



# Solar.web Query API

## Specification

© Fronius International GmbH

Version 52.0 2025-01-27

Solar Energy

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.

## Table of Contents

<b>1</b>	<b><i>Version History</i></b>	<b>6</b>
<b>2</b>	<b><i>Introduction</i></b>	<b>7</b>
2.1	About Fronius Solar.web Query API	7
2.2	Usage of the Fronius Solar.web Query API	8
2.2.1	Target audience	8
2.2.2	How to start with Fronius Solar.web Query API	8
2.2.3	Data plans and pricing	9
<b>3</b>	<b><i>General information</i></b>	<b>10</b>
3.1	Key managment	10
3.2	User impersonation	12
3.3	Identification of PV systems	16
3.4	Pagination	17
3.5	Date and time formats	17
3.5.1	Use time in the calling URL	18
3.5.2	Time in response objects	18
<b>4</b>	<b><i>API Management Portal</i></b>	<b>19</b>
4.1	Development section	19
4.1.1	Release information	19
4.1.2	System availability	20
4.1.3	Key management	21
4.2	Account section	22
4.2.1	Monthly reports	22
4.2.2	Billing information	23
<b>5</b>	<b><i>Swagger UI</i></b>	<b>23</b>
<b>6</b>	<b><i>API Reference</i></b>	<b>26</b>
6.1	User impersonation calls	26

6.1.1 Impersonate: Receive a JWT using userId and password.....	26
6.1.2 Impersonate: Refresh a JWT .....	29
6.1.3 Impersonate: Revoke a JWT.....	31
6.2 Generic information calls.....	33
6.2.1 Info: Get release information.....	33
6.2.2 Info: Get user information.....	34
6.3 Metadata calls.....	37
6.3.1 Metadata: Get PV system information.....	37
6.3.2 Metadata: Count PV systems .....	41
6.3.3 Metadata: Enumerate PV system IDs.....	43
6.3.4 Metadata: Get device information.....	45
6.3.5 Metadata: Count devices .....	59
6.3.6 Metadata: Enumerate device IDs .....	60
6.4 Aggregation calls .....	62
6.4.1 Aggrdata: Aggregated energy data for a PV system .....	62
6.4.2 Aggrdata: Aggregated energy data for a device .....	71
6.5 Historical data calls.....	77
6.5.1 Histdata: Historical data for a PV system .....	77
6.5.1 Histdata: Historical data for a device .....	81
6.6 Realtime data calls.....	92
6.6.1 Flowdata: Realtime power flow data of a PV system .....	92
6.6.2 Flowdata: Realtime power flow data of a device .....	96
6.7 Weather data calls .....	100
6.7.1 Limitations .....	101
6.7.2 Weather: Current weather for a PV system .....	101
6.7.3 Weather: Weather forecast for a PV system .....	104
6.7.4 Weather: Energy forecast for a PV system.....	107
6.8 System messages calls.....	110
6.8.1 Messages: Get PV system messages .....	110
6.8.2 Messages: Count PV system messages.....	113
6.8.3 Messages: Get device system messages.....	114
6.8.4 Messages: Count device system messages .....	116

7 Appendix .....	118
7.1 Response and error codes .....	118
7.1.1 HTML error codes .....	118
7.1.2 Detailed error codes .....	119
7.2 Response Headers .....	123
7.2.1 Rate limit headers .....	123
7.2.2 Cache control headers (max-age).....	123
7.3 Channels .....	126
7.3.1 Channel list .....	126
7.3.2 Channel types .....	139
7.4 Meteorological weather symbols .....	140
7.4.1 List of weather symbols .....	140
7.5 Languages .....	143
7.6 Best practices and how-tos .....	144
7.6.1 Use filters for channels .....	144
7.6.2 Determine power values from energy values from historical data .....	144
7.6.3 Determine PV Energy and Load Energy .....	144
7.6.4 Determine if new systems were added to account .....	145
7.6.5 Grant permissions in Solar.web .....	145

# 1 Version History

Version	Modified	Description
52.0	Jan 27, 2025	<ul style="list-style-type: none"> <li>• Note to metadata filtering added</li> <li>• Fixed endpoint description for weather calls</li> <li>• Added filter information to GetUserInfo call</li> <li>• Endpoint cleanup (<a href="https://swqapi.solarweb.com">swqapi.solarweb.com</a> vs. the older <a href="https://api.solarweb.com/swqapi">api.solarweb.com/swqapi</a>)</li> <li>• Added information about API Management Portal</li> </ul>
51.0	Oct 12, 2023	<ul style="list-style-type: none"> <li>• Add new battery channels (powerflow)</li> <li>• Updated battery metadata example</li> </ul>
50.0	Aug 01, 2023	<ul style="list-style-type: none"> <li>• Added channels for Submeters.</li> <li>• Corrected "PowerPurchased" channel to "PowerPurchase".</li> <li>• Updated EnergyExported and EnergyImported channels for secondary meters</li> <li>• Pilot example added</li> <li>• Updated data plan details</li> </ul>
49.0	Oct 12, 2022	<ul style="list-style-type: none"> <li>• Updated sections Metadata calls, Aggregation calls, Historical data calls, Realtime data calls, System messages calls:               <ul style="list-style-type: none"> <li>• Added Wattpilot data.</li> <li>• Updated some example responses.</li> </ul> </li> <li>• Minor corrections in section Weather data calls.</li> <li>• Updated General information section.</li> <li>• Updated Supporting UIs section.</li> <li>• Updated channel list:               <ul style="list-style-type: none"> <li>• Added Wattpilot data</li> </ul> </li> <li>• Removed reference to outdated Aggregation calls</li> </ul>

Version	Modified	Description
48.0	Dec 09, 2021	<ul style="list-style-type: none"> <li>• Added notes to GenerateJwt call.</li> <li>• Added status code information for maintenance windows.</li> <li>• Updated device metadata: <ul style="list-style-type: none"> <li>• Provided lists for Smart Meter locations and categories.</li> <li>• Provided list for sensor types.</li> </ul> </li> <li>• Updated device metadata: <ul style="list-style-type: none"> <li>• Added missing datalogger IDs.</li> <li>• Improved examples, e.g. a GEN24 PV system.</li> </ul> </li> <li>• Added revoke JWT call.</li> <li>• Updated metadata calls with meteo filter.</li> <li>• Clarifications in error tables.</li> <li>• Removed "PowerBattDischarge" channel from channel list, and updated description of "PowerBattCharge".</li> <li>• Clarified positive/negative values for power flow data.</li> <li>• Clarified 3301 error code for historical data.</li> <li>• Added datalogger and Ohmpilot examples to device metadata.</li> <li>• Removed Fronius SSO login method.</li> </ul>
47.0	Apr 21, 2021	<ul style="list-style-type: none"> <li>• Updated camelCase notation in examples for system messages.</li> </ul>
46.0	Apr 02, 2021	<ul style="list-style-type: none"> <li>• Changed supported channel information for aggregated and historical data requests.</li> <li>• Updated Solar.web screenshots, error code lists and channel list.</li> </ul>
45.0	Feb 02, 2021	<ul style="list-style-type: none"> <li>• Version history added.</li> <li>• Added additional information about Solar.web Premium to chapter "User impersonation".</li> <li>• Updated chapter "Determine power values from energy values" in Appendix.</li> </ul>

## 2 Introduction

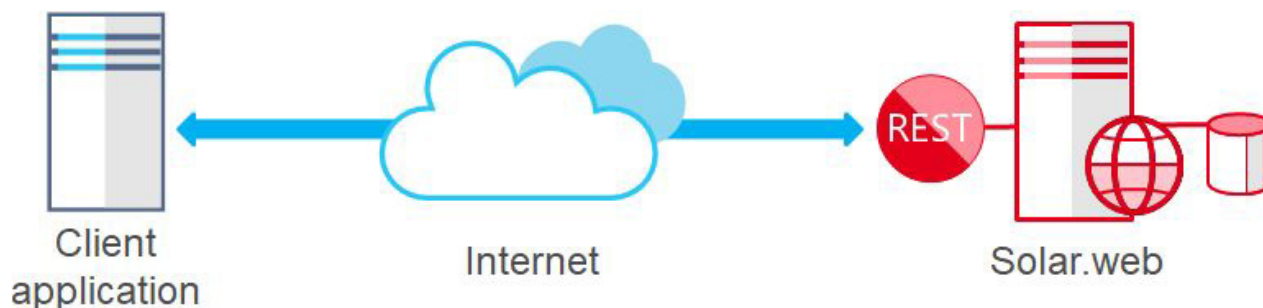
### 2.1 About Fronius Solar.web Query API

#### 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.



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

### 2.2.1 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.

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

#### **Trial access**

A Fronius Sales Representative can enable the trial access for interested customers which contains the same PV systems that are available in the Solar.web demo portal. Using the trial keys, 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 SWQAPI customers need to have an active Solar.web account (<https://www.solarweb.com>) and complete and sign an order form. More details and contact information can be found at <https://www.fronius.com/en/solarweb-query-api>.

After completing the registration, please make yourself familiar with the key management. You need to create at least one key in the Solar.web user 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 the 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 either by registering a new PV system in Solar.web for this account or by adding guest or supervisor permission for this account 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.

## Beta environment

For preview of new functionalities, Fronius provides a Beta environment (available at the URL <https://swqapi-beta.solarweb.com/>). It works like the Production environment, i.e. uses the same API keys to access the same PV systems, so you can test your applications against it.

## 2.2.3 Data plans and pricing

Trial 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

Information about pricing and data plans, more details and example calculations can be found at <https://www.fronius.com/en/solarweb-query-api>. Please contact your local sales representative for further details about pricing.

## 3 General Information

### 3.1 Key management

API users must include valid API keys in the header of their requests. These keys are generated by authorized users within the [Fronius API Management Portal](#). Each authorized user can have multiple API keys, which are restricted according to their permissions in Solar.web.

API keys have the following attributes:

Access key ID	A unique ID for the API key, e.g. "FKIAFEF58CFEFA94486F9C804CF6077A01AB". Access keys are 36 characters long and start with the "FKIA" prefix.
Access key value	A secret value (GUID), e.g. "47c076bc-23e5-4949-37a6-4bcfcf8d21d6", which you need to know for authorization of API calls. 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.
Active status	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.
Expiry date	You can set a validity period for a key, e.g. if you want to enforce key renewals. 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.
Last used date	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 require the access key ID and access key value to be included in the HTTP header.

Examples:

#### HTTP header example

```
GET https://api.solarweb.com/swqapi/pvsystems HTTP/1.1
AccessKeyId: FKIAFEF58CFEFA94486F9C804CF6077A01AB
AccessKeyValue: 47c076bc-23e5-4949-37a6-4bcfcf8d21d6
```

#### 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"
```

## 3.2 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.

### JWT token attributes

JWT token value	<p>A long string, identifying the customer and providing access to him, for example:</p> <pre>eyJ4NXQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiLCJraWQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoljoiS2hLVZSc0lPXY1tWDhvZkZJSzdJZyIsImF1ZCI6ImljNHhtcElyVnlyR2phcUlraGoxXJEOFZ6VWEiLCJzdWliOiJodWV0dG5lci50aG9tYXNybmRAZnJvbmI1c3y5jb20iLCJuYmYiOiJlODUyOTQ2NjcslmF6cCI6ImljNHhtcElyVnlyR2phcUlraGoxXJEOFZ6VWEiLCJhbXliOiIsicGFzc3dvcmQiXSwiaXNzIjoiJHtjYXJib24ucHJvdG9jb2x9OlwvXC8ke2NhcmJvbi5ob3N0fVwvb2F1dGgyXC9vaWRjZGlzY292ZXJ5liwiZXhwljoxNTg1Mjk4MjY3LCJpYXQiOiJlODUyOTQ2Njd9.HUXi1sySzyLqx2e0dLprOsszi-Yil3nGNB4GZDDwlvVHUHC4s6ED8BqfvkfFn3s45LkvJQEvqb_Wd3QtMGnzOLEZ3RdK3A8GWdsDChVq_nzIP4FGC6b5IPoz9Xi6mH_pcxt36rzA2-vjl_e6cTOrTXslelzOjZVNSZRAJ4-A5HpmEuvraoArAGUqc_yTntbfALhfJQkfsjoDAJRAfZXLknTvDKm2vMd0-uXjTQHM2dKAWGAz6r39cLQ24sFIIC7MDglp4GpNVBCLFSNzkK7mV3fSEQvgIdAFMhEP4CY4IMTItLxdfRKxcf5SA7o2fUO-_710frdFYvrkesorDCiyfg</pre>
JWT expiry date	<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>
Refresh token	A token which can be used to refresh a JWT.



GET <https://{{SWQAPIURL}}/swqapi/pvsystems> **Send** **Save**

Params **Authorization** Headers (10) Body Pre-request Script Tests Settings Cookies Code

**TYPE**  
Bearer Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

**Preview Request**

! Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Token: `eyJ4NXQiOiJOR1psTUR5bFkyRXlaR1kzTkRjNU1UVm1PR0UwWWp...`

**Body** Cookies Headers (5) Test Results Status: 200 OK Time: 45ms Size: 22.79 KB **Save Response**

Pretty Raw Preview Visualize **BETA** JSON

```
1  {
2    "pvSystems": [
3      {
4        "pvSystemId": "08ddaa09-d8d0-495d-8d1c-eb66fc7188fa",
5        "name": "Fronius AT SAT 1PN H",
6        "address": {
7          "country": "AT",
8          "zipCode": "4600",
9          "street": "Günter Fronius Straße 1",
10         "city": "Thalheim bei Wels",
11         "state": "OO"
12       },
13       "pictureURL": "https://www.solarweb.com/Image/Show?pvSystemId=08ddaa09-d8d0-495d-8d1c-eb66fc7188fa&pictureId=e594903f-49a5-ed47-9f5a-4d685da7a233".
```

```
GET https://api.solarweb.com/swqapi/pvsystems HTTP/1.1
AccessKeyId: FKIAFEF58CFEFA94486F9C804CF6077A01AB
AccessKeyValue: 47c076bc-23e5-4949-37a6-4bcfcf8d21d6
Authorization: Bearer
eyJ4NXQiOiJPVEZsT1RCbE9HSmhZak15TlRfNU5XVTJpVGd6TnpVd04yTmp0VFV5WlRFeU1tRTNZVEZoTmciLCJ3raWQ0iJPVEZsT1RCbE9HSmhZak15TlRfNU5XVTJpVGd6TnpVd04yTmp0VFV5WlRFeU1tRTNZVEZoTmciLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJzdHJlZXRfYWRkcWVzcyI6IkZyb25pdXNwbGF0eiAxIiwiaWF0Ij0iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhcnd1Yl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMjAtMDUwMTJUMjI6MDA6MDBaIiwic2lkIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVudG9yTG9jYXN1Ym91L0sImIzcyI6Imh0dHBz0lwlwXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2t1biIsImNvbRlR5Y3RfYWQ0iJmYzY0NWVjNi00NmQ1LWU5MTEtOTEyOS0wMDUwNTZHMjYxNDAlLCJzb2xhc
```

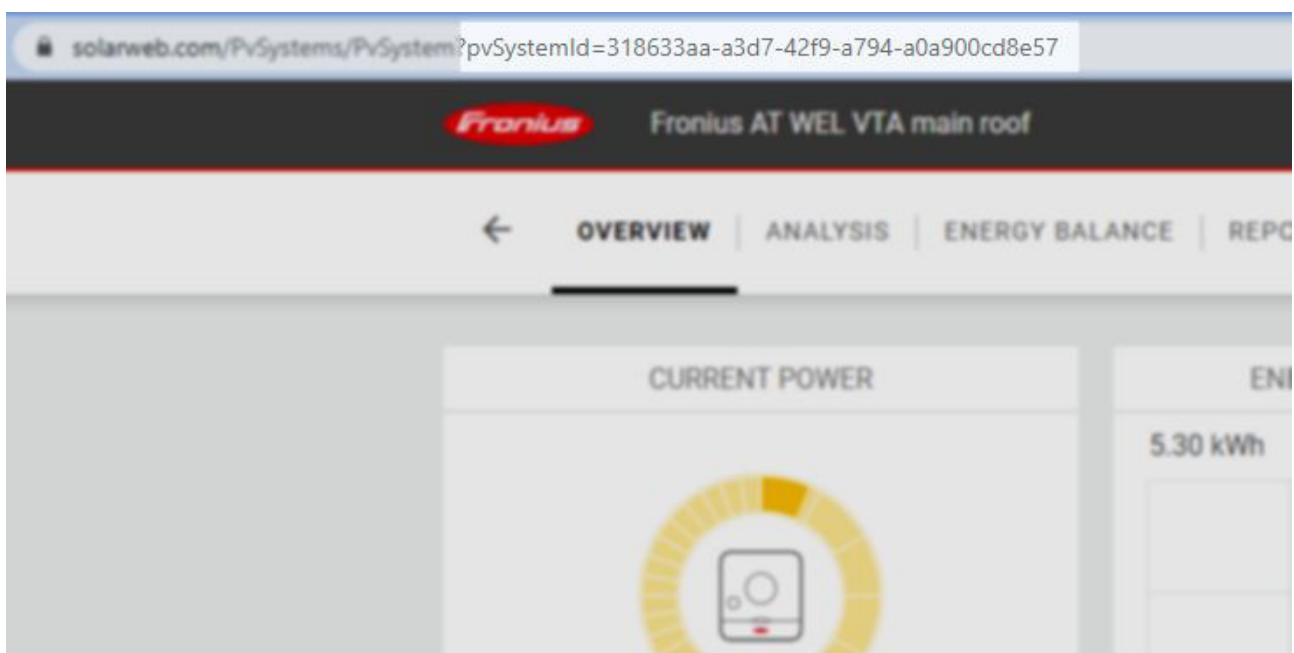
[illegible]



```
iQWxleGFuZGVyIiwic29sYXJ3ZWJfcHJlbWl1bV9yb2xlIjoimSIsIm5vbmNlIjoiyXNkZiIsImRhdGFfY29u
dGFjdF9jb21wbGV0ZSI6InllcyIsImF1ZCI6Im1mX285aVRBeUtlbU5MUVRhNlNwNkhZb25DSWEiLCJjX2hhc
2giOiJFWDZS3NwaUxHY2tDc3RFNW02VmtnIiwibmJmIjoixNTcyMjU0MDg3LCJjb3VudHJ5X2lzb19jb2RlIj
oiQVQiLCJsb2NhdGlvbiI6IldlbHMlLCJwb3N0YWxfY29kZSI6IjQ2MDAiLCJmYW1pbHlfbmFtZSI6IktyZW5
odWJlciIsImRhdGFfYWNjb3VudF92YWxpZCI6InllcyJ9.Qzywri3WV6RHjv9Ng9fDCkzxPRprNBx63bvoGk
bxa0hhyJVIdT132ylCyvmp84t_agvRG7Gk8HFcN5XrWcJ126mM-
CMQ5VFSLIyCmZm_vwoPXu0sk_pC1clX890WcqKDy3uaA3UdLLGY0ke8kg-
ueQIxfkme1gSb2q0LEATaS8wdYYW-ODakMFd7zQvLRlJnXVXICeHwXrZu68fDRjdUlulu_13Ggi-
yHrtZTji_My_J57iMgJHTLaf7Gw3QzMMaZ85Kyz3jvqQgLFPylZgNBoz6ztzEdYd5Fhu-
kccibD662q9D7R5PHN88zY9ieVL9RYPyJZ5Rjk6qIYxb5Km6w"
```

### 3.3 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 (highlighted 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).



### 3.4 Pagination

When returning many results, Fronius APIs make use of HATEOAS principles to support pagination. Additionally, the APIs return 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://swqapi.solarweb.com/pvsystems-list?offset=0&limit=50",
    "prev": null,
    "self": "https://swqapi.solarweb.com/pvsystems-list?offset=0&limit=50",
    "next": "https://swqapi.solarweb.com/pvsystems-list?offset=50&limit=50",
    "last": "https://swqapi.solarweb.com/pvsystems-list?offset=150&limit=50",
    "totalItemsCount": 173
  }
}
```

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

### 3.5 Date and time formats

SWQAPI supports extended UTC time formats (ISO 8601).

The principle format is either "yyyyMMddThhmmssTZD" 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

### 3.5.1 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 swqapi.solarweb.com/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 swqapi.solarweb.com/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 swqapi.solarweb.com/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 swqapi.solarweb.com/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
```

### 3.5.2 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)
```

## 4 API Management Portal

The Fronius API Management is available at <https://api-mgmt.solarweb.com/>.

### 4.1 Development section

#### 4.1.1 Release information

An overview about the latest release API version is provided on the **Release information** tab.

The screenshot shows the 'Release information' tab in the Fronius REST API interface. The left sidebar contains a navigation menu with categories: Development (System availability, Release information, Key management, Message queues) and Account (Monthly reports, Remote control, Billing information). The main content area displays a list of releases with columns for date, version, and changes. The 'Query' tab is selected.

Date	Version	Changes
2024-10-12	SWQAPI 2.3.4	Bugfixes Internal Improvements
2024-11-28	SWQAPI 2.3.3	Bugfixes
2024-11-27	SWQAPI 2.3.2	Bugfixes
2024-11-21	SWQAPI 2.3.1	Bugfixes
2024-10-30	SWQAPI 2.1.11	Internal Improvements
2024-10-17	SWQAPI 2.1.10	

© 2024 Fronius International GmbH | Imprint | Terms of use | Privacy | Cookies

## 4.1.2 System availability

The operational history for the last few months can be checked on the **System availability** tab. Here you can see the average system availability for the **last 30 days**, **last 6 months** or **last 12 months**.

The screenshot shows the 'System availability' tab in the Fronius REST API interface. The left sidebar is the same as in the previous screenshot. The main content area displays a bar chart showing system availability over time. The 'Query' tab is selected. The chart shows a baseline of 99% and bars representing daily availability for the last 11 days of December. To the right of the chart, there are buttons for 'Last 30 Days', 'Last 6 Months', and 'Last 12 Months'. Below the chart, there is a section for 'Planned maintenance' which states: 'There is currently no maintenance planned or active.' A link to [solarweb.status.io](https://solarweb.status.io) is provided for more information.

For more information, go to: [solarweb.status.io](https://solarweb.status.io)

Planned maintenance

There is currently no maintenance planned or active.

© 2024 Fronius International GmbH | Imprint | Terms of use | Privacy | Cookies

On the right hand side, there are two more sections.

- A link to [solarweb.status.io](https://solarweb.status.io) is provided.

Status.io is an external service which also works when Solar.web is down. If an incident happens and Solar.web is down, Fronius will post status updates here, because Solar.web services may not be able to show appropriate messages.

Additionally, Status.io keeps incident history information.

- Announcements for **Planned maintenance** will be published here as well.

### 4.1.3 Key management

For managing API keys, go to **Key management**. Here you can view and manage your keys. Please note that the time information (creation date, expiry date, and last used date) is given in UTC zulu time.

**Key management**

+ New key

Show only active keys

Search...

Status	Key ID	Name	Description	Created on	Expires on	Last used
Active	FKIA6B3EEB7ED8A1406DA1184B303FBC3B84			2023-11-02 12:46 UTC		2024-12-03 08:25 UTC
Active	FKIAD1FCE758D6B44E58BD9EE8BCFF99A1D2			2024-04-03 07:03 UTC		2024-12-05 06:18 UTC
Active	FKIAF6E83676899CA067B73CBED1ADF0DA3C			2024-07-11 10:09 UTC		2024-07-11 10:10 UTC
Active	FKIA0E7B64AE3A004B29BCD7A80F7E901AE1			2024-02-13 09:17 UTC		2024-12-12 13:57 UTC
Active	FKIA97DA361F4E8AA4CAE9D6218C5AFCE5149			2022-07-18 13:13 UTC		2022-07-18 13:15 UTC
Active	FKIAD4E3952781BE44F08A7D7CE8056C30C7			2022-05-20 07:37 UTC		2022-05-20 08:08 UTC
Active	FKIA593C8654AA4BA55A77AEEF9E3A9572			2023-08-22 09:11 UTC		2024-07-11 05:24 UTC
Active	FKIAE13F614B27F04EEE153587702965998			2022-10-25 12:18 UTC		
Active	FKIA2BF21A70AA094B31A7A021F897C028D0			2022-08-31 12:31 UTC		2024-03-22 10:31 UTC

© 2024 Fronius International GmbH

Imprint | Terms of use | Privacy | Cookies

https://api-mgmt.solarweb.com/home/key-management

#### Actions:

- If you want to create a new key, please press the **+ New key** button. This will create a new key. You can either copy and paste the API key ID and its secret value, or download a JSON file containing the API key ID and its secret value.
- You can give a name and description to each key. To do so, please press the three dots in the 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 menu column 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, press the three dots in the menu and then select **Deactivate**. To activate an inactive key, use **Activate** in the same menu.

- Only expired or inactive keys can be deleted. To do so, please press the three dots in the menu column and then select **Delete**.

#### Recommendations:

- If you have multiple developers, create separate keys for each developer. Create another key for automated testing, 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.

## 4.2 Account section

### 4.2.1 Monthly reports

You can get information about monthly consumption on this page.

**Fronius REST API** Test Query in Swagger Test Control in Swagger

Development System availability Release Information Key management Message queues Account **Monthly reports** Remote control Billing Information

### Monthly reports

Query Control

Period ↑	Data points	
2025-01	0	↓
2024-12	9840	↓
2024-11	698	↓
2024-10	1063	↓
2024-09	626	↓
2024-08	0	↓
2024-07	13000	↓
2024-06	1800	↓
2024-05	0	↓
2024-04	247	↓

© 2024 Fronius International GmbH Imprint Terms of use Privacy Cookies f @ y x

## 4.2.2 Billing information

For double-checking contractual information, go to **Billing information**. Here you can view your contract start and end time. Some licenses can be initially unlimited, indicated by the infinite symbol in the **End date** column (as in the example below).

**Billing information**

Customer information Edit

Purchase order number

Contract information

Status	Start date ↑	End date
Active	2022-05-03	∞

© 2024 Fronius International GmbH | Imprint | Terms of use | Privacy | Cookies

### Actions:

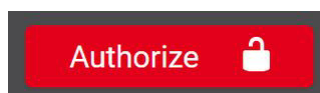
- If your Finance/Purchase department needs a **Purchase order number** on the invoice, you can enter one. To do so, press the **Edit** button on the right and enter the purchase order number in the following dialog.

## 5 Swagger UI

If you'd like to explore the API and experiment with it without writing any code, you can do so in the Swagger UI.

When working with the Swagger UI, you need to enter the API keys only once.

Press the *Authorize* button in Swagger UI.



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

**Available authorizations** ✕

**AccessKeyId (apiKey)**  
AccessKeyId: FKIA7887176DB53301DD759A472495059064  
Name: AccessKeyId  
In: header  
Value:

**AccessKeyValue (apiKey)**  
AccessKeyValue: a8c5aa35-2a41-4814-b85a-3330d700ba79  
Name: AccessKeyValue  
In: header  
Value:

**Bearer (apiKey)**  
JWT Authorization header using the Bearer scheme. Enter 'Bearer' [space] and then your token in the text input below. Example: 'Bearer ey123456...'  
Name: Authorization  
In: header  
Value:



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

Available authorizations

AccessKeyId (apiKey)

Authorized

AccessKeyId: FKIA7887176DB53301DD759A472495059064

Name: AccessKeyId

In: header

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

Logout

AccessKeyValue (apiKey)

Authorized

AccessKeyValue: a8c5aa35-2a41-4814-b85a-3330d700ba79

Name: AccessKeyValue

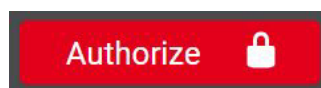
In: header

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

Logout

Bearer (apiKey)

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



### 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.

Impersonation / Bearer key:

If you want to use the impersonation feature you need to get an JWT for the user you want to impersonate (by using the /jwt endpoint first, see section *Impersonate: Receive a JWT using the Fronius login*).

Press the *Authorize* button in Swagger UI again. Under Bearer enter the JWT and press Authorize. Close the dialog using the x button on the top right corner.

**Available authorizations** ✕

---

**AccessKeyValue (apiKey)**

Authorized

AccessKeyValue: a8c5aa35-2a41-4814-b85a-3330d700ba79  
 Name: AccessKeyValue  
 In: header  
 Value: \*\*\*\*\*

**Logout**

---

**Bearer (apiKey)**

Authorized

JWT Authorization header using the Bearer scheme. Enter 'Bearer' [space] and then your token in the text input below. Example: 'Bearer ey123456...'

Name: Authorization  
 In: header  
 Value: \*\*\*\*\*

**Logout**

Please note that authentication is reset if a page refresh is done.

The Swagger UI for the SWQAPI can be found here: <https://swqapi.solarweb.com/index.html>

## 6 API Reference

### 6.1 User impersonation calls

#### 6.1.1 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.

### Developer best practice

Please use JWT tokens carefully:

- A JWT token is valid for one hour.
- After this hour is passed you will get a 401 http error code. Please use the refresh mechanism to get an updated JWT token instead of creating a new one.
- Only if the refresh call gives you an error again, you should generate a new JWT token using this call.
- Note: Refresh tokens are valid for 40 days.

Make use of the scope parameter!

- Use a unique scope for you application.
- If you implement an application for mobile devices, a good idea is to use UUIDs to prevent that one login on device A logs out users on device B.

### Methods

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

### Filters and parameters

Filter	Description
?scope=<scope>	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). Scopes allow users to be logged in from multiple devices and in Solar.web in parallel.

### Example calls

```
POST swqapi.solarweb.com/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:

Type	Objects
token information	<ul style="list-style-type: none"> <li>• refreshToken (String)</li> <li>• jwtToken (String)</li> <li>• jwtTokenExpiration (UTC time)</li> </ul>

## Example responses

```
{
  "refreshToken": "98a47454-b650-34b8-9a8c-27adae447ab7",
  "jwtToken":
  "eyJ4NXQiOiJ0R1psTURScFkyRXlaR1kzTkrjNU1UVm1PR0UwWpGaVpXWTBaamcxWVd0a09EWmtNRE5rTVEiLCJraWQiOiJ0R1psTURScFkyRXlaR1kzTkrjNU1UVm1PR0UwWpGaVpXWTBaamcxWVd0a09EWmtNRE5rTVEiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiS2hLVZSc0lPXy1tWDhvZkZJSzdJZyIsImF1ZCI6ImljNHhtcEiYVnlyR2phcUlraGoxbXJE0FZ6VWEiLCJzdWIiOiJodWV0dG5lci50aG9tYXNybmRAZnJvbmlcy5jb20iLCJuYmYiOiJlODUyOTQ2NjcsImF6cCI6ImljNHhtcEiYVnlyR2phcUlraGoxbXJE0FZ6VWEiLCJhbXUiOi0lsicGFzc3dvcmQiXSwiaXNzIjoiejhtjYXJib24ucHJvdG9jb2x90lwvXC8ke2NhcmJvbi5ob3N0fVwvb2F1dGgyXC9vaWRjZGlzY292ZXJ5IiwiaXhwIjojNTg1Mjk4MjY3LCJpYXQiOiJlODUyOTQ2Njd9.HUXi1sySzyLqx2e0dLpr0sszi-YiI3nGNB4GZDDwIwVHUHC4s6ED8BqfvkfFn3s45LkvJQEvqb_Wd3QtMGnz0LEZ3RdK3A8GWdsDChVq_nzLP4FGC6b5lPoz9Xi6mH_pcxt36rzA2-vjl_e6cT0rTXsIeIz0jZVNSZRAJ4-A5HpmEuvraoArAGUqc_yTntbfALhfJQkfsjoDAJRAfZXLknTvDKm2vMd0-
```

```
uXjTQHM2dKAWGAz6r39cLQ24sFIIC7MDgIp4GpNVBCLFSNzkK7mV3fSEQvgIdAFMhEP4CY4lMTItLxdfRKxcF
5SA7o2fU0-_710frdFYvrkesorDCiyfg",
  "jwtTokenExpiration": "2020-03-27T08:37:48.8710788Z"
}
```

## 6.1.2 Impersonate: Refresh a JWT

### Use cases for web developers

- I want to refresh an expired JWT token.

#### Developer best practice

Please use refresh tokens instead of logging in again and again

- A refresh token is valid for 40 days and allows you to create new JWT tokens without having the user to provide the login credentials.
- Creating new login tokens are costly. Therefore each refresh saves computing power and storage memory on the Fronius side, thus also login performance.

#### Notes:

- After a refresh, you get a new refresh token. The old refresh token gets invalidated.

### Methods

Method	End point and objects	Event name	Description
PATCH	/iam/jwt/{refresh-token}	RefreshJwt	Refreshes a JWT and refresh token pair.  Please note that the old refresh token gets invalidated by this call.

### 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> <p>Must not include any scope values not originally granted, and if omitted is treated as equal to the originally granted scope.</p> </div>

## Example calls

```
PATCH swqapi.solarweb.com/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	<ul style="list-style-type: none"><li>String</li></ul>
jwtToken	<ul style="list-style-type: none"><li>String</li></ul>
jwtTokenExpiration	<ul style="list-style-type: none"><li>String (UTC time)</li></ul>

### Example responses

[illegible]

```

Z3VhZ2U0iJkZSI6ImxvY2FsaXR5IjoizGUlLCJzb2xhcndlYl91c2VyaWQiOiI4RkM3NzVFR03Qj1BLTREM
zYtQTcwNi1BQTKzMDA5RjNGRjkiLCJncm91cHM0IjJbnRlcm5hbFwvZXZlcnlvbmUiLCJnaXZlbn9uYW1lIj
oiQWxleGFuZGVyIiwic29sYXJ3ZWJfcHJlbWl1bV9yb2x1IjoimSIsIm5vbmNlIjoiyXNkZiIsImRhdGFfY29
udGFjdF9jb21wbGV0ZSI6InllcyIsImF1ZCI6Im1mX285aVRBeUtlbU5MUVRhNlNwNkhZb25DSWEiLCJjX2hh
c2giOiJFWDZS3NwaUxHY2tDc3RFNW02VmtnIiwibmJmIjoxNTcyMjU0MDg3LCJjb3VudHJ5X2lzb19jb2RlI
joiQVQiLCJsb2NhdGlvbiI6IldlbHMiLCJwb3N0YWxfY29kZSI6IjQ2MDAiLCJmYW1pbHlfbmFtZSI6IktyZW
5odWJlciIsImRhdGFfYWNjb3VudF92YWxpZCI6InllcyJ9.Qzywri3WV6RHjv9Ng9fDCkzxPRprNBx63bvoG
kboxa0hhyJVIdT132ylCyvmp84t_agvRG7Gk8HFcn5XrWcJ126mM-
CMQ5VFSLIyCmZm_vwoPXu0sk_pC1clX890WcqKDy3uaA3UdlLGyOke8kg-
ueQIxfkme1gSb2q0LEATaS8wdYYW-ODakMFd7zQvLRlJnXVXICeHwXrZu68fDRjdUlulu_13Ggi-
yHrtZTji_My_J57iMgJHTLaf7Gw3QzMMaZ85Kyz3jvqQgLFPylZgNBoz6ztzEdYd5Fhu-
kccibD662q9D7R5PHN88zY9ieVL9RYPyJZ5Rjk6qIYxb5Km6w",
  "jwtTokenExpiration": "2019-12-18T13:24:19Z"
}

```

## 6.1.3 Impersonate: Revoke a JWT

### Use cases for web developers

- I want to revoke a refresh token, e.g. when the user logs out from my app

Note: Technically it is not possible to revoke the JWT token, too, but it expires automatically within one hour. If you want to "revoke" a JWT token, please discard it instead.

### Methods

Method	End point and objects	Event name	Description
DELETE	/iam/jwt/{refresh-token}	RevokeJwt	Revokes a JWT refresh token

### Filters and parameters

n/a

### Example calls

```

DELETE swqapi.solarweb.com/iam/jwt/98a47454-b650-34b8-9a8c-27adae447ab7
// revokes an existing refresh token

```

### Response objects

n/a

### Example responses

n/a



## 6.2 Generic information calls

### 6.2.1 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	/info/release	GetInfoRelease	Retrieves version and release date information about the REST API.

#### Filters and parameters

n/a

#### Example calls

```
GET swqapi.solarweb.com/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"
}
```

## 6.2.2 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	/info/user	GetInfoUser	Returns information about either the SWQAPI user or the impersonated user.

### Filters and parameters

Filter	Description
?details=<"name", "address", "contractInformation", "settings", "accountAttributes", "favoritePvSystems">	Specifies which details of the JSON response object shall be returned by the GetUserInfo call.

### Example calls

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

GET swqapi.solarweb.com/info/user?details=name,address
// returns user's name and address 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.", "Mrs."</li> <li>• firstName (String)</li> <li>• lastName (String)</li> </ul> </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>• 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", or "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> <li>• premiumMembershipEnddate (String; DateTime)</li> </ul> </li> <li>• favoritePvSystems (Array of GUIDs, optional)</li> </ul>

### Example responses

```
{
  "name": {
    "title": "Mr.",
    "firstName": "John",
    "lastName": "Doe"
  },
  "address": {
    "street": "Froniusplatz 1",
    "zipCode": "4600",
    "city": "Wels",
    "state": "Upper Austria",
    "country": "Austria"
  },
  "contractInformation": {
```

```
    "telephone": "+123456789",
    "email": "john.doe@SWQAPI.com"
  },
  "settings": {
    "timeZone": "Europe/Berlin",
    "dateFormat": "DD.MM.YYYY",
    "timeFormat": "24h",
    "language": "en"
  },
  "accountAttributes": {
    "premiumMembership": true,
    "termsAcceptedLatest": true,
    "premiumMembershipEnddate": "2024-12-31T23:59:59Z",
    "termsAcceptedVersion": 2
  }
}
```

In the JSON object part "contactInformation" is unfortunately a typo: For not breaking the API, we keep "contractInformation" (with the "r"). Please excuse any inconvenience.

## 6.3 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

### 6.3.1 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	/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	/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.

Filter	Description
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices that a PV system should contain: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>
?meteo=<level>	Filters for systems with extended (meteo=pro) weather data or with just basic (meteo=light) weather data.

Multiple filters will be processed with OR logic, e.g. ?type=sensor,battery will return PV systems that contain at least a sensor OR a battery.

### Example calls

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

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

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

GET swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689
// returns metadata of a specific PV system

GET swqapi.solarweb.com/pvsystems?meteo=pro
// returns metadata of all PV systems which have a "pro" weather information
available (which is normally only one PV system in the account)
```

### 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, ISO 3166-1 alpha-2 format)</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": "00"
        },
        "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"
    }
]
}

```

### 6.3.2 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	/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=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>
?meteo=<level>	Filters for systems with extended (meteo=pro) weather data or with just basic (meteo=light) weather data.

Multiple filters will be processed with OR logic, e.g. ?type=sensor,battery will return PV systems that contain at least a sensor OR a battery.

## Example calls

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

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

## Response objects

JSON object answer construction:

Type	Objects
n/a	<ul style="list-style-type: none"> <li>count (Number)</li> </ul>

## Example responses

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

### 6.3.3 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	/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. The limit parameter is limited to 1000. If limit is not set, 50 is being used.
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>
?meteo=<level>	Filters for systems with extended (meteo=pro) weather data or with just basic (meteo=light) weather data.

Multiple filters will be processed with OR logic, e.g. ?type=sensor,battery will return PV systems that contain at least a sensor OR a battery.

## Example calls

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

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

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

GET swqapi.solarweb.com/pvsystems-list?meteo=light
// returns PV system IDs which have "light" weather information available
```

## Response objects

JSON object answer construction:

Type	Objects
n/a	<ul style="list-style-type: none"> <li>• pvSystemIds (Array of Strings)</li> </ul>

## Example responses

```
{
  "pvSystemIds": [
    "7eb46213-e165-44a2-82aa-88969f11847f",
    "2b16033e-b842-4bf5-ae10-a3e4a14d297b",
    "f51f544a-65f5-40e7-add5-0256fc3ca660",
    "9e52c557-5fd5-41e0-ac6d-ad62f4556a27",
    "3e0f7c5e-fa06-4346-937f-b21d6c903a9b",
    "e187f87b-98ff-475e-b6fd-bfa3445cd848",
    "0ae70a7d-6448-4fc7-9425-8610ddccd0f0",
    "28603795-20ac-4456-a6cc-4bc6002deb56",
    "fdc077a9-ca23-4edc-866a-99d38f4d8042",
    "db2252b2-8ade-438c-8da5-889cccfa4036",
    "e5bf41b6-d2df-446e-83d6-a41c0100041b",
    "bcb9c5b6-cae8-40d2-9a4e-865f1365e29d",
    "fa5f45cd-9b86-4709-bda8-4e0183c4f379",
    "cc648cd7-70df-4e64-a215-ac2d2e556508",
    "ef6fe5dc-4bb9-4d18-97eb-a1b600b6cc3e",
    "046173e7-2d74-4e9b-ade3-a5e7014db4dc",
    "26d01aef-fe32-4676-a9b9-4f59263c2ba5"
  ],
  "links": {
    "first": "https://api.solarweb.com/swqapi/pvsystems-list?offset=0&limit=1000",
    "prev": null,
    "self": "https://api.solarweb.com/swqapi/pvsystems-list?offset=0&limit=1000",
    "next": null,
    "last": "https://api.solarweb.com/swqapi/pvsystems-list?offset=0&limit=1000",
    "totalItemsCount": 17
  }
}
```

totalItemsCount does not count items within this response, it counts overall available number of systems.

## 6.3.4 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	/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.
GET	/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=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices that a PV system should contain: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>
?isActive=<state>	Filters for active (isActive=true) or inactive (isActive=false) devices only.

### Example calls

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

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

```
GET swqapi.solarweb.com/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 swqapi.solarweb.com/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>• deviceTypeDetails (String)</li> <li>• dataloggerId (String)</li> <li>• nodeType (Integer)</li> <li>• numberMPPTTrackers (Number)</li> <li>• numberPhases (Number - 1 to 3)</li> <li>• peakPower (Dictionary) <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>• 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 (String)</li> <li>• dataloggerId (String)</li> <li>• maxChargePower (Number)</li> <li>• maxDischargePower (Number)</li> <li>• maxSOC (Number)</li> <li>• minSOC (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>



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>• 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>• dataloggerId (String)</li> <li>• sensors (Array of objects) <ul style="list-style-type: none"> <li>• sensorType (String)</li> <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>• dataloggerId (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>• sensorType (String)</li> <li>• sensorName (String - "Temperature")</li> <li>• isActive (Boolean)</li> <li>• activationDate (String, UTC timestamp)</li> <li>• deactivationDate (String, UTC timestamp)</li> </ul> </li> </ul>

Type	Objects
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>• deviceCategory (String)</li> <li>• deviceLocation (String)</li> <li>• serialNumber (String)</li> <li>• dataloggerId (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> </ul>

Type	Objects
EVCharger	<ul style="list-style-type: none"> <li>• deviceType (String - "EVCharger")</li> <li>• deviceId (String)</li> <li>• deviceName (String - "WattPilot")</li> <li>• deviceManufacturer (String - "Fronius")</li> <li>• serialNumber (String)</li> <li>• dataloggerId (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>• isOnline (Boolean)</li> </ul>

Type	Objects
Datalogger	<ul style="list-style-type: none"><li>• deviceType (String - "Datalogger")</li><li>• deviceId (String)</li><li>• deviceName (String)</li><li>• deviceManufacturer (String - "Fronius")</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>• dataloggerId (String)</li><li>• isOnline (Boolean)</li></ul>

Example responses

Example for an inverter

```
{
  "deviceType": "Inverter",
  "deviceId": "aff28818-5cd0-4075-8def-a3e3014b43c2",
  "deviceName": "Symo Hybrid 5.0-3-S",
  "deviceManufacturer": "Fronius",
  "serialNumber": null,
  "deviceTypeDetails": "Fronius Symo Hybrid 5.0-3-S",
  "dataloggerId": "239.14294",
  "nodeType": 97,
  "numberMPPTTrackers": 1,
  "numberPhases": 3,
  "peakPower": {
    "dc1": 6510.0
  },
  "nominalAcPower": 5000.0,
  "firmware": {
    "updateAvailable": false,
    "installedVersion": null,
  }
}
```

```

    "availableVersion": "fro27372"
  },
  "isActive": true,
  "activationDate": "2014-11-14T00:00:00Z",
  "deactivationDate": null
}

```

#### Example for another inverter

```

{
  "deviceType": "Inverter",
  "deviceId": "8b6a8f4b-57cb-48c3-899f-71ca78f14627",
  "deviceName": "Primo 5.0-1",
  "deviceManufacturer": "Fronius",
  "serialNumber": "27185462",
  "deviceTypeDetails": "Fronius Primo 5.0-1",
  "dataLoggerId": "240.196164",
  "nodeType": 97,
  "numberMPPTTrackers": 2,
  "numberPhases": 1,
  "peakPower": {
    "dc1": 5000.0,
    "dc2": null
  },
  "nominalAcPower": 5000.0,
  "firmware": {
    "updateAvailable": false,
    "installedVersion": null,
    "availableVersion": "fro27372"
  },
  "isActive": true,
  "activationDate": "2021-09-22T00:00:00Z",
  "deactivationDate": null
}

```

#### Example for a sensor

```

{
  "deviceType": "Sensor",
  "deviceId": "484d8603-64db-44d3-9b54-3de5895054c1",
  "deviceName": "Sensor Card / Box (2)",
  "deviceManufacturer": "Fronius",
  "firmware": {
    "updateAvailable": false,
    "installedVersion": "",
    "availableVersion": ""
  },
  "isActive": true,
}

```

```

"activationDate": null,
"deactivationDate": null,
"dataloggerId": "240.347585",
"sensors": [
  {
    "sensorName": "Insolation",
    "sensorType": "Insolation",
    "isActive": true,
    "activationDate": "2014-05-29T00:00:00Z",
    "deactivationDate": null
  },
  {
    "sensorName": "Temperature1",
    "sensorType": "Temperature",
    "isActive": true,
    "activationDate": "2014-05-29T00:00:00Z",
    "deactivationDate": null
  }
]
}

```

#### Example for a datalogger (SnapInverter DataManager)

```

{
  "deviceType": "Datalogger",
  "deviceId": "2e303432-3931-3136-3634-000000000000",
  "deviceName": "Datalogger",
  "deviceManufacturer": "Fronius",
  "firmware": {
    "updateAvailable": true,
    "installedVersion": "3.16.7-1",
    "availableVersion": "3.18.7-1"
  },
  "isActive": true,
  "activationDate": "2021-09-22T15:55:10Z",
  "deactivationDate": null,
  "dataloggerId": "240.196164",
  "isOnline": true
}

```

#### Example for a datalogger (Gen24 Pilot)

```

{
  "deviceType": "Datalogger",
  "deviceId": "6f6c6978-2574-2e30-3524-2d3830363933",
  "deviceName": "Datalogger",
  "deviceManufacturer": "Fronius",
  "firmware": {

```

```

        "updateAvailable": null,
        "installedVersion": null,
        "availableVersion": null
    },
    "isActive": true,
    "activationDate": "2020-01-11T1100:00Z",
    "deactivationDate": null,
    "dataLoggerId": "pilot-0.5d-80697371431225991_1593083600",
    "isOnline": true
}

```

#### Example for an Ohmpilot

```

{
  "deviceType": "Ohmpilot",
  "deviceId": "be3dac1d-8c30-4e0a-ae88-2649407cb593",
  "deviceName": "Ohmpilot",
  "deviceManufacturer": "Fronius",
  "serialNumber": "27193272",
  "firmware": {
    "updateAvailable": true,
    "installedVersion": "40000000051",
    "availableVersion": "1.0.25.3"
  },
  "isActive": false,
  "activationDate": null,
  "deactivationDate": "2017-08-16T17:05:01Z",
  "dataLoggerId": "240.196164",
  "sensors": [
    {
      "sensorName": "Temperature",
      "sensorType": null,
      "isActive": true,
      "activationDate": null,
      "deactivationDate": null
    }
  ]
}

```

#### Example for a Wattpilot

```

{
  "deviceType": "EVCharger",
  "deviceId": "cee3e54d-0191-4700-8504-aea000d0d839",
  "deviceName": "Car Charger 2 ",
  "deviceManufacturer": "Fronius",
  "serialNumber": "32719074",
  "firmware": {

```

```

        "updateAvailable": false,
        "installedVersion": null,
        "availableVersion": null
    },
    "isActive": true,
    "activationDate": "2000-01-01T00:00:00Z",
    "deactivationDate": null,
    "dataLoggerId": "240.196164",
    "isOnline": true
}

```

#### Example for a battery

```

{
  "deviceType": "Battery",
  "deviceId": "68f9a0d4-c50a-43e7-84b0-11c81ef98657",
  "deviceName": "",
  "deviceManufacturer": "Fronius",
  "serialNumber": "25441120",
  "capacity": 9600,
  "dataLoggerId": "239.34376",
  "maxChargePower": 6656,
  "maxDischargePower": 6656,
  "maxSOC": 100,
  "minSOC": 0,
  "firmware": {
    "updateAvailable": false,
    "installedVersion": "",
    "availableVersion": ""
  },
  "isActive": true,
  "activationDate": "2023-05-29T11:30:41Z",
  "deactivationDate": null
}

```

#### Example for a Smart Meter

```

{
  "deviceType": "SmartMeter",
  "deviceId": "957d94d1-229c-4fab-be84-f06e21c8811a",
  "deviceName": "METER_CAT_OTHER",
  "deviceManufacturer": "",
  "deviceCategory": "Primary Meter",
  "deviceLocation": "Grid",
  "serialNumber": "",
  "dataLoggerId": "239.14294",
  "firmware": {
    "updateAvailable": false,

```

```

    "installedVersion": "",
    "availableVersion": ""
  },
  "isActive": true,
  "activationDate": null,
  "deactivationDate": null
}

```

#### Example for multiple devices (of a GEN24 PV system)

```

{
  "devices": [
    {
      "deviceType": "Inverter",
      "deviceId": "6ac1b645-9210-48ac-b6ec-34e6df517dd9",
      "deviceName": "Symo GEN24 10.0",
      "deviceManufacturer": "Fronius",
      "serialNumber": "31393232",
      "deviceTypeDetails": "Symo GEN24 10.0 Plus",
      "dataloggerId": "pilot-0.5d-80697371430285991_1593083345",
      "nodeType": 254,
      "numberMPPTTrackers": 2,
      "numberPhases": 3,
      "peakPower": {
        "dc1": 7680.0,
        "dc2": 2560.0
      },
      "nominalAcPower": 10000.0,
      "firmware": {
        "updateAvailable": false,
        "installedVersion": null,
        "availableVersion": "fro27372"
      },
      "isActive": true,
      "activationDate": "2020-07-14T00:00:00Z",
      "deactivationDate": null
    },
    {
      "deviceType": "Battery",
      "deviceId": "3ad0653b-7a19-4884-b668-088b9ce9181c",
      "deviceName": "",
      "deviceManufacturer": "BYD",
      "serialNumber": "P030T020Z1912231837",
      "dataloggerId": "pilot-0.5d-80697371430285991_1593083345",
      "capacity": 22464,
      "firmware": {
        "updateAvailable": false,
        "installedVersion": "",
        "availableVersion": ""
      }
    }
  ],
}

```



```

    "isActive": true,
    "activationDate": null,
    "deactivationDate": null
  },
  {
    "deviceType": "SmartMeter",
    "deviceId": "f0926c18-c254-4ccc-ae59-9d28cbf0896a",
    "deviceName": "PowerMeter",
    "deviceManufacturer": "Fronius",
    "deviceCategory": "Primary Meter",
    "deviceLocation": "Grid",
    "serialNumber": "17410258",
    "dataloggerId": "pilot-0.5d-80697371430285991_1593083345",
    "firmware": {
      "updateAvailable": false,
      "installedVersion": "",
      "availableVersion": ""
    },
    "isActive": true,
    "activationDate": "2019-03-25T00:00:00Z",
    "deactivationDate": null
  },
  {
    "deviceType": "Ohmpilot",
    "deviceId": "6b108c81-d378-46a1-aeba-9a151435c4cb",
    "deviceName": "Ohmpilot",
    "deviceManufacturer": "Fronius",
    "serialNumber": "29209059",
    "firmware": {
      "updateAvailable": true,
      "installedVersion": "1.0.13.1",
      "availableVersion": "1.0.25.3"
    },
    "isActive": true,
    "activationDate": "2019-03-25T00:00:00Z",
    "deactivationDate": null,
    "dataloggerId": "pilot-0.5d-80697371430285991_1593083345",
    "sensors": [
      {
        "sensorName": "Temperature",
        "sensorType": null,
        "isActive": true,
        "activationDate": null,
        "deactivationDate": null
      }
    ]
  },
  {
    "deviceType": "Datalogger",
    "deviceId": "6f6c6970-2d74-2e30-3564-2d3830363937",
    "deviceName": "Datalogger",
    "deviceManufacturer": "Fronius",
    "firmware": {

```

```

        "updateAvailable": null,
        "installedVersion": null,
        "availableVersion": null
    },
    "isActive": true,
    "activationDate": "2020-07-11T07:00:01Z",
    "deactivationDate": null,
    "dataLoggerId": "pilot-0.5d-80697371430285991_1593083345",
    "isOnline": true
}
]
}

```

Smart Meter categories:

deviceLocation	deviceCategory
<ul style="list-style-type: none"> <li>• AC battery</li> <li>• External</li> <li>• Generation meter</li> <li>• Grid</li> <li>• Load meter</li> <li>• Sub meter</li> </ul>	<ul style="list-style-type: none"> <li>• AC storage unit</li> <li>• Building services</li> <li>• Climate control / cooling systems</li> <li>• Combined heat and power station (CHP)</li> <li>• Electric vehicle</li> <li>• Heat pump</li> <li>• LGC Meter</li> <li>• Other</li> <li>• Other heating system</li> <li>• Photovoltaic inverter</li> <li>• Photovoltaic inverter + storage unit</li> <li>• Primary Meter</li> <li>• Pumps</li> <li>• White goods</li> <li>• Wind turbine</li> </ul>

Sensor types:

sensorType
<ul style="list-style-type: none"> <li>• Energy</li> <li>• Insolation</li> <li>• Irradiation</li> <li>• Precipitation</li> <li>• Temperature</li> <li>• Velocity</li> </ul>

## 6.3.5 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	/pvsystems/{pv-system-id}/devices-count	GetDeviceCount	Returns the count of all devices for a given PV system.

### Filters and parameters

Filter	Description
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>
?isActive=<state>	Filters for active (isActive=true) or inactive (isActive=false) devices only.

### Example calls

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

```
GET swqapi.solarweb.com/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	<ul style="list-style-type: none"> <li>count (Number)</li> </ul>

### Example responses

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

## 6.3.6 Metadata: Enumerate device IDs

### Use cases for web developers

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

### Methods

Method	End point and objects	Event name	Description
GET	/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=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: <ul style="list-style-type: none"> <li>inverter</li> <li>sensor</li> <li>battery</li> <li>smartmeter</li> <li>ohmpilot</li> <li>datalogger</li> <li>evcharger</li> </ul>

Filter	Description
?isActive=<state>	Filters for active (isActive=true) or inactive (isActive=false) devices only.

### Example calls

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

```
GET swqapi.solarweb.com/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 swqapi.solarweb.com/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	<ul style="list-style-type: none"> <li>deviceIds (Array of Strings)</li> </ul>

### 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"
  ]
}
```

## 6.4 Aggregation calls

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

From these you can create diagrams like the following one:



### 6.4.1 Aggrdata: Aggregated energy data 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	/pvsystems/{pv-system-id}/aggrdata	GetSystemAggregatedData	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
?from=<start>&to=<end>	Limits the time series for the query. Period types in <from> and <to> need to match (i.e. both need to be either years, months, or days). Encoding: <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>	Limits the time series for the query. <duration> is measured in years, months or days - depending on the <from> parameter. A <duration> of 1 means that start and end are equal. Encoding: <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>

Filter	Description
?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, then &lt;from&gt;, &lt;to&gt; and &lt;duration&gt; parameters are not 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 swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=total
```

```
// get aggregated total energy flow values of this system for total lifetime
```

```
GET swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=years
```

```
// get aggregated annual energy flow values of this system for all years since the
installation
```

```
GET swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017&duration=1
```

```
// get the aggregated annual energy flow values of this system for 2017
```

```
GET swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=2017
```

```
// get the aggregated monthly energy flow values of this system for 2017
```

```
GET swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
period=2017-12
```

```
// get the aggregated daily energy flow values of this system for December 2017
```

```
GET swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/aggrdata?
from=2017-12&duration=1&channel=EnergyFeedIn
```

```
// get the EnergyFeedIn value of this system for December 2017
```

```
GET swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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	<ul style="list-style-type: none"> <li>• EnergyFeedIn</li> <li>• EnergyPurchased</li> <li>• EnergySelfConsumption</li> <li>• EnergyDirectConsumption</li> </ul>	Requires Smart Meter; otherwise NULL
	<ul style="list-style-type: none"> <li>• EnergyBattCharge</li> <li>• EnergyBattDischarge</li> <li>• EnergyBattChargeGrid</li> <li>• EnergyBattDischargeGrid</li> </ul>	Requires Smart Meter and battery; otherwise NULL
	<ul style="list-style-type: none"> <li>• OhmpilotEnergy</li> </ul>	Uses ChannelType.FromGenToOhmPilot, NOT OhmPilotEnergy! Requires Smart Meter and Ohmpilot; otherwise NULL
	<ul style="list-style-type: none"> <li>• EnergyEVCCCharge</li> <li>• EnergyEVCCChargeGrid</li> <li>• EnergyEVCCChargeBatt</li> </ul>	Requires Smart Meter and Wattpilot; otherwise NULL
PV system output without Smart Meter	<ul style="list-style-type: none"> <li>• EnergyOutput</li> </ul>	Only if there is no Smart Meter; NULL with Smart Meter
PV system totals	<ul style="list-style-type: none"> <li>• EnergyProductionTotal</li> <li>• EnergyConsumptionTotal</li> <li>• EnergySelfConsumptionTotal</li> </ul>	Requires Smart Meter; otherwise NULL
PV system CO2 savings	<ul style="list-style-type: none"> <li>• SavingsCO2</li> <li>• SavingsTrees</li> <li>• SavingsTravelCar</li> <li>• SavingsTravelPlane</li> </ul>	
PV system savings	<ul style="list-style-type: none"> <li>• Profits</li> <li>• Earnings</li> <li>• Savings</li> </ul>	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",
    "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",
        "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": "EnergyEVCCChargeGrid",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
    },
    {
        "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
      }
    ]
  }
]
}

```

#### 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
        }
      ]
    }
  ]
}

```

```

    ]
  }
]
}

```

#### 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",
          "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
      }
    ]
  }
]
}

```

## 6.4.2 Aggrdata: Aggregated energy data 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	/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 types in <from> and <to> need to match (i.e. both need to be either years, months, or days). Encoding: <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>	Limits the time series for the query. <Duration> is measured in years, months or days - depending on the <from> parameter. A <duration> of 1 means that start and end are equal. Encoding: <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>	Limits the time series for the query to a period. Encoding: <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> Note: If <period> parameter is used, then <from>, <to> and <duration> parameters are not allowed, and vice versa.



Filter	Description
?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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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 swqapi.solarweb.com/pvsystems/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	<div><ul style="list-style-type: none"><li>• EnergyExported</li><li>• EnergyDC1</li><li>• EnergyDC2</li><li>• ...</li></ul></div> <div>New inverter types can have more than two strings. If there are more than two strings, their energy channels will be added as "EnergyDCn" (where n is the number of the string).</div>

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
        }
      ]
    }
  ]
}
```

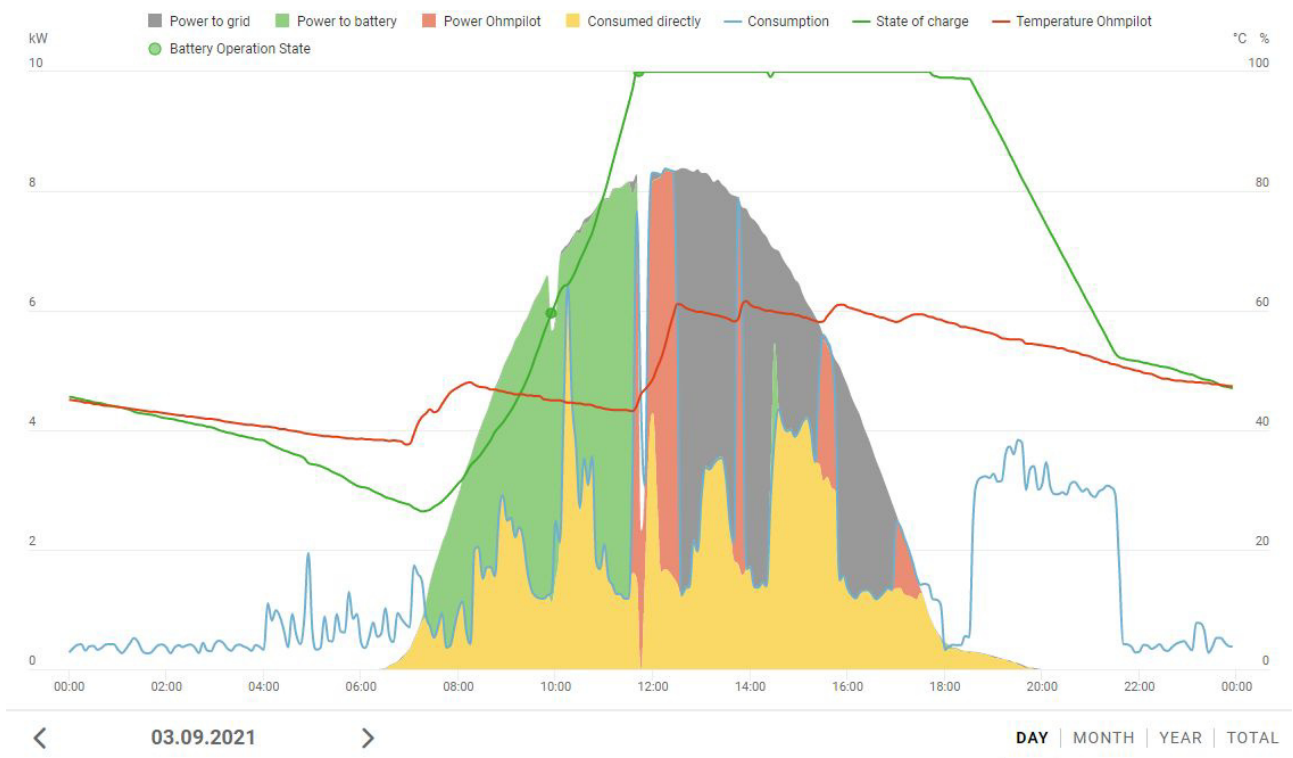
```
    ]  
  }
```

## 6.5 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). By default the resolution of these data points is 5 minutes. Please note that power values are logged as energy values. To calculate power values from energy values (to display curves as shown below) please refer to section *Best practices and how-tos* at the end of this document.

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



### 6.5.1 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.

Inverters report production data to Solar.web only once per hour. Data channels for the current hour therefore cannot be accessed by applications.

## Methods

Method	End point and objects	Event name	Description
GET	/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: <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>
?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 swqapi.solarweb.com/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

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

Supported value channels:

Type	Objects	Remarks
PV system power flow with Smart Meter	<ul style="list-style-type: none"> <li>• EnergyFeedIn</li> <li>• EnergyPurchased</li> <li>• EnergySelfConsumption</li> </ul>	Requires Smart Meter; otherwise NULL
	<ul style="list-style-type: none"> <li>• EnergyBattCharge</li> <li>• EnergyBattChargeGrid</li> <li>• EnergyBattDischarge</li> <li>• EnergyBattDischargeGrid</li> </ul>	Requires Smart Meter and battery; otherwise NULL
	<ul style="list-style-type: none"> <li>• EnergyEVCCCharge</li> <li>• EnergyEVCCChargeBatt</li> <li>• EnergyEVCCChargeGrid</li> </ul>	Requires Smart Meter and EV charger; otherwise NULL
PV system power flow without Smart Meter	<ul style="list-style-type: none"> <li>• EnergyOutput</li> </ul>	Only if there is no Smart Meter; NULL with Smart Meter
PV system totals	<ul style="list-style-type: none"> <li>• EnergyProductionTotal</li> <li>• EnergyConsumptionTotal</li> <li>• EnergySelfConsumptionTotal</li> <li>• EnergyEVCCChargeTotal</li> </ul>	

## Example responses

```
{
  "pvSystemId": "73733052-6f10-47a5-9746-7f7e76ebcb8c",
  "deviceId": null,
  "data": [
    {
      "logDateTime": "2022-06-27T10:00:00Z",
```

```

"logDuration": 599,
"channels": [
  {
    "channelName": "EnergySelfConsumption",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyFeedIn",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyBattCharge",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyBattDischargeGrid",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyBattDischarge",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyPurchased",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 14.02
  },
  {
    "channelName": "EnergyBattChargeGrid",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyOutput",
    "channelType": "Energy",
    "unit": "Wh",
    "value": 0.0
  },
  {
    "channelName": "EnergyEVCCCharge",
    "channelType": "Energy",
    "unit": "Wh",

```



```

        "value": 0.0
      },
      {
        "channelName": "EnergyEVCCChargeBatt",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
      },
      {
        "channelName": "EnergyEVCCChargeGrid",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
      },
      {
        "channelName": "EnergyProductionTotal",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
      },
      {
        "channelName": "EnergySelfConsumptionTotal",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
      },
      {
        "channelName": "EnergyConsumptionTotal",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 14.02
      },
      {
        "channelName": "EnergyEVCCChargeTotal",
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
      }
    ]
  }
}

```

## 6.5.2 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.

Inverters report production data to Solar.web only once per hour. Data channels for the current hour therefore cannot be accessed by applications.

## Methods

Method	End point and objects	Event name	Description
GET	/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.
?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>
?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 swqapi.solarweb.com/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 swqapi.solarweb.com/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	<ul style="list-style-type: none"> <li>• EnergyExported</li> <li>• EnergyImported</li> <li>• EnergyDC1</li> <li>• EnergyDC2</li> <li>• ...</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, their "EnergyDCn", "CurrentDCn" and "VoltageDCn" channels will be added (where n is the number of the string).</p> </div> <ul style="list-style-type: none"> <li>• CurrentA</li> <li>• CurrentB</li> <li>• CurrentC</li> <li>• CurrentDC1</li> <li>• CurrentDC2</li> <li>• ...</li> <li>• VoltageA</li> <li>• VoltageB</li> <li>• VoltageC</li> <li>• VoltageAB</li> <li>• VoltageBC</li> <li>• VoltageCA</li> <li>• VoltageDC1</li> <li>• VoltageDC2</li> <li>• ...</li> <li>• ApparentPower</li> <li>• ReactivePower</li> <li>• PowerFactor</li> <li>• StandardizedPower</li> </ul>

Type	Channels
Smart Meter	<ul style="list-style-type: none"> <li>• GridPowerA</li> <li>• GridPowerB</li> <li>• GridPowerC</li> <li>• GridApparentPowerA</li> <li>• GridApparentPowerB</li> <li>• GridApparantPowerC</li> <li>• GridVoltageA</li> <li>• GridVoltageB</li> <li>• GridVoltageC</li> <li>• LoadPowerA</li> <li>• LoadPowerB</li> <li>• LoadPowerC</li> <li>• LoadApparentPowerA</li> <li>• LoadApparentPowerB</li> <li>• LoadApparentPowerC</li> <li>• LoadVoltageA</li> <li>• LoadVoltageB</li> <li>• LoadVoltageC</li> <li>• ExtEnergyExportedAbs</li> <li>• ExtEnergyExported</li> <li>• EnergyLoadAbs</li> <li>• EnergyLoad</li> <li>• EnergyImported</li> <li>• EnergyExported</li> <li>• GridEnergyExportedAbs</li> <li>• GridEnergyImportedAbs</li> <li>• GridEnergyExported</li> <li>• GridEnergyImported</li> </ul>

Type	Channels
Sensor	<ul style="list-style-type: none"> <li>• Temp1</li> <li>• Temp2</li> <li>• Insolation</li> <li>• Digital1</li> <li>• Digital2</li> <li>• Digital3</li> <li>• Digital1Energy</li> <li>• Digital2Energy</li> <li>• Digital3Energy</li> </ul>
Battery	<ul style="list-style-type: none"> <li>• BattOpState</li> <li>• BattSOC</li> </ul>
Ohmpilot	<ul style="list-style-type: none"> <li>• OhmpilotTemp</li> <li>• OhmpilotEnergyAbs</li> <li>• OhmpilotEnergy</li> <li>• OhmpilotError</li> </ul>
EV charger	<ul style="list-style-type: none"> <li>• EnergyChargeTotal</li> <li>• VoltageA</li> <li>• VoltageB</li> <li>• VoltageC</li> </ul>

### Example responses

Inverter
<pre>{   "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         }       ]     }   ] }</pre>

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

```

## 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
      }
    ]
  }
}

```

### Smart Meter (primary)

```

{
  "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",
        "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
      }
    ]
  }
}

```

### Smart Meter (consumption)

```

{
  "pvSystemId": "85896da3-bb2a-47f7-9c6e-2909dd44832c",
  "deviceId": "d2c78b92-1d96-4fad-93cb-ec43169c3ed0",
  "data": [
    {
      "logDateTime": "2021-01-01T09:00:00Z",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "EnergyExported",
          "channelType": "Energy",
          "unit": "Wh",
          "value": 0.0
        },
        {
          "channelName": "EnergyImported",

```

```

        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
      }
    ]
  }
]
}

```

## 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
        }
      ]
    }
  ]
}

```

## EV charger

```

{
  "pvSystemId": "73733052-6f10-47a5-9746-7f7e76ebcb8c",
  "deviceId": "cee3e54d-0191-4700-8504-aea000d0d839",
  "data": [
    {
      "logDateTime": "2022-08-14T11:00:00+10:00",
      "logDuration": 300,
      "channels": [
        {
          "channelName": "EnergyChargeTotal",

```

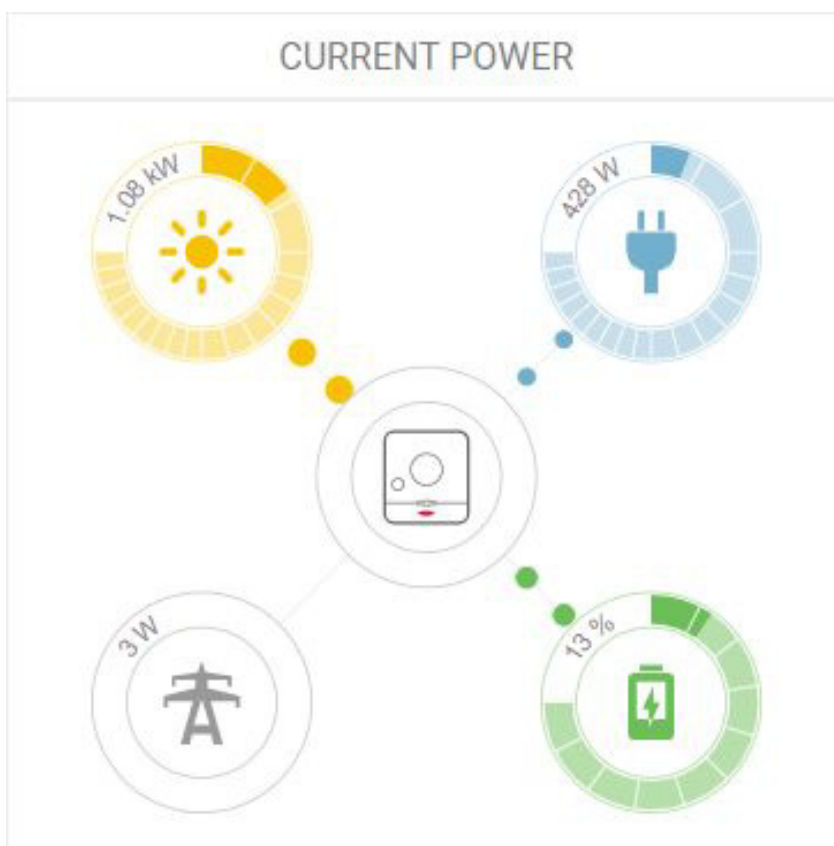
```
        "channelType": "Energy",
        "unit": "Wh",
        "value": 0.0
    },
    {
        "channelName": "VoltageA",
        "channelType": "Voltage",
        "unit": "V",
        "value": 241.81
    },
    {
        "channelName": "VoltageB",
        "channelType": "Voltage",
        "unit": "V",
        "value": 241.84
    },
    {
        "channelName": "VoltageC",
        "channelType": "Voltage",
        "unit": "V",
        "value": 243.51
    }
]
}
]
```

## 6.6 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. The data is updated every few seconds. To make use of the full potential of the power flow data the PV system is required to have a Fronius Smart Meter installed. With such a meter Solar.web can determine the directions of the power flows (e.g. to the grid, from the battery, etc.). Without a meter only the power generated by the PV system can be provided.

Power flows can be shown like in this diagram:



### 6.6.1 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	/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: <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>

### Example calls

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

### Response objects

JSON object answer construction:

Objects
<ul style="list-style-type: none"> <li>• pvSystemId (String)</li> <li>• status <ul style="list-style-type: none"> <li>• isOnline (Boolean)</li> <li>• battMode (Number)</li> </ul> </li> <li>• data <ul style="list-style-type: none"> <li>• logDateTime (UTC time)</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, String)</li> </ul> </li> </ul> </li> </ul>

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> <li>PowerEVCTotal</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. Positive values are going to the inverter, negative values from the inverter to somewhere else.</p> <p>If there is no Smart Meter, the inverter returns PowerOutput instead.</p> <ul style="list-style-type: none"> <li>PowerBattCharge is negative when charging</li> </ul>
	<ul style="list-style-type: none"> <li>isOnline</li> <li>battMode</li> </ul>	<p>when is the system shown online? timeout</p> <p>when loosing contact?</p> <p>which battmodes are available and what do they mean</p>

Sometimes NULL values are returned, but another call a few seconds later returns valid data.

This is because Solar.web implements an asynchronous architecture. For the powerflow data Solar.web has to request the data from the PV systems' devices. Depending on the round-trip time it can happen that Solar.web internally runs into a timeout while waiting for the data to be reported from the devices. Hence, SWQAPI then will return no values.

However, the data arrives usually after a few seconds and will then be returned by another call.

## Example responses

```
{
  "pvSystemId": "04d81b82-7861-4e36-8e7f-41036ce711a4",
  "status": {
    "isOnline": true,
    "battMode": "1.0"
  },
  "data": {
    "logDateTime": "2019-06-18T14:01:57Z",
    "channels": [
      {
        "channelName": "PowerFeedIn",
        "channelType": "Power",
        "unit": "W",
        "value": -496.01
      }
    ]
  }
}
```

```

{
  "channelName": "PowerLoad",
  "channelType": "Power",
  "unit": "W",
  "value": -186.89
},
{
  "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
},
{
  "channelName": "PowerEVCTotal",
  "channelType": "Power",
  "unit": "W",
  "value": -1000.0
}
]
}

```

## 6.6.2 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	/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: <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>

### Example calls

```
GET swqapi.solarweb.com/pvsystems/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	<ul style="list-style-type: none"> <li>• PowerOutput</li> </ul>
sensor	<ul style="list-style-type: none"> <li>• Temp1, Temp2</li> <li>• Insolation</li> <li>• Velocity</li> <li>• Digital1, Digital2, Digital3</li> </ul>



Type	Objects
battery	<ul style="list-style-type: none"> <li>• BattSOC</li> <li>• BattMode</li> <li>• PowerBattCharge</li> <li>• BattSOH</li> <li>• BattEnergyChargedTotal</li> <li>• BattEnergyDischargedTotal</li> <li>• MaxChargePower</li> <li>• MaxDischargePower</li> </ul>
Smart Meter	<ul style="list-style-type: none"> <li>• PowerFeedIn</li> <li>• PowerLoad</li> <li>• Power</li> <li>• PowerExt</li> <li>• PowerPurchase</li> </ul>
Ohmpilot	<ul style="list-style-type: none"> <li>• PowerOhmpilot</li> <li>• OhmpilotTemp</li> <li>• OhmpilotState</li> </ul>
Wattpilot (EV Charger)	<ul style="list-style-type: none"> <li>• PowerTotal</li> <li>• EVCMODE</li> </ul>

Sometimes NULL values are returned, but another call a few seconds later returns valid data.

This is because Solar.web implements an asynchronous architecture. For the powerflow data Solar.web has to request the channel data from the devices first. Depending on the round-trip time it can happen that Solar.web internally runs into a timeout while waiting for the data to be reported from the devices. Hence, SWQAPI then will return no values.

However, the data arrives usually after a few seconds and will then be returned by another call.

## Example responses

### Example for an inverter

```
{
  "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
        }
      ]
    }
  }
}

```

#### Example for a battery

```

{
  "pvSystemId": "5845cdf8-ae05-4cf8-a111-f1dc5665cae3",
  "deviceId": "0ae28f7f-6983-422d-9615-56f168419074",
  "status": {
    "isOnline": true,
    "battMode": 1.0
  },
  "data": {
    "logDateTime": "2023-09-05T05:24:49Z",
    "channels": [
      {
        "channelName": "BattSOC",
        "channelType": "Percentage",
        "unit": "%",
        "value": 32.0
      },
      {
        "channelName": "BattMode",
        "channelType": "",
        "unit": "",
        "value": "Normal"
      },
      {
        "channelName": "BattSOH",
        "channelType": "Percentage",
        "unit": "%",
        "value": null
      },
      {
        "channelName": "PowerBattCharge",
        "channelType": "Power",
        "unit": "W",
        "value": null
      }
    ]
  }
}

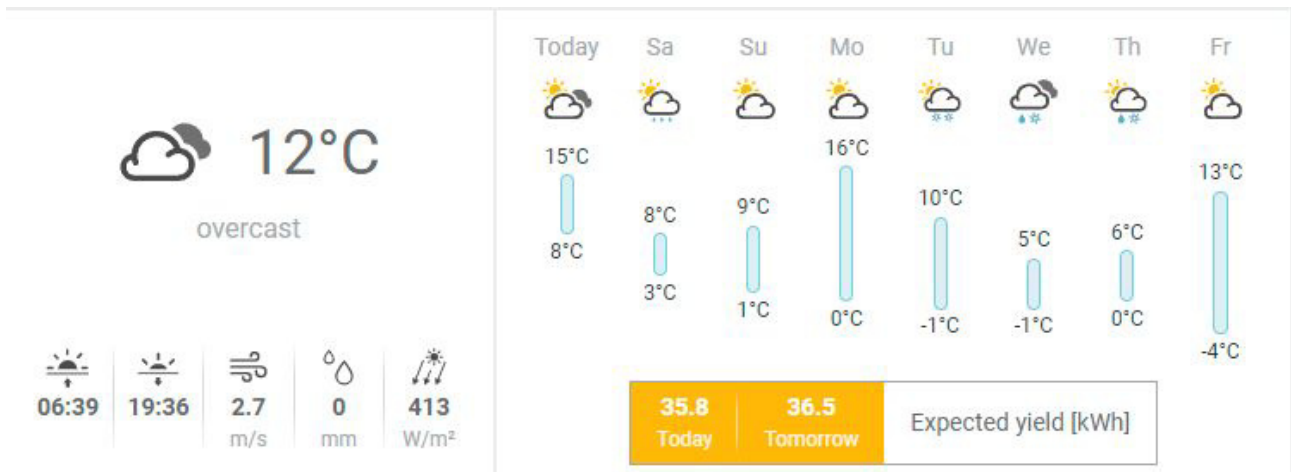
```

```
{
  "channelName": "BattEnergyChargedTotal",
  "channelType": "Energy",
  "unit": "kWh",
  "value": null
},
{
  "channelName": "BattEnergyDischargedTotal",
  "channelType": "Energy",
  "unit": "kWh",
  "value": null
},
{
  "channelName": "MaxChargePower",
  "channelType": "Power",
  "unit": "W",
  "value": null
},
{
  "channelName": "MaxDischargePower",
  "channelType": "Power",
  "unit": "W",
  "value": null
}
]
}
```

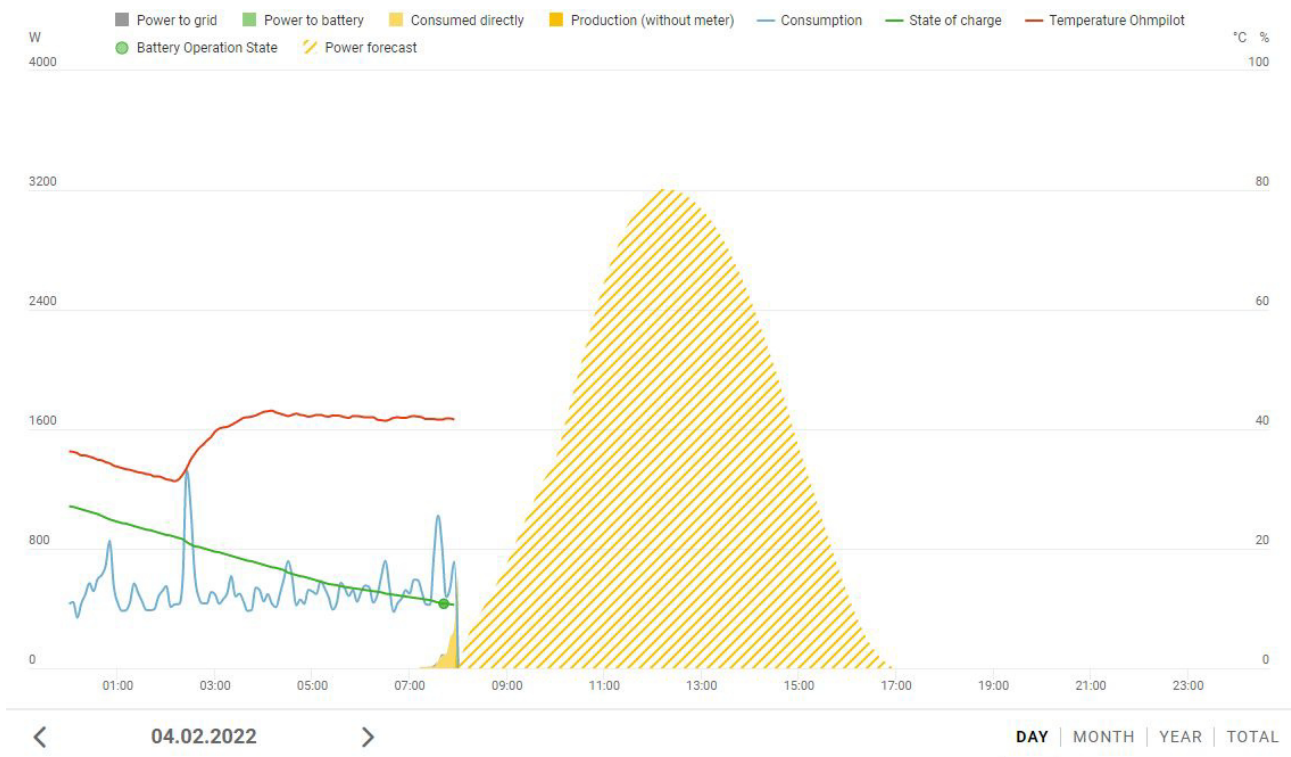
## 6.7 Weather data calls

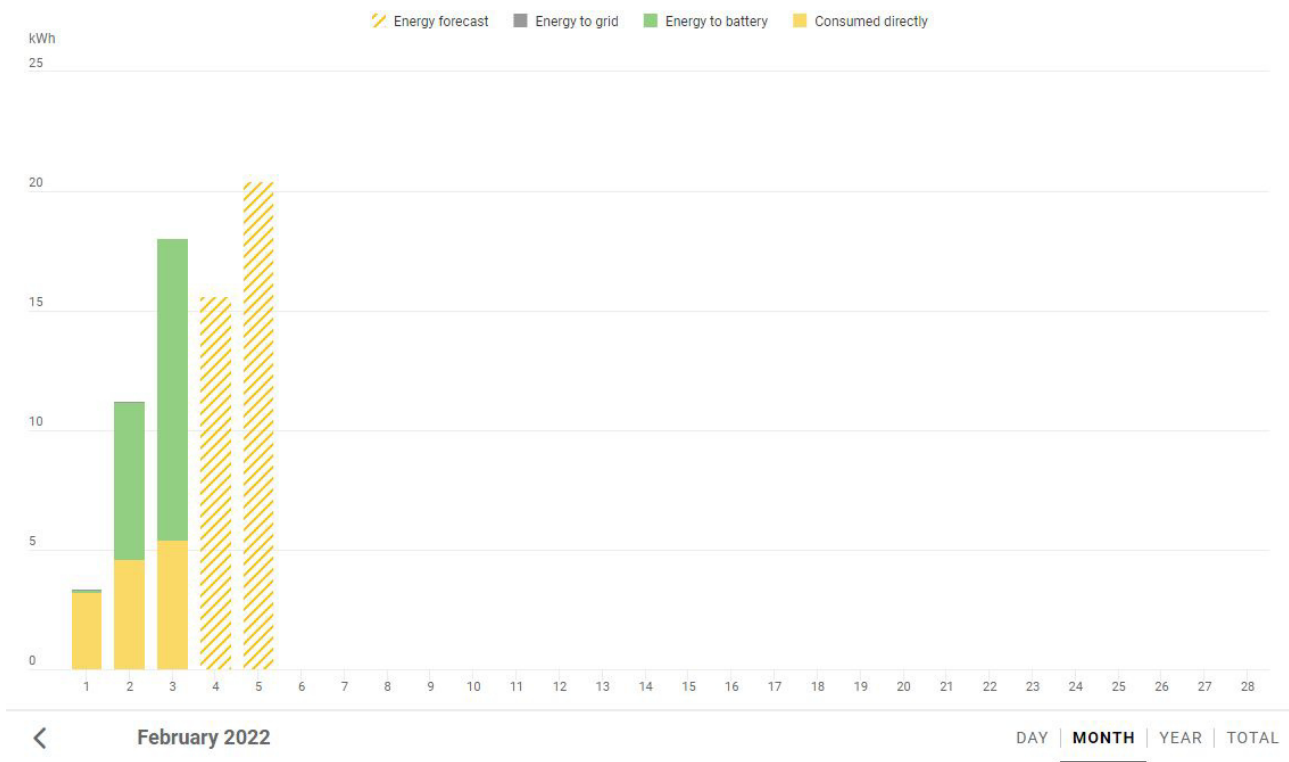
This set provides weather and energy forecast information for PV systems.

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):





### 6.7.1 Limitations

Only Solar.web 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).

### 6.7.2 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 Solar.web Premium 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	/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: <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>

## Example calls

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

```
GET swqapi.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 swqapi.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

- pvSystemId (String)
- data
  - logDateTime (DateTime)
  - channels (Array)
    - channelName (String)
    - channelType (String)
    - unit (String)
    - value (Number or Object)

Supported value channels:

Type	Channels	Remarks
PV system	<ul style="list-style-type: none"> <li>• Temp</li> <li>• WindSpeed</li> <li>• Precipitation</li> <li>• CloudCover</li> <li>• Daylight</li> <li>• Symbol</li> </ul>	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
      }
    ]
  }
}

```

### 6.7.3 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 Solar.web Premium 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	/pvsystems/{pv-system-id}/weather/forecast	GetSystemWeatherForecast	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. <ul style="list-style-type: none"> <li>• Alternatively, &lt;start&gt; can also be "today" (default) or "tomorrow"; &lt;end&gt; can be "tomorrow".</li> </ul> </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. <ul style="list-style-type: none"> <li>• Alternatively, &lt;start&gt; can also be "today" (default) or "tomorrow".</li> </ul> </li> <li>• &lt;duration&gt; is the number of days. <ul style="list-style-type: none"> <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> </ul> </li> <li>• Date format encoding: "yyyyMMdd", "yyyy-MM-dd".</li> </ul>

## Example calls

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

GET swqapi.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 swqapi.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
	<ul style="list-style-type: none"> <li>• pvSystemId (String)</li> <li>• data               <ul style="list-style-type: none"> <li>• logDateTime (Date)</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 or Object)</li> </ul> </li> </ul> </li> </ul>

Supported value channels:

Type	Channels	Remarks
PV system	<ul style="list-style-type: none"> <li>• Temp</li> <li>• Daylight</li> <li>• Symbol</li> <li>• EnergyExpected</li> </ul>	Temperatures with temperatureMin and temperatureMax values. Daylight with sunrise and sunset times. Precipitation and Daylight channels only for "pro" systems.

## 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"
      }
    ]
  }
]
}

```

## 6.7.4 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 Solar.web Premium 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	/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 swqapi.solarweb.com/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/weather/energyforecast
// gets energyforecast information for PV system for the next 48 hours

GET swqapi.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	<ul style="list-style-type: none"> <li>EnergyExpected</li> </ul>	

### 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"
        }
      ]
    }
  ]
}
```

## 6.8 System messages calls

### 6.8.1 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	/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	/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.

Filter	Description
?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".
?statecode=<code>	Filters by StateCode.
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>

### Example calls

```
GET swqapi.solarweb.com/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"
    }
  ]

```

## 6.8.2 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	/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. <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".
?statecode=<code>	Filters by StateCode.

Filter	Description
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: <ul style="list-style-type: none"> <li>• inverter</li> <li>• sensor</li> <li>• battery</li> <li>• smartmeter</li> <li>• ohmpilot</li> <li>• datalogger</li> <li>• evcharger</li> </ul>

### Example calls

```
GET swqapi.solarweb.com/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	<ul style="list-style-type: none"> <li>• count (Number)</li> </ul>

### Example responses

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

## 6.8.3 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	/pvsystems/{pv-system-id}/ devices/{device-id}/ messages	GetDeviceMessages	Returns service messages for the requested device of a PV system with the given ID in English (default language). The service messages are provided as JSON objects.
GET	/pvsystems/{pv-system-id}/ devices/{device-id}/ messages/{ISO-country-code}	GetDeviceMessages	Returns service messages for the requested 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: <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".
?statecode=<code>	Filters by StateCode.

### Example calls

```
GET swqapi.solarweb.com/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	<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": "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"
  }
]
```

## 6.8.4 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	/pvsystems/{pv-system-id}/devices/{device-id}/messages-count	GetDeviceMessagesCount	Returns number of service messages for the requested 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. <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".
?statecode=<code>	Filters by StateCode.

## Example calls

```
GET swqapi.solarweb.com/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	<ul style="list-style-type: none"> <li>count (Number)</li> </ul>

### Example responses

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

## 7 Appendix

### 7.1 Response and error codes

#### 7.1.1 HTML error codes

Used HTML response codes by SWQAPI:

Code	Short description	Description
200	OK	Successful
201	Created	Request was successful and a new resource has been created
204	No content	Successful request, no data returned
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
409	Conflict	The request could not be processed because of a state-conflict
415	Unsupported media type	Payload format is not supported
422	Unprocessable content	The server cannot process the content
429	Too many requests	Request is rejected due to rate limiting
500	Internal server error	An unexpected server error happened

Code	Short description	Description
503	Service not available	The server is temporarily unavailable

## 7.1.2 Detailed error codes

If an error happens, Fronius APIs indicate 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.

Fronius Response Code	Verbose Description	Comment
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.
1012	The API is not available due to server maintenance. The maintenance window lasts until {0}.	Maintenance end time is added to the message.
1013	User not authorized for PV system.	
1015	PV system not found.	
1016	Error while processing request.	
<b>AUTHENTICATION (11xx)</b>		
1101	AccessKeyId and Value not sent.	
1102	AccessKey not found.	
1103	AccessKey is not active.	
1104	AccessKey expired.	
1105	User blocked.	
1106	Authentication failed.	
1107	Invalid request.	JWT generation failed, usually because of invalid data.
1108	User did not accept latest Terms of Use.	
1110	Invalid JWT format.	
1111	Invalid JWT signature.	
1112	Invalid JWT issuer.	
1113	JWT expired.	



Fronius Response Code	Verbose Description	Comment
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.	
1118	Invalid JWT.	
1120	Refresh token invalid.	
1121	Refresh token expired.	
1122	Account temporarily blocked - too many login attempts.	
1123	There was no Solar.web user found, User needs to log in to Solar.web Portal once.	
<b>Metadata calls (30xx)</b>		
3001	Type filters invalid.	List of invalid type filters is appended at the end.
3002	Invalid meteo parameter.	Wrong meteo filter. Only "pro" or "light" values are valid.
3003	Error fetching Data from Database.	An unexpected error happened while retrieving data from database.
<b>Power flow data (flowdata) calls (31xx)</b>		
31xx		
<b>Aggregation data (aggrdata, aggrdata) calls (32xx)</b>		
3201	Invalid date format for "from" parameter.	
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.	

Fronius Response Code	Verbose Description	Comment
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>Historical data (histdata) calls (33xx)</b>		
3301	Date range max is 24 hours.	The maximum time range that can be queried is 24 hours - mainly because calls would take too long otherwise. If you need data for more than a day, please split the query into multiple calls with no more than 24 hours each.
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.	
3405	Invalid state severity.	
<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.)
<b>Info calls (36xx)</b>		
3601	TermsAcceptedLatest cannot be set to false.	

## 7.2 Response Headers

The API returns HTTP response headers to clients which can provide more detail context.

Name	Description	Example
Cache-Control	Indicates that the response can be cached for N seconds before getting refreshed	max-age=3600
X-Rate-Limit-Limit	Request limit per period; the limit defines the time window length	1h
X-Rate-Limit-Remaining	Number of requests left for the time window	21942
X-Rate-Limit-Reset	The time when the rate limit resets (in UTC)	2024-03-28T13:00:00.000000Z

### 7.2.1 Rate limit headers

To avoid DOS attacks or overloading, the API implements rate limiting. Rate limit is implemented very flexible; there are rate limits applied per hour, per minute, and per second.

Typical rate limits are e.g. 30,000 calls per hour, 1,000 calls per minute and 100 calls per second.

If you need different rate limits for your application, please contact Fronius.

Whenever a rate limit is hit, the API will return a 429 error (together with a verbose error response description).

Interpreting the rate limit response headers allows callers to find out how many calls are still possible. If the rate limit already applied, it is possible to find out when the next call can be done.

**Example (1h rate limit is typically 30,000 calls):**

X-Rate-Limit-Limit		1h
X-Rate-Limit-Remaining		29999
X-Rate-Limit-Reset		2024-07-25T12:00:00.000000Z

### 7.2.2 Cache control headers (max-age)

Cache-Control headers are added to the responses of the following endpoints upon successful requests. If a request is unsuccessful, no Cache-Control header will be added.

## Historical Data

- GET /pvsystems/{pv-system-id}/histdata
- GET /pvsystems/{pv-system-id}/devices/{device-id}/histdata

Historical data is expected to be received once per hour. Allowing a two-minute tolerance, the `maxAge` for these endpoints can be set to up to 3720 seconds. If historical data for multiple devices is requested, `maxAge` is calculated based on the oldest import. If the last import is overdue, meaning it is older than the expected hourly interval, `maxAge` will be reduced to 120 seconds to enable more frequent updates. If the last import time for the requested device(s) cannot be verified, `maxAge` will be set to 600 seconds (10 minutes).












## Weather forecast (energy forecast)

- GET /pvsystems/{pv-system-id}/weather/current
- GET /pvsystems/{pv-system-id}/weather/forecast
- GET /pvsystems/{pv-system-id}/weather/energyforecast

For calculating `maxAge`, different intervals are considered based on the type of PV system:

- "Pro" Pv Systems (Solar.web Premium "Pro" PP System)  
Weather updates occur every two hours, allowing for a `maxAge` of up to 7320 seconds with a two-minute tolerance.  
Power forecast updates occur every hour, with a `maxAge` of up to 3720 seconds, including tolerance.  
`maxAge` will be calculated until the next weather or power update is expected. If the last update is overdue, `maxAge` will be set to 120 seconds to enable frequent updates.
- "Light" Pv Systems (Solar.web Premium "Light PV System")  
Weather updates occur every four hours, allowing for a `maxAge` of up to 14640 seconds, including tolerance.  
`maxAge` will be calculated until the next weather update is expected. If the last update is overdue, `maxAge` will be set to 120 seconds.
- "Basic" Pv Systems with a weather POI  
Weather updates occur every four hours. `maxAge` will be calculated until the next weather update is expected. If the last update is overdue, `maxAge` will be set to 120 seconds.
- "Basic" Pv Systems without a weather POI  
MaxAge will be 14640 seconds.

## Example Response Headers

Body	Cookies	Headers (10)	Test Results	 200 OK 1438 ms 1.07 KB
		Key		Value
		Content-Length		743
		Content-Type		application/json; charset=utf-8
		Date		Thu, 04 Apr 2024 08:36:43 GMT
		Server		Kestrel
		Cache-Control		max-age=386
		X-Content-Type-Options		nosniff
		Referrer-Policy		origin-when-cross-origin
		X-Frame-Options		SAMEORIGIN
		X-XSS-Protection		1; mode=block
		Content-Security-Policy		default-src 'self'

## 7.3 Channels

### 7.3.1 Channel list

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggregatedata	historicaldata	weather
ApparentPower	Apparent power	inverter	VA	Apparent power (S)			x	
BattEnergyChargedTotal		battery	kWh	Total lifetime battery charge	x			
BattEnergyDischargedTotal		battery	kWh	Total lifetime battery discharge	x			
BattMode		battery		Battery operating state. State values are: <ul style="list-style-type: none"> <li>• 0 = Disabled</li> <li>• 1 = Normal</li> <li>• 2 = ServiceMode</li> <li>• 3 = ChargeBoost</li> <li>• 4 = NearlyDepleted</li> <li>• 5 = SuspendedOnPurpose</li> <li>• 6 = Calibrate</li> <li>• 7 = GridSupport</li> <li>• 8 = DepletedRecovery</li> <li>• 9 = NonOperableTemperature</li> <li>• 10 = NonOperableVoltage</li> <li>• 11 = Preheating</li> <li>• 12 = Startup</li> <li>• 13 = AwakeButNonOperableTemperature</li> <li>• 14 = BatteryFull</li> <li>• 90 = ForcedStandby</li> </ul>	x		x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregated data, aggregated data	historical data	weather
BattSOC	State of charge	battery, general	%	Battery state of charge	x		x	
BattSOH		battery	%	Battery state of health	x			
CloudCover			%	Cloud coverage				x
CurrentA, CurrentB, CurrentC	Current AC L1, Current AC L2, Current AC L3	inverter, evcharger	A	Mean current of 3-phase devices on the AC side of the device 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	
Daylight (sunrise, sunset)			time	Sunrise and sunset times				x
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	
Earnings	Earning	general	<currency>	Energy fed into grid multiplied by a tariff.		x		



Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggregatedata	historicaldata	weather
EnergyBattCharge	Energy to battery	general	Wh	Calculated energy flowing from generators to batteries		x	x	
EnergyBattChargeGrid		general	Wh	Calculated energy flowing from grid to battery		x	x	
EnergyBattDischarge	Energy from battery	general	Wh	Calculated energy flowing from batteries to consumer		x	x	
EnergyBattDischargeGrid		general	Wh	Calculated energy flowing from battery to grid		x	x	
EnergyChargeTotal		evcharger	Wh	Total energy consumed by Watterpilot			x	
EnergyConsumptionTotal	Consumption	general	Wh	Calculated consumed energy (EnergyPurchased + EnergySelfConsumption + EnergyBattDischarge)		x	x	
EnergyDC1, EnergyDC2, ...	Power MPP1, Power MPP2	inverter	Wh	Calculated energy flowing from generator into inverter on DC1 / DC2 input		x	x	
EnergyDirectConsumption	Consumed directly (Production tab)	general	Wh	Calculated energy flowing from generators to consumers (excluding Ohmpilot + Watterpilot)		x		
EnergyEVCharge		general	Wh	Energy flowing from generators to Watterpilot		x	x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	aggregated data, aggregated data	historical data	weather
EnergyEVCCChargeBatt		general	Wh	Energy flowing from battery to Wattpilot		x	x	
EnergyEVCCChargeGrid		general	Wh	Energy flowing from grid to Wattpilot		x	x	
EnergyEVCCChargeTotal		general	Wh	Total energy consumed by Wattpilot			x	
EnergyExpected		general	Wh	Energy forecast				x
EnergyExported	Total power Production (for smartmeter)	general, inverter, smartmeter	Wh	Total energy flowing out of the main inverter (generators sum + battery)		x	x	
EnergyFeedIn	Power to grid	general	Wh	Calculated energy flowing from generators to grid		x	x	
EnergyImported	Consumption	inverter, smartmeter	Wh	Total energy flowing into the main (hybrid) inverter. This energy comes either from the grid or from another inverter, and usually it is stored in a battery.			x	
EnergyLoad		smartmeter	Wh	Energy flowing into the consumers			x	
EnergyLoadAbs		smartmeter	Wh	Energy flowing into the consumers (absolute value)			x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregdata, aggregated	histdata	weather
EnergyOutput		general	Wh	Calculated total energy produced (NULL if smart meter is connected)		x	x	
EnergyProduction Total	Production	general	Wh	Calculated energy flowing from generators to consumer, battery and grid (EnergySelfConsumption + EnergyBattCharge + EnergyFeedIn + EnergyOutput)		x	x	
EnergyPurchased	Power from grid	general	Wh	Calculated energy flowing from grid to consumer		x	x	
EnergySelfConsumption	Consumed directly (Consumption tab)	general	Wh	Calculated energy flowing from generators to consumers (including Ohmpilot + Wattpilot)		x	x	
EnergySelfConsumptionTotal	Own consumption	general	Wh	Calculated energy flowing from generators to consumers and battery (EnergySelfConsumption + EnergyBattCharge)		x	x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggregatedata	historicaldata	weather
EVCMODE		evcharger		Active charging mode of Wattpilot (refer to Wattpilot documentation for details): <ul style="list-style-type: none"> <li>EcoMode</li> <li>NextTripMode</li> <li>StandardMode</li> <li>NoCar</li> <li>NotCharging</li> <li>WaitingForPrice</li> <li>FullyCharged</li> <li>ConnectionLost</li> <li>PvConnectionLost</li> </ul>	x			
ExtEnergyExported		smartmeter	Wh	Energy flowing from external AC source (e.g. wind power generator) to the point of common coupling			x	
ExtEnergyExportedAbs		smartmeter	Wh	Energy flowing from external AC source (e.g. wind power generator) to the point of common coupling (absolute value)			x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggrdata	histdata	weather
GridApparentPowerA, GridApparentPowerB, GridApparentPowerC	Apparent power L1 feed-in-point, Apparent power L2 feed-in-point, Apparent power L3 feed-in-point	smartmeter	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	
GridEnergyExported		smartmeter	Wh	Energy flowing from the house into grid			x	
GridEnergyExportedAbs		smartmeter	Wh	Energy flowing from the house into grid (absolute value)			x	
GridEnergyImported		smartmeter	Wh	Energy imported from grid			x	
GridEnergyImportedAbs		smartmeter	Wh	Energy imported from grid (absolute value)			x	
GridPowerA, GridPowerB, GridPowerC	Effective power L1 feed-in point, Effective power L2 feed-in point, Effective power L3 feed-in point	smartmeter	W	Mean power of 3-phase devices of the grid phase 1 / 2 / 3 measured by primary grid meter in the given interval			x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggregatedata	historicaldata	weather
GridVoltageA, GridVoltageB, GridVoltageC	Voltage AC L1 feed-in point, Voltage AC L2 feed-in point, Voltage AC L3 feed-in point	smartmeter	V	Mean voltages of 3-phase devices of the grid phase 1 / 2 / 3 measured by primary load meter in the given interval			x	
Insolation	Insolation	sensor	W/m <sup>2</sup>	Insolation			x	
IsOnline		general	-	Online status of the PV system	x			
LoadApparentPowerA, LoadApparentPowerB, LoadApparentPowerC		smartmeter	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	
LoadPowerA, LoadPowerB, LoadPowerC		smartmeter	W	Mean 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		smartmeter	V	Mean voltage of 3-phase devices of the load phase 1 / 2 / 3 measured by primary load meter in the given interval			x	
MaxChargePower		battery	W	Maximum power for battery charging	x			

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggregatedata	histdata	weather
MaxDischargePower		battery	W	Maximum power for battery discharging	x			
OhmpilotEnergy	Power	Ohmpilot	Wh	Energy used by Ohmpilot		x	x	
OhmpilotEnergyAbs		Ohmpilot	Wh	Absolute energy used by Ohmpilot			x	
OhmpilotError		Ohmpilot	-	Ohmpilot error code			x	
OhmpilotTemp	Temperature Ohmpilot	Ohmpilot	°C	Temperature measured on Ohmpilot device	x		x	
Power		general	W	Power measured by a submeter (requires a Smart Meter)	x			
PowerBattCharge		general, battery	W	Power flowing from inverter to the battery or from battery to inverter (requires a Smart Meter and a battery)	x			
PowerEVCTotal		general	W	Total power consumed by Wattpilot	x			
PowerExt		general	W	Power generated by external inverter (requires a Smart Meter)	x			
PowerFactor	Power factor	inverter, evcharger	-	Power factor			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	flowdata	aggregata, aggregatedata	historicaldata	weather
PowerLoad		general	W	Power flowing from inverter to the consumer (requires a Smart Meter)	x			
PowerOhmpilot		general	W	Power flowing from inverter to an Ohmpilot (requires a Smart Meter and an Ohmpilot)	x			
PowerOutput		general	W	Power generated by inverter (NULL if Smart Meter is connected)	x			
PowerPurchase		general	W	Power flowing in from grid (requires a Smart Meter)	x			
PowerPV		general	W	Power flowing from generators to the inverter	x			
PowerTotal		evcharger	W	Power consumed by Wattpilot	x			
Precipitation			mm	Precipitation				x
Profits		general	<currency>	Total financial benefit of the PV system (Earnings + Savings).		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			
ReactivePower	Reactive power	inverter	VAr	Reactive power (Q)			x	
Savings		general	<currency>	Directly consumed energy multiplied by a tariff.		x		



Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	aggregated data, aggregated data	historical data	weather
SavingsCO2	CO2 savings	general	kg	Equivalent of saved CO <sub>2</sub> 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		
SavingsTrees	CO2 savings	general	tree	Equivalent of saved average trees due to PV production.		x		
StandardizedPower	Standardized power	inverter	%	Percentage of power generated in relation to peak power (i.e. kWh/kWp)			x	
Symbol			symbol	Weather forecast symbol				x
Temp			°C	Current temperature				x
Temp (min, max)			°C	Temperature forecast				x
Temp1, Temp2	Module temperature	sensor	°C	Temperature			x	
VoltageA, VoltageB, VoltageC	Voltage AC L1, Voltage AC L2, Voltage AC L3	inverter, evcharger	V	Mean voltage of 3-phase devices on the AC side of the device for line L1 / L2 / L3			x	
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	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flowdata	aggregata, aggregatedata	historicaldata	weather
VoltageDC1, VoltageDC2, ...	Voltage DC MPP1, Voltage DC MPP2	inverter	V	Mean voltage on the DC side of the inverter for DC1 / DC2			x	
WindSpeed		sensor	m/s	Wind speed			x	x

### 7.3.2 Channel types








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

## 7.4 Meteorological weather symbols

### 7.4.1 List of weather symbols

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

Symbol description	Icon (example)
overcast with thunderstorms	
sunny with strong thunderstorms	
cloudy with strong thunderstorms	
overcast with strong thunderstorms	
light snowfall	
snowfall	
heavy snow	
sunny with light snow showers	
cloudy with light snow showers	
overcast with light snow showers	
sunny with heavy snow showers	
cloudy with heavy snow showers	
overcast with heavy snow showers	
light sleet	
sleet	
heavy sleet	
sunny with light sleet showers	
cloudy with light sleet showers	
overcast with light sleet showers	
sunny with heavy sleet showers	
cloudy with heavy sleet showers	
sunny with heavy sleet showers	
cloudy with heavy sleet showers	
overcast with heavy sleet showers	

Symbol description	Icon (example)
duststorm / sandstorm	
drifting snow	
graupel	
fog patches	
low clouds, sun	
freezing fog	
sun and high clouds	

## 7.5 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

ISO Code	Language
FI	Finnish
FR	French
HU	Hungarian
IT	Italian
NL	Dutch
PL	Polish

ISO Code	Language
PT	Portuguese
RU	Russian
SK	Slovakian
SV	Swedish
TR	Turkish

## 7.6 Best practices and how-tos

### 7.6.1 Use filters for channels

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

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

Power [W] = Energy [Wh] \* 3600 / 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
        }
      ],
      ...
    }
  ]
}
```

### 7.6.3 Determine PV Energy and Load Energy

PV Energy = EnergySelfConsumption + EnergyFeedIn + EnergyBattCharge

Load Energy = EnergySelfConsumption + EnergyPurchased + EnergyBattDischarged



## 7.6.4 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:

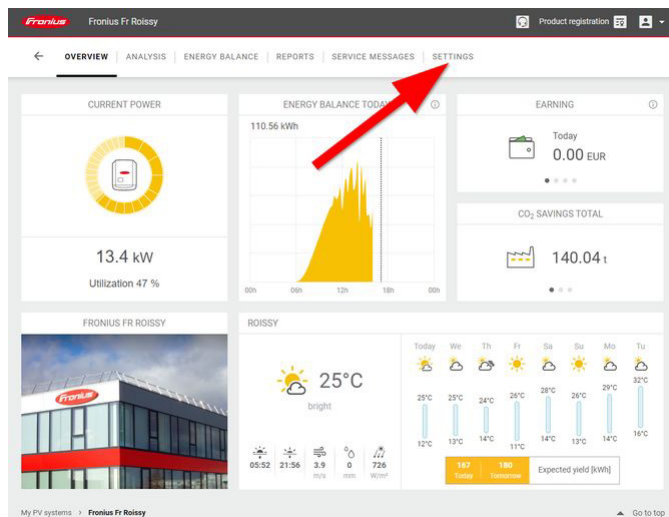
1. Use the /pvsystems-list endpoint to get the PV system IDs.
2. Compare the IDs with your current list of IDs in order to find the ID of the new system.
3. Update your list of IDs.
4. Use the /pvsystems-list endpoint repeatedly to determine if a PV system is removed from account.
5. If an ID does not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

## 7.6.5 Grant permissions in Solar.web

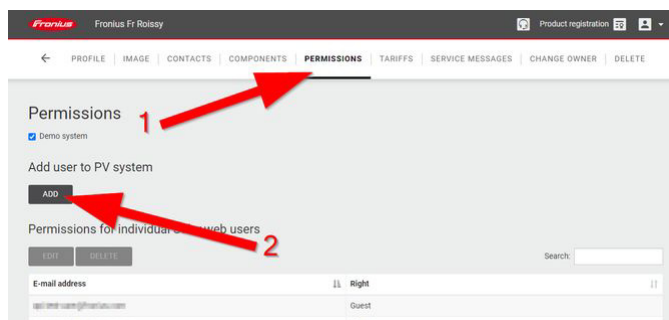
When using the regular authentication method your user needs permission to access a PV system's data through the API if you don't own it. Here's a short guide on how to grant permission.

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 (1) and click on ADD (2)



4. Enter the E-mail address of the account that shall get access to the system. Select a permission level:

- Guest: in Solar.web read only and no change of settings possible; service messages cannot be read via SWQAPI.
- Supervisor: in Solar.web change of settings possible; service messages can be read via SWQAPI.

Click OK

