How to use Data Sharing API

V4

Date	2023/03/24
Author	Gopi Ande
Verified	
Revision	4

Table of Contents

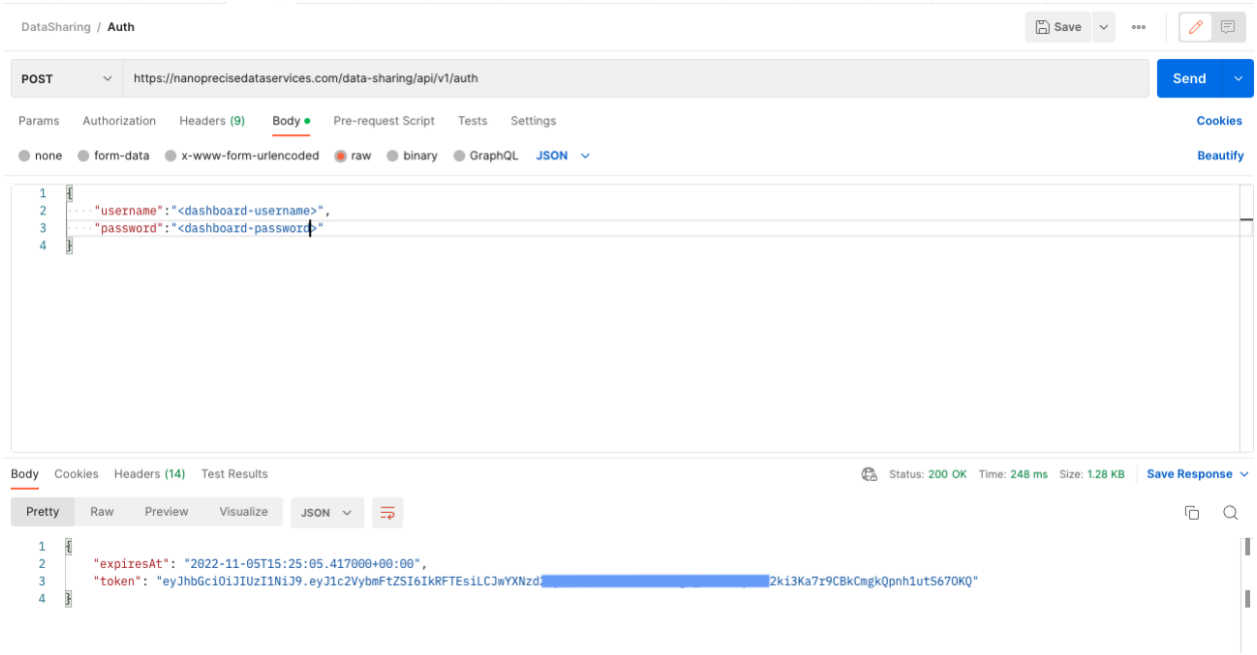
1. Requirements

- You need an HTTP client.
- The base url for all requests is https://nanoprecisedataservices.com/data-sharing/api/v1. you need to append the end point to this url to get the data
- You need dashboard login credentials

2. Authentication

2.1. /auth [POST]

1. Make a request to /auth endpoint using the username and password given to you to get the bearer token. The credentials for this are **same as** our **dashboard login credentials**.
2. You will get a bearer token along with expiry time of that token. So, you can make as multiple requests using the same token before it expires. After that you have to get the new token to request the data again.
3. Example (Requesting auth token from postman):



3. EndPoints Descriptions

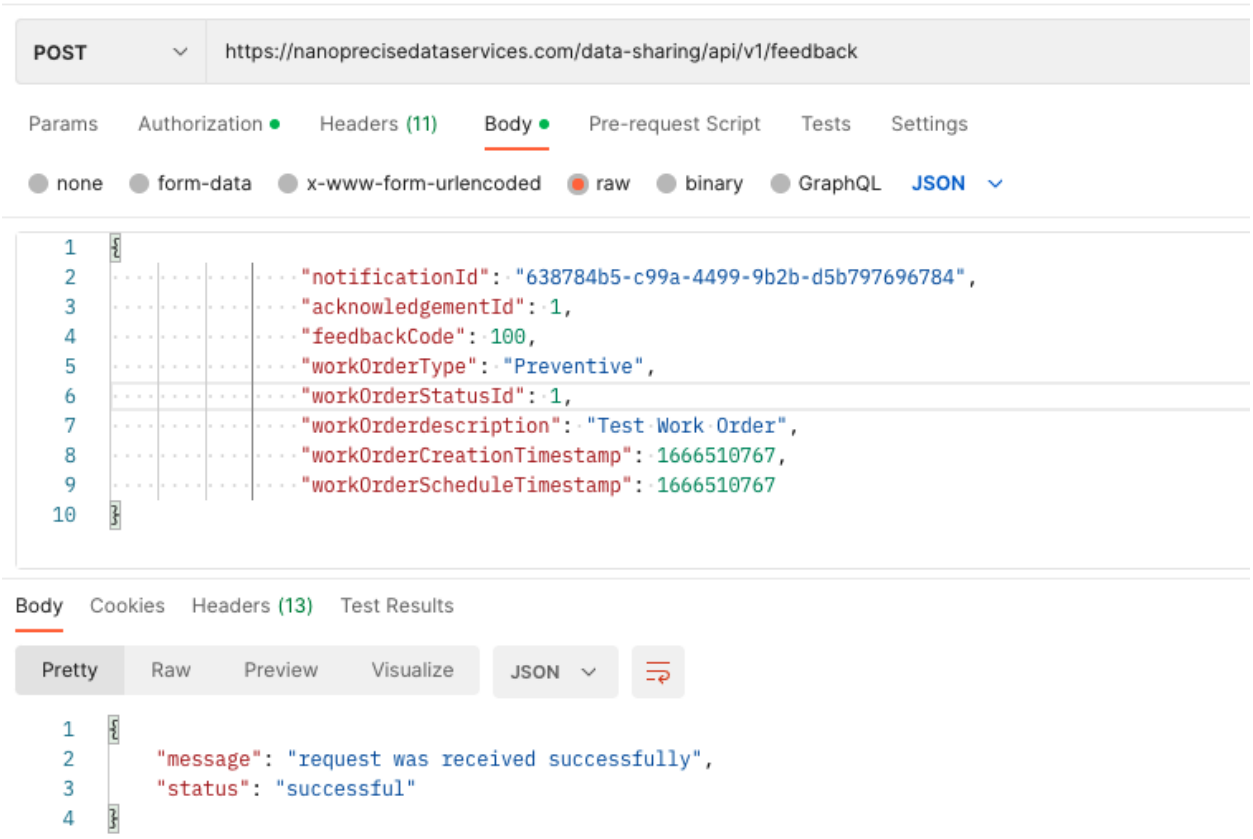
3.1. /feedback [POST]

To send feedback to Nanoprecise regarding the given notification shared through notification endpoint to improve the fault detection

Request body should contain the following parameters:

Parameter	Description	Type	enum
feedbackCode	To differentiate between feedback for different companies. Each company will receive a feedback code offline to use here	integer	
acknowledgementId	Integer value between [0,1,2] 0 New, 1 Acknowledged, 2 Closed/Resolved type: integer	integer	[1, 2, 3]
notificationId	Associating the feedback to specific notification	string	
workOrderType	Once value from Set of work order types. Types: ['Preventive', 'Damage', 'Corrective', 'Safety', 'Upgrade', 'Electrical', 'Project', 'Inspection', 'Meter reading', 'Other']	string	
workOrderStatusId	Integer value between [0,1,2] 0 Open, 1 in progress, 2 closed	integer	[1, 2, 3]
workOrderScheduleTimestamp	When work order is scheduled to happen	integer	
workOrderCreationTimestamp	When work order was created	integer	
workOrderdescription	Description of work order	string	

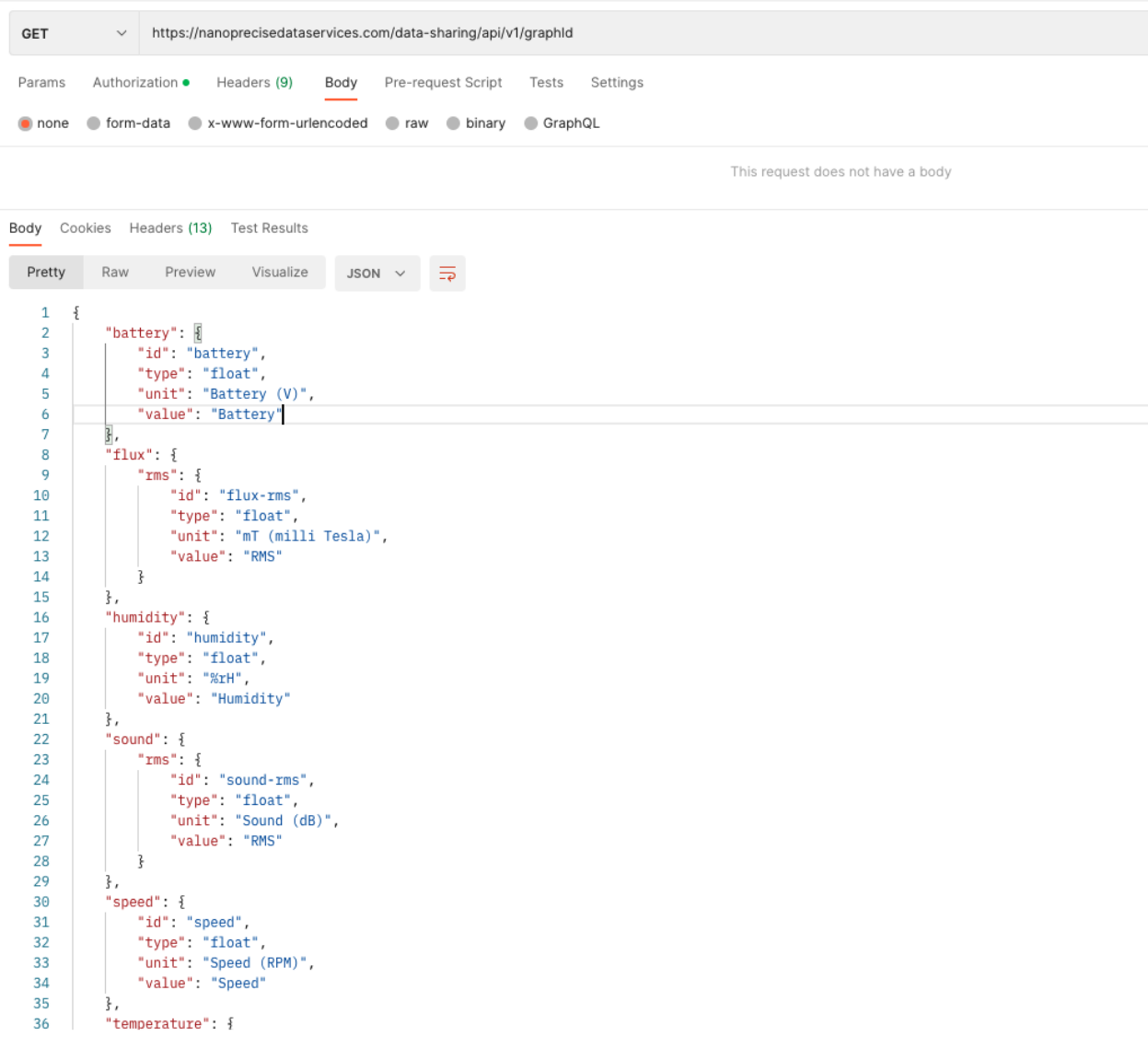
Example:


POST
https://nanoprecisedataservices.com/data-sharing/api/v1/feedback
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"notificationId": "638784b5-c99a-4499-9b2b-d5b797696784",
"acknowledgementId": 1,
"feedbackCode": 100,
"workOrderType": "Preventive",
"workOrderStatusId": 1,
"workOrderdescription": "Test Work Order",
"workOrderCreationTimestamp": 1666510767,
"workOrderScheduleTimestamp": 1666510767
}
Body
Cookies
Headers (13)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"message": "request was received successfully",
"status": "successful"
}

3.2. /graphId [GET]

It provides the Id to use for getting the data from `/analytics/graph` endpoint. The returned object `id` field's value needs to be used as `graphId` in `/analytics/graph`. You shouldn't pass anything in the request body for this request.

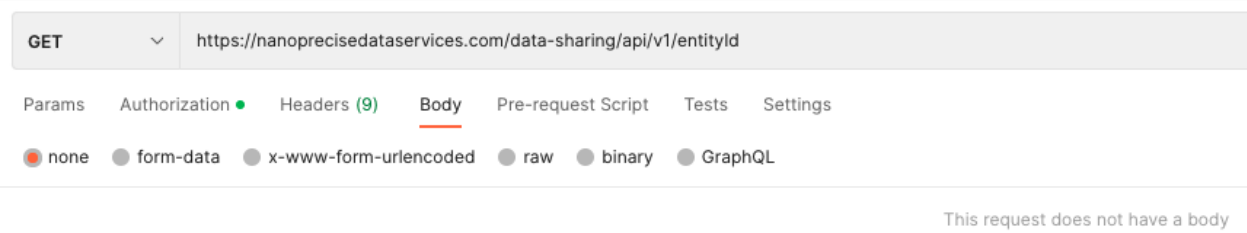
Example:

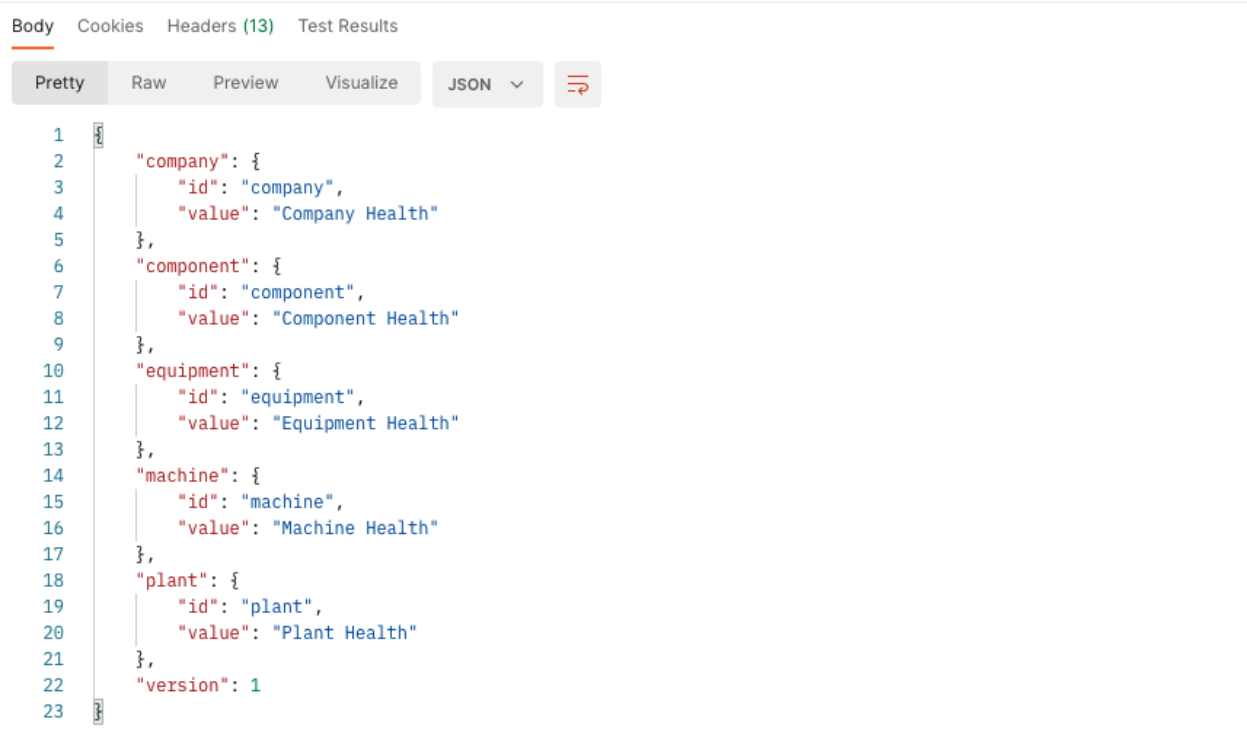


3.3. /entityId [GET]

It provides the Id to use for getting the data from `/analytics/health` endpoint. The returned object `id` field's value needs to be used as `entityTypeId` in `/analytics/health`. You shouldn't pass anything in the request body for this request.

Example:





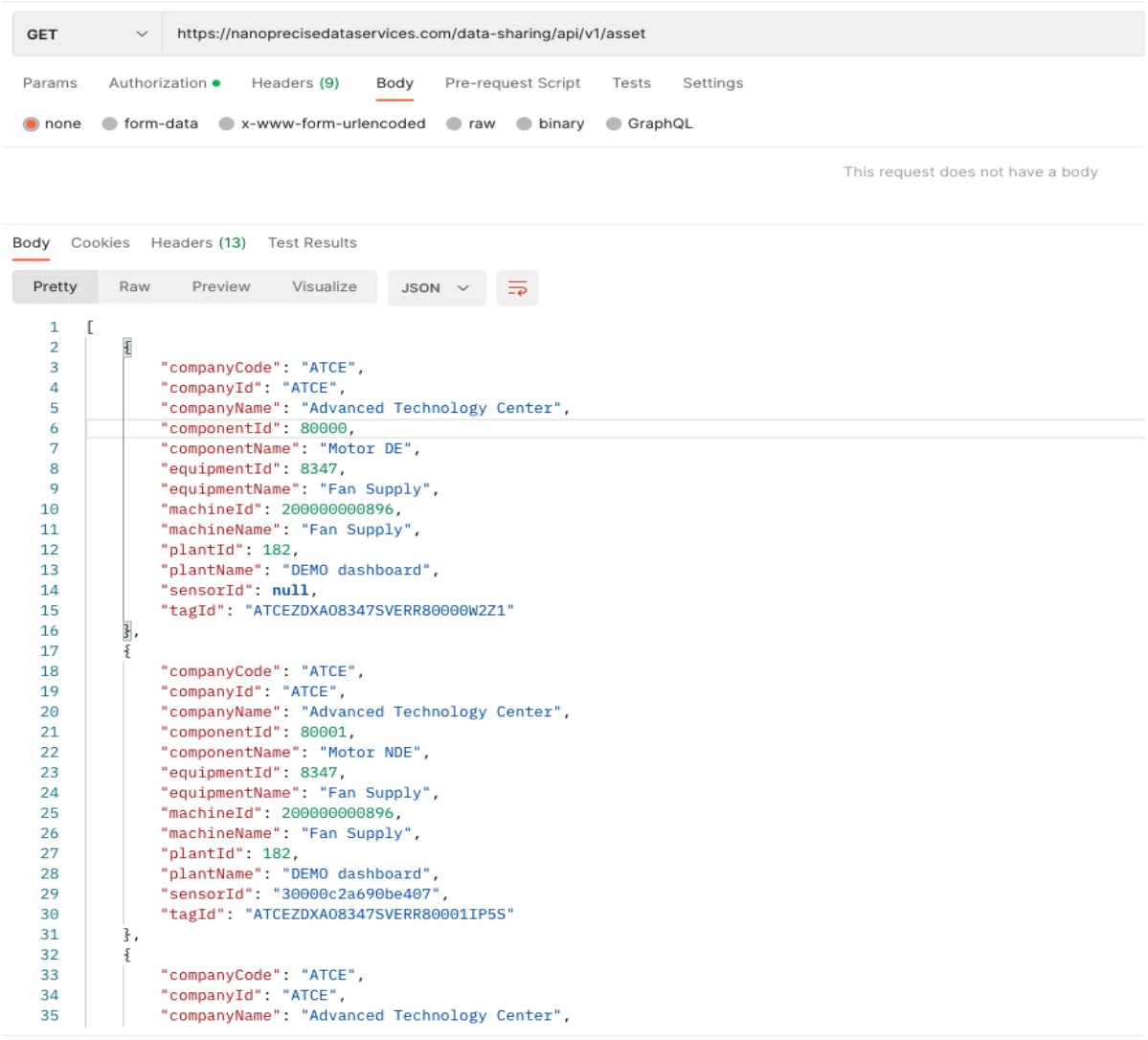
3.4. /asset [GET]

Returns an array of JSON objects. Each return object has name fields for company, plant, equipment and component along with tagId. tagId is a unique identifier for each component. You shouldn't pass anything in the request body for this request.

If the assets in Nanoprecise system are linked to customer's assets, then you also will get a key called `externalId` and the value will be the assetId of the asset in customer's system. Note that this id has to be setup in the Nanoprecise system through configuration portal or the DIY mobile app.

The results of this endpoint are required for all `/analytics/` endpoints where tagIds or equipment, component, plantIds are used in the request body.

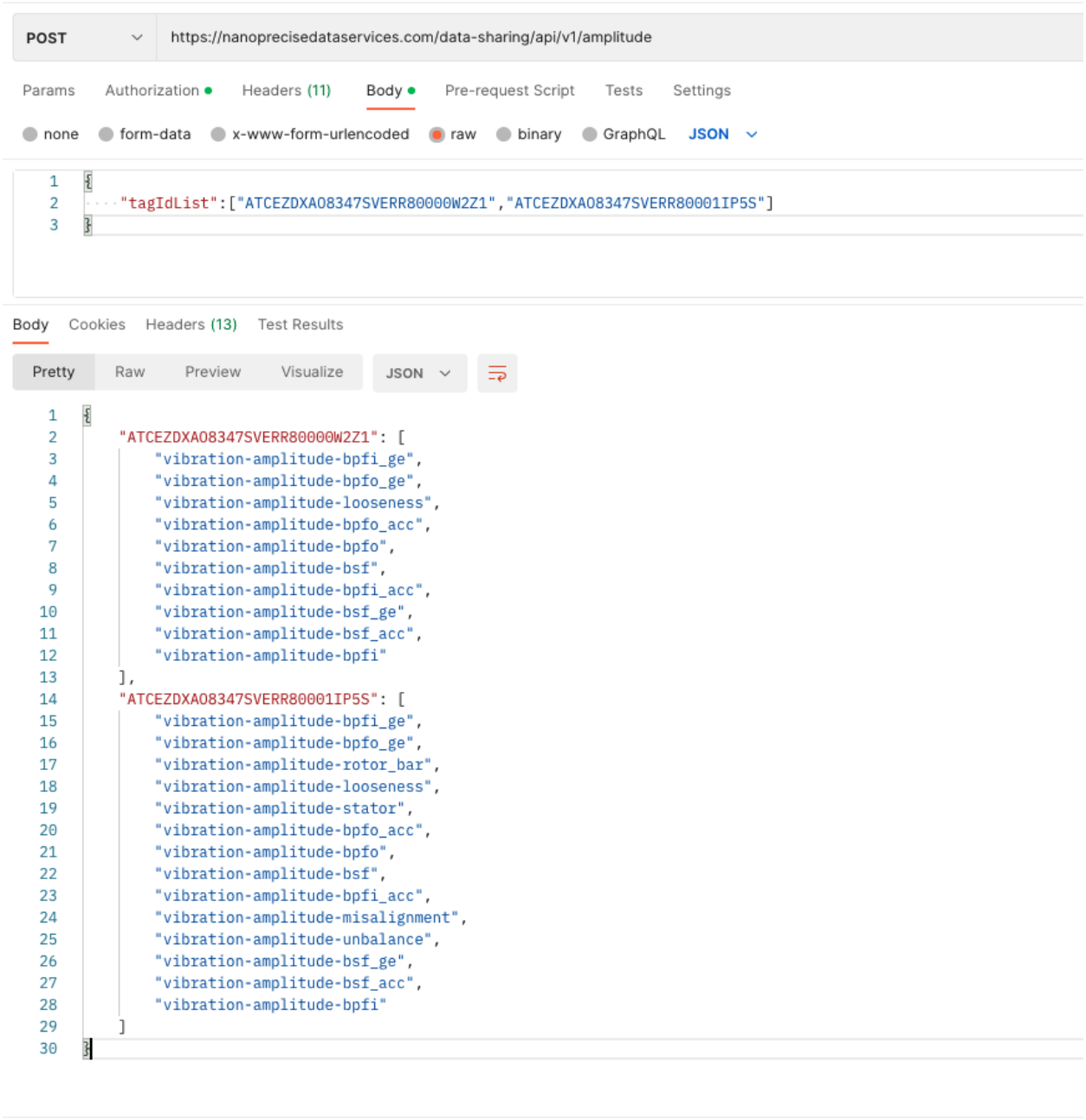
Example:



3.5. /amplitude [POST]

Returns the a JSON object with tagIds as properties and an array of active amplitude keys as values. This gives the types of amplitude data enabled for each tagId. The request body must have **tagIdList** field with a list of tag values as value as shown in the below screenshot.

Example:

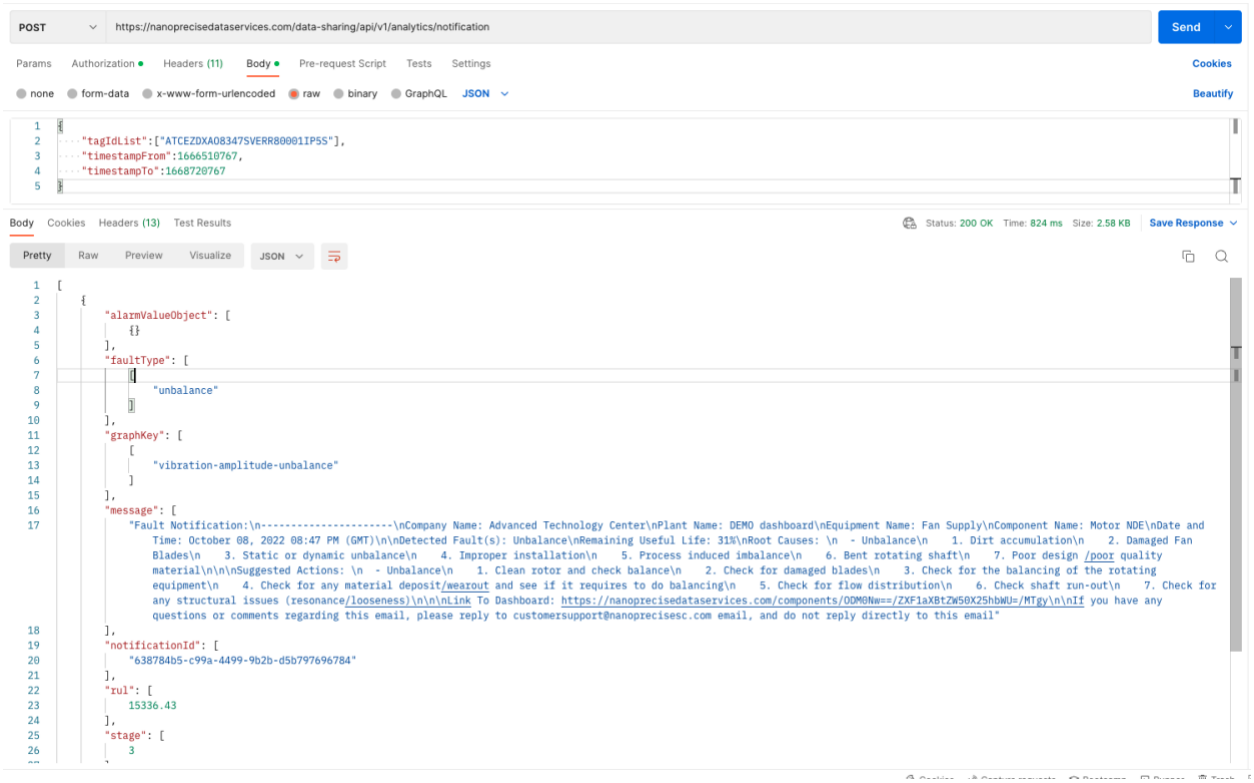


3.6. /analytics/notification [POST]

Returns an array of notifications for each tagId in the `tagIdList` passed in the request body.

Required fields in the request body are `tagIdList`, `timestampFrom`, `timestampTo` as shown in the below example.

Example:



3.7. /analytics/graph [POST]

Returns the data for a specific graph. For each graph a separate request needs to be sent. Get the graphIds from /graphId endpoint and tagIds from /asset endpoint.

Required fields in the request body are `tagIdList`, `graphId`, `timestampFrom`, `timestampTo` as shown in the below example.

Example:

POST https://nanoprecisedataservices.com/data-sharing/api/v1/analytics/graph

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON


```
{
    "tagIdList":["ACME1696a724b2d2421783095d3fbfc9da91", "ACME041c05efb4104064b29c08a9ebbbb556"],
    "graphId":"vibration-rms-acceleration",
    "timestampFrom":1666510767,
    "timestampTo":1668720767
}
```


Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "data": {
            "x": [
                0.008555498173744003,
                0.012922139391076734,
                0.008637158566876432,
                0.008623904349474866,
                0.008887945998913403,
                0.009709019278851636,
                0.008603193913493179,
                0.00864388157684881,
                0.008657740674768733,
                0.0086407679400159,
                0.008632840656947532
            ],
            "y": [
                0.008436793489759906,
                0.008681540418236983,
                0.00846291400446809,
                0.00845165174688458,
                0.008704928644060412,
                0.008869626883346476,
                0.00842114341883828,
                0.008428519080118905,
                0.008450331485398173,
                0.008414818604219169,
                0.008405908515989964
            ],
            "z": [
                0.00927696133803654,
```

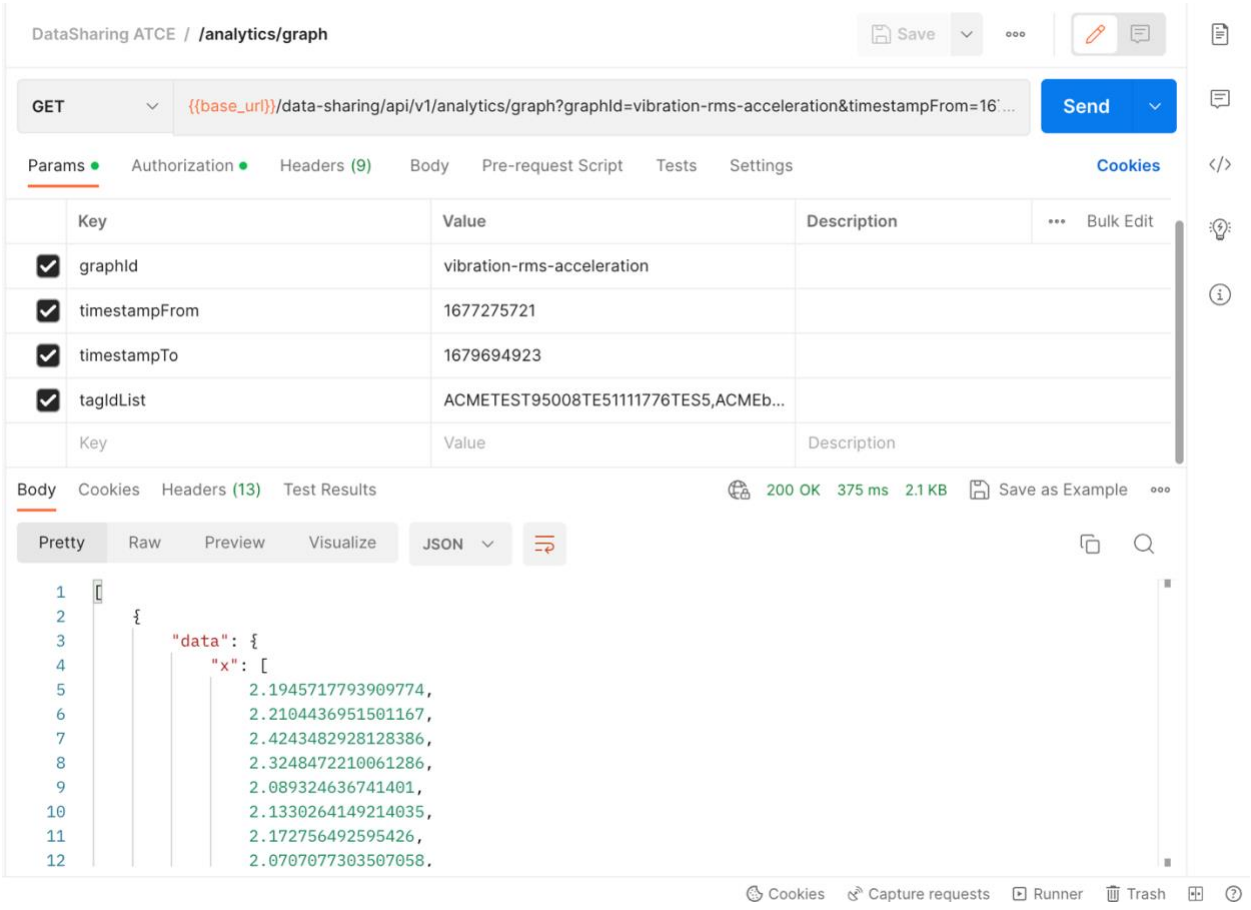
3.8. /analytics/graph [GET]

This returns the same results as the POST request for /analytics/graph endpoint. But the request type is different, and the request parameters are passed in the URL.

You must pass same the parameters in the URL here.

Note that, you cannot pass more than 10 tagIds in each request for the GET request type where is in POST request, you can pass up to 50 tagIds.

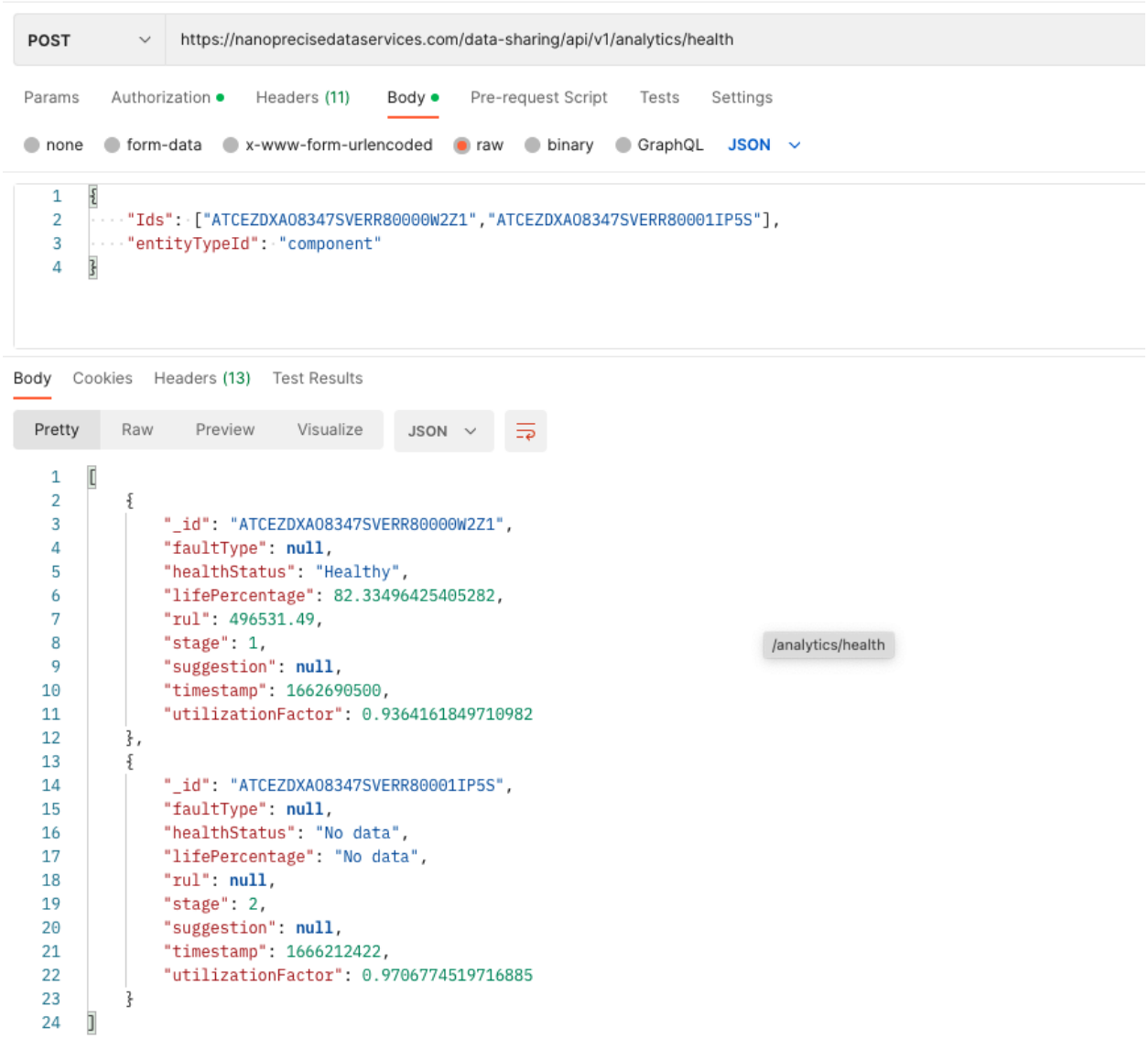
Example:



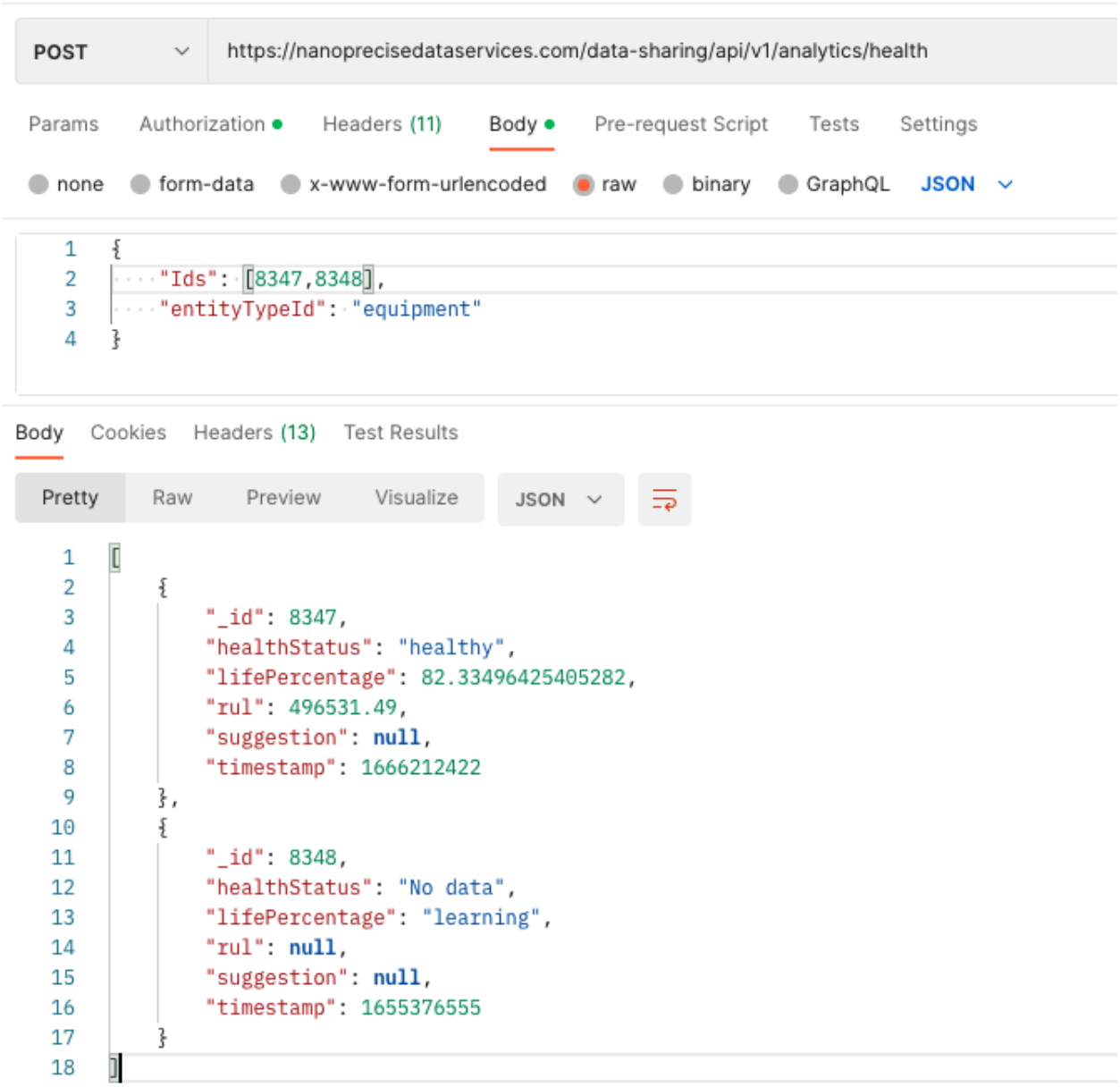
3.9. /analytics/health [POST]

To get the health status of a certain entity. Entities are different levels of a company, which includes Company, Plant, Machine (if enabled), Equipment, and Component. You can get thes.e Ids from /asset endpoint. The following examples are given for component and equipment. You can also query similarly at plant level and machine level(if enabled)

Example: querying health data of components by sending componentId via `Ids` field and the `entityTypeId` as **component**



Example: querying health data of equipment by sending equipment Ids via `Ids` field and the `entityTypeId` as **equipment**



4. Limitations

- One cannot request for more than 30 days of data at a time
- When time duration is required, starting time should not be older than 31 days ago
- “timestampFrom” in the requests wherever applicable can neither be older than 31 days nor be a future date
- Maximum number of API requests is 15000/day per each user
- When number of tags are specified, it should not be more than 50 tags in each request