

/ Perfect Welding / Solar Energy / Perfect Charging



# **SOLAR.WEB QUERY API SPECIFICATION**

© Fronius International GmbH

Version 47.0 2021-04-21

BU SE

Fronius reserves all rights, in particular rights of reproduction, distribution and translation.

No part of this work may be reproduced in any way without the written consent of Fronius. It must not be saved, edited, reproduced or distributed using any electrical or electronic system.

You are hereby reminded that the information published in this document, despite exercising the greatest of care in its preparation, is subject to change and that neither the author nor Fronius can accept any legal liability.

Gender-specific wording refers equally to female and male form.

.

## TABLE OF CONTENTS

<b>1</b>	<b>Introduction .....</b>	<b>7</b>
1.1	About Fronius Solar.web Query API.....	7
1.2	Usage of the Fronius Solar.web Query API.....	7
1.2.1	Target audience.....	7
1.2.2	How to start with Fronius Solar.web Query API.....	7
1.2.3	Data plans and pricing.....	8
<b>2</b>	<b>General Information .....</b>	<b>10</b>
2.1	Key management .....	10
2.2	User impersonation .....	11
2.2.1	JWT token attributes .....	11
2.2.2	Creating a JWT .....	12
2.2.3	Using a JWT in a SWQAPI call .....	12
2.2.4	Examples:.....	13
2.3	Identification of PV systems .....	17
2.4	Pagination .....	17
2.5	Date and time formats .....	18
2.5.1	Use time in the calling URL .....	19
2.5.2	Time in response objects.....	19
<b>3</b>	<b>Supporting UIs.....</b>	<b>21</b>
3.1	API key management in Solar.web .....	21
3.2	Working with API keys in Swagger UI .....	22
<b>4</b>	<b>API Reference .....</b>	<b>24</b>
4.1	User impersonation calls .....	24
4.1.1	Impersonate: Receive a JWT using the Fronius login.....	24
4.1.2	Impersonate: Receive a JWT using userId and password .....	25
4.1.3	Impersonate: Refresh a JWT .....	27
4.2	Generic information calls.....	30
4.2.1	Info: Get release information .....	30
4.2.2	Info: Get user information.....	30
4.3	Metadata calls .....	33
4.3.1	Metadata: Get PV system information.....	33

4.3.2 Metadata: Count PV systems.....	37
4.3.3 Metadata: Enumerate PV system IDs .....	39
4.3.4 Metadata: Get device information .....	40
4.3.5 Metadata: Count devices.....	45
4.3.6 Metadata: Enumerate device IDs .....	47
4.4 Aggregation calls.....	49
4.4.1 Aggdata (outdated): Aggregated energy production for a PV system .....	49
4.4.2 Aggrdata: Aggregated energy production for a PV system .....	57
4.4.3 Aggrdata: Aggregated energy production for a device .....	65
4.5 Historical data calls .....	70
4.5.1 Histdata: Historical data for a PV system .....	70
4.5.2 Histdata: Historical data for a device.....	74
4.6 Realtime data calls .....	81
4.6.1 Flowdata: Realtime power flow data of a PV system .....	81
4.6.2 Flowdata: Realtime power flow data of a device .....	85
4.7 Weather data calls.....	87
4.7.1 Limitations .....	88
4.7.2 Weather: Current weather for a PV system.....	88
4.7.3 Weather: Weather forecast for a PV system .....	91
4.7.4 Weather: Energy forecast for a PV system .....	93
4.8 System messages calls.....	97
4.8.1 Messages: Get PV system messages.....	97
4.8.2 Messages: Count PV system messages.....	100
4.8.3 Messages: Get device system messages .....	101
4.8.4 Messages: Count device system messages .....	103
<b>5 Appendix.....</b>	<b>105</b>
5.1 Response and error codes .....	105
5.1.1 HTML error codes.....	105
5.1.2 Detailed error codes .....	105
5.2 Channels .....	109
5.2.1 Channel list.....	109
5.2.2 Channel types .....	119
5.3 Meteorological weather symbols .....	120
5.3.1 List of weather symbols .....	120

5.4	Languages.....	122
5.5	Best practices and how-tos .....	123
5.5.1	Use filters for channels.....	123
5.5.2	Determine power values from energy values from historical data .....	123
5.5.3	Determine PV Energy and Load Energy .....	123
5.5.4	Determine if new systems were added to account.....	123
5.5.5	Grant permissions in Solar.web.....	124
5.5.6	Migration from Solar.web Third Party API to Solar.web Query API .....	125

Version	Modified	Description
47.0	Apr 21, 2021	/ Updated camelCase notation in examples for system messages.
46.0	Apr 02, 2021	/ Changed supported channel information for aggregated and historical data requests. / Updated Solar.web screenshots, error code lists and channel list.
45.0	Feb 02, 2021	/ Version history added. / Added additional information about Solar.web Premium to chapter "User impersonation". / Updated chapter "Determine power values from energy values" in Appendix.

# 1 INTRODUCTION

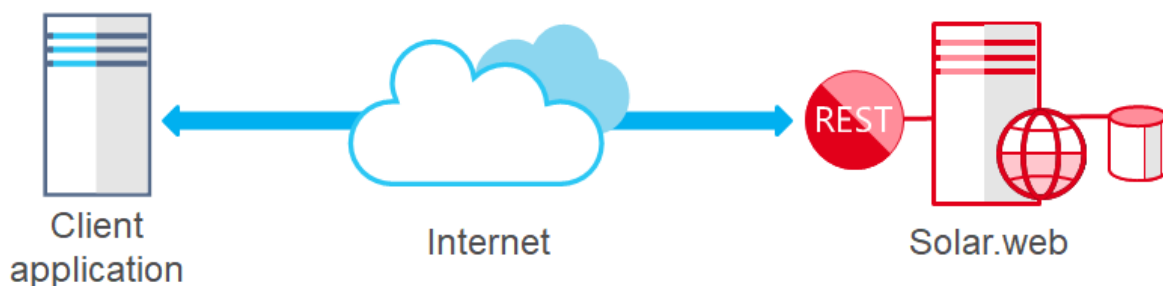
## 1.1 About Fronius Solar.web Query API

### Fronius Solar.web

The Fronius Solar.web online portal allows users to easily and conveniently monitor, analyze and compare their photovoltaic systems by visualizing energy flows and displaying PV (photovoltaic) yields. Intelligent analysis functions ensure that yield losses are reliably avoided.

### Fronius Solar.web Query API

The Solar.web Query Application Programming Interface (SWQAPI) is an application-to-application interface for accessing the raw data of PV systems stored on Solar.web servers. Two applications (client requesting data and Solar.web delivering data) are interacting via API to each other without any user intervention, so that the client application can e.g. display information to end users or do detailed analysis of the data.



## 1.2 Usage of the Fronius Solar.web Query API

### 1.2.1 Target audience

SWQAPI is intended to be used by customers who want to have their own visualization of their PV systems or integrate the data into their existing applications.

For example, a utility which, next to its core business (electricity supply), offers PV systems to its customers, most likely already provides an online portal or an app where customers can check their electricity consumption. The utility might want to extend the functionality of the portal and also show the data of the customers' PV systems. The utility has just to fetch the PV data from the Solar.web servers via the API and then visualize it for its customers in its portal.

Another example would be an O&M (operation and monitoring) company which offers extensive monitoring solutions to their customers. Often an O&M company supports PV systems from different vendors and does not want to use multiple monitoring portals. By fetching the PV data from Solar.web via API the O&M company can easily integrate the data in its monitoring solution.

### 1.2.2 How to start with Fronius Solar.web Query API

#### Demo

A Fronius Sales Representative can give interested customers access to the demo, which contains the same PV systems that are available in the Solar.web demo portal. Using the demo key credentials, interested customers can use the Swagger UI to test the SWQAPI. In that case an interested customer does not need to have his/her own Solar.web account or any PV system linked to a Solar.web account.

## Unlimited access

In order to access and use the API customers need to be registered in Solar.web (<https://www.solarweb.com>) and sign a contract with Fronius to get access to SWQAPI.

After completing the registration, please make yourself familiar with the key management. You need to create at least one key in the Solar.web Settings, which you can then use programmatically. We recommend getting started by using the Swagger UI (<https://api.solarweb.com/swqapi/index.html>) and test a few calls first, e.g. by enumerating PV systems and showing their metadata (unique ID, name of system, address, etc). Once you are more familiar with SWQAPI, have a look at the energy flows which show the current status of PV systems. Fronius also recommends comparing the API results with information and diagrams you see in the Solar.web UI.

Note: If you don't see any PV systems in your account at all, you likely need to add PV systems to your account by adding guest or supervisor permission for this user to an already existing PV system. Guest permissions are sufficient to see most of the data. However, if you want to see service messages for a certain PV system you need supervisor permission to view them.

From there, continue with exploration. If you want to show power curves for the last few days, have a look at the historical data which gives you 5 min granularity to draw production and consumption diagrams. If you want to go further into the past, use the aggregation method which has daily, monthly or annual energy data available for you.

### 1.2.3 Data plans and pricing

Demo access is free but there are costs for unlimited access. The costs depend on the number of queried data points per month. Below is a list showing how the end points and data points are billed.

Response to the following calls are not billed:

- / Release information
- / PV system information
  - / Count of systems
  - / List of system IDs
  - / Count of devices
  - / List of device IDs

Each response to the following calls counts as just one data point:

- / PV system information
  - / Detailed information about systems
  - / Detailed information about devices
- / Power flow data
- / Current weather data

The responses for the following calls count as multiple data points:

- / Aggregation data: per data point/channel and timestamp; CO2 savings (4 channels) count as one (per timestamp); Profits (3 channels) count as one (per timestamp)
- / Historical data: per data point/channel and timestamp (e.g. one hour of EnergyExported with 5 minute log interval counts as 12 data points)
- / Service messages: each service message counts as data point
- / Energy forecast: each 15min/1hr EnergyExported forecast counts as one data point
- / Weather forecast: one data point per day



Data plan	Data points / month
Free (not billed)	0 - 500,000
Small	500,000 - 2,500,000
Medium	2,500,001 - 10,000,000
Large	10,000,001 - 30,000,000
Advanced	30,000,001 - 60,000,000
Pro	> 60,000,000

Please contact your local sales representative for further details about pricing.

## 2 GENERAL INFORMATION

### 2.1 Key management

In SWQAPI users have to provide valid API keys in the header of an API request. API keys are generated in Solar.web by authorized API users. Each authorized API user in Solar.web can have one or more API keys. Of course, those API keys are limited to the user's permissions in Solar.web.

API keys have the following attributes:

<b>Access key ID</b>	A unique ID for the API key, e.g. "FKIAFEF58CFEFA94486F9C804CF6077A01AB". Access keys are 36 characters long and start with the "FKIA" prefix.
<b>Access key value</b>	<p>A secret value (GUID), e.g. "47c076bc-23e5-4949-37a6-4bcfcf8d21d6", which you need to know for authorization of API calls.</p> <p>Please note: When you create a key, please save it to a secure key store. Fronius does not have means to recover a lost key. If you lose a key, you need to recreate a new one.</p>
<b>Active status</b>	A key can be active or passive, and you can toggle its status. Active keys can be used, passive keys cannot be unless you toggle them.
<b>Expiry date</b>	<p>You can set a validity period for a key, e.g. if you want to enforce key renewals.</p> <p>By default, new keys do not have an expiry date; they can be used as long as you do not delete them, or set an expiry date and the expiry date is not yet reached. Once you define an expiry date, you cannot delete the expiry date any longer nor extend the expiry date into the future.</p>
<b>Last used date</b>	This attribute indicates the time and date when the key was last used for an API call. This way you can identify unused keys and delete or deactivate them for security reasons.

API calls expect to receive access key ID and access key value data in the HTTP header.

Examples:

<b>HTTP header example</b>
<pre>GET https://api.solarweb.com/swqapi/pvsystems HTTP/1.1 AccessKeyId: FKIAFEF58CFEFA94486F9C804CF6077A01AB AccessKeyValue: 47c076bc-23e5-4949-37a6-4bcfcf8d21d6</pre>

### CURL example

```
curl -X GET "https://api.solarweb.com/swqapi/pvsystems" -H "accept: application/json"
-H "AccessKeyId: FKIAFEF58CFEFA94486F9C804CF6077A01AB" -H "AccessKeyValue:
47c076bc-23e5-4949-37a6-4bcfcf8d21d6"
```

## 2.2 User impersonation

There are situations for applications which require the applications to see PV systems in context of another user, e.g. a service provider might want to show and analyze the data of their customers. For such use cases SWQAPI supports impersonation using JWT tokens in addition to API keys.



### Solar.web Premium

Please note that access to Solar.web Premium features through the API is only possible if the impersonated user owns a Solar.web Premium membership.

### 2.2.1 JWT token attributes

<b>JWT token value</b>	<p>A long string, identifying the customer and providing access to him, for example:</p> <p>eyJ4NXQiOiJ0R1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWpGaVpXWTBaamcxWVd0a09EWmtNRE5rTVEiLCJraWQiOiJ0R1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWpGaVpXWTBaamcxWVd0a09EWmtNRE5rTVEiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjois2hLZVZsc0lPXy1tWDhvZkZSzdJZyIsImF1ZCI6I1ljNHhtcEIyVnlyR2phcUlaGoxbXJE0FZ6VWEiLCJzdWIiOiJodWV0dG5lci50aG9tYXNybmRAZnJvbml1cy5jb20iLCJuYmYiOiJlODUyOTQ2NjcsImF6cCI6I1ljNHhtcEIyVnlyR2phcUlaGoxbXJE0FZ6VWEiLCJhbXI0lsicGFzc3dvcmQiXSwiaXNzIjoiejhtjYXJib24ucHJvdG9jb2x90lwvXC8ke2NhcmJvbi5ob3N0fVwvb2F1dGgyXC9vaWRjZGlzY292ZXJ5IiwiaXhwIjojNTg1Mjk4MjY3LCJpYXQiOiJlODUyOTQ2Njld9.HUXi1sySzyLqx2e0dLpr0sszi-YiI3nGNB4GZDDwIwVHUHC4s6ED8BqfvkfFn3s45LkvJQEvqb_Wd3QtMGnz0LEZ3RdK3A8GWdsDChVq_nzLP4FGC6b5lPoz9Xi6mH_pcxt36rzA2-vj_l_e6cT0rTXsIeIz0jZVNSZRAJ4-A5HpmEuvraoArAGUqc_yTntbfALhfJQkfsjoDAJRAfZXLknTvDKm2vMd0-uXjTQHM2dKAWGAz6r39cLQ24sFIIC7MDgIp4GpNVBCLFSNzkk7mV3fSEQvgIdAFMhEP4CY4lMTItLxdfrKxcF5SA7o2fU0-_710frdFYvrkesorDCiyfg</p>
<b>JWT expiry date</b>	<p>An expiry time (UTC time) for the JWT; a Fronius JWT is valid for exactly one hour and needs to be refreshed periodically.</p> <p>Note: A JWT stays valid even if the customer changes his password (for a maximum of one hour).</p>
<b>Refresh token</b>	<p>A token which can be used to refresh a JWT.</p>

### 2.2.2 Creating a JWT

JWT can be created in two ways.

<b>Option 1: User ID and password</b>	You can pass another user's Fronius Solar.web user ID and password. If you use this option, please be careful about the user's password and protect it against leaking.
<b>Option 2: Fronius login</b>	<p>You can have the user provide their Fronius Solar.web user ID and password in a Fronius login dialog in a browser-like window. This way, you never get in touch with the user's Fronius login credentials.</p> <p>Notes:</p> <ul style="list-style-type: none"><li>/ You need to provide a redirection URL to Fronius and parse the redirection header for the parameters returned.</li><li>/ Because of CORS protection, you cannot use a standard browser with Ajax to use this option. However, you might want to use toolkits for implementation of this option.</li></ul>

### 2.2.3 Using a JWT in a SWQAPI call

When you call SWQAPI, you need to pass the JWT in addition to your API key credentials. It is not possible to call without the API key credentials.

### 2.2.4 Examples:

## HTTP header example

[illegible]

**CURL example**

[illegible]

### Example with Postman (option 1):

In the following example we see the token values returned in the body. You need the refreshToken and the jwtToken. Copy the jwtToken value.

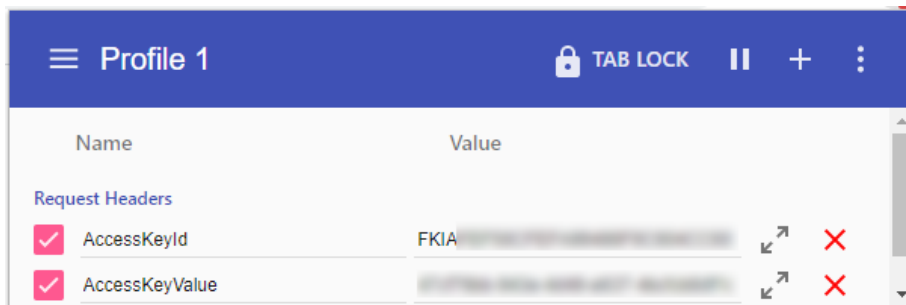
In the Authorization section select "Bearer" and then paste the `jwtToken` value from previous call.

Solar.web Query API manual 15/128

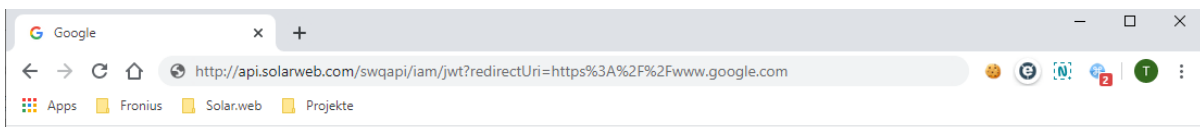
### Example with Chrome (option2):

It is not possible to test this function in Swagger or Postman directly, because the redirection to the Fronius login page sends them a JavaScript page which they cannot execute correctly. You need to test the GET call directly in a browser window or in a web app.

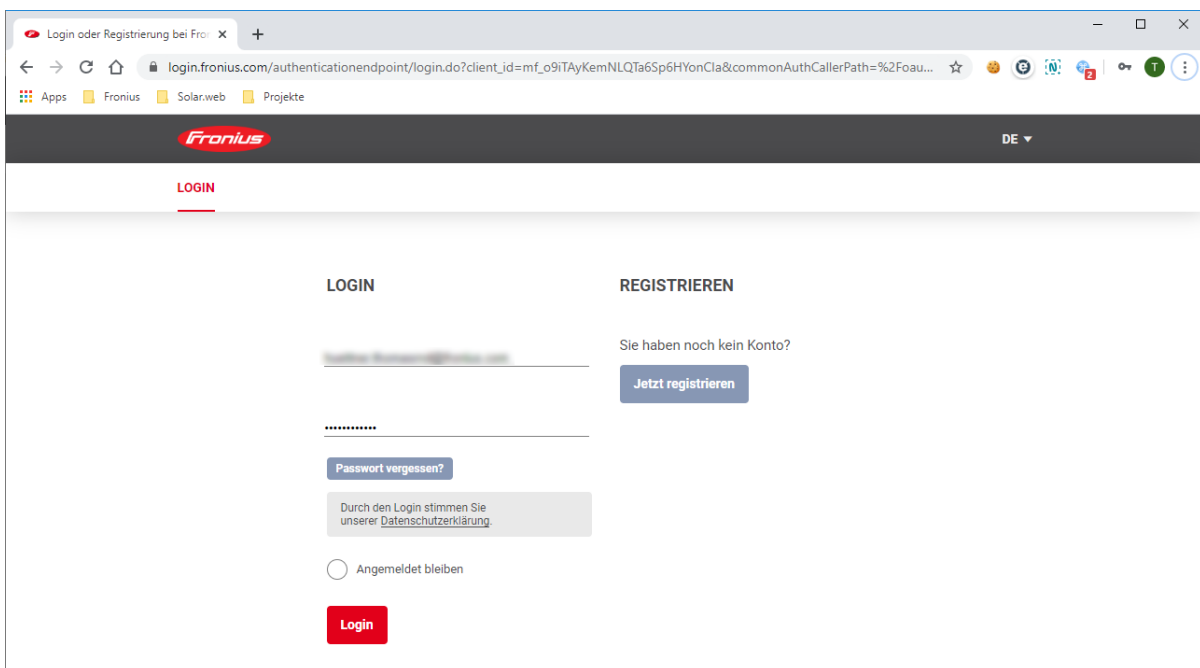
For testing this functionality in the browser, you need to pass the API key and its value. To do so please install a browser extension like "Modify Headers" and enter the API key and its value there.



Then enter the complete URL. In this example's redirectUri parameter we use [www.google.com](https://www.google.com), but you can use anything.



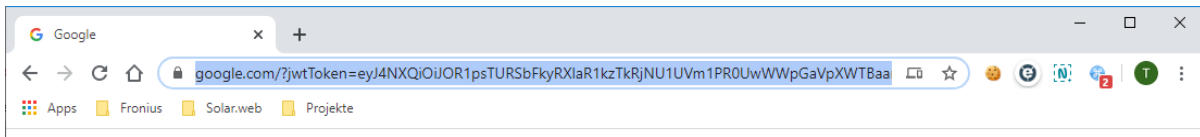
After pressing Enter on your keyboard, a Fronius login will appear in the browser.





Please enter your Fronius credentials.

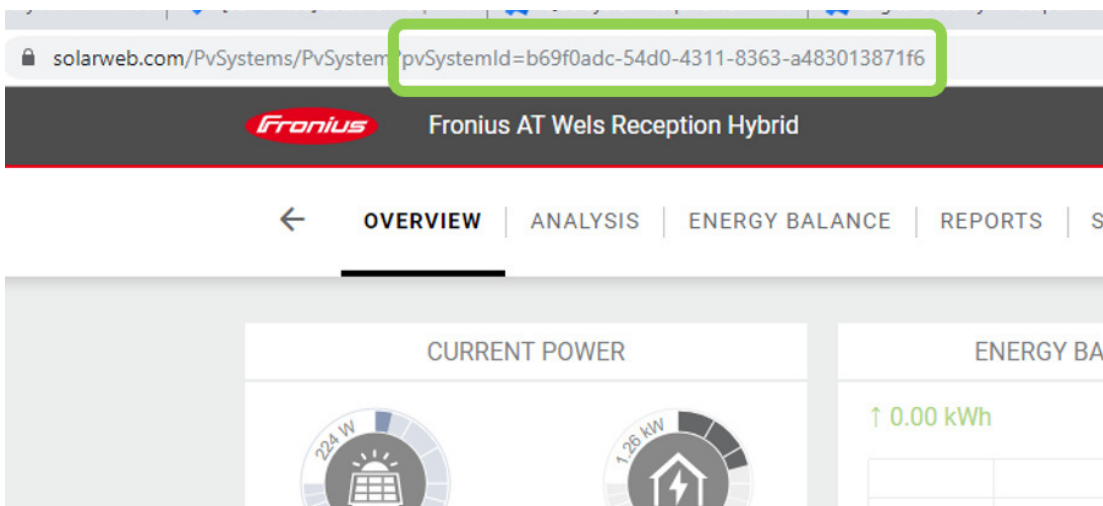
You are then redirected to the target URI, in our example this is [www.google.com](http://www.google.com). Please note that you can see the result parameters in the URL. You need to parse `jwtToken`, `refreshToken` and `jwtTokenExpiration` from the URI.



Copy the string after `jwtToken` and use it for your testing.

## 2.3 Identification of PV systems

Each PV system has its own unique ID (PV system ID) which is a mandatory parameter for many API endpoints. A PV system ID can be determined by using the `pvsystems` API endpoints that provide information about all PV systems linked to the user's account. The PV system ID can also be found in the URL of the system in Fronius Solar.web (marked green in screenshot below).



Please note that a PV system can only be accessed through the API if the user has access to it (by ownership or access permission).

## 2.4 Pagination


When returning many results, SWQAPI makes use of HATEOAS principles to support pagination. Additionally, SWQAPI returns a `"totalItemsCount"` object.

The default pagination limit is 50, the maximum pagination limit is 1000 currently.

Example:

#### Example JSON return object with pagination information (see the "links" object type)

```
{
  "pvSystemIds": [
    ...
  ],
  "links": {
    "first": "https://api.solarweb.com/swqapi/pvsystems-list?offset=0&limit=50",
    "prev": null,
    "self": "https://api.solarweb.com/swqapi/pvsystems-list?offset=0&limit=50",
    "next": "https://api.solarweb.com/swqapi/pvsystems-list?offset=50&limit=50",
    "last": "https://api.solarweb.com/swqapi/pvsystems-list?offset=150&limit=50",
    "totalItemsCount": 173
  }
}
```

 Please note that, for better readability, we do not show the paging objects in the command reference.

## 2.5 Date and time formats

SWQAPI supports extended UTC time formats (ISO 8601).

The principle format is either "yyyyMMddThh:mm:ssTZD" or "yyyy-MM-ddThh:mm:ssTZD" - where TZD is a timezone designator (either "Z" or an offset).

### http encoding speciality

If you are using a positive timezone offset, please use "%2b" instead of "+". Negative timezone offsets are not affected by the http encoding.

Examples:

```
/ 2018-10-11T13:00:00%2b01:00 instead of 2018-10-11T13:00:00+01:00
/ 2018-10-11T13:00:00-01:00
```

### 2.5.1 Use time in the calling URL

#### Example URIs, all showing the same time request

```
// get all historical temperature values (Temp1 channel) using different timezones in
the URL
// all examples below are for October 10th, 2018, from 11am to 12am zulu time

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/histdata?
from=2018-10-10T11:00:00Z&to=2018-10-11T12:00:00Z?channel=Temp1
    // zulu time notation

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/histdata?
from=20181010T120000%2b01:00&to=20181011T130000%2b01:00?channel=Temp1
    // CET (+01:00 offset to zulu time), compact encoding
    // please note that the "+" in the offset needs to be encoded with "%2b"

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/histdata?
from=2018-10-10T06:00:00-05:00&to=2018-10-11T07:00:00-05:00?channel=Temp1
    // EST (-05:00 offset to zulu time)
```

Additionally, you can also use local time of the PV system.

#### Example URIs, all showing the same time request

```
// get all historical temperature values (Temp1 channel) for October 10th, 2018, from
11am to 12am local time of the PV system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/histdata?
from=2018-10-10T11:00:00&to=2018-10-11T12:00:00?channel=Temp1
    // local time, depending on where PV system 20bb600e-019b-4e03-9df3-a0a900cda689 is
located
```

### 2.5.2 Time in response objects

When returning data, SWQAPI will either return zulu or local UTC time, extended encoding.

By default SWQAPI delivers zulu time, but you can use the "timezone" parameter to request conversion to the local time zone where the PV system is located. Local time (i.e. without timezone offset) is not returned, but when you ignore the offset, you have the system's local time.

### Example responses, all showing the same time

```
// data for August 31st, 2019; 12am zulu time

// timezone=zulu
"logDateTime": "2019-07-31T12:00:00Z"
  // zulu time notation

// timezone=local variations, depending on the timezone location of the queried PV
system
"logDateTime": "2019-07-31T12:00:00+00:00"
  // assuming PV system is in zulu time (without offset time)

"logDateTime": "2019-07-31T13:00:00+01:00"
  // assuming PV system is in CET (+01:00 offset to zulu time)

"logDateTime": "2019-07-31T07:00:00-05:00"
  // assuming PV system is in EST (-05:00 offset to zulu time)
```

## 3 SUPPORTING UIS

### 3.1 API key management in Solar.web

In Solar.web you can manage the API keys which are required for working with SWQAPI.

For managing API keys in Solar.web, go to **User Settings** and then got to the **REST API** tab. This tab has two views, one of them is **Key management**. Here you get an overview of your keys. Please note that the time information (creation date, expiry date, and last used date) is given in UTC zulu time.

The screenshot shows the Solar.web interface with the 'REST API' tab selected. Under 'Key management', there is a search bar and a 'Hide expired keys' checkbox. A table lists API keys with columns for Name, Description, Status, Creation date, Expiry date, Last used date, and Actions. The table contains 6 entries. Below the table, there is a 'Show 10 entries' dropdown, a 'Showing 1 to 6 of 6 entries' status, and pagination buttons (First, Previous, Next, Last). At the bottom, there are buttons for 'CREATE NEW KEY' and 'TEST REST API IN SWAGGER'.

Name	Description	Status	Creation date	Expiry date	Last used date	Actions
Test key	Test key	<input checked="" type="checkbox"/>	21.04.2020 / 02:00		21.04.2020 / 14:14	⋮
Test key	Test key with expiry date	<input type="checkbox"/>	20.04.2020 / 02:00	21.04.2020 / 01:59		⋮
Test key	Test key with expiry date much in the fut...	<input checked="" type="checkbox"/>	20.04.2020 / 02:00	31.12.2099 / 00:59		⋮
Test key	Test key, disabled	<input type="checkbox"/>	16.10.2019 / 08:46			⋮
Test Key	Test key for various purposes	<input checked="" type="checkbox"/>	23.07.2019 / 10:39			⋮
Primary Key		<input checked="" type="checkbox"/>	13.05.2019 / 18:39		21.04.2020 / 13:51	⋮

Actions:

- / If you want to create a new key, please press the **CREATE NEW KEY** button. This will create a new key, which will be downloaded in a JSON file containing the API key ID and its secret value.
- / You can give a name and description to each key, how it most convenes to you. To do so, please press the three dots in the **Action** menu column, and then select **Edit**.  
Please note: Expired keys cannot be renamed.
- / If you want to set an expiry date please press the three dots in the **Action** menu column, too, and then select **Edit**.  
Please note: Expiry dates cannot be removed or changed to a later date once they are set. If a key is expired, you can no longer edit it.
- / To deactivate a key, toggle its status in the **Status** column.
- / Expired or inactive keys can be deleted. To do so, please press the three dots in the **Action** menu column, and then select **Delete**.

Recommendations:

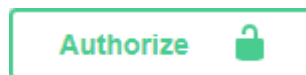
- / If you have multiple developers, create separate keys for each developer. Create another key for automated tested, staging and production systems.
- / If a key is compromised, especially keys for production, please renew the key:

- / Create a new key.
- / Set an expiry date for the old key or deactivate the old key as soon as the new one is deployed.
- / If you want to implement periodic key renewals for security reasons:
  - / Create a new key.
  - / Set an expiry date for the old key which gives you enough overlapping time to push the new key to all relevant systems.

### 3.2 Working with API keys in Swagger UI

If you are using the Swagger UI (<https://api.solarweb.com/swqapi/index.html>), you need to enter the API keys only once.

Press the *Authorize* button in Swagger UI.



Enter both the accesskey ID and its values. Press the *Authorize* button in both sections.

**Available authorizations** x

**AccessKeyId (apiKey)**  
AccessKeyId: FKIA123456789  
Name: AccessKeyId  
In: header  
Value:

**AccessKeyValue (apiKey)**  
AccessKeyValue: 123456789  
Name: AccessKeyValue  
In: header  
Value:

Finally, close the dialog using the x button on the top right corner.

Available authorizations

AccessKeyId (apiKey)

Authorized

AccessKeyId: FKIA123456789

Name: AccessKeyId

In: header

Value: \*\*\*\*\*

Logout

AccessKeyValue (apiKey)

Authorized

AccessKeyValue: 123456789


Name: AccessKeyValue

In: header

Value: \*\*\*\*\*

Logout

Result: The *Authorize* button shows a closed lock now.

Authorize 



#### Swagger UI does not verify the keys

The closed lock in the *Authorize* button does not mean that your access keys are correct. They are verified directly by the APIs you call.

Please note that the authentication key is reset if you do a page refresh.

## 4 API REFERENCE

### 4.1 User impersonation calls

#### 4.1.1 Impersonate: Receive a JWT using the Fronius login

 This call only works when executed by a browser. It does not work in browser apps using AJAX.

##### Use cases for web developers

- / I want to login to Fronius using the customer's user ID and password. The end user enters his user ID and password in a browser using a Fronius web form, I never see them.

##### Methods

Method	End point and objects	Event name	Description
GET	swqapi/iam/jwt	GenerateJwtExternal	Generates a JWT and refresh token pair using the Fronius login page.

##### Filters and parameters

Filter	Description
?redirectUri=<link>	Mandatory https address of a website to which the user is redirected after this call.

##### Example calls

```
GET api.solarweb.com/swqapi/iam/jwt?redirectUri=https://www.my-little-app.com/
user=425675472645/dashboard
// generates a new JWT for impersonation and redirects to my-little-app page

GET api.solarweb.com/swqapi/iam/jwt?redirectUri=myapp://dashboard
// example for a mobile app to generate a new JWT for impersonation, and then have
the browser redirect back to the app's dashboard based on a custom URL scheme
```

##### Input objects

n/a

##### Response objects

Because of the redirection, the response is passed to the redirected URL in the query string. The redirected page needs to parse the response parameters.



The following information is returned:

Type	Objects
refreshToken	/ String
jwtToken	/ String
jwtTokenExpiration	/ String (UTC time)

#### Example responses

```
https://www.my-little-app.com/user=425675472645/dashboard?
jwtToken=eyJ4NXQiOiJ0R1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWpGaVpXWTBaamcxWVd0a09EWmtN
RE5rTVEiLCJraWQiOiJ0R1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWpGaVpXWTBaamcxWVd0a09EWmtNR
E5rTVEiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoIYjRYOTFCTmo5M2Rld3pXdGZlZWRuZyIsImF1ZCI6I
lljNHhtcEiYVnlyR2phcUlrGoxbXJE0FZ6VWEiLCJjX2hhc2giOiJtYlRmbzdXMnZRei11SkFFNGRvbHZ3Ii
wic3ViIjoIYm9zbmphaW5tYXJpb0Bmcm9uaXVzLmNvbSIsIm5iZiI6MTU4MDEyODYyMSwiYXpwIjoIWM0eG1
wQjJWexJHAmFwSwtoajFtckQ4VnpVYSIsImFtcCI6WyJDbDdXN0b21BdXRoZW50aWNhdG9yTG9jYWxNYWluIl0s
ImZlcyI6IiR7Y2FyYm9uLnByb3RvY29sfTpcL1wvJHtjYXJib24uaG9zdH1cL29hdXR0Mlwwb2lkY2Rpc2Nvd
mVyeSIsImV4cCI6MTU4MDEzMjIyMSwiaWF0IjoxNTgwMTI4NjIxLCJub25jZSI6ImFzZGYiLCJzaWQiOiI1Ym
VhYTU0Zi1i0ThjLTRhMzgtODY5ODU1ZjgyOWUyNGY1MjMifQ.1wkCu6Mqo-Fs5l3QYFg3B5kWLIF_rRd-
bEPvXhFqNt_cKsi8xZGrHlU0ndN0LBxsN6iC0Ky12-pLKjJ4I7iU0ilPkzbVHPW-0ZYUVz2-
n6C3TLQAUU6kHPpu4h8JhM0rV5Tg_R1FDhZ6cdVac1uweNhs6VJ9HSTZ55mvMpdPzfRvH-
xdtjmW0GpIhz7-
eDL6hrR06i6h0TomydcFscr_Vtft4e0L3iuYp4xVTMpQTPeSZSdUrjrV0ESQR7RhAB2ijdInWFYMBv5pcdFvc
fNGgj7f2fRPIWu0uzl0eYTasIke0FC4PucUgZKYH40Qj5RyrxnzlyU1gX7DlqWwvtmg&refreshToken=16a4
9996-47f2-3c47-b11e-33b681428e4f&jwtTokenExpiration=27.1.2020.%2013:37:04
```

#### 4.1.2 Impersonate: Receive a JWT using userID and password

##### Use cases for web developers

- / I want to login to Fronius using a customer's user ID and password. (E.g. the customer enters the Fronius login credentials on my website and I am able to store it.)

##### Security notes

Storing customer passwords needs to be carefully designed, because passwords can easily leak. Please make use of operating system capabilities, such as iCloud Keychain, Android Keystore, Credential Manager in Windows, etc.

Additionally, please consider GDPR and other PII regulation.

- / I used the former "ThirdParty API" (predecessor of SWQAPI) and used the customer's credentials to login. When I migrate from ThirdParty API to SWQAPI I can reuse the credentials, thus the customer does not notice a change.

## Methods

Method	End point and objects	Event name	Description
POST	swqapi/iam/jwt	GenerateJwt	Generates a JWT and refresh token pair by passing credentials.

## Filters and parameters

Filter	Description
?scope=<scope>	<p>Optional but recommended: Scope of the token for multiple sessions. The scope can be generic or specific for a user agent (e.g. a device or app ID).</p> <p>Scopes allow users to be logged in from multiple devices and in Solar.web in parallel.</p>

## Example calls

```
POST api.solarweb.com/swqapi/iam/jwt?scope=my-app.23423af9afe0af0
// generates a new JWT for impersonation in a specific scope
```

## Input objects

JSON object input construction:

Type	Objects
credentials	<ul style="list-style-type: none"><li>/ userId (String)</li><li>/ password (String)</li></ul>

## Example input


```
{
  "userId": "mike@thisisme.com",
  "password": "thisIsMyVeryPrivatePassword!"
}
```

## Response objects

JSON object answer construction:



## Filters and parameters

Filter	Description
?scope=<scope>	<p>Optional but recommended: Scope of the token for multiple sessions, which needs to be the same scope when the original token was created. The scope can be generic or specific for a user agent (e.g. a device or app ID).</p> <p>Scopes allow users to be logged in from multiple devices and in Solar.web in parallel.</p> <div> Must not include any scope values not originally granted, and if omitted is treated as equal to the originally granted scope.</div>

## Example calls

```
PATCH api.solarweb.com/swqapi/iam/jwt/98a47454-b650-34b8-9a8c-27adae447ab71?scope=my-app.23423af9afe0af0
// refreshes an existing token and generates a new one, uses the original scope
```

## Response objects

JSON object answer construction:

Type	Objects
refreshToken	/ String
jwtToken	/ String
jwtTokenExpiration	/ String (UTC time)

### Example responses

[illegible]

## 4.2 Generic information calls

### 4.2.1 Info: Get release information

#### Use cases for web developers

- / I want to know the version number of the REST API which I am using.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/info/release	GetInfoRelease	Retrieves version and release date information about the REST API.

#### Filters and parameters

n/a

#### Example calls

```
GET api.solarweb.com/swqapi/info/release
// retrieves the API's full version number and release date
```

#### Response objects

JSON object answer construction:

Type	Objects
info/release	<ul style="list-style-type: none"><li>/ ReleaseVersion (String)</li><li>/ ReleaseDate (String, Date)</li></ul>

#### Example responses

```
{
  "releaseVersion": "1.0.0.0",
  "releaseDate": "2019-10-07T"
}
```

### 4.2.2 Info: Get user information

#### Use cases for web developers

- / I want to show the customer's address data.
- / I want to know if the customer is a premium customer.
- / I want to know if the customer has accepted the (latest) Terms of Use.

## Methods

Method	End point and objects	Event name	Description
GET	swqapi/info/user	GetInfoUser	Returns user information of either the logged in user or the impersonated user.

## Filters and parameters

n/a

## Example calls

```
GET api.solarweb.com/swqapi/info/user
// returns user information
```

## Response objects

JSON object answer construction:

Type	Objects
user	<ul style="list-style-type: none"><li>/ name (Object)<ul style="list-style-type: none"><li>/ title (String) - e.g. "Mr." or "Mrs."</li><li>/ firstName (String)</li><li>/ lastName (String)</li></ul></li><li>/ address (Object)<ul style="list-style-type: none"><li>/ country (String)</li><li>/ zipCode (String)</li><li>/ city (String)</li><li>/ street (String)</li></ul></li><li>/ contactInformation (Object)<ul style="list-style-type: none"><li>/ telephone (String)</li><li>/ email (String)</li></ul></li><li>/ settings (Object)<ul style="list-style-type: none"><li>/ timeZone (String) - in Olson format</li><li>/ dateFormat (String) - "DD.MM.YYYY", "MM.DD.YYYY", "YYYY.MM.DD", "DD/MM/YYYY", "MM/DD/YYYY", "YYYY/MM/DD"</li><li>/ timeFormat (String) - "12h", "24h"</li><li>/ language (String) - in ISO language code</li></ul></li><li>/ accountAttributes (Object)<ul style="list-style-type: none"><li>/ premiumMembership (Boolean)</li><li>/ termsAcceptedLatest (Boolean)</li><li>/ termsAcceptedVersion (Integer)</li></ul></li></ul>

## Example responses

```
{
  "name": {
    "title": "Mr.",
    "firstName": "Thomas",
    "lastName": "Tester"
  },
  "address": {
    "country": "AT",
    "zipCode": "4600",
    "street": "Froniusplatz 1",
    "city": "Wels"
  },
  "contactInformation": {
    "telephone": null,
    "email": "tester.thomas@fronius.com"
  },
  "settings": {
    "timeZone": "Europe/Berlin",
    "dateFormat": "DD.MM.YYYY",
    "timeFormat": "24h",
    "language": "EN"
  },
  "accountAttributes": {
    "premiumMembership": true,
    "termsAcceptedLatest": true,
    "termsAcceptedVersion": 2
  }
}
```



## 4.3 Metadata calls

This set contains methods to retrieve

- / the number of PV systems linked to the user's account,
- / a list of all PV system IDs,
- / detailed information about the PV systems,
- / the number of devices of a given PV system,
- / a list of all devices of a given PV system and
- / detailed information about the devices of a PV system

### 4.3.1 Metadata: Get PV system information

This call returns detailed meta information for one or more PV systems that are linked to the user's account (e.g. through ownership or guest/supervisor permission).

#### Use cases for web developers

- / I want to get the meta information of all or specific PV systems: such as name, location, peak power, picture, activation date etc.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/pvsystems	GetSystemMetaDataList	Returns a list of all PV systems for the user. The list is a JSON array containing PV systems and their metadata. Parameters allow pagination ("offset" & "limit"), or filtering for specific PV system attributes ("type"; for example, if "type"="ohmpilot" you only get PV systems which have an Ohmpilot).
GET	swqapi/ pvsystems/{pv- system-id}	GetSystemMetaData	Returns metadata of the PV system with the given ID, including metadata for the system's devices. Filters do not apply.

#### Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns pv-systems from a starting <offset> and returning not more than <limit> items.
?type=inverter	Filters for systems and devices with inverters.
?type=sensor	Filters for systems and devices with sensors.
?type=battery	Filters for systems and devices with batteries.
?type=smartmeter	Filters for systems and devices with Smart Meters.

Filter	Description
?type=ohmpilot	Filters for systems and devices with Ohmpilots.
?type=datalogger	Filters for systems and devices with dataloggers.

#### Example calls

```
GET api.solarweb.com/swqapi/pvsystems
// retrieves metadata of all PV systems

GET api.solarweb.com/swqapi/pvsystems?offset=200&limit=50
// returns metadata of 50 PV systems, starting at offset 200

GET api.solarweb.com/swqapi/pvsystems?type=battery
// returns metadata of all PV systems which have a battery (caution: does not
retrieve the battery device info)

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689
// returns metadata of a specific PV system including metadata for all its devices
```

#### Response objects

JSON object answer construction:

Type	Objects
PV systems	<ul style="list-style-type: none"> <li>/ lastImport (String, UTC timestamp)</li> <li>/ installationDate (String, UTC timestamp)</li> <li>/ pvSystemId (String)</li> <li>/ name (String)</li> <li>/ address (Object) <ul style="list-style-type: none"> <li>/ street (String)</li> <li>/ zipCode (String)</li> <li>/ city (String)</li> <li>/ state (String)</li> <li>/ country (String)</li> </ul> </li> <li>/ timezone (String, Olson format)</li> <li>/ pictureURL (String, URL)</li> <li>/ peakPower (Number)</li> <li>/ meteoData (String)</li> </ul>

## Example responses

### Example for a single PV system

```
{
  "pvSystemId": "04d81b82-7861-4e36-8e7f-41036ce711a4",
  "name": "Fronius AT Wels Reception Hybrid",
  "address": {
    "country": "AT",
    "zipCode": "4600",
    "street": "Günter Fronius Straße 1",
    "city": "Thalheim bei Wels",
    "state": "00"
  },
  "pictureURL": "https://www.solarweb.com/Image/Show?pvSystemId=04d81b82-7861-4e36-8e7f-41036ce711a4&pictureId=20991838-16d7-4a9a-83fd-a75b00b34211",
  "peakPower": 5000.0,
  "installationDate": "2000-01-01T00:00:00Z",
  "lastImport": "2020-03-27T06:03:42Z",
  "meteoData": "light",
  "timeZone": "Europe/Berlin"
}
```

## Example for multiple PV systems

```
{
  "pvSystems": [
    {
      "pvSystemId": "20bb600e-019b-4e03-9df3-a0a900cda689",
      "name": "Fronius AUS Melbourne",
      "address": {
        "country": "AU",
        "zipCode": "3043",
        "street": " _",
        "city": "Tullamarine",
        "state": null
      },
      "pictureURL": "https://www.solarweb.com/Image/Show?pvSystemId=20bb600e-019b-4e03-9df3-a0a900cda689&pictureId=dbe22d74-02cd-480d-8565-410b3dffccce",
      "peakPower": 12880.0,
      "installationDate": "2011-06-01T00:00:00Z",
      "lastImport": "2020-02-14T02:36:08Z",
      "meteoData": "light",
      "timeZone": "Australia/Sydney"
    },
    {
      "pvSystemId": "83535831-3e55-46b4-a48c-a4e500ddcd1b",
      "name": "Fronius AT Sattledt Hybrid",
      "address": {
        "country": "AT",
        "zipCode": "4",
        "street": "Froniusstrasse 1",
        "city": "Sattledt",
        "state": "00"
      },
      "pictureURL": "https://www.solarweb.com/Image/Show?pvSystemId=83535831-3e55-46b4-a48c-a4e500ddcd1b&pictureId=bb4af026-540a-fb66-e053-0204ff0a5ac0",
      "peakPower": 5001.0,
      "installationDate": "2000-01-01T00:00:00Z",
      "lastImport": "2018-01-16T00:25:04Z",
      "meteoData": "light",
      "timeZone": "Europe/Berlin"
    },
    {
      "pvSystemId": "04d81b82-7861-4e36-8e7f-41036ce711a4",
      "name": "Fronius AT Wels Reception Hybrid",
      "address": {
        "country": "AT",
        "zipCode": "4600",
        "street": "Günter Fronius Straße 1",
        "city": "Thalheim bei Wels",
        "state": "00"
      },
    },
  ],
}
```

```

        "pictureURL": "https://www.solarweb.com/Image/Show?
pvSystemId=04d81b82-7861-4e36-8e7f-41036ce711a4&pictureId=20991838-16d7-4a9a-83fd-
a75b00b34211",
        "peakPower": 5000.0,
        "installationDate": "2000-01-01T00:00:00Z",
        "lastImport": "2020-03-27T06:03:42Z",
        "meteoData": "light",
        "timeZone": "Europe/Berlin"
    },
    {
        "pvSystemId": "0794d488-1d9e-467c-91c1-d89342949c60",
        "name": "Fronius AT SAT Testraum 1PN",
        "address": {
            "country": "AT",
            "zipCode": "4650",
            "street": "Bahnhofstraße 4/4 ",
            "city": "Lambach",
            "state": "OÖ"
        },
        "pictureURL": "https://www.solarweb.com/Image/Show?
pvSystemId=0794d488-1d9e-467c-91c1-d89342949c60&pictureId=e594903f-49a5-
ed47-9f5a-4d685da7a233",
        "peakPower": 175000.0,
        "installationDate": "2000-01-01T00:00:00Z",
        "lastImport": "2020-01-13T07:36:36Z",
        "meteoData": "light",
        "timeZone": "Europe/Berlin"
    },
    {
        "pvSystemId": "85896da3-bb2a-47f7-9c6e-2909dd44832c",
        "name": "Sippi 1",
        "address": {
            "country": "AU",
            "zipCode": "2600",
            "street": "Perth Ave",
            "city": "Canberra",
            "state": "ACT"
        },
        "pictureURL": "https://www.solarweb.com/Image/Show?pvSystemId=85896da3-
bb2a-47f7-9c6e-2909dd44832c&pictureId=a6bfb87e-2263-4c68-bc46-a7a00065859c",
        "peakPower": 41901.0,
        "installationDate": "2000-01-01T00:00:00Z",
        "lastImport": "2020-03-27T06:30:53Z",
        "meteoData": "light",
        "timeZone": "Europe/Berlin"
    }
}
]
}

```

#### 4.3.2 Metadata: Count PV systems

This call returns the number of all PV systems that are linked to the user's account (e.g. through ownership or guest/supervisor permission).

#### Use cases for web developers

- / I want to know how many PV systems I can access. (Needed for subsequent enumeration and detail calls.)
- / I want to know how many PV systems I need to show in the UI, so I can prepare memory and pagination.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems-count	GetSystemCount	Returns the count of all PV systems. Filters can be applied to return only the number of PV systems with certain devices (e.g. inverters, Ohmpilots, batteries etc).

#### Filters and parameters

Filter	Description
?type=inverter	Filters for systems with inverters.
?type=sensor	Filters for systems with sensors.
?type=battery	Filters for systems with batteries.
?type=smartmeter	Filters for systems with Smart Meters.
?type=ohmpilot	Filters for systems with Ohmpilots.
?type=datalogger	Filters for systems with dataloggers.

#### Example calls

```
GET api.solarweb.com/swqapi/pvsystems-count
// counts all PV systems

GET api.solarweb.com/swqapi/pvsystems-count?type=battery,smartmeter
// counts all PV systems with batteries or smartmeters
```

#### Response objects

JSON object answer construction:

Type	Objects
n/a	/ count (Number)

## Example responses

```
{  
  "count": 4  
}
```

### 4.3.3 Metadata: Enumerate PV system IDs

This call returns a list of the PV system IDs of all PV systems that are linked to the user's account (e.g. through ownership or guest/supervisor permission). The IDs are required for other calls to query data from those systems.

#### Use cases for web developers

- / I want to enumerate all PV systems which I can access. With the IDs I can scan these systems in subsequent calls.
- / I want to know how many devices I need to show in the UI, so I can prepare memory and pagination.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems-list	GetSystemIdList	Returns the IDs of PV systems. Parameters allow pagination ("offset" & "limit"), or filtering for specific PV system attributes ("type"; for example, if "type"="battery" you only get PV systems which have a battery).

#### Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns pv-systems from a starting <offset> and returning not more than <limit> items.
?type=inverter	Filters for systems with inverters.
?type=sensor	Filters for systems with sensors.
?type=battery	Filters for systems with batteries.
?type=smartmeter	Filters for systems with Smart Meters.
?type=datalogger	Filters for systems with dataloggers.
?type=ohmpilot	Filters for systems with Ohmpilots.

### Example calls

```
GET api.solarweb.com/swqapi/pvsystems-list
// returns all PV system IDs

GET api.solarweb.com/swqapi/pvsystems-list?offset=200&limit=50
// returns PV system IDs, starting at offset 200 and returning a page of 50 items

GET api.solarweb.com/swqapi/pvsystems-list?type=battery
// returns PV system IDs which have a battery
```

### Response objects

JSON object answer construction:

Type	Objects
n/a	/ pvSystemIds (Array of Strings)

### Example responses

```
{
  "pvSystemIds": [
    "20bb600e-019b-4e03-9df3-a0a900cda689",
    "83535831-3e55-46b4-a48c-a4e500ddcd1b",
    "04d81b82-7861-4e36-8e7f-41036ce711a4",
    "0794d488-1d9e-467c-91c1-d89342949c60"
  ]
}
```

## 4.3.4 Metadata: Get device information

### Use cases for web developers

- / I want to get the meta information of all or specific devices a specific PV systems has: such as device types, capabilities, device detail information, attached sensors etc.

### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/devices	GetDeviceMetaDataTable	Returns a list of PV components for a given PV system. The list is a JSON array containing devices and their metadata. Filters allow pagination and filtering of device types.



Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv-system-id}/ devices/{device-id}	GetDeviceMetaData	Returns the metadata information of the requested PV device of a PV system with the given ID. (Serial number, model (inverter type, Ohmpilot, battery, Smart Meter), manufacturer, ...)

#### Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns devices from a starting <offset> and returning not more than <limit> items.
?type=inverter	Filters for devices with inverters.
?type=sensor	Filters for devices with sensors.
?type=battery	Filters for devices with batteries.
?type=smartmeter	Filters for devices with Smart Meters.
?type=ohmpilot	Filters for devices with Ohmpilots.
?type=datalogger	Filters for devices with dataloggers.
?status=active	Filters for active devices only.
?status=inactive	Filters for inactive devices only.

#### Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices
// returns metadata of all devices in the given PV system


GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices?
type=inverter
// returns metadata for all inverters in the given PV system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices?
offset=0&limit=5
// returns the metadata of the first five devices in the given PV system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
d2e61bf2-8dd7-4ba1-8733-d55d738c4679
// returns the metadata of a specific device
```

## Response objects

JSON object answer construction:

Type	Objects
Inverter	<ul style="list-style-type: none"> <li>/ deviceType (String - "Inverter")</li> <li>/ deviceId (String)</li> <li>/ deviceName (String)</li> <li>/ deviceManufacturer (String - usually "Fronius")</li> <li>/ serialNumber (String)</li> <li>/ numberMPPTTrackers (Number)</li> <li>/ numberPhases (Number - 1 to 3)</li> <li>/ peakPower (Object) <ul style="list-style-type: none"> <li>/ dc1 (Number)</li> <li>/ dc2 (Number)</li> <li>/ ...</li> </ul> </li> </ul> <div style="border: 1px solid #f0e68c; padding: 10px; margin: 10px 0;"> <p> New inverter types can have more than two strings. If there are more than two strings, they will be added to the peakPower object as "dcN" (where N is the number of the string).</p> </div> <ul style="list-style-type: none"> <li>/ nominalAcPower (Number)</li> <li>/ nodeType (Integer)</li> <li>/ firmware (Object) <ul style="list-style-type: none"> <li>/ updateAvailable (Boolean)</li> <li>/ installedVersion (String)</li> <li>/ availableVersion (String)</li> </ul> </li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> </ul>

Type	Objects
Battery	<ul style="list-style-type: none"> <li>/ deviceType (String - "Battery")</li> <li>/ deviceId (String)</li> <li>/ deviceName (String)</li> <li>/ deviceManufacturer (String)</li> <li>/ serialNumber (Number)</li> <li>/ capacity (Number)</li> <li>/ firmware (Object) <ul style="list-style-type: none"> <li>/ updateAvailable (Boolean)</li> <li>/ installedVersion (String)</li> <li>/ availableVersion (String)</li> </ul> </li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> </ul>
Smart Meter	<ul style="list-style-type: none"> <li>/ deviceType (String - "SmartMeter")</li> <li>/ deviceId (String)</li> <li>/ deviceName (String)</li> <li>/ deviceManufacturer (String)</li> <li>/ serialNumber (Number)</li> <li>/ firmware (Object) <ul style="list-style-type: none"> <li>/ updateAvailable (Boolean)</li> <li>/ installedVersion (String)</li> <li>/ availableVersion (String)</li> </ul> </li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> </ul>

Type	Objects
Sensor	<ul style="list-style-type: none"> <li>/ deviceType (String - "Sensor")</li> <li>/ deviceId (String)</li> <li>/ deviceName (String)</li> <li>/ deviceManufacturer (String)</li> <li>/ serialnumber (String)</li> <li>/ firmware (Object) <ul style="list-style-type: none"> <li>/ updateAvailable (Boolean)</li> <li>/ installedVersion (String)</li> <li>/ availableVersion (String)</li> </ul> </li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> <li>/ sensors (Array of objects) <ul style="list-style-type: none"> <li>/ sensorName (String)</li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> </ul> </li> </ul>

Type	Objects
Ohmpilot	<ul style="list-style-type: none"> <li>/ deviceType (String - "OhmPilot")</li> <li>/ deviceId (String)</li> <li>/ deviceName (String)</li> <li>/ deviceManufacturer (String - "Fronius")</li> <li>/ serialnumber (String)</li> <li>/ firmware (Object) <ul style="list-style-type: none"> <li>/ updateAvailable (Boolean)</li> <li>/ installedVersion (String)</li> <li>/ availableVersion (String)</li> </ul> </li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> <li>/ sensors (Array of objects) <ul style="list-style-type: none"> <li>/ sensorName (String - "Temperature")</li> <li>/ isActive (Boolean)</li> <li>/ activationDate (String, UTC timestamp)</li> <li>/ deactivationDate (String, UTC timestamp)</li> </ul> </li> </ul>

#### Example responses

### Example for an inverter

```
{
  "deviceType": "Inverter",
  "deviceId": "52a44bc2-3697-4339-9437-6d077c44aac4",
  "deviceName": "Fronius Galvo 3.0-1",
  "deviceManufacturer": "Fronius",
  "serialNumber": "28102747",
  "nodeType": 97,
  "numberMPPTTrackers": 1,
  "numberPhases": 1,
  "peakPower": {
    "dc1": 3000,
    "dc2": null
  },
  "nominalAcPower": 3000,
  "firmware": {
    "updateAvailable": false,
    "installedVersion": null,
    "availableVersion": "fro27372"
  },
  "isActive": true,
  "deactivationDate": null
}
```

### Example for multiple devices

```
[
  {
    "deviceType": "Inverter",
    "deviceId": "c883f93f-6661-426f-a2c5-0f381ff86c89",
    "deviceName": "Fronius Symo Hybrid 5.0-3-S",
    "deviceManufacturer": "Fronius",
    "serialNumber": "25441119",
    "nodeType": 97,
    "numberMPPTTrackers": 1,
    "numberPhases": 3,
    "peakPower": {
      "dc1": 5000,
      "dc2": null
    },
    "nominalAcPower": 5000,
    "firmware": {
      "updateAvailable": false,
      "installedVersion": null,
      "availableVersion": "fro27372"
    },
    "isActive": true,
    "deactivationDate": null
  },
  {
    "deviceType": "Battery",
    "deviceId": "83129be8-alec-48b1-a8a8-7c5accd6b64e",
    "deviceName": "Ext. Device",
    "deviceManufacturer": "Fronius International",
    "serialNumber": "26073337",
    "capacity": 3600,
    "firmware": {
      "updateAvailable": false,
      "installedVersion": "",
      "availableVersion": ""
    },
    "isActive": true,
    "deactivationDate": null
  },
  {
    "deviceType": "SmartMeter",
    "deviceId": "c573dde0-be38-4ba9-973e-66f54a4b4646",
    "deviceName": "Fronius Smart Meter (#1)",
    "deviceManufacturer": "Fronius",
    "firmware": {
      "updateAvailable": false,

```

#### Example for a sensor

```
{
  "deviceType": "Sensor",
  "deviceId": "9df7c03d-e008-42f8-8ad2-a1d400ccbf2c",
  "deviceName": "Sensorkarte",
  "deviceManufacturer": "Fronius",
  "updateAvailable": false,
  "firmware": {
    "updateAvailable": false,
    "installedVersion": "",
    "availableVersion": ""
  },
  "isActive": true,
  "deactivationDate": null,
  "sensors": [
    {
      "sensorName": "Temperature1",
      "isActive": true,
      "deactivationDate": null
    },
    {
      "sensorName": "Temperature2",
      "isActive": true,
      "deactivationDate": null
    },
    {
      "sensorName": "Energy Consumption",
      "isActive": true,
      "deactivationDate": null
    },
    {
      "sensorName": "Galvo Generation",
      "isActive": true,
      "deactivationDate": null
    }
  ]
}
```

```
    "installedVersion": "",
    "availableVersion": ""
  },
  "isActive": true,
  "deactivationDate": null
}
]
```

#### 4.3.5 Metadata: Count devices

##### Use cases for web developers

- / I want to know how many devices a specific PV systems has. (Needed for subsequent enumeration and detail calls.)
- / I want to know how many devices I need to show in the UI, so I can prepare memory and pagination.

## Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ devices-count	GetDeviceCount	Returns the count of all devices for a given PV system.

## Filters and parameters

Filter	Description
?type=inverter	Filters devices which are inverters.
?type=sensor	Filters devices which are sensors.
?type=battery	Filters devices which are batteries.
?type=smartmeter	Filters devices which are Smart Meters.
?type=ohmpilot	Filters devices which are Ohmpilots.
?type=datalogger	Filters devices which are dataloggers.
?status=active	Filters for active devices only.
?status=inactive	Filters for inactive devices only.

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices-count
// counts all devices in the given PV system
```

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices-count?type=smartmeter,ohmpilot
// counts all Smart Meter and Ohmpilot devices in the given PV system
```

## Response objects

JSON object answer construction:

Type	Objects
n/a	/ count (Number)

## Example responses

```
{
  "count": 4
}
```

### 4.3.6 Metadata: Enumerate device IDs

#### Use cases for web developers

- / I want to enumerate all devices a specific PV systems has. With the IDs I can scan these devices in subsequent calls.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ devices-list	GetDeviceIdList	Returns the IDs of devices in a PV system. Filters allow pagination and filtering of device types.

#### Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns devices from a starting <offset> and returning not more than <limit> items.
?type=inverter	Filters devices which are inverters.
?type=sensor	Filters devices which are sensors.
?type=battery	Filters devices which are batteries.
?type=smartmeter	Filters devices which are Smart Meters.
?type=ohmpilot	Filters devices which are Ohmpilots.
?type=datalogger	Filters devices which are dataloggers.
?status=active	Filters for active devices only.
?status=inactive	Filters for inactive devices only.

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices-  
list
```

```
// returns the device IDs of all devices in the given PV system
```

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices-  
list?type=inverter,sensor,battery
```

```
// returns the device IDs of all inverters, sensors and batteries in the given PV  
system
```

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices-  
list?offset=0&limit=5
```

```
// returns the device IDs of the first five devices in the given PV system
```

## Response objects

JSON object answer construction:

Type	Objects
n/a	/ deviceId (Array of Strings)

## Example responses

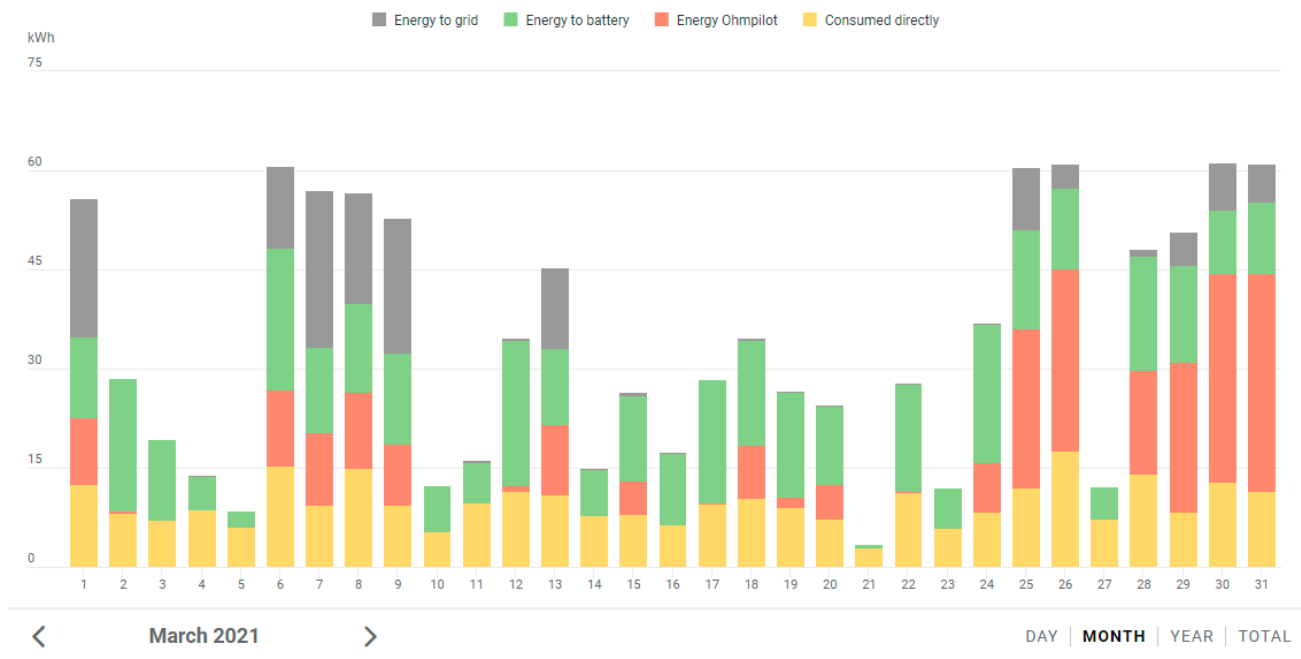
```
{  
  "deviceIds": [  
    "52a44bc2-3697-4339-9437-6d077c44aac4",  
    "58099f2e-56ab-415f-bcc4-a1d400ccbf56",  
    "6e089afa-280f-483d-b4f1-a1d600ae2582",  
    "fbd0af74-6b5b-4a02-bd32-8f91447225ae",  
    "9df7c03d-e008-42f8-8ad2-a1d400ccbf2c",  
    "ddef5593-76f6-41e4-9e4d-a1d400ccbf15",  
    "675570a3-7395-43d9-a45e-ffc4c5bf5390",  
    "6f1361c7-2003-4380-b2d5-d78645bcb07e",  
    "0a8a3b70-ae7e-4e7c-82f2-9007ce65b8ba"  
  ]  
}
```



## 4.4 Aggregation calls

This set contains methods to retrieve the energy production values of a PV system, aggregated over the whole lifetime, years, months, and/or days.

From this you can create diagrams like the following one:



### 4.4.1 Aggdata (outdated): Aggregated energy production for a PV system

Please note that this call is outdated and replaced by the aggrdata call.

#### Use cases for web developers

- / I want to show total/lifetime aggregated energy (production) values to an end user.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ aggdata	GetSystemAggregatedDataTotal	Gets the total lifetime aggregated data for a given PV system. The data is returned as a JSON object. Filters allow limiting to specific PV system energy values.
GET	swqapi/ pvsystems/{pv- system-id}/ aggdata/years	GetSystemAggregatedDataYears	Gets annual aggregated data for a given PV system for all years since installation. The data is returned as a JSON object. The data is separated by years. Filters allow limiting to specific PV system energy values.

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv-system-id}/ aggdata/years/ {year}	GetSystem AggregatedDataSpecificYear	Gets annual aggregated data for a given PV system for a specific <u>year</u> . The data is returned as a JSON object. Filters allow limiting to specific PV system energy values.
GET	swqapi/ pvsystems/{pv-system-id}/ aggdata/years/ {year}/months	GetSystem AggregatedDataMonths	Gets monthly aggregated data for a given PV system for a specific year. The data is returned as a JSON object. The data is separated by months. Filters allow limiting to specific PV system energy values.
GET	swqapi/ pvsystems/{pv-system-id}/ aggdata/years/ {year}/months/ {month}	GetSystem AggregatedDataSpecificMonth	Gets monthly aggregated data for a given PV system for a specific month of a year. The data is returned as a JSON object. Filters allow limiting to specific PV system energy values.
GET	swqapi/ pvsystems/{pv-system-id}/ aggdata/years/ {year}/months/ {month}/days	GetSystem AggregatedDataDays	Gets daily aggregated data for a given PV system for a specific month of a year. The data is returned as a JSON object. The data is separated by days. Filters allow limiting to specific PV system energy values.
GET	swqapi/ pvsystems/{pv-system-id}/ aggdata/years/ {year}/months/ {month}/days/ {day}	GetSystem AggregatedDataSpecificDay	Gets daily aggregated data for a given PV system for a specific date. The data is returned as a JSON object. Filters allow limiting to specific PV system energy values.

#### Filters and parameters

Filter	Description
?channel=<channel>	One of the detail data channels.

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata
// get aggregated total energy flow values lifetime of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years
// get aggregated annual energy flow values for lifetime of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017
// get the aggregated annual energy flow values for 2017 of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017/months
// get the aggregated monthly energy flow values for 2017 of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017/months/12
// get the aggregated monthly energy flow values for December 2017 of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017/months/12/days
// get the aggregated daily energy flow values for December 2017 of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017/months/12/days/24
// get the aggregated daily energy flow values for December 24, 2017 of this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017/months/12?channel=EnergyFeedIn
// get the aggregated monthly energy flow values to the grid for December 2017 of
this system

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggdata/
years/2017/months/12?channel=EnergyBattCharge,EnergyBattDischarge
// get the aggregated monthly energy flow values from and to batteries for December
2017 of this system
```

## Response objects

JSON object answer construction:

## Objects

```

/ pvSystemId (String)
/ data
  / logPeriod (Object; optional)
    / type (String, "total", "year", "month", or "day")
    / year (Number, required for month/day resolution)
    / month (Number, required for day resolution)
  / channels (Array)
    / channelName (String)
    / channelType (String)
    / unit (String)
    / values (Array)
    / total or year or month or day (Number)

```

Supported value channels:

Type	Channels	Comment
PV system energy flow with Smart Meter	/ EnergyFeedIn / EnergyPurchased / EnergySelfConsumption / EnergyDirectConsumption	Requires Smart Meter; otherwise NULL
	/ EnergyBattCharge / EnergyBattDischarge / EnergyBattChargeGrid / EnergyBattDischargeGrid	Requires Smart Meter and battery; otherwise NULL
	/ OhmpilotEnergy	Requires Smart Meter and Ohmpilot; otherwise NULL
PV system energy flow without Smart Meter	/ EnergyOutput	Only if there is no Smart Meter; NULL with Smart Meter
PV system totals	/ EnergyProductionTotal / EnergyConsumptionTotal / EnergySelfConsumptionTotal	Requires Smart Meter; otherwise NULL
PV system CO2 savings	/ SavingsCO2 / SavingsTrees / SavingsTravelCar / SavingsTravelPlane	
PV system savings	/ Profits / Earnings / Savings	Requires profit settings in Solar.web; otherwise NULL

## Example responses

#### Example for retrieving multiple channels for all years

```
{
  "pvSystemId":
"a2954a1c-2157-4c03-936d-a0a900cd9a2e",
  "data": {
    "logPeriod": {
      "type": "year"
    },
    "channels": [
      {
        "channelName":
"EnergyBattDischarge",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
          "2016":
3336178.6281,
          "2017":
2096504.6716,
          "2018": 1408162.612,
          "2019": 2099497.054,
          "2020": 755442.1511
        }
      },
      {
        "channelName":
"EnergyBattDischargeGrid",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
          "2016": 50105.5535,
          "2017": 21565.7363,
          "2018": 12622.5136,
          "2019": 10993.7361,
          "2020": 18829.2124
        }
      },
      {
        "channelName":
"EnergyBattCharge",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
          "2016":
3953241.3575,
          "2017":
2320728.3568,
          "2018":
1573378.3272,
          "2019":
2332528.6707,
```

#### Example for retrieving EnergyBattDischarge channel for total lifetime

```
{
  "pvSystemId": "5b72b205-b698-4243-a68a-a39200e0e9d8",
  "data": {
    "logPeriod": {
      "type": "total"
    },
    "channels": [
      {
        "channelName":
"EnergyBattDischarge",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
          "total": 8041797.7611
        }
      }
    ]
  }
}
```

#### Example for retrieving EnergyBattDischarge channel for November 2018 only

```
{
  "pvSystemId": "5b72b205-b698-4243-a68a-a39200e0e9d8",
  "data": {
    "logPeriod": {
      "type": "month",
      "year": 2018
    },
    "channels": [
      {
        "channelName":
"EnergyBattDischarge",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
          "11": 125526.5462
        }
      }
    ]
  }
}
```

```

        "2020": 846697.4328
    },
    {
        "channelName":
"EnergySelfConsumption",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
            "2016":
9511044.7487,
            "2017":
4959573.6777,
            "2018": 3118663.988,
            "2019":
4638646.3376,
            "2020": 501854.6066
        }
    },
    {
        "channelName":
"EnergyFeedIn",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
            "2016":
13549688.2916,
            "2017":
7356610.8996,
            "2018": 3903633.424,
            "2019":
5361550.4367,
            "2020": 2124059.859
        }
    },
    {
        "channelName":
"EnergyOutput",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
            "2016": 10546.7175,
            "2017": 8544.4002,
            "2018": 4258.0244,
            "2019": 4290.4598,
            "2020": 11693.1881
        }
    },
    {
        "channelName":
"EnergyBattChargeGrid",
        "channelType": "Energy",
        "unit": "Wh",
        "values": {
            "2016": 13190.58,

```

#### Example for retrieving EnergyBattDischarge channel for November 2018, all days

```

{
    "pvSystemId": "5b72b205-b698-4243-a68a-a39200e0e9d8",
    "data": {
        "logPeriod": {
            "type": "day",
            "year": 2018,
            "month": 11
        },
        "channels": [
            {
                "channelName":
"EnergyBattDischarge",
                "channelType": "Energy",
                "unit": "Wh",
                "values": {
                    "1": 9651.855,
                    "2": 5249.8219,
                    "3": 273.6811,
                    "4": 0.3194,
                    "5": 8590.9661,
                    "6": 8623.4647,
                    "7": 8445.7725,
                    "8": 5218.5789,
                    "9": 187.8033,
                    "10": 110.0903,
                    "11": 8196.3916,
                    "12": 105.9322,
                    "13": 5691.5314,
                    "14": 10195.6844,
                    "15": 8735.5328,
                    "16": 8734.9135,
                    "17": 8397.3319,
                    "18": 8519.988,
                    "19": 0.1819,
                    "20": 8873.1828,
                    "21": 0.2619,
                    "22": 397.4644,
                    "23": 0.0828,
                    "24": 472.7528,
                    "25": 0,
                    "26": 0,
                    "27": 0,
                    "28": 2105.0531,
                    "29": 8617.0336,
                    "30": 130.8739
                }
            }
        ]
    }
}

```

```

        "2017": 29427.9855,
        "2018": 488.944,
        "2019": 1037.4809,
        "2020": 8591.4541
    }
},
{
    "channelName":
"EnergyPurchased",
    "channelType": "Energy",
    "unit": "Wh",
    "values": {
        "2016":
4655590.4605,
        "2017":
4249585.2134,
        "2018":
4539096.3551,
        "2019":
5735676.3166,
        "2020":
1346258.6173
    }
},
{
    "channelName":
"C02Savings",
    "channelType": "C02
savings",
    "unit": "kg",
    "values": {
        "2016": 138365.55,
        "2017": 74984.74,
        "2018": 44031.66,
        "2019": 63165.52,
        "2020": 17839.64
    }
},
{
    "channelName":
"SavingsTrees",
    "channelType": "C02
savings",
    "unit": "tree",
    "values": {
        "2016": 3547.83,
        "2017": 1922.69,
        "2018": 1129.02,
        "2019": 1619.63,
        "2020": 457.43
    }
},
{
    "channelName":
"SavingsTravelCar",

```

```

    }
}

```

```

    "channelType": "C02
savings",
    "unit": "km",
    "values": {
        "2016": 922437.0,
        "2017": 499898.27,
        "2018": 293544.4,
        "2019": 421103.47,
        "2020": 118930.93
    }
},
{
    "channelName":
"SavingsTravelPlane",
    "channelType": "C02
savings",
    "unit": "mile",
    "values": {
        "2016": 461218.5,
        "2017": 249949.13,
        "2018": 146772.2,
        "2019": 210551.73,
        "2020": 59465.47
    }
},
{
    "channelName":
"Profits",
    "channelType":
"Currency",
    "unit": "EUR",
    "values": {
        "2016": 2021.4378,
        "2017": 1921.2410,
        "2018": 1181.4323,
        "2019": 1722.3882,
        "2020": 433.3368
    }
},
{
    "channelName":
"Earnings",
    "channelType":
"Currency",
    "unit": "EUR",
    "values": {
        "2016": 529.7265,
        "2017": 516.6538,
        "2018": 286.0457,
        "2019": 376.1888,
        "2020": 5.4703
    }
},
{

```



```

    "channelName":
    "Savings",
    "channelType":
    "Currency",
    "unit": "EUR",
    "values": {
      "2016": 1491.7113,
      "2017": 1404.5872,
      "2018": 895.3866,
      "2019": 1346.1994,
      "2020": 427.8665
    }
  }
]
}

```

#### 4.4.2 Aggrdata: Aggregated energy production for a PV system

##### Use cases for web developers

- / I want to show total/lifetime aggregated energy values to an end user.
- / I want to show annually, monthly or daily aggregated energy values to an end user.
- / I want to show aggregated energy values to an end user, but for custom time periods (e.g. a week or last 6 months).

##### Methods

Method	End point and objects	Event name	Description
GET	swqapi/pvsystems/{pv-system-id}/aggrdata	GetSystemAggregate dData	Gets aggregated data for a given PV system for a custom period of time. The custom period can either span years, months, or days. The data is returned as a JSON object. Filters allow limiting to specific PV system energy values.

##### Filters and parameters

Filter	Description
?channel=<channel>	One or more of the detail data channels.  Channel filters can be concatenated using commas. E.g.: ?channel=EnergyFeedIn,EnergyPurchased

Filter	Description
?from=<start>&to=<end>	<p>Limits the time series for the query.</p> <p>Period type in &lt;from&gt; and &lt;to&gt; need to match (i.e. both need to be either years, months, or days).</p> <p>Encoding:</p> <ul style="list-style-type: none"> <li>/ / yyyy for years</li> <li>/ / yyyy-MM or yyyyMM for months</li> <li>/ / yyyy-MM-dd or yyyyMMdd for days</li> </ul>
?from=<start>&duration=<length>	<p>Limits the time series for the query.</p> <p>&lt;Duration&gt; is measured in years, months or days - depending on the &lt;from&gt; parameter.</p> <p>A &lt;duration&gt; of 1 means that start and end are equal.</p> <p>Encoding:</p> <ul style="list-style-type: none"> <li>/ / yyyy for years</li> <li>/ / yyyy-MM or yyyyMM for months</li> <li>/ / yyyy-MM-dd or yyyyMMdd for days</li> </ul>
?period=<period>	<p>Limits the time series for the query to a period.</p> <p>Encoding:</p> <ul style="list-style-type: none"> <li>/ / "total" delivers total values over whole system lifetime</li> <li>/ / "years" delivers values for each year of system lifetime</li> <li>/ / yyyy delivers all months of requested year</li> <li>/ / yyyy-MM delivers all days of requested month</li> </ul> <p>Note: If &lt;period&gt; parameter is used, than &lt;from&gt;, &lt;to&gt; and &lt;duration&gt; parameters are now allowed, and vice versa.</p>
?offset=<offset>&limit=<limit>	<p>Supports pagination, returns items from a starting &lt;offset&gt; and returning not more than &lt;limit&gt; items (items = objects in the "Data" array).</p>

## Example calls

```
GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=total
    // get aggregated total energy flow values of this system for total lifetime

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=years
    // get aggregated annual energy flow values of this system for all years since the
    installation

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017&duration=1
    // get the aggregated annual energy flow values of this system for 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=2017
    // get the aggregated monthly energy flow values of this system for 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12&duration=1
    // get the aggregated monthly energy flow values of this system for December 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=2017-12
    // get the aggregated daily energy flow values of this system for December 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12-01&to=2017-12-31
    // get the aggregated daily energy flow values of this system for December 2017
    (alternative to above)

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12-01&duration=31
    // get the aggregated daily energy flow values of this system for December 2017
    (alternative to above)

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12-24&duration=1
    // get the aggregated daily energy flow values of this system for December 24, 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12-24&duration=7
    // get the aggregated daily energy flow values of this system for the week December
    24-30, 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12&duration=1&channel=EnergyFeedIn
    // get the EnergyFeedIn value of this system for December 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12&duration=1&channel=EnergyBattCharge,EnergyBattDischarge
```

```
// get the EnergyBattCharge and EnergyBattDischarge values of this system for
December 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-01-01&duration=365&channel=EnergyFeedIn&limit=7
// get the EnergyFeedIn values of this system for all days of the year 2017, in
weekly pages
```

#### Response objects

JSON object answer construction:

##### Objects

```
/ pvSystemID
/ data
  / logDate (String - date information like "yyyy", "yyyy-MM" or "yyyy-MM-dd", or "total")
  / channels (Array)
    / channelName (String)
    / channelType (String)
    / unit (String)
    / value (Number)
```

Supported value channels

Type	Channels	Comment
PV system energy flow with Smart Meter	/ EnergyFeedIn / EnergyPurchased / EnergySelfConsumption / EnergyDirectConsumption	Requires Smart Meter; otherwise NULL
	/ EnergyBattCharge / EnergyBattDischarge / EnergyBattChargeGrid / EnergyBattDischargeGrid	Requires Smart Meter and battery; otherwise NULL
	/ OhmpilotEnergy	Requires Smart Meter and Ohmpilot; otherwise NULL
PV system energy flow without Smart Meter	/ EnergyOutput	Only if there is no Smart Meter; NULL with Smart Meter
PV system totals	/ EnergyProductionTotal / EnergyConsumptionTotal / EnergySelfConsumptionTotal	Requires Smart Meter; otherwise NULL
PV system CO2 savings	/ SavingsCO2 / SavingsTrees / SavingsTravelCar / SavingsTravelPlane	
PV system savings	/ Profits / Earnings / Savings	Requires profit settings in Solar.web; otherwise NULL

#### Example responses

### Example for retrieving all channels for total lifetime

```
{
  "pvSystemId":
  "20bb600e-019b-4e03-9df3-a0a900cda689",
  "data": [
    {
      "logDateTime": "total",
      "channels": [
        {
          "channelName":
          "SavingsCO2",
          "channelType": "CO2
          savings",
          "unit": "kg",
          "value": 539552.54
        },
        {
          "channelName":
          "SavingsTrees",
          "channelType": "CO2
          savings",
          "unit": "tree",
          "value": 13834.68
        },
        {
          "channelName":
          "SavingsTravelCar",
          "channelType": "CO2
          savings",
          "unit": "km",
          "value": 3597016.94
        },
        {
          "channelName":
          "SavingsTravelPlane",
          "channelType": "CO2
          savings",
          "unit": "mile",
          "value": 1798508.46
        },
        {
          "channelName":
          "Profits",
          "channelType":
          "Currency",
          "unit": "EUR",
          "value": 11616.8943
        },
        {
          "channelName":
          "Earnings",
```

### Example for retrieving the EnergySelfConsumption channel for multiple years

```
{
  "pvSystemId":
  "20bb600e-019b-4e03-9df3-a0a900cda689",
  "data": [
    {
      "logDateTime": "2014",
      "channels": [
        {
          "channelName":
          "EnergySelfConsumption",
          "channelType":
          "Energy",
          "unit": "Wh",
          "value":
          4710304.1779
        }
      ]
    },
    {
      "logDateTime": "2015",
      "channels": [
        {
          "channelName":
          "EnergySelfConsumption",
          "channelType":
          "Energy",
          "unit": "Wh",
          "value":
          5678721.3691
        }
      ]
    },
    {
      "logDateTime": "2016",
      "channels": [
        {
          "channelName":
          "EnergySelfConsumption",
          "channelType":
          "Energy",
          "unit": "Wh",
          "value":
          9511044.7487
        }
      ]
    }
  ]
}
```

```

        "channelType":
"Currency",
        "unit": "EUR",
        "value": 14276.4026
    },
    {
        "channelName":
"Savings",
        "channelType":
"Currency",
        "unit": "EUR",
        "value": 8985.9121
    },
    {
        "channelName":
"EnergyOutput",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
10880386.5320
    },
    {
        "channelName":
"EnergyBattDischarge",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
12869207.6682
    },
    {
        "channelName":
"EnergyBattDischargeGrid",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
143905.5845
    },
    {
        "channelName":
"EnergyBattCharge",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
14609870.0412
    },
    {
        "channelName":
"EnergySelfConsumption",
        "channelType":
"Energy",
        "unit": "Wh",

```

#### Example for retrieving the EnergySelfConsumption channel for a week

```

{
    "pvSystemId":
"20bb600e-019b-4e03-9df3-a0a900cda689",
    "data": [
        {
            "logDateTime": "2020-06-29",
            "channels": [
                {
                    "channelName":
"EnergySelfConsumption",
                    "channelType":
"Energy",
                    "unit": "Wh",
                    "value": 24612.0087
                }
            ]
        },
        {
            "logDateTime": "2020-06-30",
            "channels": [
                {
                    "channelName":
"EnergySelfConsumption",
                    "channelType":
"Energy",
                    "unit": "Wh",
                    "value": 23861.9769
                }
            ]
        },
        {
            "logDateTime": "2020-07-01",
            "channels": [
                {
                    "channelName":
"EnergySelfConsumption",
                    "channelType":
"Energy",
                    "unit": "Wh",
                    "value": 23267.4185
                }
            ]
        },
        {
            "logDateTime": "2020-07-02",
            "channels": [
                {
                    "channelName":
"EnergySelfConsumption",

```

```

        "value":
35305418.9650
      },
      {
        "channelName":
"EnergyFeedIn",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
44585679.6562
      },
      {
        "channelName":
"EnergyBattChargeGrid",
        "channelType":
"Energy",
        "unit": "Wh",
        "value": 69769.6105
      },
      {
        "channelName":
"EnergyPurchased",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
27991985.4892
      },
      {
        "channelName":
"EnergyProductionTotal",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
94500968.6624
      },
      {
        "channelName":
"EnergySelfConsumptionTotal",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
49915289.0062
      },
      {
        "channelName":
"EnergyConsumptionTotal",
        "channelType":
"Energy",
        "unit": "Wh",
        "value":
77907274.4954

```

```

        "channelType":
"Energy",
        "unit": "Wh",
        "value": 28923.5102
      }
    ]
  },
  {
    "logDateTime": "2020-07-03",
    "channels": [
      {
        "channelName":
"EnergySelfConsumption",
        "channelType":
"Energy",
        "unit": "Wh",
        "value": 30986.6525
      }
    ]
  },
  {
    "logDateTime": "2020-07-04",
    "channels": [
      {
        "channelName":
"EnergySelfConsumption",
        "channelType":
"Energy",
        "unit": "Wh",
        "value": 18682.0911
      }
    ]
  },
  {
    "logDateTime": "2020-07-05",
    "channels": [
      {
        "channelName":
"EnergySelfConsumption",
        "channelType":
"Energy",
        "unit": "Wh",
        "value": 20237.0875
      }
    ]
  }
]
}

```



```

    }
  ]
}

```

#### 4.4.3 Aggrdata: Aggregated energy production for a device

##### Use cases for web developers

- / I want to show an inverter's total/lifetime aggregated energy values to an end user.
- / I want to show an inverter's annually, monthly or daily aggregated energy values to an end user.
- / I want to show an inverter's aggregated energy values to an end user, but for custom time periods (e.g. a week or last 6 months).

##### Methods

Method	End point and objects	Event name	Description
GET	swqapi/pvsystems/{pv-system-id}/devices/{device-id}/aggrdata	GetDeviceAggregated Data	Gets aggregated data for a given device of a given PV system for a custom period of time. The custom period can either span years, months, or days. The data is returned as a JSON object.

##### Filters and parameters

Filter	Description
?channel=<channel>	One or more of the detail data channels.
?from=<start>&to=<end>	Limits the time series for the query. Period type in <from> and <to> need to match (i.e. both need to be either years, months, or days). Encoding: /     / yyyy for years / yyyy-MM or yyyyMM for months / yyyy-MM-dd or yyyyMMdd for days

Filter	Description
?from=<start>&duration=<length>	<p>Limits the time series for the query.</p> <p>&lt;Duration&gt; is measured in years, months or days - depending on the &lt;from&gt; parameter.</p> <p>A &lt;duration&gt; of 1 means that start and end are equal.</p> <p>Encoding:</p> <ul style="list-style-type: none"> <li>/ / yyyy for years</li> <li>/ / yyyy-MM or yyyyMM for months</li> <li>/ / yyyy-MM-dd or yyyyMMdd for days</li> </ul>
?period=<period>	<p>Limits the time series for the query to a period.</p> <p>Encoding:</p> <ul style="list-style-type: none"> <li>/ / "total" delivers total values over whole system lifetime</li> <li>/ / "years" delivers values for each year of system lifetime</li> <li>/ / yyyy delivers all months of requested year</li> <li>/ / yyyy-MM delivers all days of requested month</li> </ul> <p>Note: If &lt;period&gt; parameter is used, than &lt;from&gt;, &lt;to&gt; and &lt;duration&gt; parameters are now allowed, and vice versa.</p>
?offset=<offset>&limit=<limit>	<p>Supports pagination, returns items from a starting &lt;offset&gt; and returning not more than &lt;limit&gt; items (items = objects in the "Data" array).</p>

## Example calls

```
GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2010&to=2015
    // get aggregated annual energy values of this device for the years 2010 to 2015

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2020-02-24&duration=7
    // get aggregated daily energy values of this device for the week of February 24th
to March 1st, 2020

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?period=total
    // get aggregated energy values of this device for its total lifetime

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?period=years
    // get aggregated annual energy values of this device for all years since the
installation

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017&duration=1
    // get aggregated annual energy values of this device for the year 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?period=2017
    // get aggregated monthly energy values of this device for 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017-12&duration=1
    // get aggregated monthly energy values of this device for the month December 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?period=2017-12
    // get aggregated daily energy values of this device for December 2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017-12-01&to=2017-12-31
    // get aggregated daily energy values of this device for December 2017 (alternative
to above)

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017-12-01&duration=31
    // get aggregated daily energy values of this device for December 2017 (alternative
to above)

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017-12-24&duration=1
    // get aggregated daily energy values of this device for the day of December 24,
2017
```

```
GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017-12-24&duration=7
// get aggregated daily energy values of this device for the week December 24-30,
2017

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
b582f1b9-95b9-49db-800b-6b042e9938b4/aggrdata?from=2017-12&duration=1
// get aggregated energy values of this device for the month of December 2017
```

#### Response objects

JSON object answer construction:

##### Objects

```
/ pvSystemID
/ deviceID
/ data
  / logDate (String - date information like "yyyy", "yyyy-MM" or "yyyy-MM-dd", or "total")
  / channels (Array)
    / channelName (String)
    / channelType (String)
    / unit (String)
    / value (Number)
```

Supported channels:

Type	Channels
Inverter	<pre>/ EnergyExported / EnergyDC1 / EnergyDC2</pre>

## Example responses

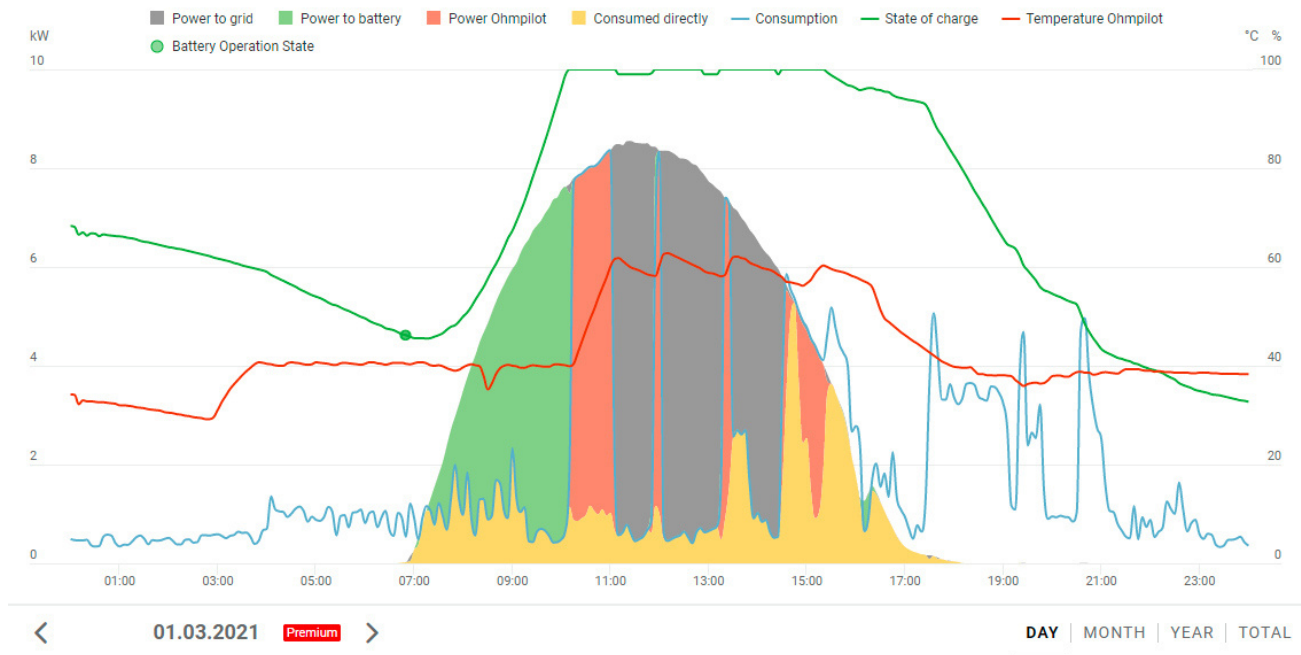
```
{
  "pvSystemId": "20bb600e-019b-4e03-9df3-a0a900cda689",
  "deviceId": "b582f1b9-95b9-49db-800b-6b042e9938b4",
  "data": [
    {
      "logDateTime": "total",
      "channels": [
        {
          "channelName": "EnergyExported",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 32154.3471
        },
        {
          "channelName": "EnergyDC1",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 155055.7984
        },
        {
          "channelName": "EnergyDC2",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0.0
        }
      ]
    }
  ]
}
```

## 4.5 Historical data calls

This set contains methods to retrieve the historical data for a PV system or for a single device of a given PV system.

Historical data points are data points that are logged at the inverter and transferred to Solar.web at regular intervals (usually every hour). The resolution (default: 5 minutes) of these data points and the times at which the log data is transferred to Solar.web depend on the settings of the inverter (Datamanager – Settings – Fronius Solar.web).

With the historical data you can create diagrams like this one:



### 4.5.1 Histdata: Historical data for a PV system

#### Use cases for web developers

/ I want to show time series graphs for a complete PV system.

#### Restrictions

Impersonated basic users can only retrieve information not older than 72 hours - like in Solar.web.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ histdata	GetSystemHistoricalData	Gets historical data for a given PV system and time range. The data resolution is 5min. The data is returned a JSON object.

## Filters and parameters

Filter	Description
?channel=<channel>	One of the detail data channels from inverter, sensor, battery, Smart Meter, Ohmpilot, or general type categories.
?timezone=<"local", "zulu">	Specifies time format in response object:  / zulu (default): returns time in UTC zulu time. / local: returns time in PV system's local UTC time (local time + UTC offset).
?from=<start>&to=<end>	Limits the time series for the query.  / <start> and <end> are ISO-8601 time values. / Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?offset=<offset>&limit=<limit>	Supports pagination, returns items from a starting <offset> and returning not more than <limit> items (items = objects in the "Data" array).

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/histdata?  
from=2018-10-10T00:00:00Z&to=2018-10-11T00:00:00Z  
// gets all historical values for 10th of October, 2018, for PV system
```

## Response objects

JSON object answer construction:

Objects
<pre>/ pvSystemId (String) / data (Array)   / logDateTime (UTC Time)   / logDuration (Integer - unit: seconds)   / channels (Array)     / channelName (String)     / channelType (String)     / unit (String)     / value (Number, String)</pre>

Supported value channels:

Type	Objects	Remarks
PV system power flow with Smart Meter	/ EnergyFeedIn / EnergyPurchased / EnergySelfConsumption	Requires Smart Meter; otherwise NULL
	/ EnergyBattCharge / EnergyBattChargeGrid / EnergyBattDischarge / EnergyBattDischargeGrid	Requires Smart Meter and battery; otherwise NULL
PV system power flow without Smart Meter	/ EnergyOutput	Only if there is no Smart Meter; NULL with Smart Meter
PV system totals	/ EnergyProductionTotal / EnergyConsumptionTotal / EnergySelfConsumptionTotal	



## Example responses

```
{
  "pvSystemId": "85896da3-bb2a-47f7-9c6e-2909dd44832c",
  "data": [
    {
      "logDateTime": "2019-07-18T19:00:00Z",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "EnergySelfConsumption",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 90.98
        },
        {
          "channelName": "EnergyFeedIn",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 127
        },
        {
          "channelName": "EnergyBattCharge",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0
        },
        {
          "channelName": "EnergyBattDischargeGrid",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0
        },
        {
          "channelName": "EnergyBattDischarge",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0
        },
        {
          "channelName": "EnergyPurchased",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0
        },
        {
          "channelName": "EnergyBattChargeGrid",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0
        },
        {
          "channelName": "EnergyOutput",
          "channelType": "Energy",
          "unit": "Wh",

```

```

        "value": NULL
    },
    {
        "channelName": "OhmPilotTemp",
        "channelType": "Temperature",
        "unit": "°C",
        "value": 64.4
    },
    {
        "channelName": "OhmPilotEnergy",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 69
    }
  ]
}

```

## 4.5.2 Histdata: Historical data for a device

### Use cases for web developers

- / I want to show time series graphs for a PV system, but only for a certain type or device (inverters, sensors, batteries, smartmeters, ohmpilots).

### Restrictions

Impersonated basic users can only retrieve information not older than 72 hours - like in Solar.web.

### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems{pv- system-id}/ devices/{device- id}/histdata	GetDeviceHistoricalData	Gets historical data for a given PV device of a given PV system and time range. The data resolution is 5 min. The data is returned a JSON object.

### Filters and parameters

Filter	Description
?channel=<channel>	One of the detail data channels.

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: / zulu (default): returns time in UTC zulu time. / local: returns time in PV system's local UTC time (local time + UTC offset).
?from=<start>&to=<end>	Limits the time series for the query. / <start> and <end> are ISO-8601 time values. / Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?offset=<offset>&limit=<limit>	Supports pagination, returns items from a starting <offset> and returning not more than <limit> items (items = objects in the "Data" array).

#### Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
d2e61bf2-8dd7-4ba1-8733-d55d738c4679/histdata?from=2018-10-10T00:00:00Z&to=2018-10-11
T00:00:00Z
// gets all historical values for 10th of October, 2018, for given device

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
d2e61bf2-8dd7-4ba1-8733-d55d738c4679/histdata?
from=20181010T000000Z&to=20181011T000000Z?channel=Temp1
// gets all historical temperature values (Temp1 channel) for the 10th of October,
2018, for given device
```

#### Response objects

JSON object answer construction:

Objects
/ pvSystemId (String) / deviceId (String) / data (Array) / logDateTime (UTC Time) / logDuration (Integer - unit: seconds) / channels (Array) / channelName (String) / channelType (String) / unit (String) / value (Number or String, depending on channel)

Supported channels:

Type	Channels
Inverter	/ EnergyExported / EnergyImported / EnergyDC1 / EnergyDC2 / CurrentA / CurrentB / CurrentC / CurrentDC1 / CurrentDC2 / VoltageA / VoltageB / VoltageC / VoltageAB / VoltageBC / VoltageCA / VoltageDC1 / VoltageDC2 / ApparentPower / ReactivePower / PowerFactor / StandardizedPower
Type	Channels
Sensor	/ Temp1 / Temp2 / Insolation / Digital1 / Digital2 / Digital3 / Digital1Energy / Digital2Energy / Digital3Energy
Battery	/ BattOpState / BattSOC
Ohmpilot	/ OhmpilotTemp / OhmpilotEnergyAbs / OhmpilotEnergy / OhmpilotError

Type	Channels
Smart Meter	/ GridPowerA / GridPowerB / GridPowerC / GridApparentPowerA / GridApparentPowerB / GridApparantPowerC / GridVoltageA / GridVoltageB / GridVoltageC / LoadPowerA / LoadPowerB / LoadPowerC / LoadApparentPowerA / LoadApparentPowerB / LoadApparentPowerC / LoadVoltageA / LoadVoltageB / LoadVoltageC / ExtEnergyExportedAbs / ExtEnergyExported / EnergyLoadAbs / EnergyLoad / GridEnergyExportedAbs / GridEnergyImportedAbs / GridEnergyExported / GridEnergyImported

#### Example responses

## Inverter

```
{
  "pvSystemId": "85896da3-
bb2a-47f7-9c6e-2909dd44832c",
  "deviceId": "ea5e207e-84c0-49fd-a9e0-
ed7234a84c63",
  "data": [
    {
      "logDateTime":
"2019-07-31T12:00:00Z",
      "logDuration": 300,
      "channels": [
        {
          "channelName":
"GridEnergyExported",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 313.11
        },
        {
          "channelName":
"StandardizedPower",
          "channelType": "Percentage",
          "unit": "%",
          "value": 1.3
        },
        {
          "channelName": "VoltageA",
          "channelType": "Voltage",
          "unit": "V",
          "value": 236.7
        },
        {
          "channelName": "VoltageB",
          "channelType": "Voltage",
          "unit": "V",
          "value": 226.1
        },
        {
          "channelName": "VoltageC",
          "channelType": "Voltage",
          "unit": "V",
          "value": 238.5
        },
        {
          "channelName": "CurrentA",
          "channelType": "Current",
          "unit": "A",
          "value": 5.37
        },
        {
          "channelName": "CurrentB",
```

## Smart Meter

```
{
  "pvSystemId": "85896da3-
bb2a-47f7-9c6e-2909dd44832c",
  "deviceId": "f2adce80-e9f2-43cb-
b6ae-26ab2e39cd8f",
  "data": [
    {
      "logDateTime":
"2019-07-31T12:00:00Z",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "GridPowerA",
          "channelType": "Power",
          "unit": "W",
          "value": -1323.92
        },
        {
          "channelName": "GridPowerB",
          "channelType": "Power",
          "unit": "W",
          "value": 220.4
        },
        {
          "channelName": "GridPowerC",
          "channelType": "Power",
          "unit": "W",
          "value": -2384.07
        },
        {
          "channelName":
"GridApparentPowerA",
          "channelType": "Apparent
Power",
          "unit": "VA",
          "value": 1364.11
        },
        {
          "channelName":
"GridApparentPowerB",
          "channelType": "Apparent
Power",
          "unit": "VA",
          "value": 1052.78
        },
        {
          "channelName":
"GridApparentPowerC",
          "channelType": "Apparent
Power",
          "unit": "VA",
```

```

        "channelType": "Current",
        "unit": "A",
        "value": 5.27
    },
    {
        "channelName": "CurrentC",
        "channelType": "Current",
        "unit": "A",
        "value": 5.38
    },
    {
        "channelName": "VoltageDC1",
        "channelType": "Voltage",
        "unit": "V",
        "value": 574.7
    },
    {
        "channelName": "CurrentDC1",
        "channelType": "Current",
        "unit": "A",
        "value": 5.22
    },
    {
        "channelName": "VoltageDC2",
        "channelType": "Voltage",
        "unit": "V",
        "value": 491
    },
    {
        "channelName": "CurrentDC2",
        "channelType": "Current",
        "unit": "A",
        "value": 1.87
    },
    {
        "channelName":
"ReactivePower",
        "channelType": "Reactive
Power",
        "unit": "VAr",
        "value": -78.9
    },
    {
        "channelName":
"ApparentPower",
        "channelType": "Apparent
Power",
        "unit": "VA",
        "value": 3758.15
    },
    {
        "channelName": "PowerFactor",
        "channelType": "",
        "unit": "",
        "value": 1
    }

```

```

        "value": 2411.02
    },
    {
        "channelName": "GridVoltageA",
        "channelType": "Voltage",
        "unit": "V",
        "value": 235.66
    },
    {
        "channelName": "GridVoltageB",
        "channelType": "Voltage",
        "unit": "V",
        "value": 225.61
    },
    {
        "channelName": "GridVoltageC",
        "channelType": "Voltage",
        "unit": "V",
        "value": 237.92
    }
    ]
}
}
}

```

```

    },
    {
      "channelName": "EnergyDC1",
      "channelType": "Energy",
      "unit": "Wh",
      "value": 239.72
    },
    {
      "channelName": "EnergyDC2",
      "channelType": "Energy",
      "unit": "Wh",
      "value": 73.39
    }
  ]
}

```

### Ohmpilot

```

{
  "pvSystemId": "85896da3-
bb2a-47f7-9c6e-2909dd44832c",
  "deviceId": "17280720-2079-495d-92cb-
fa3ce2afa305",
  "data": [
    {
      "logDateTime":
"2019-07-31T12:00:00Z",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "OhmPilotTemp",
          "channelType": "Temperature",
          "unit": "°C",
          "value": 57.3
        },
        {
          "channelName":
"OhmPilotEnergy",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 85
        }
      ]
    }
  ]
}

```

## Battery

```
{
  "pvSystemId": "85896da3-
bb2a-47f7-9c6e-2909dd44832c",
  "deviceId": "83129be8-a1ec-48b1-
a8a8-7c5accd6b64e",
  "data": [
    {
      "logDateTime":
"2019-07-31T12:00:00Z",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "BattSOC",
          "channelType": "Percentage",
          "unit": "%",
          "value": 97
        }
      ]
    }
  ]
}
```

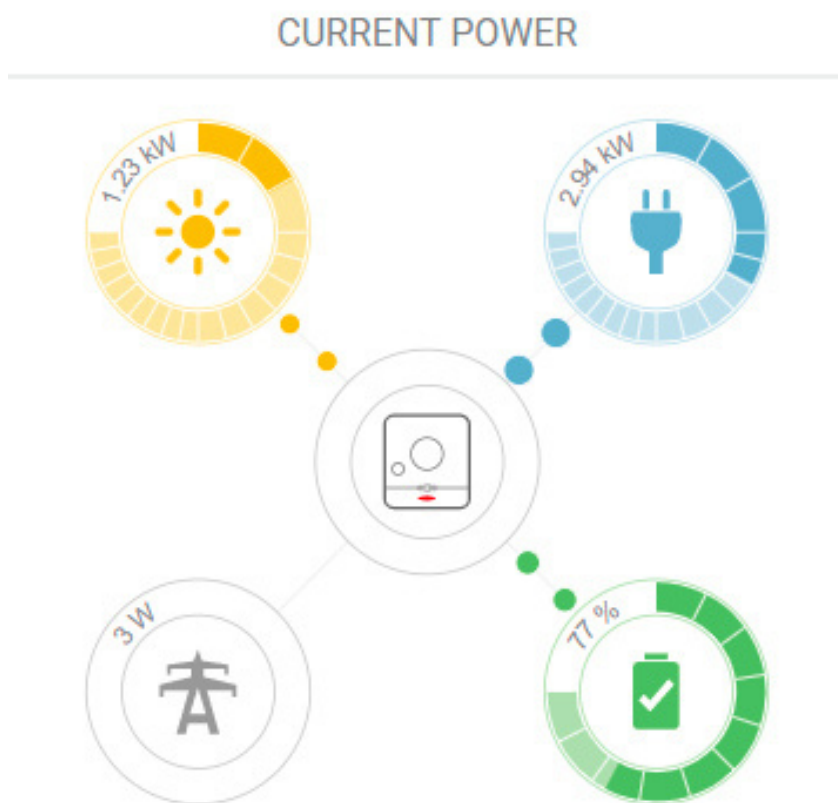


## 4.6 Realtime data calls

This set contains methods to retrieve the power flow data for a PV system or for a single device of a given PV system.

The power flow data points are real time data. Solar.web first has to collect the data from the inverter when this request is initiated: Solar.web forwards the request to the PV system which responds with the data in real time; Solar.web then passes the data on to the calling application. This data is not stored in Solar.web, it is just forwarded to the calling application. The periodic use of this call at short intervals can cause a lot of traffic and high CPU load on the inverter. It is therefore recommend to use it only rarely.

Power flows can be shown like in this diagram:



### 4.6.1 Flowdata: Realtime power flow data of a PV system

Use cases for web developers

- / I want to show current power flow for a complete PV system.

## Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ flowdata	GetSystemFlowData	Gets the realtime power flow data of a given PV system. The data is returned as a JSON object.

## Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object:  / zulu (default): returns time in UTC zulu time / local: returns time in PV system's local UTC time (local time + UTC offset)

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/flowdata
// gets current power flow information for the given PV system
```

## Response objects

JSON object answer construction:

Objects
<pre>/ pvSystemId (String) / status   / isOnline (Boolean)   / battMode (String) / data   / logDateTime (UTC time)   / channels (Array)     / channelName (String)     / channelType (String)     / unit (String)     / value (Number, String)</pre>

Supported value channels:

Type	Channels	Remarks
PV system	<ul style="list-style-type: none"> <li>/ PowerFeedIn</li> <li>/ PowerLoad</li> <li>/ PowerBattCharge</li> <li>/ PowerOhmpilot</li> <li>/ PowerPV</li> <li>/ PowerOutput</li> <li>/ BattSOC</li> <li>/ RateSelfConsumption</li> <li>/ RateSelfSufficiency</li> </ul>	<p>If a Smart Meter is attached, it provides the values for PowerFeedIn, PowerLoad, PowerBattCharge, PowerOhmpilot, PowerPV. PowerOutput will return NULL in this case.</p> <p>If there is no Smart Meter, the inverter returns PowerOutput instead.</p>

## Example responses

```
{
  "pvSystemId": "04d81b82-7861-4e36-8e7f-41036ce711a4",
  "status": {
    "isOnline": true,
    "battMode": "Normal"
  },
  "data": {
    "logDateTime": "2019-06-18T14:01:57Z",
    "channels": [
      {
        "channelName": "PowerFeedIn",
        "channelType": "Power",
        "unit": "W",
        "value": -596.01
      },
      {
        "channelName": "PowerLoad",
        "channelType": "Power",
        "unit": "W",
        "value": -1086.89
      },
      {
        "channelName": "PowerBattCharge",
        "channelType": "Power",
        "unit": "W",
        "value": 0
      },
      {
        "channelName": "PowerPV",
        "channelType": "Power",
        "unit": "W",
        "value": 1682.9
      },
      {
        "channelName": "PowerOhmpilot",
        "channelType": "Power",
        "unit": "W",
        "value": null
      },
      {
        "channelName": "BattSOC",
        "channelType": "Percent",
        "unit": "%",
        "value": 99
      },
      {
        "channelName": "RateSelfSufficiency",
        "channelType": "Percent",
        "unit": "%",
        "value": 100
      },
      {
        "channelName": "RateSelfConsumption",
```

```

        "channelType": "Percent",
        "unit": "%",
        "value": 64.58
    }
  ]
}

```

## 4.6.2 Flowdata: Realtime power flow data of a device

### Use cases for web developers

/ I want to show current power flow for a a specific device.

### Methods

Method	End point and objects	Event name	Description
GET	swqapi/pvsystems/{pv-system-id}/devices/{device-id}/flowdata	GetDeviceFlowData	Gets the real-time power flow data of the requested PV device of a PV system with the given ID. The data is returned as a JSON object.

### Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: / zulu (default): returns time in UTC zulu time / local: returns time in PV system's local UTC time (local time + UTC offset)

### Example calls

```

GET api.solarweb.com/swqapi/pv-systems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
d2e61bf2-8dd7-4ba1-8733-d55d738c4679/flowdata
// gets current power flow information for the given device

```

### Response objects

JSON object answer construction:

Type	Objects
inverter	/ PowerOutput

Type	Objects
sensor	/ Temp1, Temp2 / Insolation / Velocity / Digital1, Digital2, Digital3
battery	/ BattSOC / BattMode
Smart Meter	/ PowerFeedIn / PowerLoad / PowerExt / PowerPurchase
Ohmpilot	/ PowerOhmpilot / OhmpilotTemp / OhmpilotState

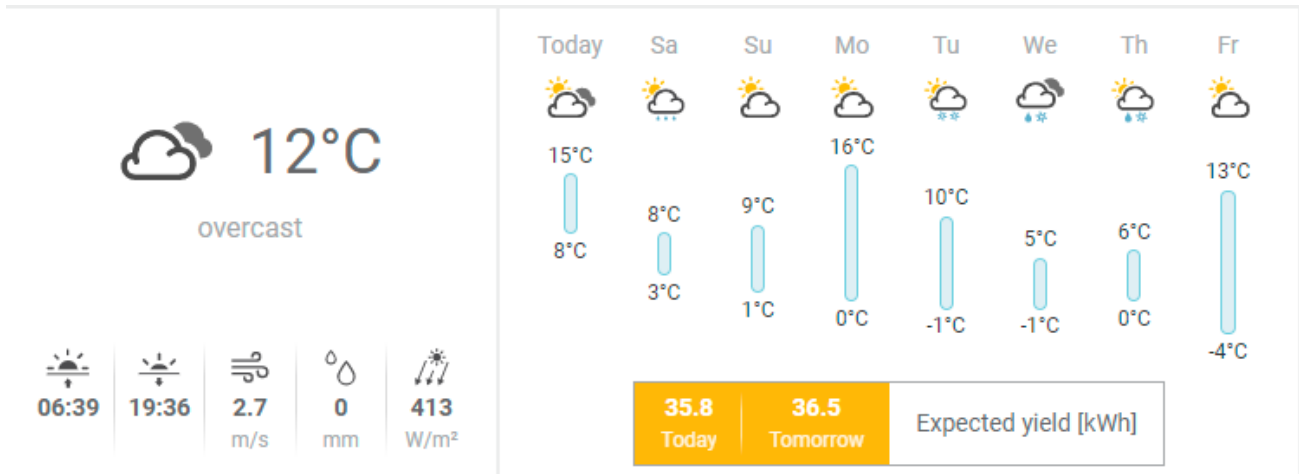
#### Example responses

```
{
  "pvSystemId": "04d81b82-7861-4e46-8e7f-41036ce711a4",
  "deviceId": "c883f93f-6661-425f-a2c5-0f381ff86c89",
  "status": {
    "isOnline": true,
    "battMode": 1.0
  },
  "data": {
    "logDateTime": "2020-03-26T14:34:20Z",
    "channels": [
      {
        "channelName": "PowerOutput",
        "channelType": "Power",
        "unit": "W",
        "value": 1041.0
      }
    ]
  }
}
```

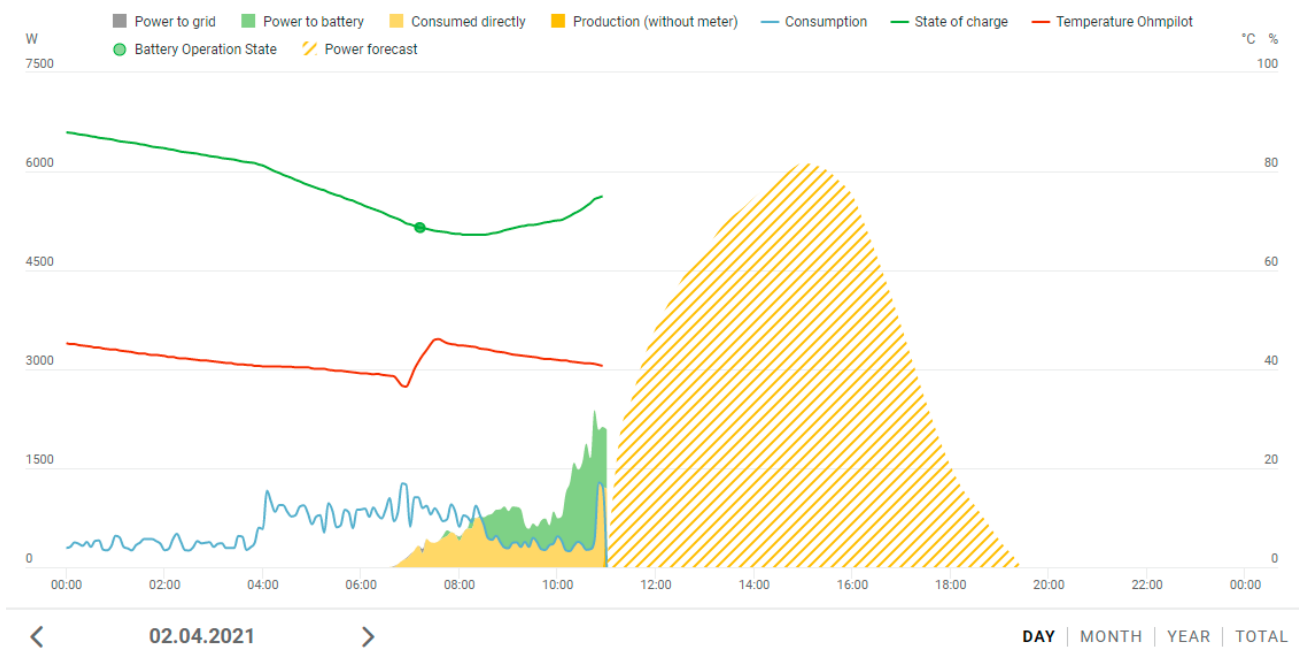
## 4.7 Weather data calls

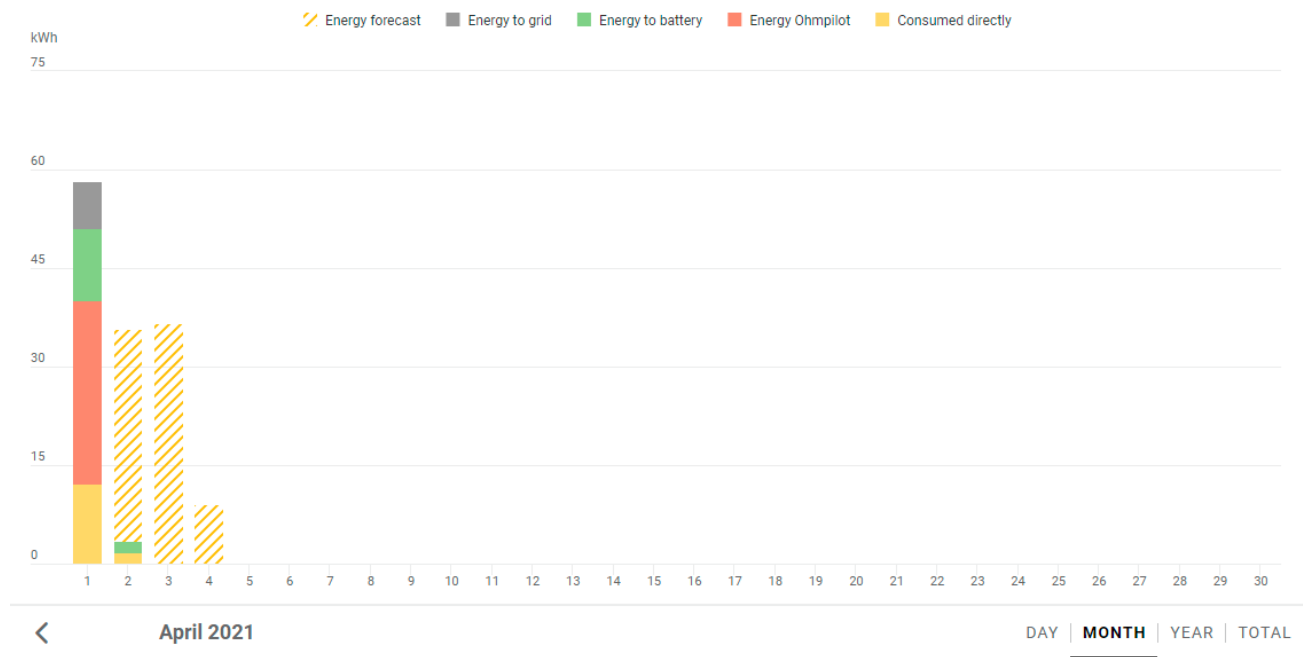
This set provides weather and energy forecast information for PV systems.

Two calls can check the current and future weather. The following info graphic gives you an example:



A third call allows you to predict energy production for the next two days. You could might want to create some diagrams like the following ones (see the hatched areas):





#### 4.7.1 Limitations

Only "Premium" users can access weather forecast information. For "Basic" users the weather forecast APIs will return no data (403 error code).

Premium users receive full weather information for one "pro" system. For all other systems they receive "light" information ("pro" channels will return null).

#### 4.7.2 Weather: Current weather for a PV system

##### Use cases for web developers

- / I want to show the current temperature, wind speed, precipitation for a given PV system.
- / I want to know the time of sunrise and sunset.
- / I want to show weather forecast symbols in my app.

##### Restrictions

- / This call only works for a pro customer who is entitled to get weather forecast information, not for a basic customer.
- / The precipitation, cloud coverage and daylight time channels are not available for "light" PV systems.

##### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ weather/current	GetSystemWeatherCurrent	Gets the current weather information for a given PV system.



### Filters and parameters

Filter	Description
?channel=<channel>	One or more of the detail data channels, e.g. temperature or velocity.
?timezone=<"local", "zulu">	Specifies time format in response object:  / zulu (default): returns time in UTC zulu time / local: returns time in PV system's local UTC time (local time + UTC offset)

### Example calls

```
GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/current  
// gets current weather information for PV system
```

```
GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/current?  
channel=daylight&time=local  
// gets today's sunrise and sunset times in local UTC time
```

```
GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/current?  
channel=temp,symbol  
// gets current temperature and symbol information
```

### Response objects

JSON object answer construction:

Objects
<pre>/ pvSystemId (String) / data   / logDateTime (DateTime)   / channels (Array)     / channelName (String)     / channelType (String)     / unit (String)     / value (Number or Object)</pre>

Supported value channels:

Type	Channels	Remarks
PV system	<pre>/ Temp / WindSpeed / Precipitation / CloudCover / Daylight / Symbol</pre>	Daylight with sunrise and sunset times.  Precipitation, CloudCover and Daylight channels only for "pro" systems.

## Example responses

### Example for a "light" PV system

```
{
  "pvSystemId": "04d81b82-7861-4e36-8e7f-41036ce711a4",
  "data": {
    "logDateTime": "2020-03-17T08:30:00+01:00",
    "channels": [
      {
        "channelName": "Temp",
        "channelType": "Temperature",
        "value": 9.05,
        "unit": "°C"
      },
      {
        "channelName": "WindSpeed",
        "channelType": "Velocity",
        "value": 1.665,
        "unit": "m/s"
      },
      {
        "channelName": "Precipitation",
        "channelType": "Precipitation",
        "value": null,
        "unit": "mm"
      },
      {
        "channelName": "CloudCover",
        "channelType": "Cloudcoverage",
        "value": null,
        "unit": "%"
      },
      {
        "channelName": "Daylight",
        "channelType": "Daylight",
        "value": {
          "sunrise": null,
          "sunset": null
        },
        "unit": "Time"
      },
      {
        "channelName": "Symbol",
        "channelType": "Symbol",
        "value": "mostly cloudy",
        "unit": null
      }
    ]
  }
}
```

### 4.7.3 Weather: Weather forecast for a PV system

#### Use cases for web developers

- / I want to show weather forecast information for the next few days for a given PV system.
- / I want to show weather forecast symbols in my app.

#### Restrictions

- / This call only works for a pro customer who is entitled to get weather forecast information, not for a basic customer.
- / The precipitation and daylight time channels are not available for "light" PV systems.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ weather/forecast	GetSystemWeatherForeca st	Gets weather forecast information for up to next 9 days.

#### Filters and parameters

Filter	Description
?channel=<channel>	One or more of the detail data channels, e.g. temperature or velocity.
?timezone=<"local", "zulu">	Specifies time format in response object: <ul style="list-style-type: none"><li>/ zulu (default): returns time in UTC zulu time.</li><li>/ local: returns time in PV system's local UTC time (local time + UTC offset).</li></ul>
?from=<start>&to=<end>	Limits the time series for the query. <ul style="list-style-type: none"><li>/ &lt;start&gt; and &lt;end&gt; are days, local time of the PV system.</li><li>/ Alternatively, &lt;start&gt; can also be "today" (default) or "tomorrow"; &lt;end&gt; can be "tomorrow".</li><li>/ Date format encoding: "yyyyMMdd", "yyyy-MM-dd".</li></ul>
?from=<start>&duration=<days>	Limits the time series for the query. <ul style="list-style-type: none"><li>/ &lt;start&gt; is a day, local time of the PV system.</li><li>/ Alternatively, &lt;start&gt; can also be "today" (default) or "tomorrow".</li><li>/ &lt;duration&gt; is the number of days.</li><li>/ &lt;duration&gt;=1 means only one single day.</li><li>/ If &lt;duration&gt; is missing, the maximum period is assumed, i.e. full 9 days of weather forecast.</li><li>/ Date format encoding: "yyyyMMdd", "yyyy-MM-dd".</li></ul>

## Example calls

```
GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/forecast
// gets weather forecast for PV system for the next nine days, starting today

GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/forecast?
start=tomorrow&duration=7
// gets weather forecast for PV system for the next week, starting with tomorrow

GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/forecast?
channel=temp,daylight&time=local
// gets weather forecast for PV system for the next nine days, starting today
// the daylight times are returned in local UTC time
```

## Response objects

JSON object answer construction:

Type	Objects
	<pre>/ pvSystemId (String) / data   / logDateTime (Date)   / channels (Array)     / channelName (String)     / channelType (String)     / unit (String)     / value (Number or Object)</pre>

Supported value channels:

Type	Channels	Remarks
PV system	<pre>/ Temp / Daylight / Symbol / EnergyExpected</pre>	<p>Temperatures with temperatureMin and temperatureMax values.</p> <p>Daylight with sunrise and sunset times.</p> <p>Precipitation and Daylight channels only for "pro" systems.</p>

## Example responses

```
{
  "pvSystemId": "04d81b82-7861-4e36-8e7f-41036ce711a4",
  "data": [
    {
      "logDateTime": "2020-03-18T23:00:00Z",
      "channels": [
        {
          "channelName": "Temp",
          "channelType": "Temperature",
          "value": {
            "temperatureMin": 5.05,
            "temperatureMax": 16.85
          },
          "unit": "°C"
        },
        {
          "channelName": "Daylight",
          "channelType": "Daylight",
          "value": {
            "sunrise": null,
            "sunset": null
          },
          "unit": "Time"
        },
        {
          "channelName": "Symbol",
          "channelType": "Symbol",
          "value": "bright",
          "unit": null
        },
        {
          "channelName": "EnergyExpected",
          "channelType": "Energy",
          "value": null,
          "unit": "Wh"
        }
      ]
    }
  ]
}
```

### 4.7.4 Weather: Energy forecast for a PV system

#### Use cases for web developers

- / I want to extend an energy production curve (historical information) with forecast information (future prediction).

#### Restrictions

- / This call only works for a pro customer who is entitled to get weather forecast information, not for a basic customer.

- / This call only works for a pro PV system.
- / Data granularity:
  - / next 24 hours: 15min resolution
  - / following 24 hours: 1h resolution

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv-system-id}/ weather/ energyforecast	GetSystemWeatherEnergyforecast	Gets energy production forecast information for up to next 2 days.

#### Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: <ul style="list-style-type: none"> <li>/ zulu (default): returns time in UTC zulu time.</li> <li>/ local: returns time in PV system's local UTC time (local time + UTC offset).</li> </ul>
?from=<start>&to=<end>	Limits the time series for the query. <ul style="list-style-type: none"> <li>/ &lt;start&gt; and &lt;end&gt; are ISO-8601 time values.</li> <li>/ Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".</li> </ul>
?from=<start>&duration=<hours>	Limits the time series for the query. <ul style="list-style-type: none"> <li>/ &lt;start&gt; is an ISO-8601 time value.</li> <li>/ &lt;duration&gt; a number of 1 to 48 hours.               <ul style="list-style-type: none"> <li>/ If &lt;duration&gt; is missing, the maximum period is assumed, i.e. full two days of energy forecast.</li> </ul> </li> <li>/ Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".</li> </ul>

#### Example calls

```
GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/energyforecast
// gets energyforecast information for PV system for the next 48 hours

GET api.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/energyforecast?duration=8
// gets energyforecast information for PV system for the next 8 hours
```

### Response objects

JSON object answer construction:

Type	Objects
	<ul style="list-style-type: none"><li>/ pvSystemId (String)</li><li>/ data (Array)<ul style="list-style-type: none"><li>/ logDateTime (DateTime)</li><li>/ logDuration (Integer - unit: seconds)</li><li>/ channels (Array)<ul style="list-style-type: none"><li>/ ChannelName (String)</li><li>/ ChannelType (String)</li><li>/ Unit (String)</li><li>/ Value (Number)</li></ul></li></ul></li></ul>

Supported value channels:

Type	Channels	Remarks
PV system	/ EnergyExpected	

## Example responses

```
{
  "pvSystemId": "5b72b205-b698-4243-a68a-a39200e0e9d8",
  "data": [
    {
      "logDateTime": "2020-01-07T12:30:00Z",
      "logDuration": 900,
      "channels": [
        {
          "channelName": "EnergyExpected",
          "channelType": "Energy",
          "value": 24.43275,
          "unit": "Wh"
        }
      ]
    },
    {
      "logDateTime": "2020-01-07T12:45:00Z",
      "logDuration": 900,
      "channels": [
        {
          "channelName": "EnergyExpected",
          "channelType": "Energy",
          "value": 26.23989,
          "unit": "Wh"
        }
      ]
    }
  ]
}
```



## 4.8 System messages calls

### 4.8.1 Messages: Get PV system messages

Use cases for web developers

- / I want to show system and error messages to the user.

#### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv-system-id}/ messages	GetSystemMessages	Returns service messages for a given pv-system in English (default language). The service messages are provided as JSON objects.
GET	swqapi/ pvsystems/{pv-system-id}/ messages/{ISO-country-code}	GetSystemMessages	Returns service messages for a given pv-system in a certain language. The service messages are provided as JSON objects in the language defined by the <ISO-country-code> object.

Note: Since the difference is only the language, the event is the same and the language is encoded in the event log.

#### Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: <ul style="list-style-type: none"><li>/ zulu (default): returns time in UTC zulu time</li><li>/ local: returns time in PV system's local UTC time (local time + UTC offset)</li></ul>
?offset=<offset>&limit=<limit>	Supports pagination, returns messages from a starting <offset> and returning not more than <limit> items.
?from=<start>&to=<end>	Limits the time series for the query. <ul style="list-style-type: none"><li>/ &lt;start&gt; and &lt;end&gt; are ISO-8601 time values.<ul style="list-style-type: none"><li>/ If "to" is missing, then "to" is considered "now". ("from" must not be empty.)</li></ul></li><li>/ Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".</li></ul>
?statetype=<type>	Filters by StateType, e.g. "Error", "Event".
?stateseverity=<level>	Filters by StateSeverity, i.e. "Error", "Warning", "Information".

Filter	Description
?statecode=<code>	Filters by StateCode.
?type=inverter	Filters for inverters.
?type=sensor	Filters for sensors.
?type=battery	Filters for batteries.
?type=smartmeter	Filters for Smart Meters.
?type=ohmpilot	Filters for Ohmpilots.

#### Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/messages?
from=2018-10-10T00:00:00Z&to=2018-10-11T00:00:00Z
// gets all system messages for 10th of October, 2018, for PV system
```

#### Response objects

JSON object answer construction:

Type	Objects
PV systems	<ul style="list-style-type: none"> <li>/ pvSystemId (String)</li> <li>/ deviceId (String)</li> <li>/ stateType (String)</li> <li>/ stateSeverity (String)</li> <li>/ stateCode (Integer)</li> <li>/ logDateTime (String, UTC timestamp)</li> <li>/ text (String) - language depends on &lt;ISO-country-code&gt; object</li> </ul>

## Example responses

```
[
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "b6f5495d-e0e7-49c2-80d6-e0bcd0ebacdc",
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 509,
    "logDateTime": "2019-01-08T09:32:00Z",
    "text": "No Feed In For 24 Hours"
  },
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "b6f5495d-e0e7-49c2-80d6-e0bcd0ebacdc",
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 901,
    "logDateTime": "2018-12-27T23:50:00Z",
    "text": "Current Sensor Deviation On Channel 1"
  },
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": null,
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 906,
    "logDateTime": "2018-12-26T12:38:07Z",
    "text": "Heating rod 1 defective - short circuit L1"
  },
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "b6f5495d-e0e7-49c2-80d6-e0bcd0ebacdc",
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 475,
    "logDateTime": "2018-12-24T09:03:00Z",
    "text": "Isolation Error"
  },
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "b6f5495d-e0e7-49c2-80d6-e0bcd0ebacdc",
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 901,
    "logDateTime": "2018-12-27T23:50:00Z",
    "text": "Current Sensor Deviation On Channel 1"
  }
]
```

## 4.8.2 Messages: Count PV system messages

### Use cases for web developers

- / I want to know how many error messages I have to show. (Needed for subsequent enumeration and detail calls.)

### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ messages-count	GetSystemMessagesCount	Returns number of service messages for a given pv-system.

### Filters and parameters

Filter	Description
?from=<start>&to=<end>	Limits the time series for the query.  / <start> and <end> are ISO-8601 time values. / If "to" is missing, then "to" is considered "now". ("from" must not be empty.)  / Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?statetype=<type>	Filters by StateType, e.g. "Error", "Event".
?stateseverity=<level>	Filters by StateSeverity, i.e. "Error", "Warning", "Information".
?statecode=<code>	Filters by StateCode.
?type=inverter	Filters for inverters.
?type=sensor	Filters for sensors.
?type=battery	Filters for batteries.
?type=smartmeter	Filters for Smart Meters.
?type=ohmpilot	Filters for Ohmpilots.
?type=datalogger	Filters for dataloggers.

### Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/messages-  
count?from=2018-10-10T00:00:00Z&to=2018-10-11T00:00:00Z  
// counts all system messages for 10th of October, 2018, for PV system
```

### Response objects

JSON object answer construction:

Type	Objects
n/a	/ count (Number)

### Example responses

```
{  
  "count": 5  
}
```

## 4.8.3 Messages: Get device system messages

Use cases for web developers

/ I want to show system and error messages to the user.

### Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv- system-id}/ devices/{device-id} /messages	GetDeviceMessages	Returns service messages for the requested PV device of a PV system with the given ID in English (default language). The service messages are provided as JSON objects.
GET	swqapi/ pvsystems/{pv- system-id}/ devices/{device-id} /messages/{ISO- country-code}	GetDeviceMessages	Returns service messages for the requested PV device of a PV system with the given ID in a certain language. The service messages are provided as JSON objects in the language defined by the <ISO-country-code> object.

Note: Since the difference is only the language, the event is the same and the language is encoded in the event log.

## Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object:  / zulu (default): returns time in UTC zulu time / local: returns time in PV system's local UTC time (local time + UTC offset)
?offset=<offset>&limit=<limit>	Supports pagination, returns messages from a starting <offset> and returning not more than <limit> items.
?from=<start>&to=<end>	Limits the time series for the query.  / <start> and <end> are ISO-8601 time values. / If "to" is missing, then "to" is considered "now". ("from" must not be empty.)  / Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?statetype=<type>	Filters by StateType, e.g. "Error", "Event".
?stateseverity=<level>	Filters by StateSeverity, i.e. "Error", "Warning", "Information".
?statecode=<code>	Filters by StateCode.

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/  
d2e61bf2-8dd7-4ba1-8733-d55d738c4679/messages?from=2018-10-10T00:00:00Z&to=2018-10-11  
T00:00:00Z  
// gets all system messages for device for 10th of October, 2018
```

## Response objects

JSON object answer construction:

Type	Objects
Device	 / pvSystemId (String) / deviceId (String) / stateType (String) / stateSeverity (String) / stateCode (Integer) / logDateTime (String, UTC timestamp) / text (String) - language depends on <ISO-country-code> object

Example responses

```
[
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "d2e61bf2-8dd7-4ba1-8733-d55d738c4679",
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 509,
    "logDateTime": "2019-01-08T09:32:00Z",
    "text": "No Feed In For 24 Hours"
  },
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "d2e61bf2-8dd7-4ba1-8733-d55d738c4679",
    "stateType": "Error",
    "stateSeverity": "Error",
    "stateCode": 901,
    "logDateTime": "2018-12-27T23:50:00Z",
    "text": "Current Sensor Deviation On Channel 1"
  }
]
```

4.8.4 Messages: Count device system messages

Use cases for web developers

- / I want to know how many error messages I have to show. (Needed for subsequent enumeration and detail calls.)

Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv-system-id}/ devices/{device-id}/messages-count	GetDeviceMessagesCount	Returns number of service messages for the requested PV device of a PV system with the given ID.

## Filters and parameters

Filter	Description
?from=<start>&to=<end>	Limits the time series for the query.  / <start> and <end> are ISO-8601 time values. / If "to" is missing, then "to" is considered "now". ("from" must not be empty.)  / Time format encoding: "yyyyMMddTHH:mm:ssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?statetype=<type>	Filters by StateType, e.g. "Error", "Event".
?stateseverity=<level>	Filters by StateSeverity, i.e. "Error", "Warning", "Information".
?statecode=<code>	Filters by StateCode.

## Example calls

```
GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices/
d2e61bf2-8dd7-4ba1-8733-d55d738c4679/messages-count?from=2018-10-10T00:00:00Z&to=2018-1
0-11T00:00:00Z
// counts all system messages for device for 10th of October, 2018
```

## Response objects

JSON object answer construction:

Type	Objects
n/a	/ count (Number)

## Example responses

```
{
  "count": 2
}
```



## 5 APPENDIX

### 5.1 Response and error codes

#### 5.1.1 HTML error codes

Used HTML response codes by SWQAPI:

Code	Short description	Description
200	OK	Successful
204	No content	Successful request but no data
400	Bad request	Malformed request
401	Unauthorized	No or invalid authentication details are provided
403	Forbidden	Authentication succeeded but authenticated user/ API key doesn't have access to the resource
404	Not found	Non-existent resource is requested
405	Method not allowed	The request method is not supported for the requested resource
415	Unsupported media type	Payload format is not supported
429	Too many requests	Request is rejected due to rate limiting
500	Internal server error	An unexpected server error happened
503	Service not available	The server is temporarily unavailable

#### 5.1.2 Detailed error codes

If an error happens, SWQAPI indicates the error reason in the JSON response object.

##### Error code examples

```
{
  "responseError": 1008,
  "responseMessage": "Invalid channels:
EnergyBatteryDischarge,EnergyBatteryDischarge"
}
```

```
{
  "responseError": 1005,
  "responseMessage": "Invalid date and time format."
}
```

```
{
  "responseError": 1004,
  "responseMessage": "Input invalid. Unrecognized parameters: cannel"
}
```

#### List of detailed error codes

Fronius Response Code	Verbose Description	Comment
<b>GENERAL (10xx)</b>		
1001	Error while processing request.	General error for internal server.
1002	Requested resource not found.	
1003	No input set.	
1004	Input invalid.	The reason for the error is usually given.
1005	Invalid date and time format.	
1006	Invalid date format.	
1007	Invalid timezone parameter.	
1008	Invalid channels	List of invalid channels is appended at the end.
1009	Invalid language code	
1010	From date is after to date.	
1011	API calls quota exceeded.	"Maximum admitted {0} per {1}. Retry after: {2}" added to message
<b>AUTHENTICATION (11xx)</b>		
1101	AccessKeyId and Value not sent.	
1102	AccessKey not found.	

<b>Fronius Response Code</b>	<b>Verbose Description</b>	<b>Comment</b>
1103	AccessKey is not active.	
1104	AccessKey expired.	
1105	User blocked.	
1106	Authentication failed.	
1107	Invalid request.	
1108	User did not accept latest Terms of Use.	
1110	Invalid JWT format.	
1111	Invalid JWT signature.	
1112	Invalid JWT issuer.	
1113	JWT expired.	
1114	Missing parameters: UserId, Password	Missing parameters could be UserId or Password
1115	Empty parameters: UserId, Password	Empty parameters could be UserId or Password
1116	Invalid scope. Token likely expired.	
1120	Refresh token invalid.	
1121	Refresh token expired.	
<b>Metadata calls (30xx)</b>		
3001	Type filters invalid.	List of invalid type filters is appended at the end
3002	Invalid meteo parameter.	
<b>Flowdata calls (31xx)</b>		
31xx		
<b>Aggdata calls (32xx)</b>		
3201	Invalid date format for "from" parameter.	

<b>Fronius Response Code</b>	<b>Verbose Description</b>	<b>Comment</b>
3202	Invalid date format for "to" parameter.	
3203	Invalid duration format.	
3204	Invalid combination of parameters.	Only from and to, or from and duration are allowed.
3205	Invalid duration range.	Maximum range is 100 years.
3206	From and to parameters do not have same format.	
3207	Period parameter should not be used in combination with other time parameters.	Do not use from, to or duration, in combination with period parameter.
<b>Histdata calls (33xx)</b>		
3301	Date range max is 24 hours.	
3302	User unauthorized to access requested time range.	Impersonated basic users can only retrieve information not older than 72 hours - like in Solar.web.
<b>Messages calls (34xx)</b>		
3401	Invalid state type.	
3402	Invalid state code.	
3403	Date range between from and to filters too big.	
3404	From date is required.	
<b>Weather calls (35xx)</b>		
3501	Invalid parameter combination.	
3502	No POI data for PV system.	PV system does not have POI information assigned.
3503	Not a Pro PV system.	Pro user is requesting energy forecast information for a "light" PV system. (Forecast information is only available for "pro" systems.)

## 5.2 Channels

### 5.2.1 Channel list

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
EnergyExported	Total power	general, inverter	Wh	Total energy flowing out of the main inverter (generators sum + battery)		x	x	
EnergyExpected		general	Wh	Energy forecast				x
IsOnline		general	-	Online status of the PV system	x			
EnergyImported		inverter	Wh	Total energy flowing into the main (hybrid) inverter. This energy comes either from the grid or from another inverter, and it is stored in a battery.			x	
EnergyDC1, EnergyDC2, ...	Power MPP1, Power MPP2	inverter	Wh	Calculated energy flowing from generator into inverter on DC1 / DC2 input		x	x	
CurrentA, CurrentB, CurrentC	Current AC L1, Current AC L2, Current AC L3	inverter	A	Mean current of 3-phase devices on the AC side of the main inverter for line L1 / L2 / L3			x	
CurrentDC1, CurrentDC2, ...	Current DC MPP1, Current DC MPP2	inverter	A	Mean current on the DC side of the inverter for DC1 / DC2			x	
VoltageA, VoltageB, VoltageC	Voltage AC L1, Voltage AC L2, Voltage AC L3	inverter	V	Mean voltage of 3-phase devices on the AC side of the main inverter for line L1 / L2 / L3			x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
VoltageAB, VoltageBC, VoltageCA		inverter	V	Mean voltage of 3-phase devices on the AC side of the main inverter (voltages are between lines L1L2 / L2L3 / L3L1)			x	
VoltageDC1, VoltageDC2, ...	Voltage DC MPP1, Voltage DC MPP2	inverter	V	Mean voltage on the DC side of the inverter for DC1 / DC2			x	
Temp1, Temp2	Module temperature	sensor	°C	Temperature			x	
Insolation	Insolation	sensor	W/m <sup>2</sup>	Insolation			x	
WindSpeed		sensor	m/s	Wind speed			x	x
Temp			°C	Current temperature				x
Temp (min, max)			°C	Temperature forecast				x
Daylight (sunrise, sunset)			time	Sunrise and sunset times				x
CloudCover			%	Cloud coverage				x
Precipitation			mm	Precipitation				x
Symbol			symbol	Weather forecast symbol				x

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
Digital1, Digital2, Digital3		sensor	<variable>	Digital channel from Fronius Sensor Card (unit depends on sensor settings)			x	
Digital1Energy, Digital2Energy, Digital3Energy		sensor	Wh	Energy measured by sensor. Note: DigitalX and DigitalXEnergy exclude each other.			x	
BattMode		battery		Battery operating state. State values are: / 0 = Disabled / 1 = Normal / 2 = ServiceMode / 3 = ChargeBoost / 4 = NearlyDepleted / 5 = SuspendedOnPurpose / 6 = Calibrate / 7 = GridSupport / 8 = DepletedRecovery / 9 = NonOperableTemperature / 10 = NonOperableVoltage / 11 = Preheating / 12 = Startup / 13 = AwakeButNonOperableTemperature / 14 = BatteryFull / 90 = ForcedStandby	x		x	
BattSOC	State of charge	battery, general	%	Battery state of charge	x		x	



Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
GridPowerA, GridPowerB, GridPowerC	Effective power L1 feed-in point, Effective power L2 feed-in point, Effective power L3 feed-in point	Smart Meter	W	Mean power of 3-phase devices of the grid phase 1 / 2 / 3 measured by primary grid meter in the given interval			x	
GridApparentPowerA, GridApparentPowerB, GridApparentPowerC	Apparent power L1 feed-in-point, Apparent power L2 feed-in-point, Apparent power L3 feed-in-point	Smart Meter	VA	Mean apparent power of 3-phase devices of the grid phase 1 / 2 / 3 measured by primary grid meter in the given interval			x	
GridVoltageA, GridVoltageB, GridVoltageC	Voltage AC L1 feed-in point, Voltage AC L2 feed-in point, Voltage AC L3 feed-in point	Smart Meter	V	Mean voltages of 3-phase devices of the grid phase 1 / 2 / 3 measured by primary load meter in the given interval			x	
LoadPowerA, LoadPowerB, LoadPowerC		Smart Meter	W	Mean power of 3-phase devices of the load phase 1 / 2 / 3 measured by primary load meter in the given interval			x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
LoadApparentPowerA, LoadApparentPowerB, LoadApparentPowerC		Smart Meter	VA	Mean apparent power of 3-phase devices of the load phase 1 / 2 / 3 measured by primary load meter in the given interval			x	
LoadVoltageA, LoadVoltageB, LoadVoltageC		Smart Meter	V	Mean voltage of 3-phase devices of the load phase 1 / 2 / 3 measured by primary load meter in the given interval			x	
ExtEnergyExportedAbs		Smart Meter	Wh	Energy flowing from external AC source (e.g. wind power generator) to the point of common coupling (absolute value)			x	
ExtEnergyExported		Smart Meter	Wh	Energy flowing from external AC source (e.g. wind power generator) to the point of common coupling			x	
EnergyLoadAbs		Smart Meter	Wh	Energy flowing into the consumers (absolute value)			x	
EnergyLoad		Smart Meter	Wh	Energy flowing into the consumers			x	
GridEnergyImportedAbs		Smart Meter	Wh	Energy imported from grid (absolute value)			x	
GridEnergyExportedAbs		Smart Meter	Wh	Energy flowing from the house into grid (absolute value)			x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
GridEnergyImported		Smart Meter	Wh	Energy imported from grid			x	
GridEnergyExported		Smart Meter	Wh	Energy flowing from the house into grid			x	
OhmpilotTemp	Temperature Ohmpilot	Ohmpilot	°C	Temperature measured on Ohmpilot device	x		x	
OhmpilotEnergyAbs		Ohmpilot	Wh	Absolute energy used by Ohmpilot			x	
OhmpilotEnergy	Power	Ohmpilot	Wh	Energy used by Ohmpilot		x	x	
OhmpilotError		Ohmpilot	-	Ohmpilot error code			x	
EnergySelfConsumption	Consumed directly (Consumption tab)	general	Wh	Calculated energy flowing from generators to consumers (including Ohmpilot)		x	x	
EnergyDirectConsumption	Consumed directly (Production tab)	general	Wh	Calculated energy flowing from generators to consumers (excluding Ohmpilot)		x		
EnergyFeedIn	Power to grid	general	Wh	Calculated energy flowing from generators to grid		x	x	
EnergyBattCharge	Power to battery	general	Wh	Calculated energy flowing from generators to batteries		x	x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
EnergyBattDischarge	Power from battery	general	Wh	Calculated energy flowing from batteries to consumer		x	x	
EnergyBattDischargeGrid		general	Wh	Calculated energy flowing from battery to grid		x	?	
EnergyPurchased	Power from grid	general	Wh	Calculated energy flowing from grid to consumer		x	x	
EnergyBattChargeGrid		general	Wh	Calculated energy flowing from grid to battery		x	?	
EnergyOutput		general	Wh	Calculated total energy produced (NULL if smart meter is connected)		x	x	
EnergyProductionTotal	Production	general	Wh	Calculated energy flowing from generators to consumer, battery and grid (EnergySelfConsumption + EnergyBattCharge + EnergyFeedIn + EnergyOutput)		x	x	
EnergySelfConsumptionTotal	Own consumption	general	Wh	Calculated energy flowing from generators to consumers and battery (EnergySelfConsumption + EnergyBattCharge)		x	x	
EnergyConsumptionTotal	Consumption	general	Wh	Calculated consumed energy (EnergyPurchased + EnergySelfConsumption + EnergyBattDischarge)		x	x	
PowerFeedIn		general	W	Power flowing from inverter to the grid (requires a Smart Meter)	x			

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
PowerLoad		general	W	Power flowing from inverter to the consumer (requires a Smart Meter)	x			
PowerBattCharge		general	W	Power flowing from inverter to the battery (requires a Smart Meter and a battery)	x			
PowerOhmpilot		general	W	Power flowing from inverter to an Ohmpilot (requires a Smart Meter and an Ohmpilot)	x			
PowerPV		general	W	Power flowing from generators to the inverter	x			
PowerBattDischarge		general	W	Power flowing out of the battery (requires a Smart Meter and a battery)				
PowerOutput		general	W	Power generated by inverter (NULL if Smart Meter is connected)	x			
PowerExt		general	W	Power generated by external inverter (requires a Smart Meter)	x			
PowerPurchased		general	W	Power flowing in from grid (requires a Smart Meter)	x			
RateSelfConsumption		general	%	Percentage of produced energy which is directly used (compared to energy fed into the grid)	x			
RateSelfSufficiency		general	%	Percentage of produced energy of all energy used (includes energy from the grid)	x			
ApparentPower	Apparent power	inverter	VA	Apparent power (S)			x	

























Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
ReactivePower	Reactive power	inverter	VAr	Reactive power (Q)			x	
PowerFactor	Power factor	inverter	-	Power factor			x	
StandardizedPower	Standardized power	inverter	%	Percentage of power generated in relation to peak power (i.e. kWh/kWp)			x	
Profits		general	<currency>	Total financial benefit of the PV system (Earnings + Savings).		x		
Savings		general	<currency>	Directly consumed energy multiplied by a tariff.		x		
Earnings	Earning	general	<currency>	Energy fed into grid multiplied by a tariff.		x		
SavingsCO2	CO2 savings	general	kg	Equivalent of saved CO <sub>2</sub> due to PV production.		x		
SavingsTrees	CO2 savings	general	tree	Equivalent of saved average trees due to PV production.		x		
SavingsTravelCar	CO2 savings	general	km	Equivalent of saved average car travel distance due to PV production.		x		
SavingsTravelPlane	CO2 savings	general	mile	Equivalent of saved average air travel distance due to PV production.		x		

### 5.2.2 Channel types












Type	Unit
Power	W
Energy	Wh
Voltage	V
Current	A
Apparent Power	VA
Reactive Power	VA <sub>r</sub>
Temperature	°C
Percentage	%
Insolation	W/m <sup>2</sup>
Velocity (wind speed)	m/s
Precipitation (rain)	mm
Boolean	true or fals
Symbol	Symbol
Time	Time
Currency	<currency>
CO2 savings	kg, tree, km or mile













## 5.3 Meteorological weather symbols

### 5.3.1 List of weather symbols

Symbol description	Icon (example)	Symbol description	Icon (example)
clear sky		light snowfall	
bright		snowfall	
cloudy		heavy snow	
mostly cloudy		sunny with light snow showers	
overcast		cloudy with light snow showers	
fog		overcast with light snow showers	
low clouds		sunny with heavy snow showers	
light rain		cloudy with heavy snow showers	
rain		overcast with heavy snow showers	
heavy rain		light sleet	
drizzle		sleet	
light freezing rain		heavy sleet	
heavy freezing rain		sunny with light sleet showers	
sunny with scattered rain showers			



Symbol description	Icon (example)
cloudy with scattered rain showers	
overcast with scattered rain showers	
sunny with heavy rain showers	
cloudy with heavy rain showers	
overcast with heavy rain showers	
sunny with thunderstorms	
cloudy with thunderstorms	
overcast with thunderstorms	
sunny with strong thunderstorms	
cloudy with strong thunderstorms	
overcast with strong thunderstorms	

Symbol description	Icon (example)
cloudy with light sleet showers	
overcast with light sleet showers	
sunny with heavy sleet showers	
cloudy with heavy sleet showers	
overcast with heavy sleet showers	
duststorm / sandstorm	
drifting snow	
graupel	
fog patches	
low clouds, sun	
freezing fog	
sun and high clouds	

## 5.4 Languages

SWQAPI supports the following languages in system messages and Terms of Use (ISO 639-1 language codes):

ISO Code	Language
CS	Czech
DA	Danish
DE	German
EL	Greek
EN	English
ES	Spanish
FI	Finnish
FR	French
HU	Hungarian
IT	Italian
NL	Dutch
PL	Polish
PT	Portuguese
RU	Russian
SK	Slovakian
SV	Swedish
TR	Turkish

## 5.5 Best practices and how-tos

### 5.5.1 Use filters for channels

It is highly recommended to use filters whenever possible even though all available channels are requested. Fronius might add channels in the future so filters would help to get the same number of channels (and data points) without the need of changing any code.

### 5.5.2 Determine power values from energy values from historical data

$$\text{Power [W]} = \text{Energy [Wh]} \times 3600 / \text{logDuration [s]}$$

logDuration (typ. 300 seconds) can be found in the response to /histdata endpoint:

```
{
  "pvSystemId": "a69f2adc-5fd0-4321-8323-a484013871f6",
  "deviceId": null,
  "data": [
    {
      "logDateTime": "2021-01-19T09:00:00+01:00",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "EnergyProductionTotal",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 34.99
        },
        ...
      ]
    }
  ]
}
```

### 5.5.3 Determine PV Energy and Load Energy

PV Energy = EnergySelfConsumption + EnergyFeedIn + EnergyBattCharge

Load Energy = EnergySelfConsumption + EnergyPurchased + EnergyBattDischarged

### 5.5.4 Determine if new systems were added to account

**Method 1 - only systems are added:**

1. Simply request the system count (pvsystems-count) for your account once in a while (e.g. once per day).
2. Compare the value with the last one you received. If the new one is higher then there is a new system.
3. Use the pvsystems-list endpoint to get the PV system IDs.
4. Compare the IDs with your current list of IDs in order to find the ID of the new system.
5. Update your list of IDs.

**Method 2 - if there are systems that are temporarily added to your account:**

1. Use the pvsystems-list endpoint to get the PV system IDs.

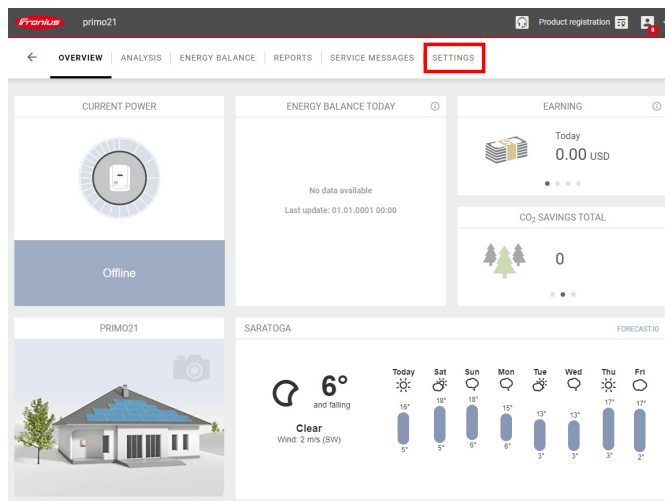
2. Compare the IDs with your current list of IDs in order to find the ID of the new system.
3. Update your list of IDs.
4. Use the pvsystems-list endpoint repeatedly to determine if a PV system is removed from account.
5. If an ID does not show up in the response anymore it can be deleted from your list.

### 5.5.5 Grant permissions in Solar.web

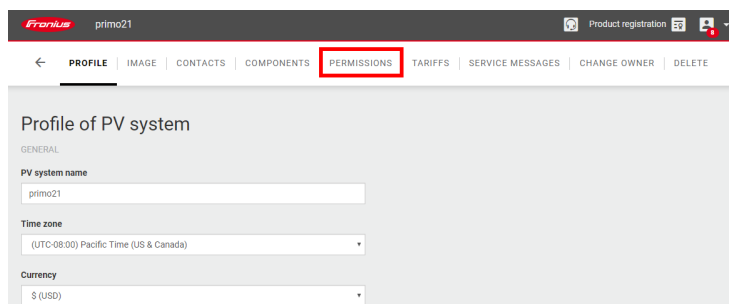
When using the regular authentication method your user needs permission to access a PV system's data through the API if you don't own it. Here's a short guide on how to grant permission.

- i** The API user has to take care of getting the permission from the owner (or a supervisor) of a PV system. Fronius does not grant anybody permission to a PV system that is not linked to the account (by ownership or permission).

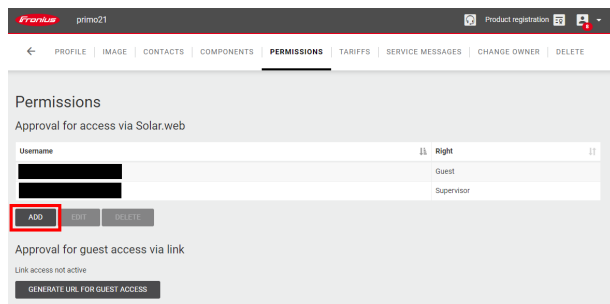
1. Log in to Solar.web and select PV the system
2. Go to SETTINGS



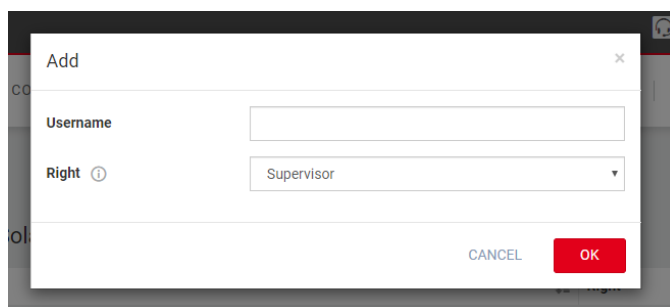
3. Go to PERMISSIONS



4. Click on ADD



5. Enter the email address of the account that shall be granted access to the system in the Username input field. Select a permission level (there is no difference for access through the Solar.web Query API): Guest (only read permission, no change of settings possible) or Supervisor (change of settings possible). Click OK



### 5.5.6 Migration from Solar.web Third Party API to Solar.web Query API

#### User authentication and access token renewal

The SWQAPI does not require an access token. Valid API keys (access key ID and access key value) are required to be sent with each request as part of the header. For more details about API key management see.

Old	New
/auth/login	n/a
/auth/refresh	n/a

#### Metadata

The old metadata endpoint returns a list of all PV systems that are linked to the user's account, including detailed information about the system and all devices. To get the same information with the new methods it would require to use *pvsystems* for the list of PV systems and then *pvsystems/<pv-system-id>/devices* for each PV system.

To simply check if a new system was added to the account *pvsystems-count* can be used. If only the PV system IDs are needed then *pvsystems-list* would be enough. For checking for new devices in a PV system *pvsystems/<pv-system-id>/devices-count* and *pvsystems/<pv-system-id>/devices-list* can be used.

Old	New
/metadata	pvsystems-count
	pvsystems-list
	pvsystems
/metadata/{pvSystemId}	pvsystems/<pv-system-id>
	pvsystems/<pv-system-id>/devices-count
	pvsystems/<pv-system-id>/devices-list
	pvsystems/<pv-system-id>/devices
	pvsystems/<pv-system-id>/devices/<device-id>

#### Aggregated data

Old	New
n/a	pvsystems/<pv-system-id>/aggdata
/aggvalues/total	pvsystems/<pv-system-id>/aggdata/years
	pvsystems/<pv-system-id>/aggdata/years/<year>
/aggvalues/year	pvsystems/<pv-system-id>/aggdata/years/<year>/months
	pvsystems/<pv-system-id>/aggdata/years/<year>/months/<month>
/aggvalues/month	pvsystems/<pv-system-id>/aggdata/years/<year>/months/<month>/days
	pvsystems/<pv-system-id>/aggdata/years/<year>/months/<month>/days/<day>

#### Detail values

To get historical data two endpoints can now be used: one that provides energy data for the whole system and one that provides the data for a single device.

Old	New
/detailvalues/{pvSystemId}	pvsystems/<pv-system-id>/histdata
	pvsystems/<pv-system-id>/devices/<device-id>/histdata

The migration for some common data points are shown in the table below.

Old channel	New endpoint	New channel
DevWork	pvsystems/<pv-system-id>/devices/<device-id>/histdata	EnergyExported
FromGenToConsumer	pvsystems/<pv-system-id>/histdata	EnergySelfConsumption
FromGenToGrid		EnergyFeedIn
FromGenToBatt		EnergyBattCharge
FromBattToConsumer		EnergyBattDischarge
FromGridToConsumer		EnergyPurchased
FromGenToSomewhere		EnergyOutput

#### Energy flow values and real time data

Old	New
/energyflow/{pvSystemId}	pvsystems/<pv-system-id>/flowdata
/realtime/{pvSystemId}	pvsystems/<pv-system-id>/flowdata
	pvsystems/<pv-system-id>/flowdata/<device-id>/flowdata

#### Service messages

Old	New
/servicemessages/{pvSystemId}	pvsystems/<pv-system-id>/messages-count

Old	New
	pvsystems/<pv-system-id>/messages
	pvsystems/<pv-system-id>/messages/<ISO-country-code>
	pvsystems/<pv-system-id>/devices/<device-id>/messages-count
	pvsystems/<pv-system-id>/devices/<device-id>/messages
	pvsystems/<pv-system-id>/devices/<device-id>/messages/<ISO-country-code>