API Resources

Version 0.1

13/11/2023

Contents

1 Disclaimer

Every attempt has been made to make this document complete, accurate and up-to-date. However, readers are cautioned that changes to local regulations or product improvements may cause SMA Germany to make changes to this document without advance notice. SMA Germany shall not be responsible for any damages, including indirect, incidental, or consequential damages, caused by reliance on the material presented, including, but not limited to, omissions, typographical errors, arithmetical errors or listing errors in the content material.

It is therefore recommended that you always check for the latest version prior to following the instructions in this document.

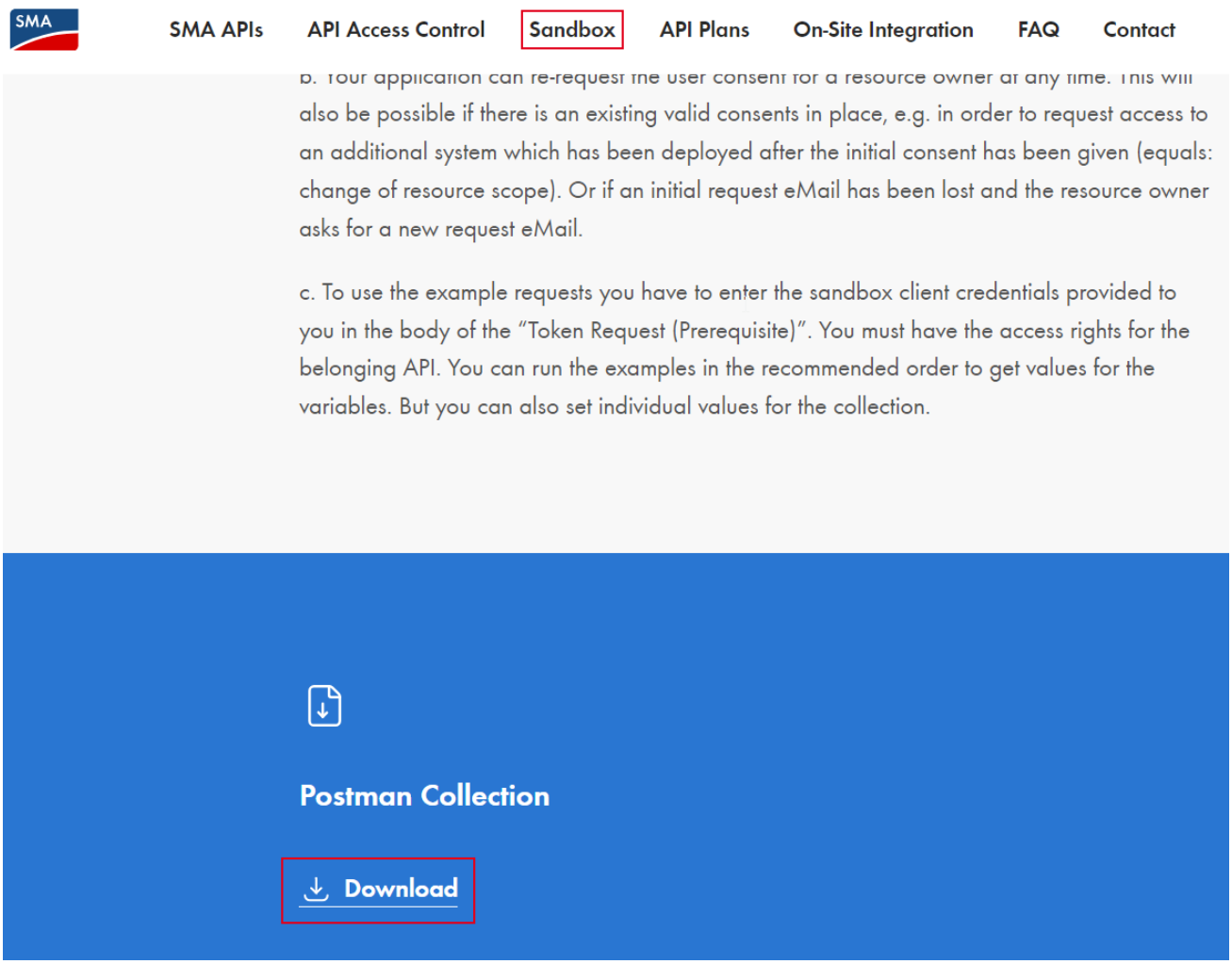
2 Scope

This document is intended to cover the following:

a. API Tutorial using POSTMAN (Through the custom workflow).
b. Common error responses that occur when making requests to the API.
c. An overview of the information you can pull from the API.

3 Postman Collection

Postman is a graphical API testing platform. A postman collection can be downloaded at the bottom of this page. However, it purely uses the sandbox URLs.



3.1 Setting up the Collection

The SMA POSTMAN Collection setup will be discussed in the following section as well as the usage and adaptation of queries to the productive environment.

3.1.1 How to import a collection

Step 1: Open Postman and navigate to the "Collections" tab.

Step 2: Click on the "Import" button.

Step 3: Choose the source type, “File”.

Step 4: Select your JSON file.

Step 5: Now you will be able to see the imported file.

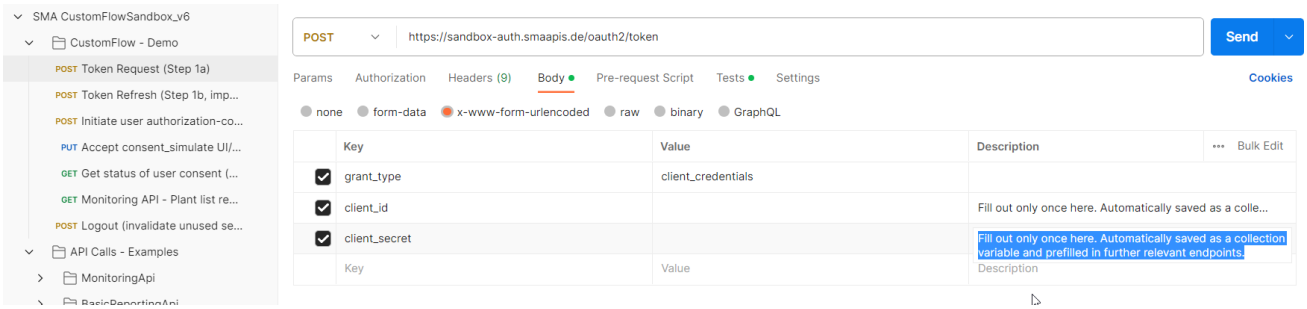


Figure A: Import Collection

3.1.2 How to adjust the Collection to Productive

The resource server on SMA production environment is named https://monitoring.smaapis.de.

3.2 Using the API with Postman

The POSTMAN collection will automatically use the most recent access and refresh tokens generated. After sending the “01 Plants List” request under the “MonitoringApi” folder which is inside “API Calls – Examples” folder, you should get the response:

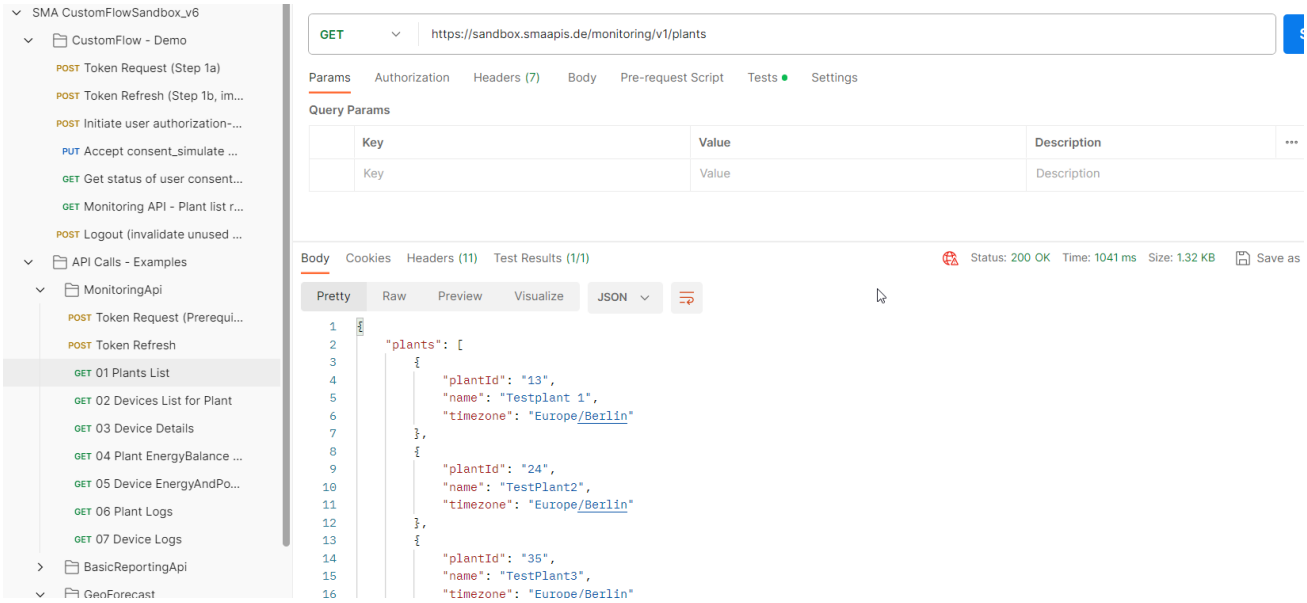


Figure B: Plant list response

You can then send the request “02 Device List for Plant” and so on.

The above request should give the response:


```
{
    "plant": {
        "plantId": "13",
        "name": "Testplant 1",
        "timezone": "Europe/Berlin"
    },
    "devices": [
        {
            "deviceId": "14",
            "name": "Satellit Sensor",
            "timezone": "Europe/Berlin",
            "type": "Sensor technology",
            "product": "Satellite Sensor",
            "productId": 200053,
            "vendor": "SMA Solar Technology AG",
            "isActive": true
        },
        {
            "deviceId": "15",
            "name": "MyTestLdm1",
            "timezone": "Europe/Berlin",
            "type": "Monitoring and control",
            "product": "EDMM-10",
            "productId": 9397,
            "serial": "234534635778",
            "vendor": "SMA Solar Technology AG",
            "isActive": true
```


Figure C: Plant list response

4 API Tutorial (Using POSTMAN)

4.1 Initial setup

To access plant data, you first need to get Client Credentials (Client ID and Client Secret).

For this, go to Contact (sma.de): Select the API's you are interested in. Then fill out and submit contact information.

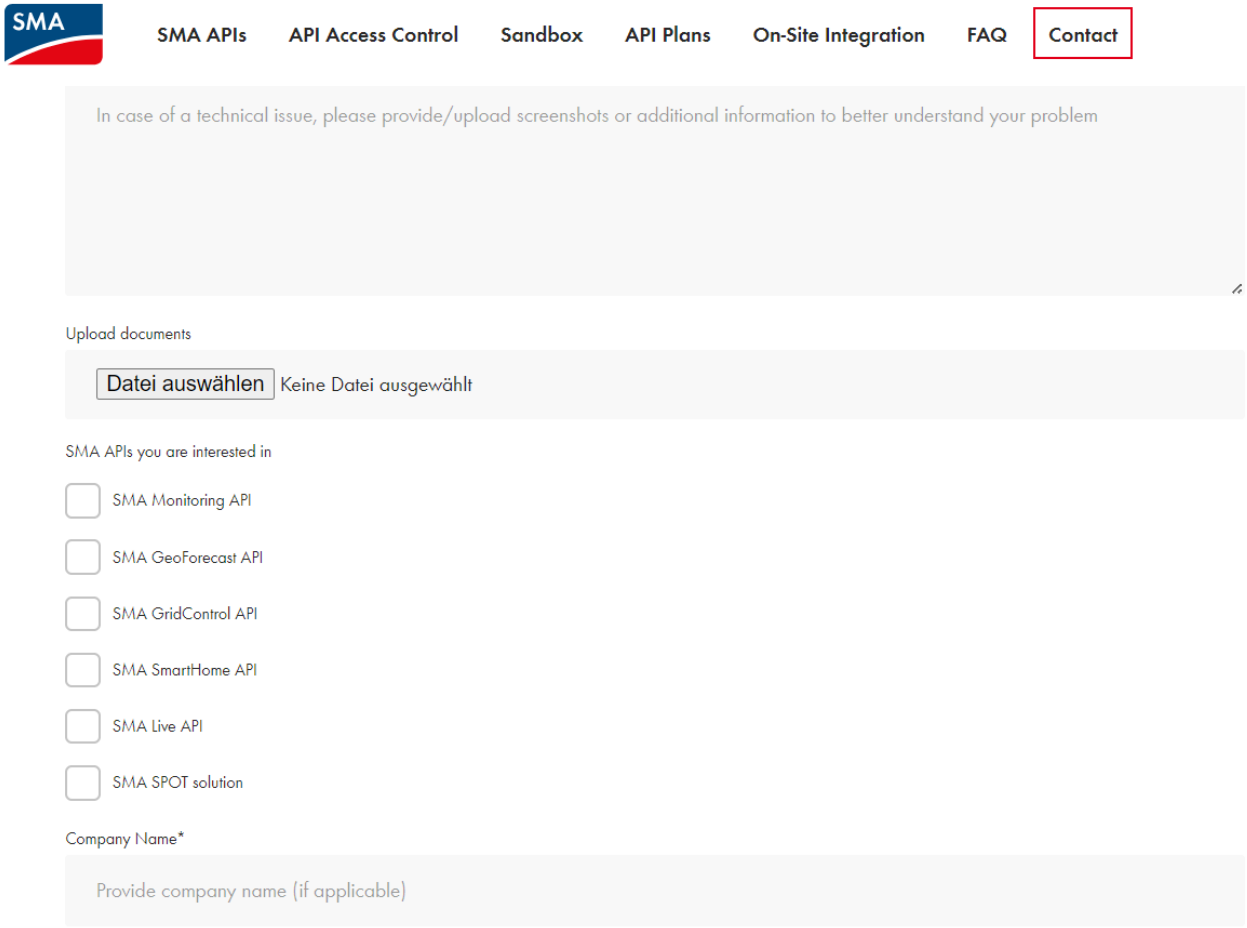


Figure 1: Contact page

Following the completion of your request and signation of contract/trials form that will be sent to you, the API Developer Support team will forward an email to you containing your Client Credentials. These credentials function as your login and password, granting access to the APIs. The client ID serves as a public

identifier for your application, while the client secret is a confidential piece of information known only to the application and the authorization server.

4.2 API Authorization: Custom flow

The technical tutorial on API Access Control | SMA Developer Portal | SMA Developer Portal is held simply and is designed for a direct communication with servers or services through API calls without the need for a graphical user interface (GUI) provided by a web browser.

The Custom Grant flow and how-to will be covered in the next section using POSTMAN for a better understanding.

4.2.1 Token Acquisition

The first step is to generate an access and refresh token. This is done by sending a post request to the authorization URL. When interacting with the sandbox environment use https://sandbox-auth.smaapis.de/oauth2/token, however for production use https://auth.smaapis.de/oauth2/token.

The following settings were used (Enter your own client id/secret):

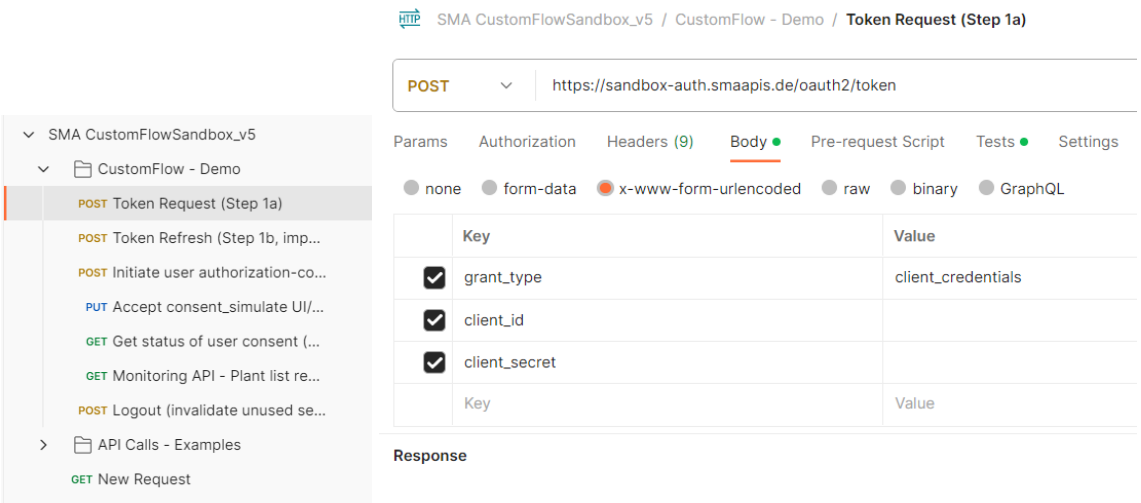


Figure 2: Authentication header settings

	Key	Value	Description	Bulk Edit
☑	grant_type	client_credentials		
☑	client_id	< Enter Client ID >	Fill out only once here. Automatically saved as a collection variable and prefilled in further relevant endpoints.	
☑	client_secret	< Enter Client Secret >	Fill out only once here. Automatically saved as a collection variable and prefilled in further relevant endpoints.	
	Key	Value	Description	

Figure 3: Authentication body settings

The Content-Type will always be set to *"application/x-www-form-urlencoded"*. The grant type is set to *"client_credentials"* when logging in. Then **press** the [send] button. The results should appear as follows:

SMA CustomFlowSandbox_v5 / CustomFlow - Demo / **Token Request (Step 1a)** Save

POST https://sandbox-auth.smaapis.de/oauth2/token Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (13) Test Results (1/1) Status: 200 OK Time: 259 ms Size: 2.72 KB Save as example

Pretty Raw Preview Visualize JSON


```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJoZXZFVzNoaGV6WlpjWkVkTTFvZUpnOW5lSFI4R3ZLMFVZZG0tbGdPbE1JIn0.
        eyJleHAiOjE3MDQyODI2MTIsImlhdCI6MTcwNDI4MjMxMiwianRpIjoiZTI4ODUxYTMtZjg4Mi00ZTZiLWIwZjYtYWMzMTVlZWM5OTMyIiwiaXNzIjoiaHR0cHM6Ly9zYW5kYm94LWF1dGguc21hYXBpcy5kZS9hdXRoL3JlYWxtcy9TTUEiLCJzdWIiOiJmZjk2Mjc2OC05MTk2LTQ2YTItYmU1NC04YzQ5YmRkNjQ0MzIiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJ0cmV2aV9hcGkiLCJzZXNzaW9uX3N0YXRlIjoiODc2ZjQ5MDEtN2QwMy00ZTE5LWI5ZTUtZDZhNGU1OGFjMjc0IiwiYWNyIjoiMSIsInNjb3BlIjoibW9uaXRvcmluZ0FwaTpyZWFkIGdyaWRDb250cm9sQXBpX0VuZXJneVRyYWRlcjp3cml0ZSBncmlkQ29udHJvbEFwaV9FbmVyZ3lUcmFkZXI6cmVhZCIsInNpZCI6Ijg3NmY0OTAxLTdkMDMtNGUxOS1iOWU1LWQ2YTRlNThhYzI3NCIsInVpZCI6IjEwNTcyIiwiY2xpZW50SWQiOiJ0cmV2aV9hcGkiLCJjbGllbnRIb3N0IjoiMTQ3LjE2MS4yNTUuMTAzIiwicHJlZmVycmVkX3VzZXJuYW1lIjoidHJldmlfYXBpX2FjZWFhMDEwQHNtYS5kZSIsImNsaWVudEFkZHJlc3MiOiIxNDcuMTYxLjI1NS4xMDMifQ.
        1R9im40Ng3ULkBv3rrm0taNZvPOsRYMgFlFJw3z-ScmiRwAzRfhtlQf9tn_g6oXMAN99_0rTbvXpBDJOZ_zMcz03W3Olzs9GdClUN0fKewT6cV5zixYWn-jZ048zvGdKbDi55
        xRfA5Z-mSzwOw5eojNLXhCBMjXH8BqIvlWUlNP6_jqhqsEutk-zZKf3hHYqusuabeSW_LTZSSPQWGfpEyzwSb_JBGuVGriPvgnVN6WsVawOLOayK6Wkm9OOnYeARff1RJtM0U
        G1bMK6Igy3euRKzxz_t-QD58YIEXoWnWFx3Gz1FtBZQBh35ySTK4VlPGrvNbLpJxaeSFufU1b-Zw",
    "expires_in": 300,
    "refresh_expires_in": 1800,
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJhNmJlZjg4NS0yNTY2LTQ2YjctYTJmMS02NGY0NjBmMzAzMmEifQ.
        eyJleHAiOjE3MDQyODQxMTIsImlhdCI6MTcwNDI4MjMxMiwianRpIjoiZGI5M2Q3OTYtNGZkMC00MjM3LTgyODktYmEyZWI0ZjU4ODM2IiwiaXNzIjoiaHR0cHM6Ly9zYW5kYm94LWF1dGguc21hYXBpcy5kZS9hdXRoL3JlYWxtcy9TTUEiLCJhdWQiOiJodHRwczovL3NhbmRib3gtYXV0aC5zbWFhcGlzLmRlL2F1dGgvcmVhbG1zL1NNQSIsInN1YiI6Im
        ZmOTYyNzY4LTkxOTYtNDZhMi1iZTU0LThjNDliZGQ2NDQzMiIsInR5cCI6IlJlZnJlc2giLCJhenAiOiJ0cmV2aV9hcGkiLCJzZXNzaW9uX3N0YXRlIjoiODc2ZjQ5MDEtN2Q
        wMy00ZTE5LWI5ZTUtZDZhNGU1OGFjMjc0Iiwic2NvcGUiOiJtb25pdG9yaW5nXBpOnJlYWQgZ3JpZENvbnRyb2xBcGlfRW5lcmd5VHJhZGVyOndyaXRlIGdyaWRDb250cm9s
        QXBpX0VuZXJneVRyYWRlcjpyZWFkIiwic2lkIjoiODc2ZjQ5MDEtN2QwMy00ZTE5LWI5ZTUtZDZhNGU1OGFjMjc0In0.
        2XoEoyZqLASL7DuZWlDxe2NVB9SDYmOoThscnVGAGIo",
    "token_type": "Bearer",
    "not-before-policy": 0,
    "session_state": "876f4901-7d03-4e19-b9e5-d6a4e58ac274",
    "scope": "monitoringApi:read gridControlApi_EnergyTrader:write gridControlApi_EnergyTrader:read"
}
```


Figure 4: Access and Request tokens

The access token will expire in 300 seconds and the refresh token will expire in 1800 seconds as shown by the output.

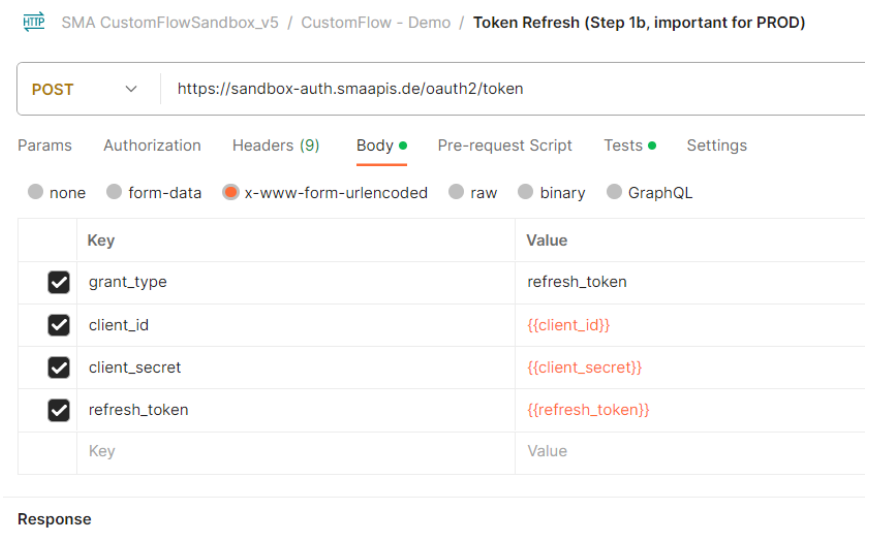


Figure 5: Refresh token retrieval

Change grant type to *"refresh_token"* and add the refresh token to body. Sending this request will make you use of the grant_type: refresh_token. This is to maintain the session and request new AccessToken & RefreshToken pair.

4.2.2 Requesting plant access.

Send the request to the plant owner. To do this, use the URL https://sandbox.smaapis.de/oauth2/v2/bc-authorize for testing (Sandbox) and https://async-auth.smaapis.de/oauth2/v2/bc-authorize for production. Use the settings:

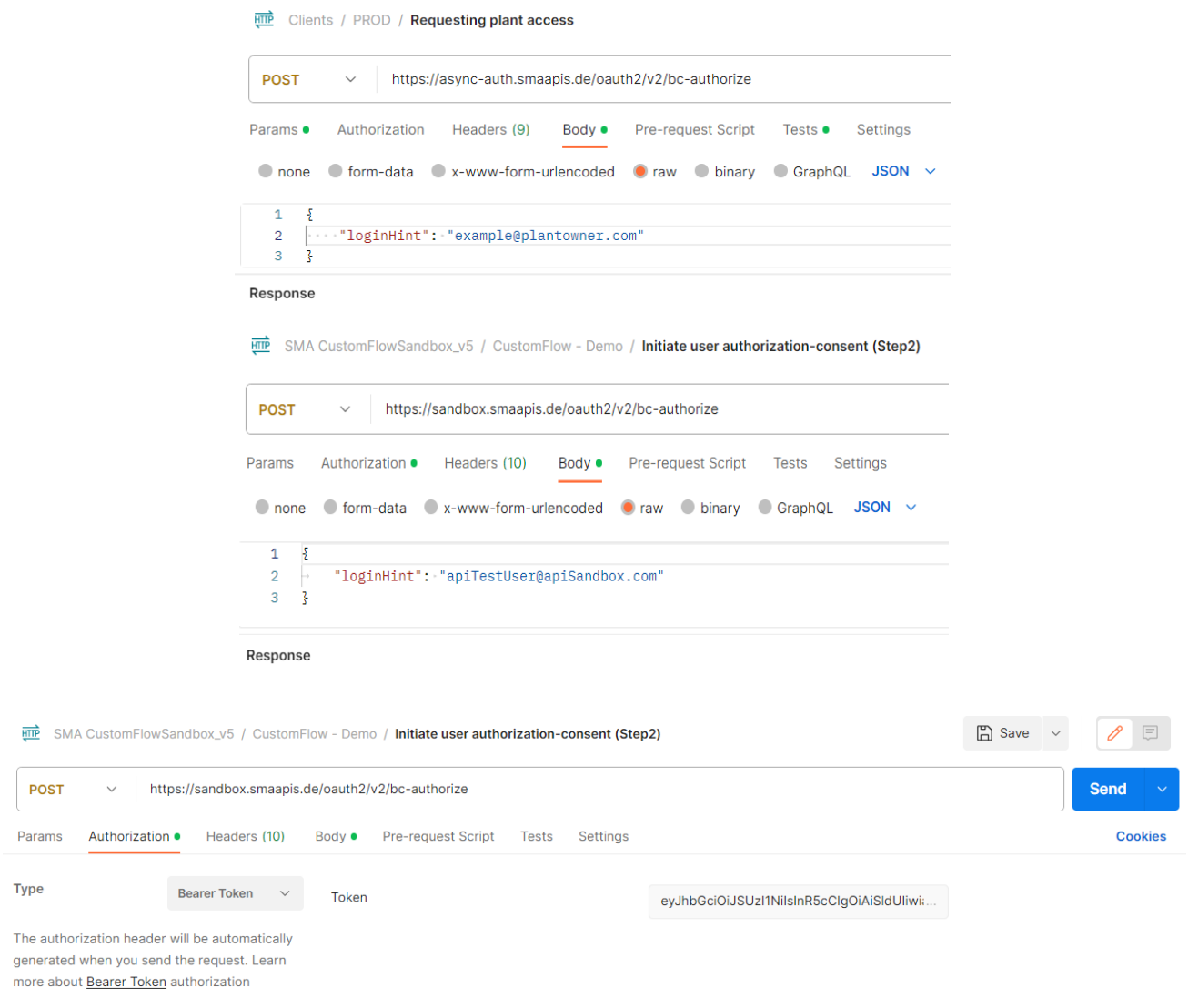


Figure 6: Plant access request

Replace the random string of characters after *"Authorization: bearer "* with your access token.

When moving into production, replace the email address of loginHint to the email address of the plant owner whose plant data you want to access/monitor.

The result should look like the following (expiring 7 days after reqest):

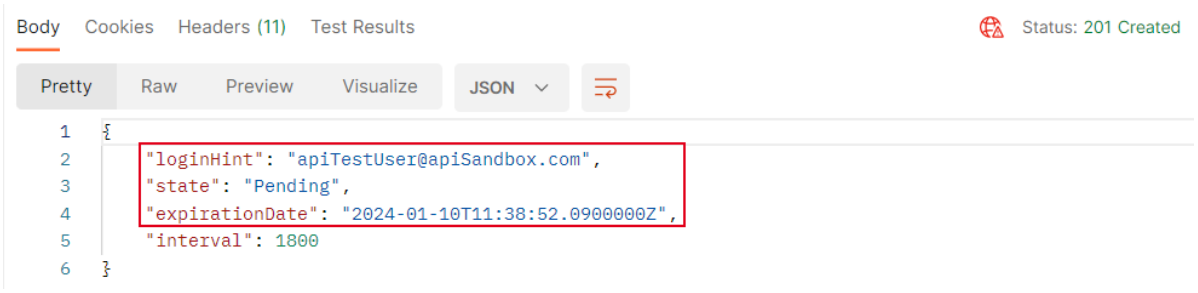


Figure 7: Plant access "pending" response

The "loginHint" must be the same email address of the plant owner to obtain plant data access permission. Now an email will be sent to the plant owner while you will be waiting for the plant owner to accept your request to access.

4.2.3 Accepting the permission request (For the Sandbox)

This request simulates *"accepting"* the permission request for plant access. The URL is: https://sandbox.smaapis.de/oauth2/v2/bc-authorize/apiTestUser@apiSandbox.com/status.

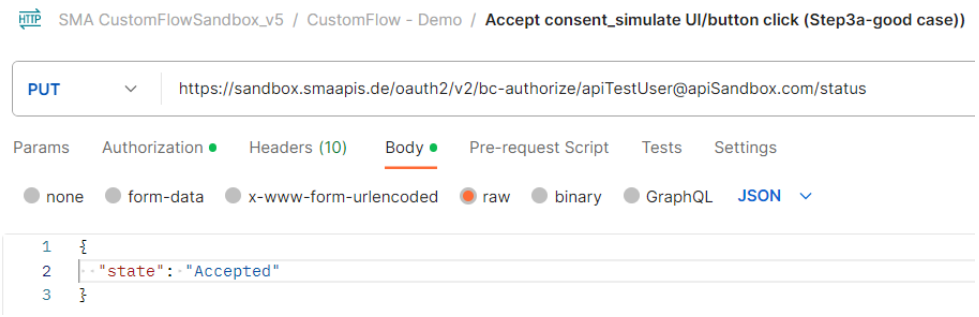


Figure 8.a.: Accepted plant access.

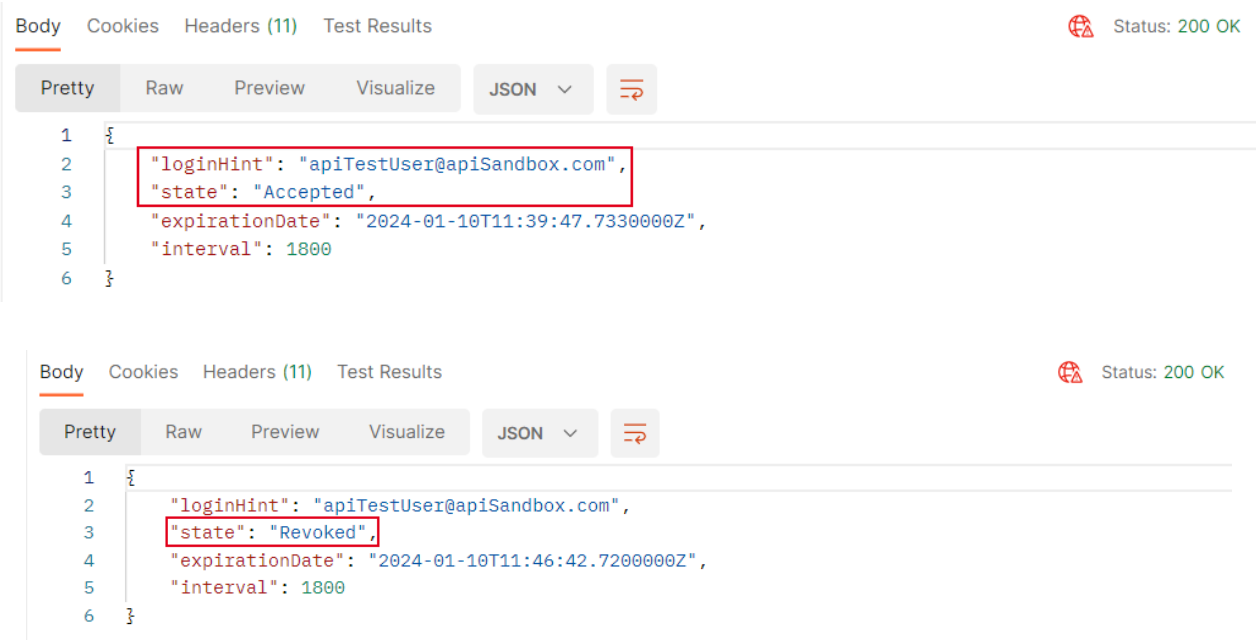


Figure 8.b.: Accepted plant access.

You can also reject or revoke access by changing the state to *"reject"* or *"revoked"*. The revoked state will remove permission from requests that were previously confirmed.

4.2.4 Checking plant access rights

Sandbox URL: https://sandbox.smaapis.de/oauth2/v2/bc-authorize/apiTestUser@apiSandbox.com,

Production URL: https://async-auth.smaapis.de/oauth2/v2/bc-authorize/{loginHint}

Figure 9: request status settings and response

Send the GET request once the plant owner has confirmed the consent email. Or resend the request every 1800 seconds until the state is no longer pending (accepted/rejected/revoked). You may need to refresh the token (or login again).

Note: In the case a request hasn't been sent before, you will get a response similar to:

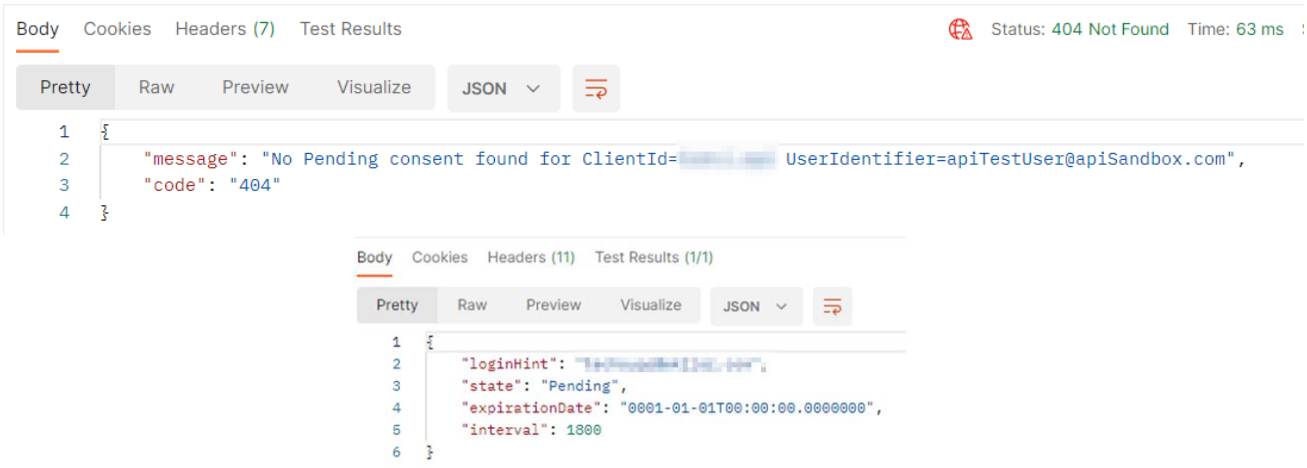


Figure 10: expirationDate = '0001-01-01T00:00:00.0000000'

You can use the expiration date to test whether an expiration date has already been noted.

The first response in figure 10 is telling that a bc-autorizing step is needed before checking the status. The second figure displays a response that can occasionally occur when sending a permission request or after investigating, you may find out, that the user (eMail address you used) is not a plant owner. Hence, no permission request will be sent out.

Normally, when the loginHint does in fact not involve to a plant owner, you will receive the following output:

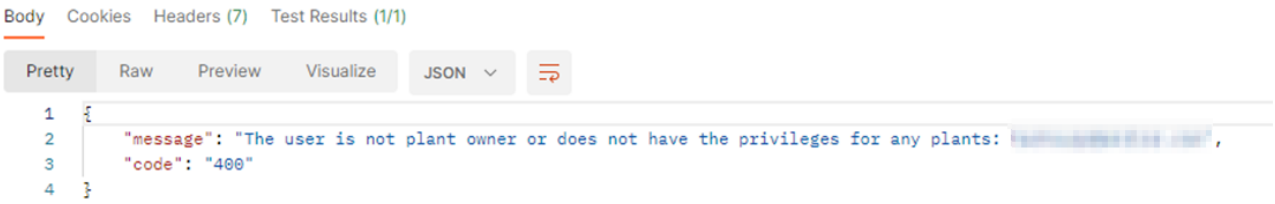


Figure 11: No privileges as plant owner

4.2.5 Logging Out

Logging out sessions once they are done being used is necessary as a maximum amount of sessions per client may be added in the future.

Sandbox URL: https://sandbox-auth.smaapis.de/oauth2/logout

Production URL: https://auth.smaapis.de/oauth2/logout

Settings:

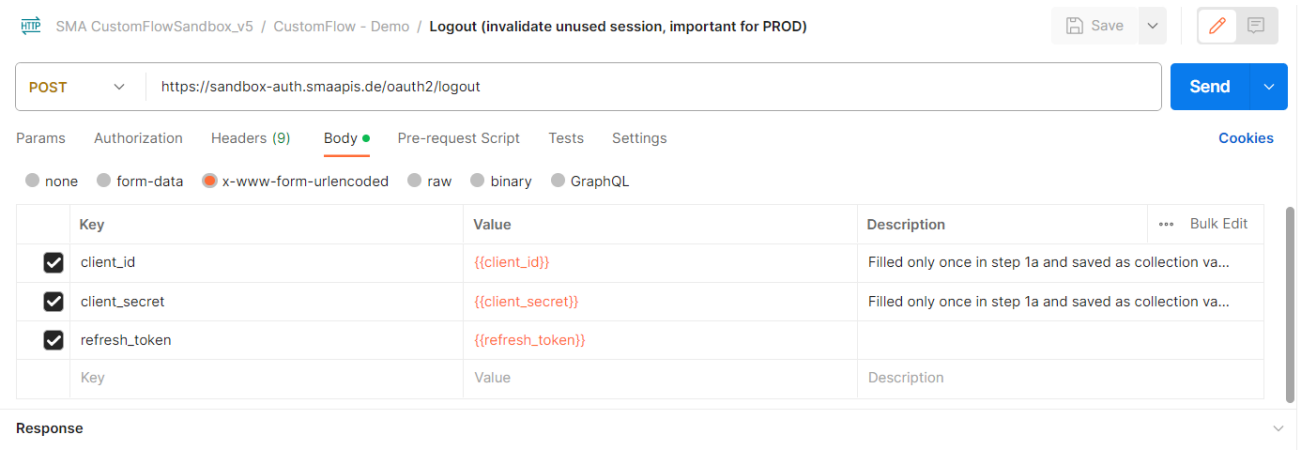


Figure 12: Logout settings

It should return an empty response with the status code 204.

Now you can use the API to access the owner's plants/plant data.

The swagger docs can be accessed here.

Use either URL depending on what your testing:

- Sandbox: https://sandbox.smaapis.de/monitoring/v1/plants
- Production: https://monitoring.smaapis.de/v1/plants

Use the following settings (with your access token after *"Bearer"*):

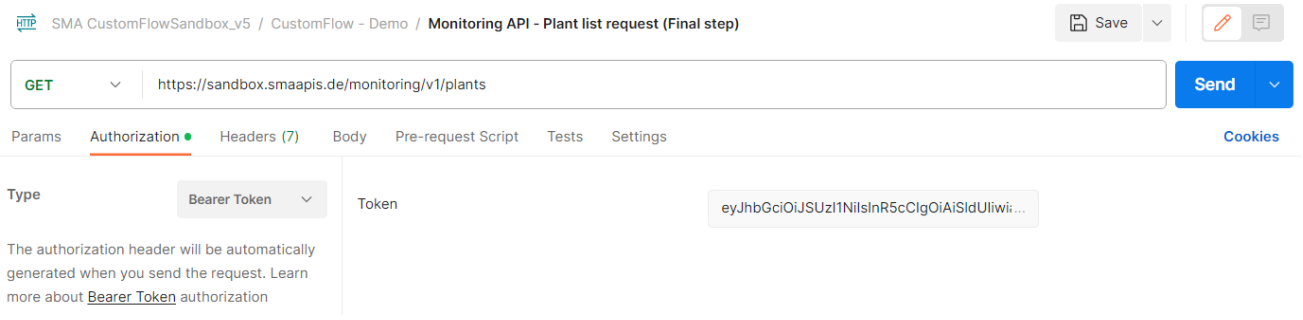


Figure 6: Get Plants API settings

You should get the response:

```
{
    "plants": [
        {
            "plantId": "13",
            "name": "Testplant 1",
            "timezone": "Europe/Berlin"
        },
        {
            "plantId": "24",
            "name": "TestPlant2",
            "timezone": "Europe/Berlin"
        },
        {
            "plantId": "35",
            "name": "TestPlant3",
            "timezone": "Europe/Berlin"
        },
        {
            "plantId": "46",
            "name": "TestPlant4",
            "timezone": "Europe/Berlin"
        },
```


Figure 7: Get Plants response

5 API Common errors

5.1 Error 415 – Unsupported Media Type (Cannot consume Content-Type)

The content type is not valid for the given request.

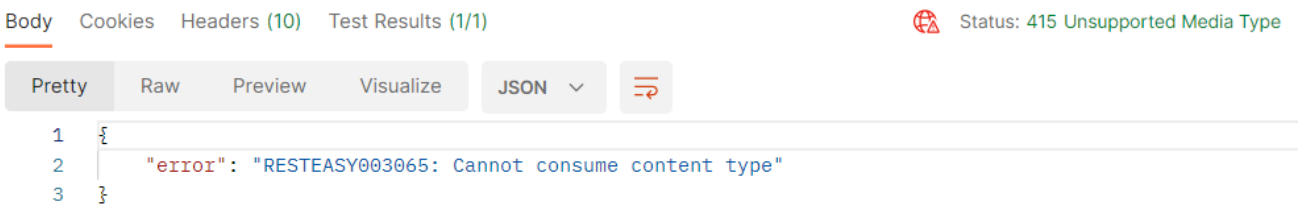


Figure 15: Unsupported Media Type

Correct content types for the following requests:

a. POST /oauth2/token ➔ *"Content-Type": "application/x-www-form-urlencoded"*
b. POST /oauth2/v2/bc-authorize ➔ *"Content-Type": "application/json"*
c. POST /oauth2/logout ➔ *"Content-Type": "application/x-www-form-urlencoded"*

5.2 Error 400 – Bad Request (Missing form parameter: grant_type)

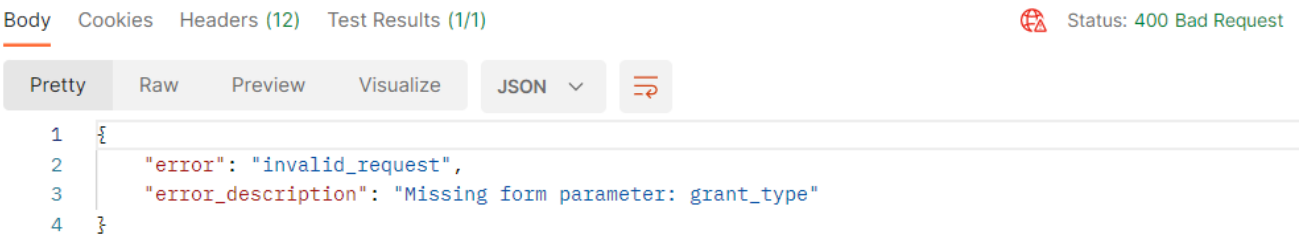


Figure 16: Grant type missing

This error can mean a couple of things. First check if *"grant_type"* was sent in the body. Another possible cause is that you did not send the *"Content-Type"* header or sent the incorrect type. For example,

"application/json" was sent instead of *"application/x-www-form-urlencoded"*. For more information about content type to use check section 4.1 for authorization API endpoints or here if you are accessing or setting plant data.

5.3 Error 400 – Bad Request (Invalid client credentials)

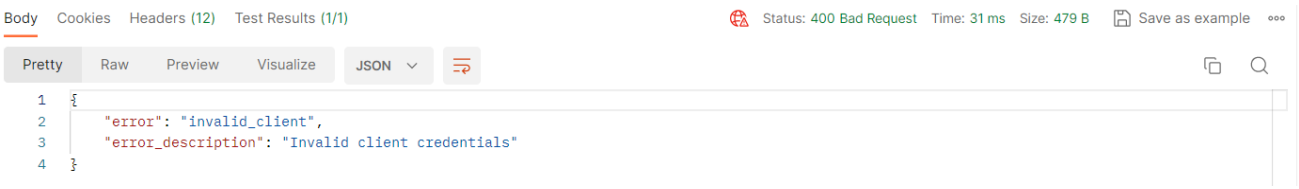


Figure 17: Invalid client credentials

This error can be caused by either an invalid id or secret, or if the client_id was not sent at all. However, this error occurs mostly when you attempted a request to an authorization server e.g. Custom flow: https://sandbox-auth.smaapis.de/oauth2/token when you actually received client credentials for the Code flow: https://sandbox-auth.smaapis.de/oauth2/auth.

5.4 Error 401 – Unauthorized (Token is not active)

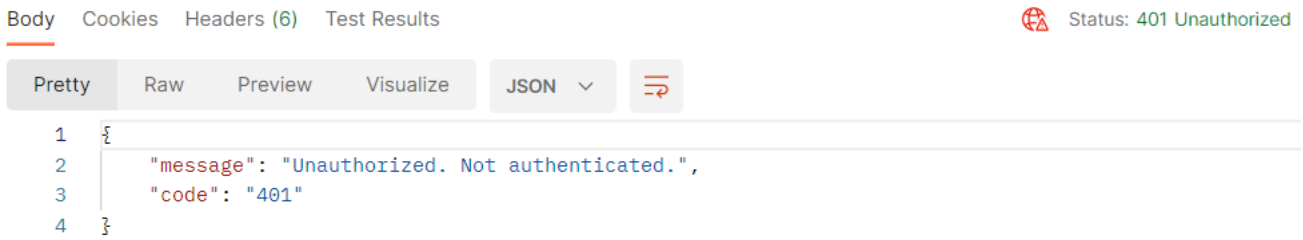


Figure 18: Session expired. Unauthorized.

This occurs when the token used is more than 1800 seconds old (i.e. expired).

6 API overview

6.1 Monitoring API

The SMA monitoring API can get information on plant groups, specific plants and specific devices within those plants. This includes:

a. The plant group's:
    i. Logs
    ii. Individual plants
b. The plant's:
    i. Id
    ii. Name
    iii. Location
    iv. Status
    v. Nominal power
    vi. Feed in tariff
    vii. CO2 savings
    viii. Orientation
    ix. Sub-plants
    x. Devices
    xi. Stakeholders
    xii. Configuration
    xiii. Power/Energy consumption
    xiv. Battery statistics (voltage, current, temperature, etc.)
    xv. Logs
c. The specific device's:
    i. Static information (Id, name, time zone)
    ii. Characteristics
    iii. Status
    iv. Capabilities (i.e. Generator power and battery capacity)
    v. Movement of power (Energy consumption, production and battery charging + dis-charging)
    vi. Battery statistics (voltage, current, temperature, etc.)
    vii. Voltage, current and power phase

6.2 Grid Control API

You can schedule commands to be run at specific times (for a specific duration) that can:

- Set the maximum power generation
- Disconnect the system from mains
- Set the power factor

You can also manage plant settings such as Active Power Ramp, Active Power Fixed Limit, Active Power Default Dynamic Target, Active Power Volt Watt Curve, Active Power Freq Watt, Reactive Power Mode Unspecific, Reactive Power Mode Injecting And Absorbing, Reactive Power Volt Var Curve, Energize, Voltage Fault Ride Through Must Trip, Frequency Fault Ride Through Must Trip Settings.

6.3 GeoForecast API

This API can do weather and power forecasts.

This API can do weather forecasts by sending the location of the plant which then returns the temperature and weather conditions (i.e. Cloudless, Heavy snow fall).

By sending both the location and generator specifications, the average AC power throughout the given day can be found.

6.4 Live API

This API provides live data from plants and their devices.

For devices, it can provide the constitution of what is supplying the power (Between the solar panels, battery and the grid). As well as the battery charge, device status and the total amount of power consumed by the device.

For the solar plants you can get their configuration, measurements, status and any errors. The measurements include the power generated by the PV inverters, total consumption and grid feed in.

6.5 Smart Home API

The Smart Home API returns detailed information about smart home device settings (Such as for EV charging or for Power sockets). This is currently only available for systems with home managers.