSignatures": [],
        "customFields": [
          {
            "id": "CUSTOM_FIELD_1",
            "name": "custom shop",
            "type": "STRING",
            "value": ""
          }
        ],
        "product": {
          "id": "f328782c-636a-6634-8bd7-35f1d503444"
        }
      }
    }
  ]
}
```

d

Download

Response headers

```
content-length: 1975
content-type: application/json; charset=utf-8
date: Mon, 03 Mar 2025 09:03:31 GMT
server: Kestrel
```

2.6 Subscription

AP Connect can proactively inform an application whenever a measurement, check, or adjustment job is completed. This notification is implemented via a REST-based subscription model, allowing a server to react immediately upon completion of selected measurement types.

2.6.1 Server requirements & setup

To integrate this notification feature, ensure your server:

- **Provides a publicly accessible HTTP/HTTPS endpoint (WebURI)** capable of accepting **POST**-requests from AP Connect.
- **Can process notifications containing the unique measurement ID** to further query AP Connect and retrieve detailed measurement data.

2.6.2 Creation and deletion of subscriptions

To create or delete a subscription a server needs to send either a **POST**- or **DELETE**- request to AP Connect.

In the request body, you must specify a **webURI** and indicate whether notifications should be sent for *measurements*, *checks*, *adjustments*, or *all* measurement types. Selecting *all* includes or removes subscriptions for all available measurement types.

After a successful subscription, AP Connect sends an empty notification to the server as confirmation. Following this, for every completed job, AP Connect sends a REST-notification to the server, including the *measurement type*, *measurement ID*, and an *href* link to access the results and metadata.

When a subscription is deleted, AP Connect does not send any confirmation response to the server.

NOTE

If the server is unavailable when AP Connect attempts to deliver a REST notification. AP Connect will retry to send the notification three additional times (after 5, 10 and 15 seconds). If these attempts fail, AP Connect stops sending notifications for that particular job. Notifications will resume as normal for subsequent completed jobs.

2.7 Behaviour using AP Connect Pharma Edition

AP Connect Pharma Edition provides the electronic signature for a review process.

- REST subscription
A notification about a new measurement is only sent after completing the review process.
- REST query for measurement data
Data of a measurement/check/adjustment is only available via a GET-request after completing the review process.

For every set action an audit trail entry is written in AP Connect.

- REST subscription
That is also the case for creating and deleting REST-subscriptions. In case measurements, checks and adjustments are subscribed by using the *all*, three individual audit trail entries are created for every measurement type.
- REST query for measurement data
For querying a specific measurement, check or adjustment by their ID, also an audit trail entry is created to document the query as export.

2.8 Typical integration use cases

This section describes common REST API use cases for integrating a LIMS with AP Connect. In most workflows, one endpoint provides the identifier required by the next step. The most relevant identifiers are the instrument items[].id, the tasklist id, and the measurement id or href.

2.8.1 Create and execute a tasklist

Use this workflow when a tasklist should be created in AP Connect and optionally started via REST API.

HINT: For tasklists, the relevant tasklist permissions must be granted to the authenticated user.

NOTICE: When using AP Connect's REST API for the creation and execution of tasklists only Basic Authentication is supported.

The workflow starts by identifying the instrument:

- GET /api/v1/instruments

Keep from the response:

- items[].id
Use this value later as instrumentId.
- items[].products[].productName
Use this information to determine which products or methods are available on the instrument.
- optional: items[].instrumentName, items[].serialNumber
Useful for selecting the correct instrument.

Tasklists are created with:

- POST /api/v1/tasklists

The request body contains the tasklist definition. Relevant fields are:

- **instrumentId**
- **tasks[]**
- optional: **name** (of tasklist)
- optional: **magazineSize** (if magazine sample changer is in use)
- optional: **autoStart** (true or false)

For measurement tasks, the task object typically contains:

- **type** (either measurement, check or pause)
- **name** (for checks and pauses) / **sampleName** (for measurements)
- **magazinePosition** (for measurements or checks)
- **Method** (for measurements or checks)
- Optional: **customDataFields**

Two typical tasklist scenarios can be differentiated LIMS integrations: depending if a tasklist should be started automatically immediately after creation or with a manual step after optional further editing.

Scenario 1: Create a complete tasklist and start it immediately

Use this approach when all samples are already known when the tasklist is created, for example when a magazine sample changer is used and samples are ready to measure. In this case, the tasklist is created with all required tasks and **autoStart = true**, so execution starts directly after creation.

Scenario 2: Create a tasklist and add tasks later

Use this approach when measurements/checks/pauses are added continuously and the full tasklist is not yet known at creation time. In this case, the tasklist is created first and additional tasks are appended later by using:

- POST /api/v1/tasklists/{id}

A tasklist is can be manually started in the AP Connect UI, or by using:

- POST /api/v1/tasklists/{id}/start

After creation, keep the returned tasklist id. This identifier is required for all follow-up operations on the tasklist.

- **id** (of the tasklist)
- **href** (of the tasklist)

After creation, keep the returned **tasklist id**. This identifier is required for all follow-up operations on the tasklist.

To monitor a tasklist, use:

- GET /api/v1/tasklists (to search for a tasklist)
- GET /api/v1/tasklists/{id} (to see the content and state of a tasklist)

The tasklist state can be used to track progress. For already completed measurements in the tasklist, the returned task information contains the related measurement reference (**measurementId**). This measurement id can be extracted and then used to access the measurement results

- GET /api/v1/measurements/completed/{id}
- GET /api/v1/checks/completed/{id}

Additional metadata such as timestamps is also available in the tasklist state.

Not all instruments support tasklist creation and execution. This must be checked separately for the target device.

2.8.2 Search for several measurements

Use this workflow when the LIMS needs a list of completed measurements, for example to collect all measurements from a defined time range, instrument, method, or sample.

The main endpoint is:

- GET /api/v1/measurements/completed

This endpoint returns a sequence of completed measurements. Each returned item contains reduced information, mainly metadata, but not the full detailed results. The request supports filtering and pagination, and the returned sequence is ordered from oldest to newest.

Typical query parameters for narrowing the result set are:

- **Limit** (find details here: 2.9 Filtering, sorting, and paging)
- **Offset** (find details here: 2.9 Filtering, sorting, and paging)
- **FromTimestamp**
- **apc_ToTimestamp**
- **IncludeExported**
- **apc_Status**
- **apc_SampleName**
- **apc_Method** (name of method as stored on the instrument)
- **apc_Product** (name of product as stored on the instrument)
- **apc_InstrumentAlias** (used in AP Connect)
- **apc_InstrumentUser**
- **apc_InstrumentType**
- **apc_InstrumentSerialNumber**
- **apc_FromMeasurementCompletionNo** (is part of response after querying e.g. GET /api/v1/measurements/completed/latest)

From the list response, keep in particular:

- **items[].metadata.id**
- **items[].metadata.href**

These values are then used to query one specific measurement in full detail.

- GET /api/v1/measurements/completed/{id}

- GET /api/v1/checks/completed/{id}
- GET /api/v1/adjustments/completed/{id}

If **no subscription** and **no tasklist workflow** is used, completed datasets can still be located directly through the **completed-data** endpoints. **Besides filtering** with GET /api/v1/measurements/completed, AP Connect also provides:

- GET /api/v1/measurements/completed/latest

This endpoint returns the latest completion counter. The same concept is used across all measurement types and is also available for checks and adjustments. The returned completion number can be used to detect newly completed datasets and continue with a filtered request to the completed list endpoint.

Equivalent endpoint patterns exist for:

- GET /api/v1/checks/completed/latest
Received back **apc_FromMeasurementCompletionNo** can be used as query parameter for GET /api/v1/checks/completed
- GET /api/v1/adjustments/completed/latest
Received back **apc_FromMeasurementCompletionNo** can be used as query parameter for GET /api/v1/adjustments/completed

2.8.3 Retrieve the results of one measurement

Use this workflow when the LIMS already knows which completed measurement is relevant and needs the full result dataset.

The detail endpoint is:

- GET /api/v1/measurements/completed/{id}

This endpoint returns the complete measurement object, including metadata, specification, environment, and results. It is used to extract the actual result values of a completed measurement.

The required path parameter is:

- **{id}** (completed measurement id)

This id typically comes from one of the following sources:

- **items[].metadata.id** from GET /api/v1/measurements/completed
- **measurement id** obtained from a tasklist state
- **id** from a subscription callback
- **href** from a subscription callback, which already points to the completed object resource.

Fields that are typically relevant for follow-up processing are:

- metadata.id
- metadata.href
- metadata.flags.exported
- results

Result values are returned with the unit and precision configured in AP Connect.

2.8.4 Subscribe to completed measurements, checks, or adjustments

Use this when the LIMS should be notified automatically as soon as a new measurement, check, or adjustment is completed.

Subscriptions are configured by using:

- POST /api/v1/subscriptions/measurements
- DELETE /api/v1/subscriptions/measurements

To create a subscription, provide at least:

- **webhookUri** (a URI of a receiving server where a notification can be sent to)
- **measurementType** (individually for measurements, checks or adjustments, or for all: All)

The receiving server must provide a publicly accessible HTTP or HTTPS endpoint that can accept notifications from AP Connect. After a subscription has been created successfully, AP Connect sends an empty notification as confirmation. For each completed object, AP Connect then sends a callback containing the object reference. Relevant callback fields are:

- **id** (of the measurement, check or adjustment)
- **href** (which already points to the completed object resource)
- **measurementType**

The callback should be acknowledged successfully by the receiving server. In case of delivery failure, 3 retries are attempted after 5, 10 and 15 seconds each.

After receiving the callback, the LIMS typically continues with:

- GET /api/v1/measurements/completed/{id}

or directly by using the returned **href** to request the full dataset. Equivalent detail endpoints are available for checks and adjustments.

2.8.5 Mark measurements as exported

Use this workflow after a completed measurement has been processed successfully and should be marked as exported in AP Connect.

The endpoint is:

- PATCH /api/v1/measurements/completed/{id}

Equivalent endpoints are also available for checks and adjustments:

- PATCH /api/v1/checks/completed/{id}
- PATCH /api/v1/adjustments/completed/{id}

The required path parameter is:

- {id} = completed object id

The relevant request body field is:

- action = EXPORT

This workflow is used to document that the record has already been transferred or processed. The exported state is then available as part of the measurement metadata and can be considered in later search and filtering routines.

If data should be removed from AP Connect after processing, deletion is possible only after the object has been marked as exported.

2.9 Filtering, sorting, and paging

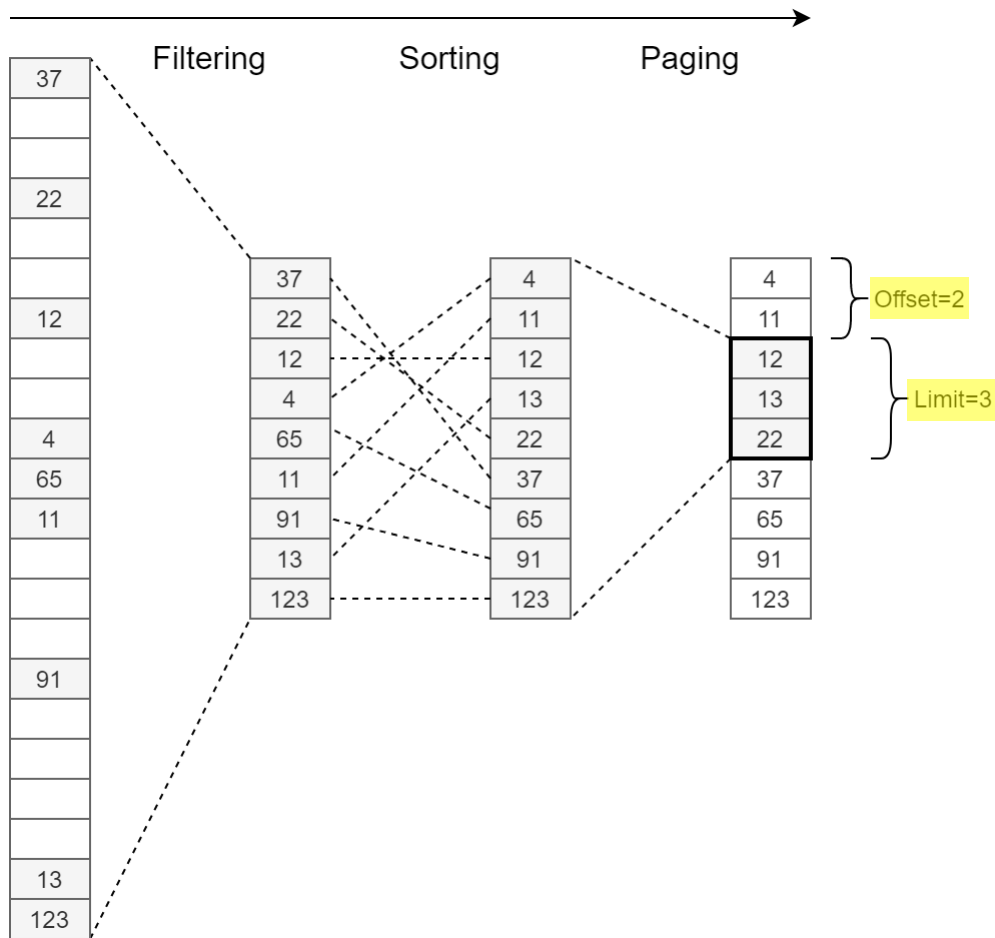
API endpoints that return a sequence of items support **sorting (decending)** , **filtering**, and **pagination**.

Filtering means reducing a larger set of items to a smaller one by removing (filter) items based on some criteria.

Sorting means ordering the sequence of items. The sorting algorithm must be stable, meaning it must return the same result order for the same input.

Paging means to return a sub-set of the sorted and filtered sequence of items, a so-called page.

Those three operations are applied in the following order: filtering, sorting (decending), paging as shown in the following illustration.



Each API endpoint usually defines its own filter criteria and sort options. But, all API endpoints use the same pagination concept.

3 Appendix

3.1.1 API Versioning

The API is generally versioned on two levels.

The **major version** (currently 1) is part of the URI, e.g. “api/v1”. If a second major version is being introduced in the future, then this will be reachable under “api/v2”, while “api/v1” is still being maintained. Major versions usually contain breaking changes and require manual migration.

For the major version “api/v1”, we also have a so-called **API revision**, for example, “API 100”. Higher numbers represent newer API versions which contain more features while still being compatible with older API revisions. If an API revision is specified it usually indicates the minimum API revision necessary for a feature or field.

3.2 Well-known tables

3.2.1 WellKnownSystemState

Part of object: *System/state*

Value	Remarks
“IDLE”	Indicates that the system is idle
“BUSY”	Indicates that the system is performing a measurement, check, adjustment or something similar

3.2.2 WellKnownAdjustmentStatus

Part of object: *AdjustmentMetadata/status*

Value	Remarks
“SUCCESS”	Indicates that the adjustment completed successfully without any warning or error.
“SUCCESS_WITH_WARNING”	Indicates that the adjustment completed successfully with at least one warning.
“SUCCESS_WITH_ERROR”	Indicates that the adjustment completed successfully with at least one error.
“CANCELED”	Indicates that the adjustment has been canceled by the user.
“FAILURE”	Indicates that the adjustment failed to complete. This usually indicates a technical problem.

3.2.3 WellKnownAdjustmentRecommendation

Part of object: *AdjustmentMetadata/recommendation*

Value	Remarks
“APPLY”	The system recommended to apply the new adjustment constants.
“REJECT”	The system recommended to reject the new adjustment constants.
“NONE”	The system did not give any recommendations.

3.2.4 WellKnownAdjustmentDecision

Part of object: *AdjustmentMetadata/decision*

Value	Remarks
“APPLIED”	The user decided to apply the new adjustment constants.
“REJECTED”	The user decided to reject the new adjustment constants.

3.2.5 WellKnownCheckStatus (Checkmetadata/status)

Part of object: *CheckMetadata/status*

Value	Remarks
"SUCCESS"	Indicates that the check measurement completed successfully without any warning or error.
"SUCCESS_WITH_WARNING"	Indicates that the check measurement completed successfully with at least one warning.
"SUCCESS_WITH_ERROR"	Indicates that the check measurement completed successfully with at least one error.
"CANCELED"	Indicates that the check measurement has been canceled by the user.
"FAILURE"	Indicates that the check measurement failed to complete. This usually indicates a technical problem.

3.2.6 WellKnownCheckAssessment

Part of object: *CheckMetaData/assessment*

Value	Remarks
"PASSED"	Indicates that the check passed. All results are within defined limits.
"FAILED"	Indicates that the check failed. One or more results exceed defined limits.
"UNDETERMINED"	Indicates that the outcome of the check is undetermined because the check measurement didn't complete successfully. See also status.

3.2.7 WellKnownElectronicSignatureSigningLevel

Part of object: *ElectronicSignature/signingLevel*

Value	Remarks
"SUBMITTER"	Represents the 'Submitter' level. Lowest level.
"REVIEWER"	Represents the 'Reviewer' level.
"APPROVER"	Represents the 'Approver' level. Highest level.

3.2.8 WellKnownMeasurementStatus

Part of object: *MeasurementMetadata/status*

Value	Remarks
"SUCCESS"	Indicates that the check measurement completed successfully without any warning or error.
"SUCCESS_WITH_WARNING"	Indicates that the check measurement completed successfully with at least one warning.
"SUCCESS_WITH_ERROR"	Indicates that the check measurement completed successfully with at least one error.
"CANCELED"	Indicates that the check measurement has been canceled by the user.
"FAILURE"	Indicates that the check measurement failed to complete. This usually indicates a technical problem.

3.2.9 WellKnownMeasurementAssessment

Part of object: *MeasurementMetadata/assessment*

Value	Remarks
"SUCCESS"	Indicates that the measurement completed successfully without any warning or error.
"SUCCESS_WITH_WARNING"	Indicates that the measurement completed successfully with at least one warning.
"SUCCESS_WITH_ERROR"	Indicates that the measurement completed successfully with at least one error.
"CANCELED"	Indicates that the measurement has been canceled by the user.
"FAILURE"	Indicates that the measurement failed to complete. This usually indicates a technical problem.

3.3 Reference Tables

The page contains reference tables with well-known constants used throughout the API.

3.3.1 Quantities and Units

Defines well-known physical quantities and associated units.

Physical Quantity	Unit Symbol	API	Remarks
SI-based Quantities			
"_"	" "	100	A dimensionless quantity
"t"	"s"	100	Time (Duration) in Second
"l"	"m"	100	Length in Meter
"m"	"kg"	100	Mass in Kilogram
"I"	"A"	100	Electric Current in Ampere
"T"	"K"	100	Temperature in Kelvin
"Td"	"K"	100	Temperature Difference in Kelvin
"n"	"mol"	100	Amount of Substance in Mole
"Iv"	"cd"	100	Luminous Intensity in Candela
Common SI-derived Quantities			
"ANGLE"	"rad"	100	Angle in Radian
"ENERGY"	"J"	100	Energy in Joule
"FORCE"	"N"	100	Force in Newton
"FREQUENCY"	"Hz"	100	Frequency in Hertz
"POWER"	"W"	100	Power in Watt
"PRESSURE"	"Pa"	100	Pressure in Pascal
"VOLTAGE"	"V"	100	Voltage in Volt
"AREA"	"m ² "	100	Area in Square Meter
"VOLUME"	"m ³ "	100	Volume in Cubic Meter
"DENSITY"	"kg/m ³ "	100	Density in SI units
"KINEMATICVISCOSITY"	"m ² /s"	100	Kinematic Viscosity in SI units
"DYNAMICVISCOSITY"	"Pa·s"	100	Dynamic Viscosity in SI units
"PERIOD"	"s"	110	Period (reciprocal of Frequency) in Second
"SPEED"	"m/s"	110	Speed in Meters per Second
"ROTATIONALSPEED"	"rad/s"	110	Rotational Speed in Radians per Second
"RELATIVEHUMIDITY"	"%"	110	Relative Humidity in Percent
"RELATIVEAIRSATURATION"	"%"	110	Relative Air Saturation in Percent
"PRESSUREPERTIME"	"Pa/s"	110	Pressure per Time in Pascal per Second
"MOLARITY"	"mol/L"	110	Molar Concentration (Molarity) in Mole per Liter
Special Purpose Quantities			
"AIRMASS"	"g"	110	O2 headspace air in Gram
"O2MASS"	"g"	110	O2 headspace in Gram
"MASSCONCENTRATION"	"kg/m ³ "	110	Weight (Mass) per Volume in SI units
"DILUTECONCENTRATION"	"kg/cm ³ "	110	Concentration of Dilution in Weight (Mass) per Volume in SI units
"SUGARCONCENTRATION"	"°Bx"	110	Concentration of Sugar in Degrees Brix
"ALCOHOLVOLUMECONCENTRATION"	"% v/v"	110	Volume Concentration for alcohols.
"MASSFRACTION"	"% w/w"	110	Mass fraction for general use
"ACIDMASSFRACTION"	"% w/w"	110	Mass fraction for acids
"ALCOHOLMASSFRACTION"	"% w/w"	110	Mass fraction for alcohols

AP Connect REST Interface Documentation

for AP Connect 4.0



"BASEMASSFRACTION"	"% w/w"	110	Mass fraction for bases
"SALTMASSFRACTION"	"% w/w"	110	Mass fraction for salts
"SUGARMASSFRACTION"	"% w/w"	110	Mass fraction for sugar
"SUGARMASSFRACTIONKMW"	"°KMW"	110	Mass fraction for sugar based on Klosterneuburger Mostwaage
"BEERCOLOR"	"EBC"	110	Beer Color in EBC
"DEGREEOFFERMENTATION"	"%"	110	Degree of Fermentation in Percent
"SAKEMETERVALUE"	"°SMV"	110	Sake Meter Value in Degree SMV
"RELATIVEDENSITY"	" "	110	Relative Density as Ratio
"RELATIVEDENSITYBAUME"	"B°"	110	Relative Density based on the Baumé scale
"APIGRAVITY"	"°API"	110	API Gravity in Degree API
"HAZE"	"EBC"	110	Turbidity (Haze) in EBC
"GASABSORPTION"	"g/(L·bar)"	110	Adsorption of gas
POLYMERCONCENTRATIONMASSPERMASS	mg/g	140	Lovis polymer specific quantity
POLYMERCONCENTRATIONMASSPERVOLUME	g/ml	140	Lovis polymer specific quantity
POLYMERSOLUTIONVISCOSITY	ml/g	140	Lovis polymer specific quantity
POLYMERMOLARMASS	g/mol	140	Lovis polymer specific quantity
HEATINGRATE	°C/min	140	Heating rate
SHEARRATE	s ⁻¹	140	Shear Rate
PERCENTSEAWATEREQUIVALENT	% SWE	140	Percent Sea Water Equivalent is a term that indicates that the ratio of the mass of a liquid to the mass of seawater that has the same density.

Unit symbols may contain the following non-ASCII characters.

Character	Example	Remarks
²	m ²	U+00B2 Superscript Two
³	kg/m ³	U+00B3 Superscript Three
·	Pa·s	U+00B7 Middle Dot
°	°Brix	U+00B0 Degree Sign

3.3.2 WellKnownMeasuredValues

Defines well-known key/value pairs representing measured values.

Id	Type	API	Remarks
Density			
Density/SetTemperature	Quantity / Temperature (Variant)	100	The set (defined) temperature of the density cell.
Density/CellTemperature	Quantity / Temperature (Variant)	100	The actual temperature of the density cell.
Density/Density	Quantity / Density (Variant)	100	The measured density (viscosity corrected).

3.3.3 WellKnown Metadata

Defines well-known key/value pairs.

Id	Type	API	Remarks
Camera/Image	Image (Variant)	100	The camera image taken during a measurement, a check, or an adjustment.
Report/File	File (Variant)	140	The report file for a measurement, a check, or an adjustment created by the instrument.

3.3.4 Variant

Defines how a *Variant* is represented.

A *Variant* is a flexible, generic key/value pair (a keyed value) that can express various data types and data formats.

3.3.4.1 Variant Design

The basic type Variant defines fields shared across all variants. Concrete Variant implementations (see Variant Type below) can define **additional fields** for a *Variant*.

3.3.4.2 Identification

The simplest representation contains just an id field. The id is mandatory and is used for uniquely identifying the variant in its enclosing collection.

```
{  
  "id": "foo"  
}
```

3.3.4.3 Translated Display Names

In addition to the id, a Variant can also contain a translated name. The name is used for display on user interfaces, reports, etc. It is supposed to be a human readable text.

```
{  
  "id": "Baseboard/BoardHumidity",  
  "name": "Humidity"  
}
```

The name field contains the translation for the requested language (see Accept-Language Request Header). Clients can also request the instrument to return translations for all available languages (see Include-Languages Request Header). In that case, an additional names field is present.

```
{  
  "id": "Baseboard/BoardHumidity",  
  "name": "Humidity",  
  "names": {  
    "en": "Humidity",  
    "de": "Luftfeuchtigkeit (Platine)",  
    "cs": "vlhkost (deska)",  
    "fr": "Humidity",  
    "hu": "Humidity",  
    "it": "Umidità",  
    "ja": "",  
    "ko": "Hum. (Board)",  
    "pl": "Wilgot. (Pyta)",  
    "pt": "Um. (placa)",  
    "ru": "",  
    "tr": "Humidity",  
    "zh": "(Board)",  
    "es": "Hum. (Tablero)"  
  }  
}
```

The *names* field is typically represented by object representation Translations.

3.3.4.4 Variant Type

A variant has an associated *type*.

```
{  
  "id": "foo",  
  "type": "INT32"  
}
```

If no type is specified, it defaults to "STRING".

Depending on the type of the Variant, additional fields are present. The Variant type acts as "abstract base class" for concrete Variant implementations, and concrete implementations inherit all fields defined by the basic Variant.

3.3.4.5 Variant Values

Variant values are different per Variant type, but they share common concepts:

- The JSON Null value expresses emptiness, allowing for optionality. This is equivalent to omitting the value field in the JSON.
- A value of T expresses a singleton value.
- A value of T[] expresses a list of T values.

The Quantity Variant behaves special, due to its nature, with details being explained in the respective sub-page.

3.3.4.6 Available Variants

Variant types for the following use cases are defined, and detailed information about additional fields can be found in the specific sections:

- Collection Variant
- Enum Variant
- Image Variant
- Percentage Variant
- Primitive Datatype Variants
- Quantity Variant
- Tabular Data Variant
- Timestamp and Duration Variants

3.3.4.7 Object Representation

Object	Field	Type	Obligation	API	Remarks																																									
Variant	id	String	Mandatory	100	An id used for identification.																																									
	name	String	Optional	100	The translated display name.																																									
	names	Translations	Optional	100	The translated user-friendly display names.																																									
	type	String	Optional (pre-130) Mandatory (from 130 on) Note: although the type was optional it is recommended to always send type information, in any protocol version implemented. Note: Synapse consumers should implement robust parsing and be able to deal with missing type information.	100	<div>The data type of the value. One of these values:</div> <table><thead><tr><th>Key</th><th>API</th><th>Remarks</th></tr></thead><tbody><tr><td>"COLLECTION"</td><td>100</td><td>A collection of variants.</td></tr><tr><td>"STRING"</td><td>100</td><td>A string value.</td></tr><tr><td>"BOOL"</td><td>100</td><td>A boolean value.</td></tr><tr><td>"INT32"</td><td>100</td><td>A 32-bit integer value.</td></tr><tr><td>"FLOAT64"</td><td>100</td><td>A 64-bit floating-point value.</td></tr><tr><td>"DATETIME"</td><td>100</td><td>A date/time in ISO 8601 representation.</td></tr><tr><td>"TIMESPAN"</td><td>100</td><td>A time span/duration in ISO 8601 representation.</td></tr><tr><td>"PERCENTAGE"</td><td>100</td><td>A percentage value.</td></tr><tr><td>"QUANTITY"</td><td>100</td><td>A 64-bit floating-point value in a specific unit/quantity.</td></tr><tr><td>"IMAGE"</td><td>100</td><td>An image value.</td></tr><tr><td>"EMBEDDED_IMAGE"</td><td>130</td><td>An image value, where the image is not a separate resource, but embedded in the document.</td></tr><tr><td>"ENUM"</td><td>100</td><td>An enumeration value.</td></tr><tr><td>"DATASERIES"</td><td>130</td><td>A set of series of uniform data, that will ultimately be interpreted as a table.</td></tr></tbody></table> <div>If omitted, defaults to "STRING". Please note that future API Versions might extend the set of valid values.</div>	Key	API	Remarks	"COLLECTION"	100	A collection of variants.	"STRING"	100	A string value.	"BOOL"	100	A boolean value.	"INT32"	100	A 32-bit integer value.	"FLOAT64"	100	A 64-bit floating-point value.	"DATETIME"	100	A date/time in ISO 8601 representation.	"TIMESPAN"	100	A time span/duration in ISO 8601 representation.	"PERCENTAGE"	100	A percentage value.	"QUANTITY"	100	A 64-bit floating-point value in a specific unit/quantity.	"IMAGE"	100	An image value.	"EMBEDDED_IMAGE"	130	An image value, where the image is not a separate resource, but embedded in the document.	"ENUM"	100	An enumeration value.	"DATASERIES"	130
Key	API	Remarks																																												
"COLLECTION"	100	A collection of variants.																																												
"STRING"	100	A string value.																																												
"BOOL"	100	A boolean value.																																												
"INT32"	100	A 32-bit integer value.																																												
"FLOAT64"	100	A 64-bit floating-point value.																																												
"DATETIME"	100	A date/time in ISO 8601 representation.																																												
"TIMESPAN"	100	A time span/duration in ISO 8601 representation.																																												
"PERCENTAGE"	100	A percentage value.																																												
"QUANTITY"	100	A 64-bit floating-point value in a specific unit/quantity.																																												
"IMAGE"	100	An image value.																																												
"EMBEDDED_IMAGE"	130	An image value, where the image is not a separate resource, but embedded in the document.																																												
"ENUM"	100	An enumeration value.																																												
"DATASERIES"	130	A set of series of uniform data, that will ultimately be interpreted as a table.																																												

3.3.4.8 Collection Variant

Variants can optionally belong to a collection, which is used for grouping and categorizing inputs and result values. Items in a COLLECTION are again Variants. It's even possible to nest collections into collections.

```
{
  ...
  "type": "COLLECTION",
  "items": [
    {
      "id": "Baseboard/BoardHumidity",
      "name": "Humidity",
      "type": "PERCENTAGE",
      "value": 0.75
    }
  ]
}
```

Object representation:

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
Collection Variant	type	String	Mandatory with a value of	100	The type of the Variant.

			"COLLECTION"		
	items	Variant[] Null	Optional	100 130 (for arrays and null)	An array of Variants that represent the items in this collection. Note: A value of Null expresses optionality. Null is not an empty list. Note: This Variant does not support singleton values; items is always a list or Null. Singleton values can be expressed by lists with one element.

3.3.4.9 Enumeration Values

A *Variant* can represent an enumeration value, which usually consists of a value and a corresponding display name for that value.

Example:

```
{
  ...
  "type": "ENUM",
  "value": {
    "key": "FAST",
    "name": "Fast",
    "names": {
      "en": "Fast",
      "de": "Schnell",
      ...
    }
  }
}
```

The *key* field is a technical identifier representing the enumeration value, e.g. "FAST", "PRECISE", "ULTRA-FAST". The *name(s)* field(s) contain a corresponding user-friendly display name of the key (not the Enum itself, this is defined already by the *name(s)* field for the enclosing *Variant* structure), e.g. "Fast", "Precise", "Ultra Fast".

Object representation:

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
EnumVariant	type	String	Mandatory with a value of "ENUM"	100	The type of the Variant.
	items	EnumValue EnumValue[] Null	Optional	100 130 (for arrays and null)	The number must be interpreted by consumers as 64 bit floating point number.
EnumValue	key	String	Mandatory	100	The technical identifier of the Enum.
	name	String Null	Optional	100	User-friendly display name of the key
	names	Translations Null	Optional	100	Translated user-friendly display names of the key

3.3.4.10 Image Variant

A *Variant* can represent an image, for example, a camera image taken during measurement.

Images

An *ImageVariant* can represent an image by hyperlinks to image resources.

Example:

```
{
  ...
  "type": "IMAGE",
  "href": "http://10.100.10.12:8393/api/v1/measurements/completed/5f993c39-e02c-4275-bd9c-a33cf38e3841/cameraImage/51a019a3-02f4-4938-a541-c0e24feada8a"
}
```

An image is represented by a URI pointing to the image; so, a separate download of that image is needed.

Object Representation:

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
ImageVariant	type	String	Mandatory with a value of "IMAGE"	100	The type of the Variant.
	href	String String[] Null	Optional	100 130 (for arrays and null)	An URI to the image resource. The MIME type of the resource must be of one of the supported types listed below.

Embedded Images Values

An *EmbeddedImageVariant* can represent an image, that is embedded into the enclosing measurement.

Example:

```
{
  ...
  "type": "EMBEDDED_IMAGE",
  "value": {
    "blob": "/9j/4AAQSkZJRgABAQAAQABA.....",
    "filename": "my_cell_image.jpg",
    "filetype": "image/jpeg"
  }
}
```

An image is represented by its Base64 encoded byte stream.

Object representation:

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
EmbeddedImageVariant	type	String	Mandatory with a value of "EMBEDDED_IMAGE"	130	The type of the Variant.

	value	EmbeddedImageValue EmbeddedImageValue[] Null	Optional	130	
EmbeddedImageValue	blob	String	Mandatory	130	The base64 encoded byte stream of the image.
	filename	String	Mandatory	130	The filename of the image.
	filetype	String	Mandatory	130	The MIME type specifier for the image file. Must be one of the supported types listed below.

Supported File Types

The file type of an image is a MIME type specifier. The following MIME types are supported:

MIME Type	Description
image / jpeg	A JPEG image.
Image / png	A PNG image.

Both IMAGE and EMBEDDED_IMAGE are supposed to deliver only images of the specified formats.

3.3.4.11 Percentage Variant

Variants can represent percentage values. Additionally to the basic variant fields, an additional field value exists.

Example:

```
{
  ...
  "type": "PERCENTAGE",
  "value": 0.75           // value 0.75 means 75%
},
{
  ...
  "type": "PERCENTAGE",
  "value": null
},
{
  ...
  "type": "PERCENTAGE",
  "value": [0.75, 0.80, 0.85, 0.90]
}
```

Object representation:

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
Percentage Variant	type	String	Mandatory with a value of "PERCENTAGE"	100	The type of the variant.
	value	Number Number[] Null	Optional	100 130 (for arrays and null)	The number(s) must be interpreted by consumers as 64 bit floating point number.

3.3.4.12 Primitive Datatype Variants

StringVariant

Example

```
{
  ...
  "type": "STRING",
  "value": "bar"
},
{
  ...
  "type": "STRING",
  "value": ""
},
{
  ...
  "type": "STRING",
  "value": null
},
{
  ...
  "type": "STRING",
  "value": [ "my", "first", "variant" ]
}
```

Object Representation:

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
StringVariant	type	String	Mandatory with a value of "STRING"	100	The type of the variant. Note: in previous versions it was allowed to omit the type and it defaulted to "STRING". It is now recommended that type information is always sent. Consumers of Synapse still should have robust parsing that can deal with missing type information.
	value	String String[] Null	Optional	100 130 (for arrays and null)	A string value. Note: A value of Null expresses optionality. Null is not equal to the empty string.

BoolVariant

Example

```
{
  ...
  "type": "BOOL",
  "value": true
},
{
  ...
  "type": "BOOL",
  "value": false
}
```

AP Connect REST Interface Documentation

for AP Connect 4.0



Object Representation

7.5	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
BoolVariant	type	String	Mandatory with a value of "BOOL"	100	The type of the variant.
	value	Boolean Boolean[] Null	Optional	100 130 (for arrays and null)	A boolean value or null.

Int32Variant

Example

```
{  
  ...  
  "type": "INT32",  
  "value": 123456789  
}
```

Object Representation

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
Int32Variant	type	String	Mandatory with a value of "INT3 2"	100	The type of the variant.
	value	Number Number[] Null	Optional	100 130 (for arrays and null)	Interpreted as 32 bit integer number.

Float64Variant

Example

```
{  
  ...  
  "type": "FLOAT64",  
  "value": 123.456789  
}
```

Object Representation

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	[API130] Translations Null	Optional	100	The translated user-friendly display names.
Float64Variant	type	String	Mandatory with a value of "FLOAT 64"	100	The type of the variant.
	value	Number Number[] Null	Optional	100 130 (for arrays and null)	Interpreted as 64 bit floating point number.

```
{
  "id": "Density/SystemMonitoring",    // collection id
  "name": "System Monitoring",         // collection name
  "names": {
    "en": "System Monitoring",
    "de": "Systemüberwachung",
    "cs": "System Monitoring",
    "fr": "System Monitoring",
    "hu": "System Monitoring",
    "it": "Monitor sistema",
    "ja": "",
    "ko": " ",
    "pl": "System Monitoring",
    "pt": "Monitoramento do Sistema",
    "ru": " ",
    "tr": "System Monitoring",
    "zh": "",
    "es": "System Monitoring"
  },
  "type": "COLLECTION",
  "items": [
    {
      "id": "Baseboard/BoardHumidity",
      "name": "Humidity",
      "type": "PERCENTAGE",
      "value": 0.75
    }
  ]
}
```

3.3.4.13 Quantity Variant

To transfer numeric values in a specific unit the type "QUANTITY" must be specified. In that case, the value field is expanded to an object.

```
{
  "id": "a",
  "type": "QUANTITY",                // well-known
  "value": {
    "numeric": 996.82,
    "unit": "kg/m³",
    "quantity": "DENSITY"
  }
},
{
  "id": "b",
  "type": "QUANTITY",                // well-known: dimensionless
  "value": {
    "numeric": 123.456,
    "unit": "",
    "quantity": "-"
  }
},
{
  "id": "a",
  "type": "QUANTITY",                // NOT well-known
  "value": {
    "numeric": 996.82,
    "unit": "mol/L"
  }
}
```

The *numeric* value is always a 64-bit floating-point value.

Quantities and Units defines a list of well-known physical quantities and their units. Numeric values should be represented in well-known units for defined quantities if possible, otherwise, consumers won't be able to interpret the received representations.

For cases where no well-known physical quantity exists, the *quantity* field is omitted. Here only the *numeric* value and the *unit* symbol are available.

```
{
  "id": "a",
  "type": "QUANTITY",
  "value": {
    "numeric": 996.82,
    "unit": "foos"
  }
}
```

Quantity-based values can have missing numeric values.

```
{
  ...
  "type": "QUANTITY",
  "value": {
    "empty": true,
    "unit": "kg/m³",
    "quantity": "DENSITY"
  }
},
{
  ...
  "type": "QUANTITY",
  "value": {
    "empty": true,
    "unit": "foos"
  }
}
```

This is indicated by *empty* and by omitting the field *numeric*.

Quantity-based values can also indicate *out-of-range*.


```
{
  ...
  "type": "QUANTITY",
  "value": {
    "outOfRange": true,
    "numeric": 996.82,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "ranges": {
      "lower": 900,    // kg/m³
      "upper": 999    // kg/m³
    }
  }
},
{
  ...
  "type": "QUANTITY",
  "value": {
    "outOfRange": true,
    "numeric": 996.82,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "ranges": {
      "lower": null,   // no lower limit defined
      "upper": 999    // kg/m³
    }
  }
}
```

This is indicated by `outOfRange` and by the additional field `ranges`, which optionally describe the lower and upper numeric bounds (represented in the same unit). It's possible that only an upper or only a lower bound exists. It's also possible that neither info is available. In that rare case, both lower and upper are null or omitted.

```
{
  ...
  "type": "QUANTITY",
  "value": {
    "empty": true,
    "outOfRange": true,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "ranges": {
      "lower": 900,    // kg/m³
      "upper": 999    // kg/m³
    }
  }
}
```

In some cases, for example for aggregated values, it is necessary to transfer not only the aggregated average value but also the corresponding standard deviation.

```
{
  "id": "a",
  "type": "QUANTITY",
  "value": {
    "numeric": 996.82,
    "stddev": 0.002,
    "unit": "kg/m³",
    "quantity": "DENSITY"
  }
}
```

Here the field `stddev` is used to express the standard deviation numeric value (in the same unit as the numeric value).

Similar to the numeric field also `stddev` is always a 64-bit floating-point value.

3.3.4.14 Precision of Quantity-based Values

To specify the precision of the numeric value, two different approaches are supported. First, it's possible to simply specify the number of digits after the decimal point.

```
{
  ...
  "type": "QUANTITY",
  "value": {
    "numeric": 995.69369,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "digits": "2"
  }
}
```

The field *digits* defines the number of digits after the decimal point for the given unit. The resulting density would be formatted as "995.69 kg/m³".

It's possible to specify a range of digits to express the lowest and highest possible positions after the decimal point.

```
{
  ...
  "type": "QUANTITY",
  "value": {
    "numeric": 995.696,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "digits": "0-3"
  }
}
```

Which results in "996 kg/m³" to "995.694 kg/m³".

The same information can be specified also by field *precision*.

```
{
  ...
  "type": "QUANTITY",
  "value": {
    "numeric": 995.6936,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "precision": "0.05"
  }
}
```

Here, it's also possible to specify a precision range.

```
{
  ...
  "type": "QUANTITY",
  "value": {
    "numeric": 995.6936,
    "unit": "kg/m³",
    "quantity": "DENSITY",
    "precision": "10.0-0.005"
  }
}
```

AP Connect REST Interface Documentation

for AP Connect 4.0



Object Representation

Object	Field	Type	Obligation	API	Remarks
Variant	id	String	Mandatory	100	An id used for identification.
	name	String Null	Optional	100	The translated display name.
	names	Translations Null	Optional	100	The translated user-friendly display names.
QuantityVariant	type	String	Mandatory with a value of "QUANTITY"	100	The type of the variant.
	value	QuantityValue	Mandatory	100	An object describing the value of the quantity.
QuantityValue	empty	Boolean Null	Optional	100 130 (for array behavior)	Indicates whether the value represents empty/nothing/missing. Defaults to false. Note: If <i>empty=true</i> the value of <i>numeric</i> should be completely ignored. Note: If <i>numeric</i> is an array, the value of this field should be ignored, since emptiness is defined by each element in the array.
	outOfRange	Boolean Null	Optional	100 130 (for array behavior)	Indicates whether the value is out-of-range. Defaults to false. Note: If <i>outOfRange=true</i> the field <i>numeric</i> can still contain a value that should be considered. Note: If <i>numeric</i> is an array, the value of this field should be ignored, since OutOfRange-ness is defined by each element in the array.
	numeric	Number (Number "OOR" Null)[] Null	Optional	100 130 (for arrays and null)	The numeric value(s). Should be interpreted as 64 bit floating point number. Note: a list element with value of <i>Null</i> should be treated as if it were a Quantity Variant with <i>empty=true</i> . A list element with the value of "OOR" (by case insensitive comparison) should be treated as if it were a Quantity Variant with <i>outOfRange=true</i> with the associated value of <i>numeric</i> being <i>Null</i> .
	stddev	Number (Number Null)[] Null	Optional	100 130 (for arrays and null)	The standard deviation in the same unit as <i>numeric</i> . Should be interpreted as 64 bit floating point number. Note: the dimensionality of this field must be the same as the dimensionality of <i>numeric</i> . If <i>numeric</i> is <i>Null</i> this field should be assumed <i>Null</i> too, regardless of its value.
	unit	String Null	Optional	100 130 (for array behavior)	The unit symbol. See also Quantities and Units. Note: if <i>numeric</i> is an array, this unit should be assigned to each array element when interpreted by the consumer.
	quantity	String Null	Optional	100 130 (for array behavior)	The physical quantity. See also Quantities and Units. Note: if <i>numeric</i> is an array, this quantity should be assigned to each array element when interpreted by the consumer.
	precision	String Null	Optional	100 130 (for array behavior)	The precision of the numeric value, i.e. "0.01" or a range lowest-to-highest "10.0-0.005". In the given unit. Note: if <i>numeric</i> is an array, this precision should be assigned to each array element when interpreted by the consumer.
	digits	String Null	Optional	100 130 (for array behavior)	The number of digits after the decimal point for the numeric value, i.e. "2" or a range lowest-to-highest "0-3". Applies to the given unit. Note: if <i>numeric</i> is an array, these digits should be assigned to each array element when interpreted by the consumer.
	ranges	RangeValue Null	Optional	100 130 (for array behavior)	An object describing the ranges causing the out-of-range state. Note: if <i>numeric</i> is an array, these ranges should be assigned to each array element when interpreted by the consumer.
RangeValue	lower	Number Null	Optional	100	The lower bound numeric value in the same unit as <i>numeric</i> and <i>stddev</i> . Should be interpreted as 64 bit floating point number.
	upper	Number Null	Optional	100	The upper bound numeric value in the same unit as <i>numeric</i> and <i>stddev</i> . Should be interpreted as 64 bit floating point number.

3.3.4.15 Tabular Data Variant

Synapse distinguishes between two fundamentally different tabular data representations:

- *Data series*, where large amounts of **uniform** tabular data can be expressed, and
- *Common tables*, where arbitrary tables can be constructed on a Variant-per-cell approach.

Data Series

Data series allow for expressing a set of series of uniform data, that will ultimately be interpreted as table, where a column represents one data series.

Column headers are taken from the Variant that represents a series. Variants are expected to be of dimension 1 (list-like).