Solar.web Query API - Specification

Information class: Public, Fronius International, BU SE, 01.08.2023

Version 50.0 2023-08-01
BU SE

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

1 VERSION HISTORY

Version	Modified	Description
50.0	Aug 01, 2023	/ Added channels for Submeters. / Corrected "PowerPurchased" channel to "PowerPurchase". / Updated EnergyExported and EnergyImported channels for secondary meters / Pilot example added / Updated data plan details
49.0	Oct 12, 2022	/ Updated sections Metadata calls, Aggregation calls, Historical data calls, Realtime data calls, System messages calls: / Added Wattpilot data. / Updated some example responses. / Minor corrections in section Weather data calls. / Updated General information section. / Updated Supporting UIs section. / Updated channel list: / Added Wattpilot data / Removed reference to outdated Aggregation calls
48.0	Dec 09, 2021	/ Added notes to GenerateJwt call. / Added status code information for maintenance windows. / Updated device metadata: / Provided lists for Smart Meter locations and categories. / Provided list for sensor types. / Updated device metadata: / Added missing datalogger IDs. / Improved examples, e.g. a GEN24 PV system. / Added revoke JWT call. / Updated metadata calls with meteo filter. / Clarifications in error tables. / Removed "PowerBattDischarge" channel from channel list, and updated description of "PowerBattCharge". / Clarified positive/negative values for power flow data. / Clarified 3301 error code for historical data. / Added datalogger and Ohmpilot examples to device metadata. / Removed Fronius SSO login method.
47.0	Apr 21, 2021	/ Updated camelCase notation in examples for system messages.
46.0	Apr 02, 2021	/ Changed supported channel information for aggregated and historical data requests. / Updated Solar.web screenshots, error code lists and channel list.
45.0	Feb 02, 2021	/ Version history added. / Added additional information about Solar.web Premium to chapter "User impersonation".

Version	Modified	Description
		/ Updated chapter "Determine power values from energy values" in Appendix.

2 INTRODUCTION

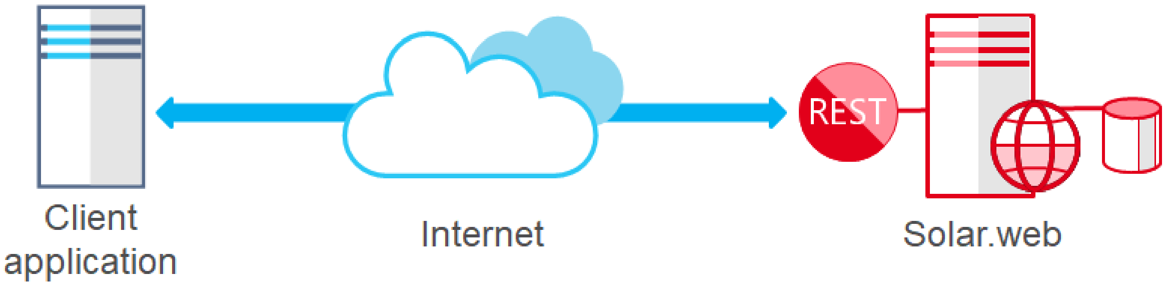
2.1 About Fronius Solar.web Query API

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

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

API keys have the following attributes:

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 expect to receive access key ID and access key value data 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.

3.2.1 JWT token attributes

JWT token value	A long string, identifying the customer and providing access to him, for example: `eyJ4NXQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiLCJraWQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiS2hLZVZsc0lPXy1tWDhvZkJZSzdJZyIsImF1ZCI6IlljNHhtcEIyVnlyR2phcUlraGoxbXJEOFZ6VWEiLCJzdWIiOiJodWV0dG5lci50aG9tYXNybmRAZnJvbml1cy5jb20iLCJuYmYiOjE1ODUyOTQ2NjcsImF6cCI6IlljNHhtcEIyVnlyR2phcUlraGoxbXJEOFZ6VWEiLCJhbXIiOlsicGFzc3dvcmQiXSwiaXNzIjoiJHtjYXJib24ucHJvdG9jb2x9OlwvXC8ke2NhcmJvbi5ob3N0fVwvb2F1dGgyXC9vaWRjZGlzY292ZXJ5IiwiZXhwIjoxNTg1Mjk4MjY3LCJpYXQiOjE1ODUyOTQ2Njd9.HUXi1sySzyLqx2e0dLpr0sszi-YiI3nGNB4GZDDwIwVHUHC4s6ED8BqfvkfFn3s45LkvJQEvqb_Wd3QtMGnzOLEZ3RdK3A8GWdsDChVq_nzlP4FGC6b5lPoz9Xi6mH_pcxt36rzA2-vjl_e6cTOrTXsIeIzOjZVNSZRAJ4-A5HpmEuvraoArAGUqc_yTntbfALhfJQkfsjoDAJRAfZXLknTvDKm2vMd0-uXjTQHM2dKAWGAz6r39cLQ24sFIIC7MDgIp4GpNVBCLFSNzkK7mV3fSEQvgIdAFMhEP4CY4lMTItLxdfRKxcf5SA7o2fU0-_710frdFYvrkesorDCiyfg`
JWT expiry date	An expiry time (UTC time) for the JWT; a Fronius JWT is valid for exactly one hour and needs to be refreshed periodically. Note: A JWT stays valid even if the customer changes his password (for a maximum of one hour).
Refresh token	A token which can be used to refresh a JWT.

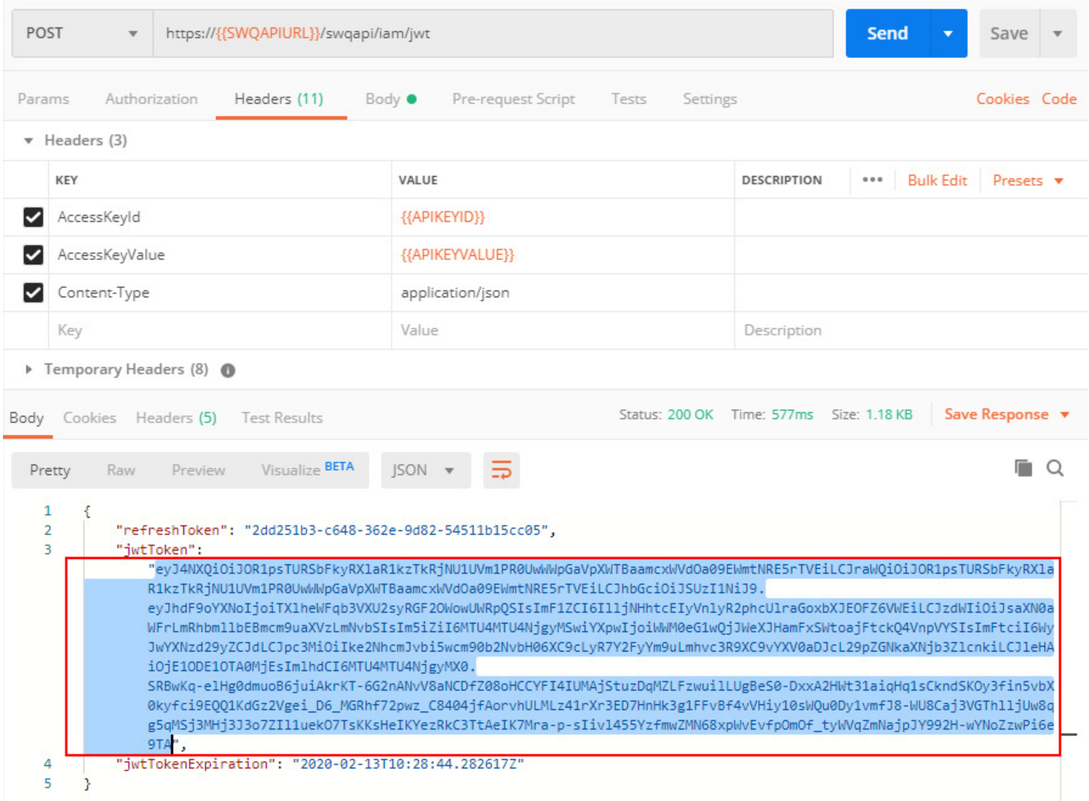
3.2.2 Creating a JWT and using it in a SWQAPI call

You can pass another user's Fronius Solar.web user ID and password, and you receive a JWT and a refresh token. Please be careful about the user's password and protect it against leaking.

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

Example with Postman how to create a JWT:

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



POST https://{{SWQAPIURL}}/swqapi/iam/jwt

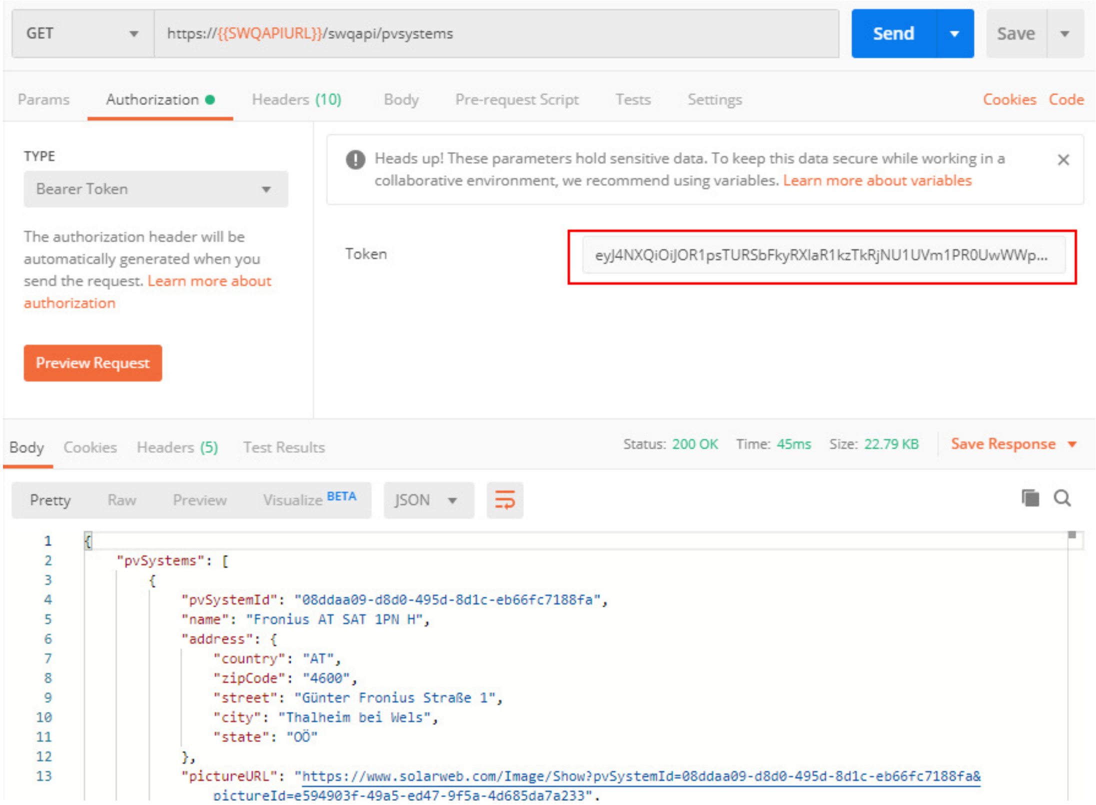
Headers (3)

	KEY	VALUE	DESCRIPTION
☑	AccessKeyId	{{APIKEYID}}	
☑	AccessKeyValue	{{APIKEYVALUE}}	
☑	Content-Type	application/json	
	Key	Value	Description

Status: 200 OK Time: 577ms Size: 1.18 KB

```
{
    "refreshToken": "2dd251b3-c648-362e-9d82-54511b15cc05",
    "jwtToken":
        "eyJ4NXQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiLCJraWQiOiJOR1psTURSbFkyRXla
        R1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiLCJhbGciOiJSUzI1NiJ9.
        eyJhdF9oYXNoIjoiTXlheWFqb3VXU2syRGF2OWowUWRpQSIsImF1ZCI6IlljNHhtcEIyVnlyR2phcUlraGoxbXJEOFZ6VWEiLCJzdWIiOiJsaXN0a
        WFrLmRhbmllbEBmcm9uaXVzLmNvbSIsIm5iZiI6MTU4MTU4NjgyMSwiYXpwIjoiWWM0eG1wQjJWeXJHamFxSWtoajFtckQ4VnpVYSIsImFtciI6Wy
        JwYXNzd29yZCJdLCJpc3MiOiIke2NhcmJvbi5wcm90b2NvbH06XC9cLyR7Y2FyYm9uLmhvc3R9XC9vYXV0aDJcL29pZGNkaXNjb3ZlcnkiLCJleHA
        iOjE1ODE1OTA0MjEsImlhdCI6MTU4MTU4NjgyMX0.
        SRBwKq-elHg0dmuoB6juiAkrKT-6G2nANvV8aNCDfZ08oHCCYFI4IUMAjStuzDqMZLFzwuilLUgBeS0-DxxA2HWt31aiqHq1sCkndSKOy3fin5vbX
        0kyfci9EQQ1KdGz2Vgei_D6_MGRhf72pwz_C8404jfAorvhULMLz41rXr3ED7HnHk3g1FFvBf4vVHiy10sWQu0Dy1vmfJ8-WU8Caj3VGThlljUw8q
        g5qMSj3MHj3J3o7ZIl1uekO7TsKKsHeIKYezRkC3TtAeIK7Mra-p-sIivl455YzfmwZMN68xpWvEvfpOmOf_tyWVqZmNajpJY992H-wYNoZzwPi6e
        9TA",
    "jwtTokenExpiration": "2020-02-13T10:28:44.282617Z"
}
```

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



https://{{SWQAPIURL}}/swqapi/pvsystems

TYPE

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

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 eyJ4NXQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWp...

Status: 200 OK Time: 45ms Size: 22.79 KB

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

3.2.3 Examples how to use a JWT:

HTTP header example

```
GET https://api.solarweb.com/swqapi/pvsystems HTTP/1.1
AccessKeyId: FKIAFEF58CFEFA94486F9C804CF6077A01AB
AccessKeyValue: 47c076bc-23e5-4949-37a6-4bcfcf8d21d6
Authorization: Bearer
eyJ4NXQiOiJPVEZsT1RCbE9HSmhZak15TlRFNU5XVTJPVGd6TnpVd04yTmpOVFV5WlRFeU1tRTNZVEZoTmciL
CJraWQiOiJPVEZsT1RCbE9HSmhZak15TlRFNU5XVTJPVGd6TnpVd04yTmpOVFV5WlRFeU1tRTNZVEZoTmciLC
JhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odW
Jlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJzdHJlZXRfYWRkcmVzcyI6IkZyb25pdXNwbGF0eiAxIiwiZ2V
uZGVyIjoiMSIsImFtciI6WyJDdXN0b21BdXRoZW50aWNhdG9yTG9jYWxNYWluIl0sImlzcyI6Imh0dHBzOlwv
XC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2tlbiIsImNvbnRhY3RfaWQiOiJmYzY0NWVjNi00NmQ1L
WU5MTEtOTEyOS0wMDUwNTZhMjYxNDAiLCJzb2xhcndlYl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMj
AtMDUtMTJUMjI6MDA6MDBaIiwic2lkIjoiOWE4NWYzMTgtMWE2Yi00NDMyLTliZGQtZGZkOGUzNWMwNjk2Iiw
iYXpwIjoibWZfbzlpVEF5S2VtTkxRVGE2U3A2SFlvbkNJYSIsImV4cCI6MTU3MjI1NzY4NywiaWF0IjoxNTcy
MjU0MDg3LCJlbWFpbCI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVmZXJyZWRfbGFuZ
3VhZ2UiOiJkZSIsImxvY2FsaXR5IjoiZGUiLCJzb2xhcndlYl91c2VyaWQiOiI4RkM3NzVFRC03QjlBLTREMz
YtQTcwNi1BQTkzMDA5RjNGRjkiLCJncm91cHMiOiJJbnRlcm5hbFwvZXZlcnlvbmUiLCJnaXZlbl9uYW1lIjo
iQWxleGFuZGVyIiwic29sYXJ3ZWJfcHJlbWl1bV9yb2xlIjoiMSIsIm5vbmNlIjoiYXNkZiIsImRhdGFfY29u
dGFjdF9jb21wbGV0ZSI6InllcyIsImF1ZCI6Im1mX285aVRBeUtlbU5MUVRhNlNwNkhZb25DSWEiLCJjX2hhc
2giOiJFWDZZS3NwaUxHY2tDc3RFNW02VmtnIiwibmJmIjoxNTcyMjU0MDg3LCJjb3VudHJ5X2lzb19jb2RlIj
oiQVQiLCJsb2NhdGlvbiI6IldlbHMiLCJwb3N0YWxfY29kZSI6IjQ2MDAiLCJmYW1pbHlfbmFtZSI6IktyZW5
odWJlciIsImRhdGFfYWNjb3VudF92YWxpZCI6InllcyJ9.Qzywri3WV6RHjv9Ng9fDCkzxPRprrNBx63bvoGk
bxa0hhyJVIdT132ylCyvmp84t_agvRG7Gk8HFcn5XrWcJ126mM-
CMQ5VFSLIyCmZm_vwoPXuOsk_pC1clX890WcqKDy3uaA3UdlLGYOke8kg-
ueQIxfkme1gSb2q0LEATaS8wdYYW-ODakMFd7zQvlRLJnXVXICeHwXrZu68fDRjdUlulu_13Ggi-
yHrtZTji_My_J57iMgJHTLaf7Gw3QzMMaZ85Kyz3jvqQgLFPylZgNBoz6ztzEdYd5Fhu-
kccibD662q9D7R5PHN88zY9ieVl9RYPyJZ5Rjk6qIYxb5Km6w
```

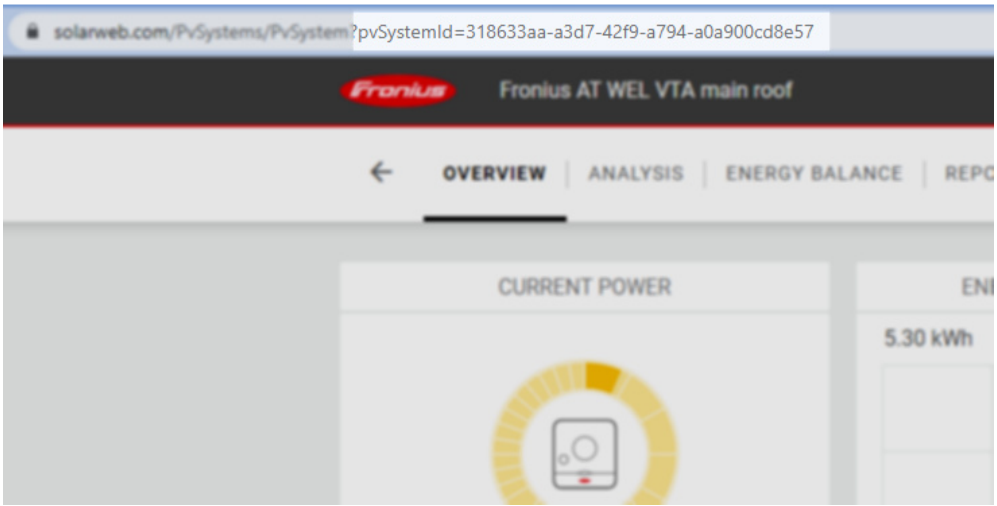
CURL example

```
curl -X GET "https://api.solarweb.com/swqapi/pvsystems" -H "accept: application/json"
 -H "AccessKeyId: FKIAFEF58CFEFA94486F9C804CF6077A01AB" -H "AccessKeyValue:
47c076bc-23e5-4949-37a6-4bcfcf8d21d6" -H "Authorization: Bearer
eyJ4NXQiOiJPVEZsT1RCbE9HSmhZak15TlRFNU5XVTJPVGd6TnpVd04yTmpOVFV5WlRFeU1tRTNZVEZoTmciL
CJraWQiOiJPVEZsT1RCbE9HSmhZak15TlRFNU5XVTJPVGd6TnpVd04yTmpOVFV5WlRFeU1tRTNZVEZoTmciLC
JhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5odW
Jlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJzdHJlZXRfYWRkcmVzcyI6IkZyb25pdXNwbGF0eiAxIiwiZ2V
uZGVyIjoiMSIsImFtciI6WyJDdXN0b21BdXRoZW50aWNhdG9yTG9jYWxNYWluIl0sImlzcyI6Imh0dHBzOlwv
XC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2tlbiIsImNvbnRhY3RfaWQiOiJmYzY0NWVjNi00NmQ1L
WU5MTEtOTEyOS0wMDUwNTZhMjYxNDAiLCJzb2xhcndlYl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwMj
AtMDUtMTJUMjI6MDA6MDBaIiwic2lkIjoiOWE4NWYzMTgtMWE2Yi00NDMyLTliZGQtZGZkOGUzNWMwNjk2Iiw
iYXpwIjoibWZfbzlpVEF5S2VtTkxRVGE2U3A2SFlvbkNJYSIsImV4cCI6MTU3MjI1NzY4NywiaWF0IjoxNTcy
MjU0MDg3LCJlbWFpbCI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVmZXJyZWRfbGFuZ
3VhZ2UiOiJkZSIsImxvY2FsaXR5IjoiZGUiLCJzb2xhcndlYl91c2VyaWQiOiI4RkM3NzVFRC03QjlBLTREMz
YtQTcwNi1BQTkzMDA5RjNGRjkiLCJncm91cHMiOiJJbnRlcm5hbFwvZXZlcnlvbmUiLCJnaXZlbl9uYW1lIjo
iQWxleGFuZGVyIiwic29sYXJ3ZWJfcHJlbWl1bV9yb2xlIjoiMSIsIm5vbmNlIjoiYXNkZiIsImRhdGFfY29u
dGFjdF9jb21wbGV0ZSI6InllcyIsImF1ZCI6Im1mX285aVRBeUtlbU5MUVRhNlNwNkhZb25DSWEiLCJjX2hhc
2giOiJFWDZZS3NwaUxHY2tDc3RFNW02VmtnIiwibmJmIjoxNTcyMjU0MDg3LCJjb3VudHJ5X2lzb19jb2RlIj
```

```
oiQVQiLCJsb2NhdGlvbiI6IldlbHMiLCJwb3N0YWxfY29kZSI6IjQ2MDAiLCJmYW1pbHlfbmFtZSI6IktyZW5
odWJlciIsImRhdGFfYWNjb3VudF92YWxpZCI6InllcyJ9.Qzywri3WV6RHjv9Ng9fDCkzxPRprrNBx63bvoGk
bxa0hhyJVIdT132ylCyvmp84t_agvRG7Gk8HFcn5XrWcJ126mM-
CMQ5VFSLIyCmZm_vwoPXuOsk_pC1clX890WcqKDy3uaA3UdlLGYOke8kg-
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).


solarweb.com/PvSystems/PvSystem?pvSystemId=318633aa-a3d7-42f9-a794-a0a900cd8e57
Fronius
Fronius AT WEL VTA main roof
OVERVIEW
ANALYSIS
ENERGY BALANCE
REPO
CURRENT POWER
ENI
5.30 kWh

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, SWQAPI makes use of HATEOAS principles to support pagination. Additionally, SWQAPI returns a "totalItemsCount" object.

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

Example:

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

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

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

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**%2b**01:00 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 api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/histdata?
from=2018-10-10T11:00:00Z&to=2018-10-11T12:00:00Z?channel=Temp1
  // zulu time notation

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

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

```
  // EST (-05:00 offset to zulu time)
```

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

Example URIs, all showing the same time request

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

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

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

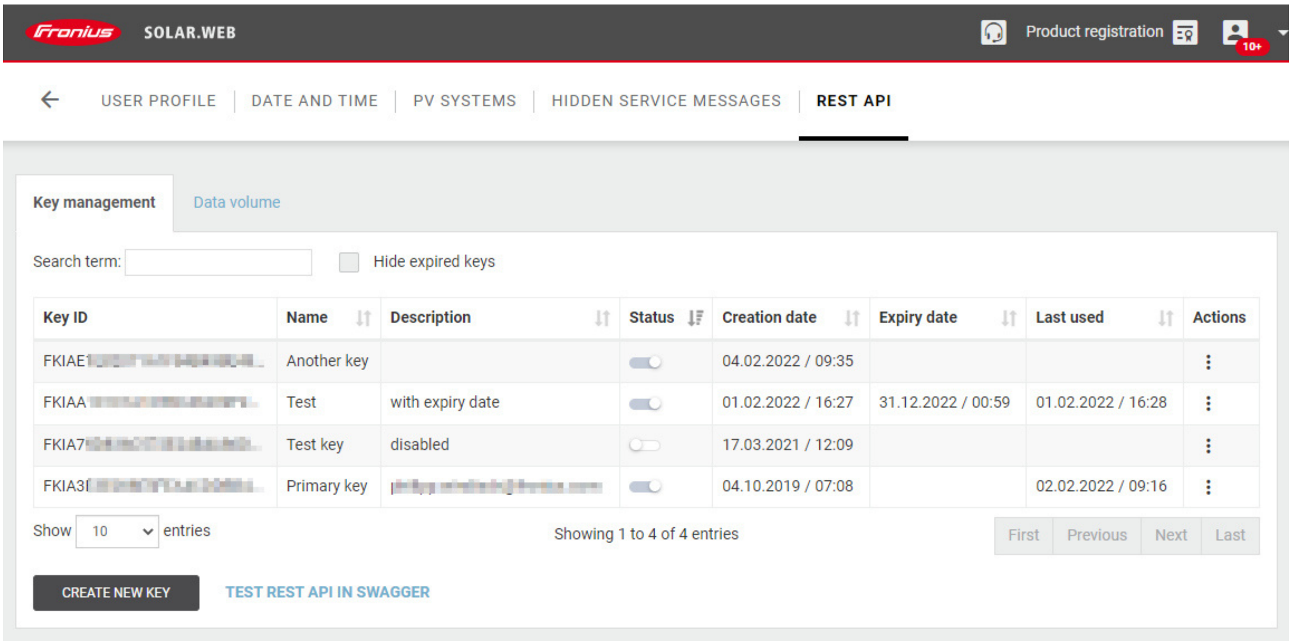
```
// data for August 31st, 2019; 12am zulu time

// 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 SUPPORTING UIS

4.1 API key management in Solar.web

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

For managing API keys in Solar.web, go to **User Settings** and then got to the **REST API** tab. This tab has two views, one of them is **Key management**. Here you 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.



Search term: □ Hide expired keys

Key ID	Name	Description	Status	Creation date	Expiry date	Last used	Actions
FKIAE1[redacted]	Another key		●	04.02.2022 / 09:35			⋮
FKIAA[redacted]	Test	with expiry date	●	01.02.2022 / 16:27	31.12.2022 / 00:59	01.02.2022 / 16:28	⋮
FKIA7[redacted]	Test key	disabled	○	17.03.2021 / 12:09			⋮
FKIA3[redacted]	Primary key	[redacted]	●	04.10.2019 / 07:08		02.02.2022 / 09:16	⋮

Show 10 entries

Showing 1 to 4 of 4 entries

Actions:

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

Recommendations:

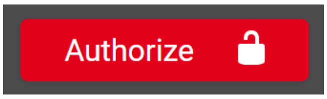
- If you have multiple developers, create separate keys for each developer. Create another key for automated tested, staging and production systems.
- If a key is compromised, especially keys for production, please renew the key:
  - Create a new key.
  - Set an expiry date for the old key or deactivate the old key as soon as the new one is deployed.
- If you want to implement periodic key renewals for security reasons:
  - Create a new key.

- Set an expiry date for the old key which gives you enough overlapping time to push the new key to all relevant systems.

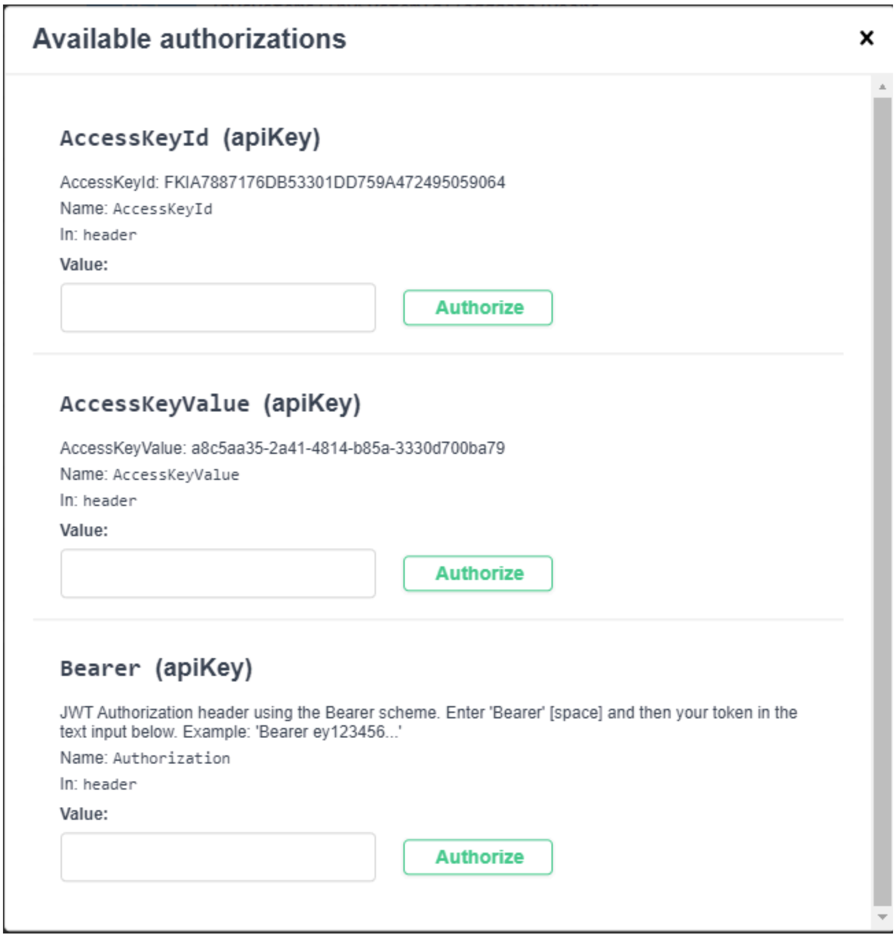
4.2 Working with API keys in Swagger UI

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

Press the *Authorize* button in Swagger UI.



Enter both the accesskey ID and its 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)

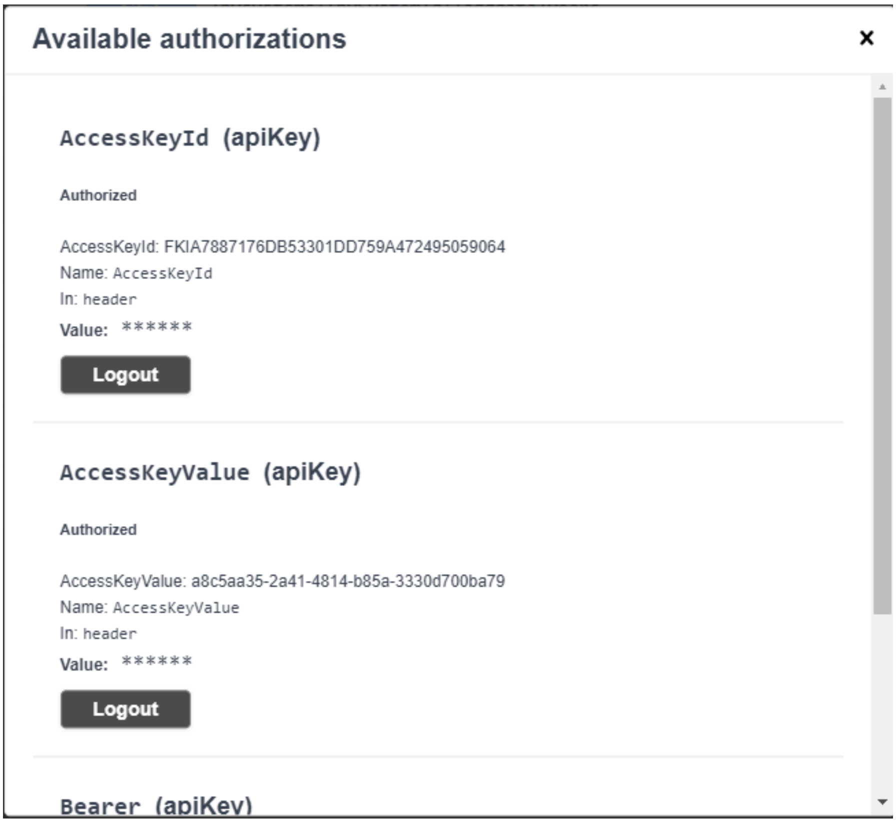
JWT Authorization header using the Bearer scheme. Enter 'Bearer' [space] and then your token in the text input below. Example: 'Bearer ey123456...'

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: ******

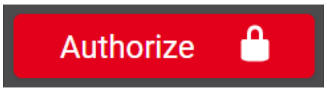
AccessKeyValue (apiKey)

Authorized

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

Bearer (apiKey)

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



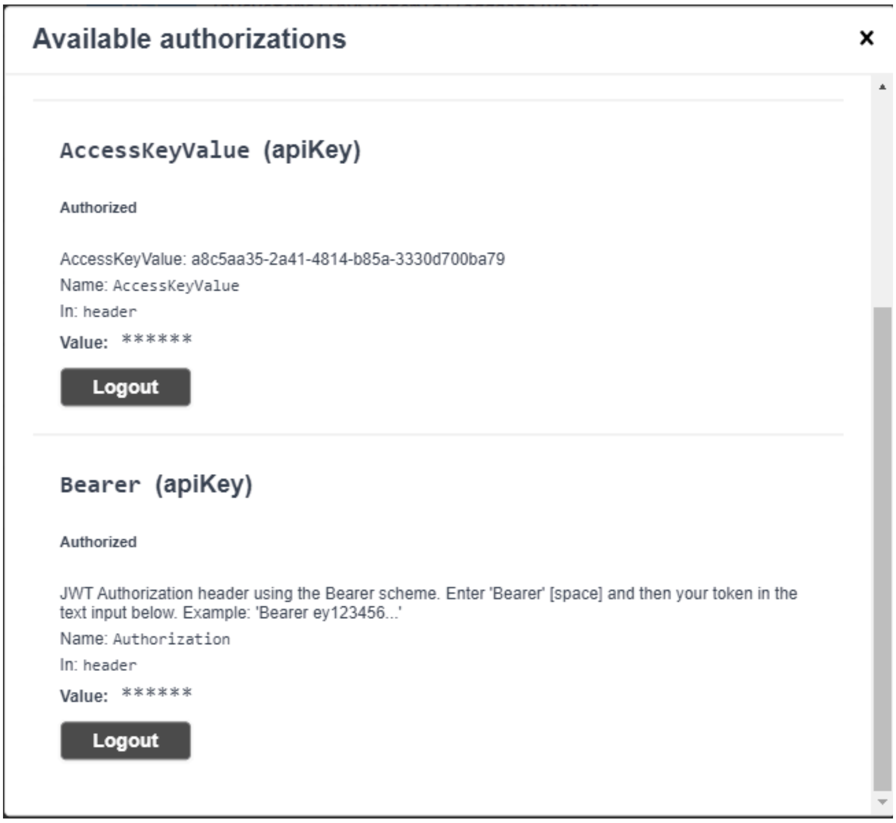
⚠ **Swagger UI does not verify the keys**

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: ******

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: ******

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

5 API REFERENCE

5.1 User impersonation calls

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

Input objects

JSON object input construction:

Type	Objects
credentials	/ userId (String) / password (String)

Example input

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

Response objects

JSON object answer construction:

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

Example responses

```
{
    "refreshToken": "98a47454-b650-34b8-9a8c-27adae447ab7",
    "jwtToken":
"eyJ4NXQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEi
LCJraWQiOiJOR1psTURSbFkyRXlaR1kzTkRjNU1UVm1PR0UwWWpGaVpXWTBaamcxWVdOa09EWmtNRE5rTVEiL
CJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiS2hLZVZsc0lPXy1tWDhvZkJZSzdJZyIsImF1ZCI6IlljNHhtc
EIyVnlyR2phcUlraGoxbXJEOFZ6VWEiLCJzdWIiOiJodWV0dG5lci50aG9tYXNybmRAZnJvbml1cy5jb20iLC
JuYmYiOjE1ODUyOTQ2NjcsImF6cCI6IlljNHhtcEIyVnlyR2phcUlraGoxbXJEOFZ6VWEiLCJhbXIiOlsicGF
zc3dvcmQiXSwiaXNzIjoiJHtjYXJib24ucHJvdG9jb2x9OlwvXC8ke2NhcmJvbi5ob3N0fVwvb2F1dGgyXC9v
aWRjZGlzY292ZXJ5IiwiZXhwIjoxNTg1Mjk4MjY3LCJpYXQiOjE1ODUyOTQ2Njd9.HUXi1sySzyLqx2e0dLpr
0sszi-
YiI3nGNB4GZDDwIwVHUHC4s6ED8BqfvkfFn3s45LkvJQEvqb_Wd3QtMGnzOLEZ3RdK3A8GWdsDChVq_nzlP4F
GC6b5lPoz9Xi6mH_pcxt36rzA2-vjl_e6cTOrTXsIeIzOjZVNSZRAJ4-
A5HpmEuvraoArAGUqc_yTntbfALhfJQkfsjoDAJRAfZXLknTvDKm2vMd0-
uXjTQHM2dKAWGAz6r39cLQ24sFIIC7MDgIp4GpNVBCLFSNzkK7mV3fSEQvgIdAFMhEP4CY4lMTItLxdfRKxcf
5SA7o2fU0-_710frdFYvrkesorDCiyfg",
    "jwtTokenExpiration": "2020-03-27T08:37:48.8710788Z"
}
```

5.1.2 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	swqapi/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>	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). Scopes allow users to be logged in from multiple devices and in Solar.web in parallel. ⚠ Must not include any scope values not originally granted, and if omitted is treated as equal to the originally granted scope.

Example calls

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

Response objects

JSON object answer construction:

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

Example responses

```
{
  "refreshToken": "c5f51670-e2ca-35b7-acc2-c32c8ceebc69",
  "jwtToken":
"eyJ4NXQiOiJPVEZsT1RCbE9HSmhZak15TlRFNU5XVTJPVGd6TnpVd04yTmpOVFV5WlRFeU1tRTNZVEZoTmci
LCJraWQiOiJPVEZsT1RCbE9HSmhZak15TlRFNU5XVTJPVGd6TnpVd04yTmpOVFV5WlRFeU1tRTNZVEZoTmciL
CJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoicXZUYi04RzJvQnR5Qko1SEt5Z1l3USIsInN1YiI6ImtyZW5od
WJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJzdHJlZXRfYWRkcmVzcyI6IkZyb25pdXNwbGF0eiAxIiwiZ2
VuZGVyIjoiMSIsImFtciI6WyJDdXN0b21BdXRoZW50aWNhdG9yTG9jYWxNYWluIl0sImlzcyI6Imh0dHBzOlw
vXC9sb2dpbi5mcm9uaXVzLmNvbVwvb2F1dGgyXC90b2tlbiIsImNvbnRhY3RfaWQiOiJmYzY0NWVjNi00NmQ1
LWU5MTEtOTEyOS0wMDUwNTZhMjYxNDAiLCJzb2xhcndlYl9wcmVtaXVtX2V4cGlyYXRpb25fZGF0ZSI6IjIwM
jAtMDUtMTJUMjI6MDA6MDBaIiwic2lkIjoiOWE4NWYzMTgtMWE2Yi00NDMyLTliZGQtZGZkOGUzNWMwNjk2Ii
wiYXpwIjoibWZfbzlpVEF5S2VtTkxRVGE2U3A2SFlvbkNJYSIsImV4cCI6MTU3MjI1NzY4NywiaWF0IjoxNTc
yMjU0MDg3LCJlbWFpbCI6ImtyZW5odWJlci5hbGV4YW5kZXJAZnJvbml1cy5jb20iLCJwcmVmZXJyZWRfbGFu
Z3VhZ2UiOiJkZSIsImxvY2FsaXR5IjoiZGUiLCJzb2xhcndlYl91c2VyaWQiOiI4RkM3NzVFRC03QjlBLTREM
zYtQTcwNi1BQTkzMDA5RjNGRjkiLCJncm91cHMiOiJJbnRlcm5hbFwvZXZlcnlvbmUiLCJnaXZlbl9uYW1lIj
oiQWxleGFuZGVyIiwic29sYXJ3ZWJfcHJlbWl1bV9yb2xlIjoiMSIsIm5vbmNlIjoiYXNkZiIsImRhdGFfY29
udGFjdF9jb21wbGV0ZSI6InllcyIsImF1ZCI6Im1mX285aVRBeUtlbU5MUVRhNlNwNkhZb25DSWEiLCJjX2hh
c2giOiJFWDZZS3NwaUxHY2tDc3RFNW02VmtnIiwibmJmIjoxNTcyMjU0MDg3LCJjb3VudHJ5X2lzb19jb2RlI
joiQVQiLCJsb2NhdGlvbiI6IldlbHMiLCJwb3N0YWxfY29kZSI6IjQ2MDAiLCJmYW1pbHlfbmFtZSI6IktyZW
5odWJlciIsImRhdGFfYWNjb3VudF92YWxpZCI6InllcyJ9.Qzywri3WV6RHjv9Ng9fDCkzxPRprrNBx63bvoG
kbxa0hhyJVIdT132ylCyvmp84t_agvRG7Gk8HFcn5XrWcJ126mM-
CMQ5VFSLIyCmZm_vwoPXuOsk_pC1clX890WcqKDy3uaA3UdlLGYOke8kg-
ueQIxfkme1gSb2q0LEATaS8wdYYW-ODakMFd7zQvlRLJnXVXICeHwXrZu68fDRjdUlulu_13Ggi-
yHrtZTji_My_J57iMgJHTLaf7Gw3QzMMaZ85Kyz3jvqQgLFPylZgNBoz6ztzEdYd5Fhu-
kccibD662q9D7R5PHN88zY9ieVl9RYPyJZ5Rjk6qIYxb5Km6w",
  "jwtTokenExpiration": "2019-12-18T13:24:19Z"
}
```

5.1.3 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	swqapi/iam/jwt/{refresh-token}	RevokeJwt	Revokes a JWT refresh token

Filters and parameters

n/a

Example calls

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

Response objects

n/a

Example responses

n/a

5.2 Generic information calls

5.2.1 Info: Get release information

Use cases for web developers

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

Methods

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

Filters and parameters

n/a

Example calls

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

Response objects

JSON object answer construction:

Type	Objects
info/release	ReleaseVersion (String) ReleaseDate (String, Date)

Example responses

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

5.2.2 Info: Get user information

Use cases for web developers

- I want to show the customer's address data.
- I want to know if the customer is a premium customer.

- I want to know if the customer has accepted the (latest) Terms of Use.

Methods

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

Filters and parameters

n/a

Example calls

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

Response objects

JSON object answer construction:

Type	Objects
user	- name (Object) - title (String) - e.g. "Mr.", "Mrs." - firstName (String) - lastName (String) - address (Object) - street (String) - zipCode (String) - city (String) - state (String) - country (String) - contactInformation (Object) - telephone (String) - email (String) - settings (Object) - timeZone (String) - in Olson format - dateFormat (String) - "DD.MM.YYYY", "MM.DD.YYYY", "YYYY.MM.DD", "DD/MM/YYYY", "MM/DD/YYYY", "YYYY/MM/DD" - timeFormat (String) - "12h", "24h" - language (String) - in ISO language code - accountAttributes (Object) - premiumMembership (Boolean) - termsAcceptedLatest (Boolean) - termsAcceptedVersion (Integer)

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,
        "termsAcceptedVersion": 2
    }
}
```

5.3 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

5.3.1 Metadata: Get PV system information

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

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns pv-systems from a starting <offset> and returning not more than <limit> items.
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices that a PV system should contain: / inverter / sensor

Filter	Description
	/ battery / smartmeter / ohmpilot / datalogger / evcharger
?meteo=<level>	Filters for systems with extended (meteo=pro) weather data or with just basic (meteo=light) weather data.

Example calls

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

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

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

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

GET api.solarweb.com/swqapi/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	/ lastImport (String, UTC timestamp) / installationDate (String, UTC timestamp) / pvSystemId (String) / name (String) / address (Object) / street (String) / zipCode (String) / city (String) / state (String) / country (String) / timezone (String, Olson format) / pictureURL (String, URL) / peakPower (Number) / meteoData (String)

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

5.3.2 Metadata: Count PV systems

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

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: / inverter / sensor / battery / smartmeter / ohmpilot / datalogger / evcharger
?meteo=<level>	Filters for systems with extended (meteo=pro) weather data or with just basic (meteo=light) weather data.

Example calls

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

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

Response objects

JSON object answer construction:

Type	Objects
n/a	/ count (Number)

Example responses

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

5.3.3 Metadata: Enumerate PV system IDs

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

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns pv-systems from a starting <offset> and returning not more than <limit> items. 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: - inverter - sensor - battery - smartmeter - ohmpilot - datalogger - evcharger
?meteo=<level>	Filters for systems with extended (meteo=pro) weather data or with just basic (meteo=light) weather data.

Example calls

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

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

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

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

Response objects

JSON object answer construction:

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

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.

5.3.4 Metadata: Get device information

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns devices from a starting <offset> and returning not more than <limit> items.
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices that a PV system should contain: / inverter / sensor / battery / smartmeter / ohmpilot / datalogger / evcharger
?isActive=<state>	Filters for active (isActive=true) or inactive (isActive=false) devices only.

Example calls

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

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

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

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

Response objects

JSON object answer construction:

Type	Objects
Inverter	/ deviceType (String - "Inverter") / deviceId (String) / deviceName (String) / deviceManufacturer (String - usually "Fronius") / serialnumber (String) / deviceTypeDetails (String) / dataloggerId (String) / nodeType (Integer) / numberMPPTrackers (Number) / numberPhases (Number - 1 to 3) / peakPower (Object) / dc1 (Number) / dc2 (Number) / ... ⚠ 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).

Type	Objects
Battery	/ deviceType (String - "Battery") / deviceID (String) / deviceName (String) / deviceManufacturer (String) / serialNumber (String) / dataloggerId (String) / capacity (Number) / firmware (Object) / updateAvailable (Boolean) / installedVersion (String) / availableVersion (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp)
Ohmpilot	/ deviceType (String - "Ohmpilot") / deviceID (String) / deviceName (String) / deviceManufacturer (String - "Fronius")

Type	Objects
	/ nominalAcPower (Number) / firmware (Object) / updateAvailable (Boolean) / installedVersion (String) / availableVersion (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp)
Sensor	/ deviceType (String - "Sensor") / deviceID (String) / deviceName (String) / deviceManufacturer (String) / firmware (Object) / updateAvailable (Boolean) / installedVersion (String) / availableVersion (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp) / dataloggerId (String) / sensors (Array of objects) / sensorType (String) / sensorName (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp)
Smart Meter	/ deviceType (String - "SmartMeter") / deviceID (String) / deviceName (String) / deviceManufacturer (String) / deviceCategory (String) / deviceLocation (String) / serialNumber (String) / dataloggerId (String)

	/ serialnumber (String)
	/ dataloggerId (String)
	/ firmware (Object)
	  / updateAvailable (Boolean)
	  / installedVersion (String)
	  / availableVersion (String)
	/ isActive (Boolean)
	/ activationDate (String, UTC timestamp)
	/ deactivationDate (String, UTC timestamp)
	/ sensors (Array of objects)
	  / sensorType (String)
	  / sensorName (String - "Temperature")
	  / isActive (Boolean)
	  / activationDate (String, UTC timestamp)
	  / deactivationDate (String, UTC timestamp)

EVCharger	/ deviceType (String - "EVCharger") / deviceID (String) / deviceName (String - "WattPilot") / deviceManufacturer (String - "Fronius") / serialnumber (String) / dataloggerId (String) / firmware (Object) / updateAvailable (Boolean) / installedVersion (String) / availableVersion (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp) / isOnline (Boolean)

Type	Objects
	/ firmware (Object) / updateAvailable (Boolean) / installedVersion (String) / availableVersion (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp)

Type	Objects
Datalogger	/ deviceType (String - "Datalogger") / deviceID (String) / deviceName (String) / deviceManufacturer (String - "Fronius") / firmware (Object) / updateAvailable (Boolean) / installedVersion (String) / availableVersion (String) / isActive (Boolean) / activationDate (String, UTC timestamp) / deactivationDate (String, UTC timestamp) / dataloggerId (String) / isOnline (Boolean)

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,
    "numberMPPTrackers": 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,
    "numberMPPTrackers": 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 DataLogger)

```
{
    "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": "4000000051",
        "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",
    "dataloggerId": "240.196164",
    "capacity": 9600,
    "firmware": {
        "updateAvailable": false,
        "installedVersion": "",
        "availableVersion": ""
    },
    "isActive": true,
    "activationDate": null,
    "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,
            "numberMPPTrackers": 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
/ AC battery / External / Generation meter / Grid	/ AC storage unit / Building services / Climate control / cooling systems / Combined heat and power station (CHP)

deviceLocation	deviceCategory
/ Load meter / Sub meter	/ Electric vehicle / Heat pump / LGC Meter / Other / Other heating system / Photovoltaic inverter / Photovoltaic inverter + storage unit / Primary Meter / Pumps / White goods / Wind turbine

Sensor types:

sensorType
/ Energy / Insolation / Irradiation / Precipitation / Temperature / Velocity

5.3.5 Metadata: Count devices

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: inverter sensor battery smartmeter ohmpilot datalogger evcharger
?isActive=<state>	Filters for active (isActive=true) or inactive (isActive=false) devices only.

Example calls

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


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

Response objects

JSON object answer construction:

Type	Objects
n/a	count (Number)

Example responses

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

5.3.6 Metadata: Enumerate device IDs

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?offset=<offset>&limit=<limit>	Supports pagination, returns devices from a starting <offset> and returning not more than <limit> items.
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: - inverter - sensor - battery - smartmeter - ohmpilot - datalogger - evcharger
?isActive=<state>	Filters for active (isActive=true) or inactive (isActive=false) devices only.

Example calls

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

GET api.solarweb.com/swqapi/pvsystems/20bb600e-019b-4e03-9df3-a0a900cda689/devices-
list?type=inverter,sensor,battery
// returns the device IDs of all inverters, sensors and batteries in the given PV
system
```

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

Response objects

JSON object answer construction:

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

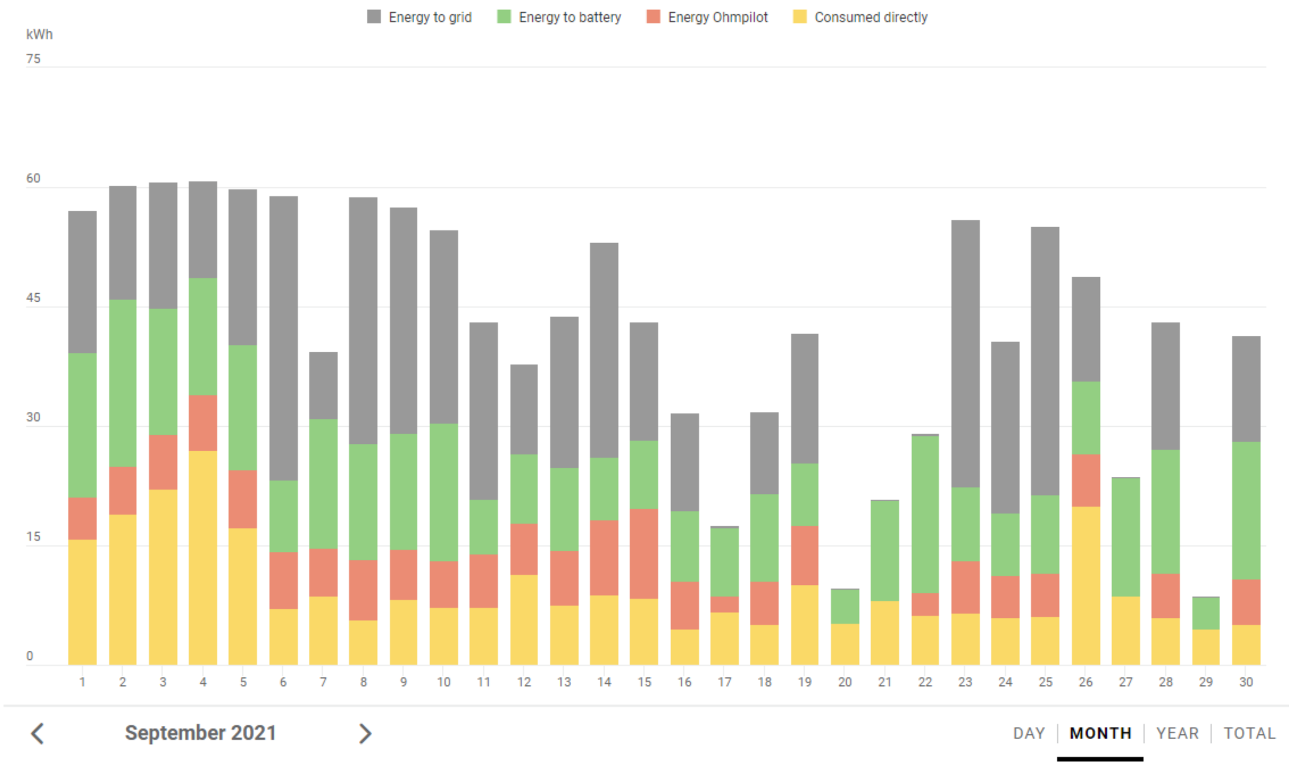
Example responses

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

5.4 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:



5.4.1 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	swqapi/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: / / yyyy for years / yyyy-MM or yyyyMM for months / yyyy-MM-dd or yyyyMMdd for days
?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: / / yyyy for years / yyyy-MM or yyyyMM for months / yyyy-MM-dd or yyyyMMdd for days
?period=<period>	Limits the time series for the query to a period. Encoding: / / "total" delivers total values over whole system lifetime / "years" delivers values for each year of system lifetime / yyyy delivers all months of requested year / yyyy-MM delivers all days of requested month Note: If <period> parameter is used, then <from>, <to> and <duration> parameters are not allowed, and vice versa.
?offset=<offset>&limit=<limit>	Supports pagination, returns items from a starting <offset> and returning not more than <limit> items (items = objects in the "Data" array).

Example calls

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

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

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

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

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

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

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

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

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

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

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

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

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

Response objects

JSON object answer construction:

Objects

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

Supported value channels

Type	Channels	Comment
PV system energy flow with Smart Meter	/ EnergyFeedIn / EnergyPurchased / EnergySelfConsumption / EnergyDirectConsumption	Requires Smart Meter; otherwise NULL
	/ EnergyBattCharge / EnergyBattDischarge / EnergyBattChargeGrid / EnergyBattDischargeGrid	Requires Smart Meter and battery; otherwise NULL
	/ OhmpilotEnergy	Uses ChannelType.FromGenToOhmPilot, NOT OhmPilotEnergy! Requires Smart Meter and Ohmpilot; otherwise NULL
	/ EnergyEVCCharge / EnergyEVCChargeGrid / EnergyEVCChargeBatt	Requires Smart Meter and Wattpilot; otherwise NULL
PV system output without Smart Meter	/ EnergyOutput	Only if there is no Smart Meter; NULL with Smart Meter
PV system totals	/ EnergyProductionTotal / EnergyConsumptionTotal / EnergySelfConsumptionTotal	Requires Smart Meter; otherwise NULL
PV system CO2 savings	/ SavingsCO2 / SavingsTrees	

Type	Channels	Comment
	/ SavingsTravelCar / SavingsTravelPlane	
PV system savings	/ Profits / Earnings / Savings	Requires profit settings in Solar.web; otherwise NULL

Example responses

Example for retrieving all channels for total lifetime

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

5.4.2 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	swqapi/pvsystems/{pv-system-id}/devices/{device-id}/aggrdata	GetDeviceAggregatedData	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: - yyyy for years - yyyy-MM or yyyyMM for months - yyyy-MM-dd or yyyyMMdd for days
?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: - yyyy for years - yyyy-MM or yyyyMM for months - yyyy-MM-dd or yyyyMMdd for days
?period=<period>	Limits the time series for the query to a period. Encoding:

Filter	Description
	/ / "total" delivers total values over whole system lifetime / "years" delivers values for each year of system lifetime / yyyy delivers all months of requested year / yyyy-MM delivers all days of requested month Note: If <period> parameter is used, then <from>, <to> and <duration> parameters are not allowed, and vice versa.
?offset=<offset>&limit=<limit>	Supports pagination, returns items from a starting <offset> and returning not more than <limit> items (items = objects in the "Data" array).

Example calls

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

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

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

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

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

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

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

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

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

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

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

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

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

Response objects

JSON object answer construction:

Objects

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

Supported channels:

Type	Channels
Inverter	/ EnergyExported / EnergyDC1 / EnergyDC2

Example responses

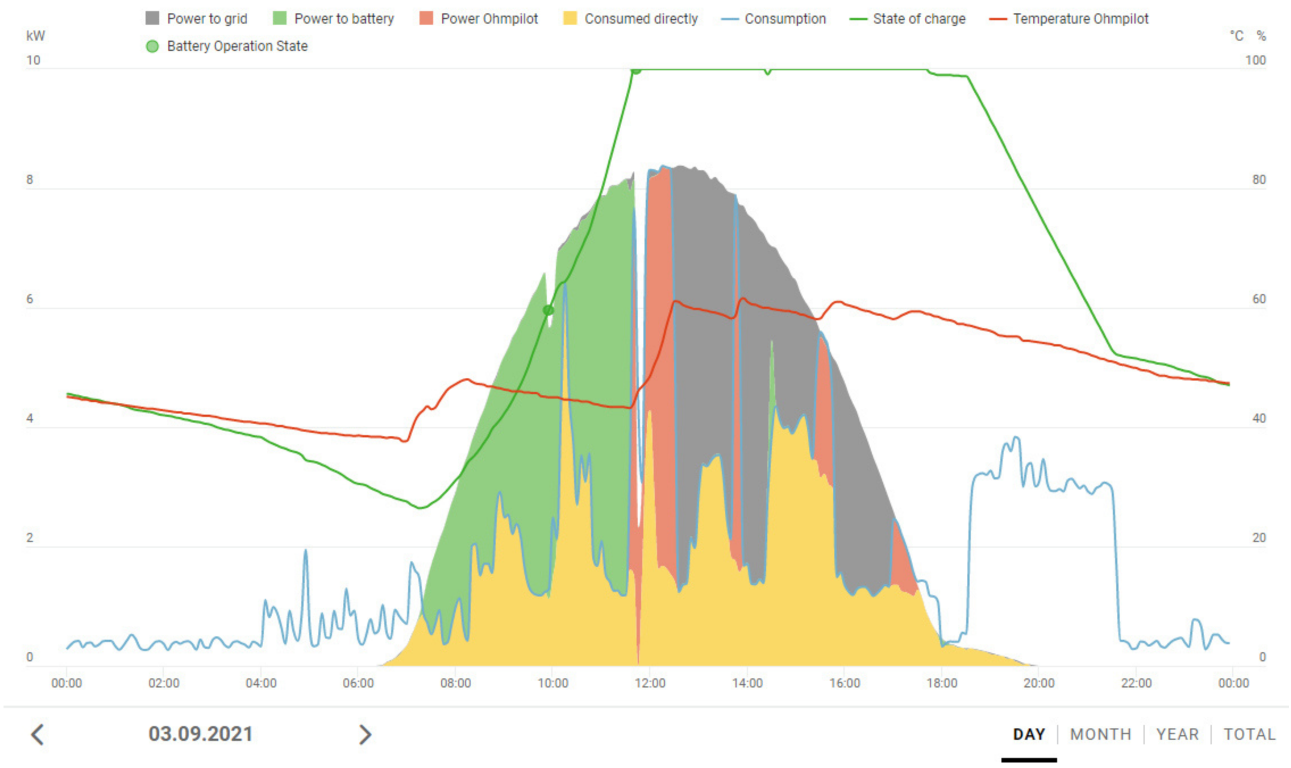
```
{
    "pvSystemId": "20bb600e-019b-4e03-9df3-a0a900cda689",
    "deviceId": "b582f1b9-95b9-49db-800b-6b042e9938b4",
    "data": [
        {
            "logDateTime": "total",
            "channels": [
                {
                    "channelName": "EnergyExported",
                    "channelType":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

5.5 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:



5.5.1 Histdata: Historical data for a PV system

Use cases for web developers

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

Restrictions

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

Methods

Method	End point and objects	Event name	Description
GET	swqapi/ pvsystems/{pv-	GetSystemHistoricalDat a	Gets historical data for a given PV system and time range. The data

Method	End point and objects	Event name	Description
	system-id}/ histdata		resolution is 5min. The data is returned a JSON object.

Filters and parameters

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

Example calls

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

Response objects

JSON object answer construction:

Objects
- pvSystemId (String) - data (Array)   - logDateTime (UTC Time)   - logDuration (Integer - unit: seconds)   - channels (Array)   - channelName (String)

Objects
/ channelType (String) / unit (String) / value (Number, String)

Supported value channels:

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

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": "EnergyEVCCharge",
                    "channelType": "Energy",
                    "unit": "Wh",
                    "value": 0.0
                },
                {
                    "channelName": "EnergyEVCChargeBatt",
                    "channelType": "Energy",
                    "unit": "Wh",
                    "value": 0.0
                },
                {
                    "channelName": "EnergyEVCChargeGrid",
                    "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": "EnergyEVCChargeTotal",
                    "channelType": "Energy",
                    "unit": "Wh",
                    "value": 0.0
                }
            ]
        }
    ]
}
```

5.5.2 Histdata: Historical data for a device

Use cases for web developers

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

Restrictions

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

Methods

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

Filters and parameters

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

Example calls

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

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

Response objects

JSON object answer construction:

Objects

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

Supported channels:

Type	Channels
Inverter	EnergyExported EnergyImported EnergyDC1 EnergyDC2 EnergyDCn ... CurrentA CurrentB CurrentC CurrentDC1 CurrentDC2 CurrentDCn ... VoltageA VoltageB VoltageC VoltageAB VoltageBC VoltageCA VoltageDC1 VoltageDC2 VoltageDCn ... ApparentPower ReactivePower PowerFactor StandardizedPower

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

Type	Channels
Sensor	Temp1 Temp2 Insolation Digital1 Digital2 Digital3 Digital1Energy Digital2Energy Digital3Energy
Battery	BattOpState BattSOC
Ohmpilot	OhmpilotTemp OhmpilotEnergyAbs OhmpilotEnergy OhmpilotError
EV charger	EnergyChargeTotal VoltageA VoltageB VoltageC

Example responses

Inverter

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

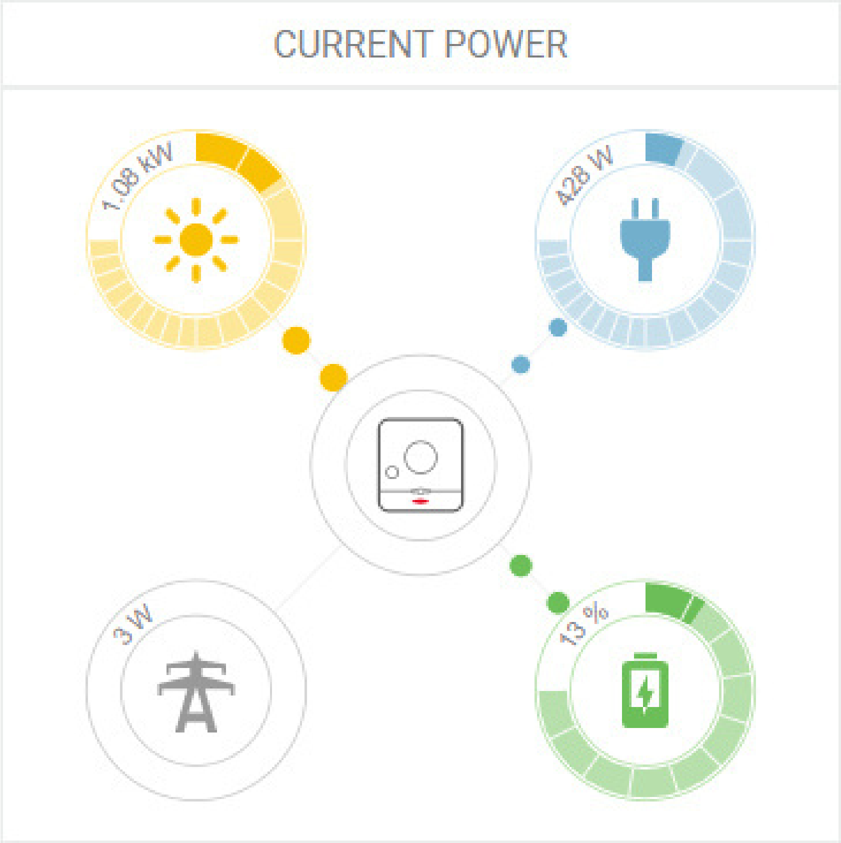
```
{
    "pvSystemId": "73733052-6f10-47a5-9746-7f7e76ebcb8c",
    "deviceId": "cee3e54d-0191-4700-8504-aea000d0d839",
    "data": [
        {
            "logDateTime": "2022-08-14T11:00:00+10:00",
            "logDuration": 300,
            "channels": [
                {
                    "channelName":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

    ]
}
```

5.6 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:



5.6.1 Flowdata: Realtime power flow data of a PV system

Use cases for web developers

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

Methods

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

Method	End point and objects	Event name	Description
	system-id}/flowdata		

Filters and parameters

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

Example calls

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

Response objects

JSON object answer construction:

<table>
<tr><th>Objects</th></tr>
<tr><td>
<ul>
<li>pvSystemId (String)</li>
<li>status
<ul>
<li>isOnline (Boolean)</li>
<li>battMode (Number)</li>
</ul>
</li>
<li>data
<ul>
<li>logDateTime (UTC time)</li>
<li>channels (Array)
<ul>
<li>channelName (String)</li>
<li>channelType (String)</li>
<li>unit (String)</li>
<li>value (Number, String)</li>
</ul>
</li>
</ul>
</li>
</ul>
</td></tr>
</table>

Supported value channels:

Type	Channels	Remarks
PV system	PowerFeedIn PowerLoad PowerBattCharge PowerOhmpilot PowerPV PowerOutput BattSOC RateSelfConsumption RateSelfSufficiency PowerEVCTotal	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. If there is no Smart Meter, the inverter returns PowerOutput instead. PowerBattCharge is negative when charging
	isOnline battMode	when is the system shown online? timeout when loosing contact? which battmodes are available and what do they mean

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
      }
    ]
  }
}
```

5.6.2 Flowdata: Realtime power flow data of a device

Use cases for web developers

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

Methods

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

Method	End point and objects	Event name	Description
			The data is returned as a JSON object.

Filters and parameters

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

Example calls

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

Response objects

JSON object answer construction:

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

Type	Objects
Wattpilot (EV Charger)	PowerTotal EVCMode

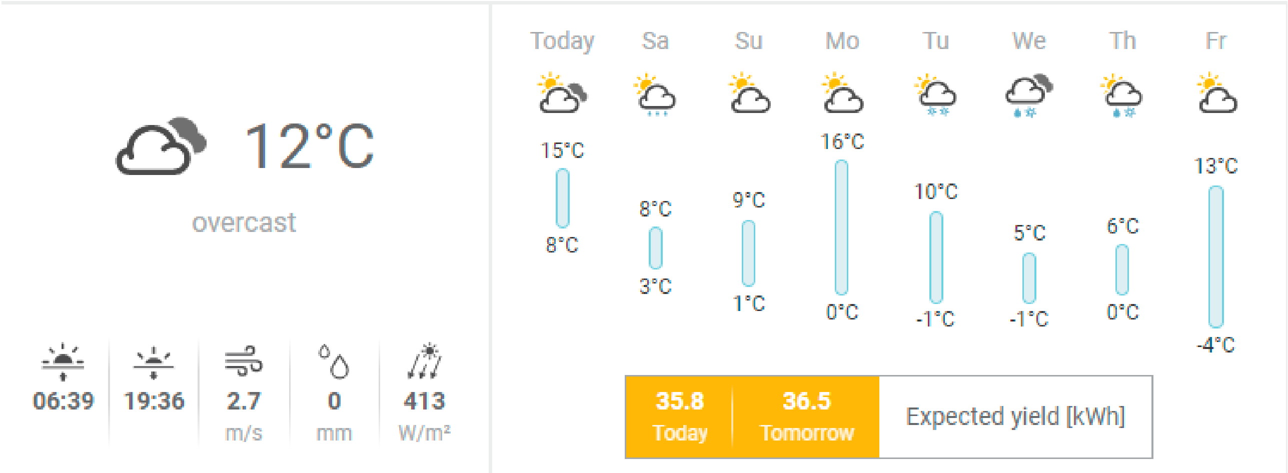
Example responses

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

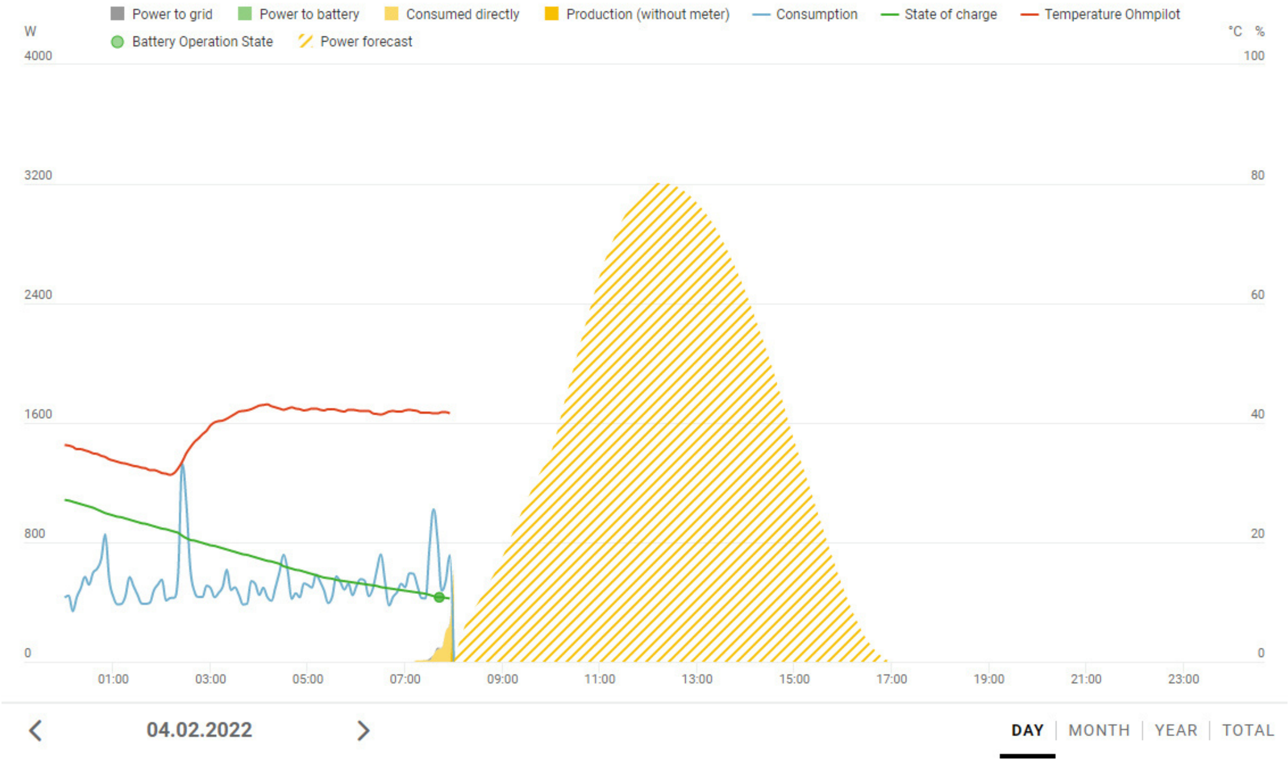
5.7 Weather data calls

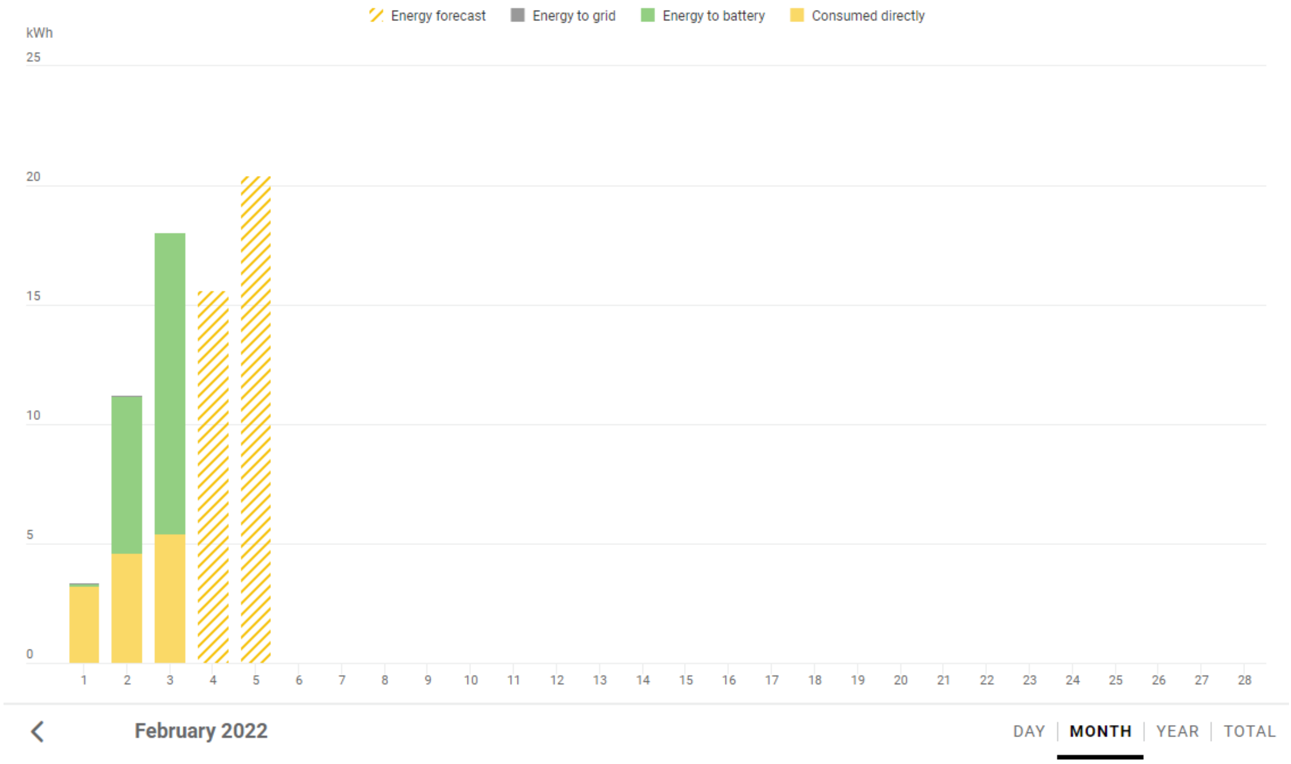
This set provides weather and energy forecast information for PV systems.

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):





5.7.1 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).

5.7.2 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	swqapi/pvsystems/{pv-system-id}/weather/current	GetSystemWeatherCurrent	Gets the current weather information for a given PV system.

Filters and parameters

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

Example calls

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

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

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

Response objects

JSON object answer construction:

Objects

- pvSystemId (String)
- data
    - logDateTime (DateTime)
    - channels (Array)
        - channelName (String)
        - channelType (String)

Objects
- unit (String) - value (Number or Object)

Supported value channels:

Type	Channels	Remarks
PV system	- Temp - WindSpeed - Precipitation - CloudCover - Daylight - Symbol	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
            }
        ]
    }
}
```

5.7.3 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	swqapi/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.

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: / zulu (default): returns time in UTC zulu time. / local: returns time in PV system's local UTC time (local time + UTC offset).
?from=<start>&to=<end>	Limits the time series for the query. / <start> and <end> are days, local time of the PV system. / Alternatively, <start> can also be "today" (default) or "tomorrow"; <end> can be "tomorrow". / Date format encoding: "yyyyMMdd", "yyyy-MM-dd".
?from=<start>&duration=<days>	Limits the time series for the query. / <start> is a day, local time of the PV system. / Alternatively, <start> can also be "today" (default) or "tomorrow". / <duration> is the number of days. / <duration>=1 means only one single day. / If <duration> is missing, the maximum period is assumed, i.e. full 9 days of weather forecast. / Date format encoding: "yyyyMMdd", "yyyy-MM-dd".

Example calls

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

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

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

Response objects

JSON object answer construction:

Type	Objects
	/ pvSystemId (String) / data / logDateTime (Date)

Type	Objects
	/ channels (Array) / channelName (String) / channelType (String) / unit (String) / value (Number or Object)

Supported value channels:

Type	Channels	Remarks
PV system	/ Temp / Daylight / Symbol / EnergyExpected	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"
                }
            ]
        }
    ]
}
```

5.7.4 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	swqapi/pvsystems/{pv-system-id}/weather/energyforecast	GetSystemWeatherEnergyforecast	Gets energy production forecast information for up to next 2 days.

Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: - zulu (default): returns time in UTC zulu time. - local: returns time in PV system's local UTC time (local time + UTC offset).
?from=<start>&to=<end>	Limits the time series for the query.

Filter	Description
	- <start> and <end> are ISO-8601 time values. - Time format encoding: "yyyyMMddTHHmmssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?from=<start>&duration=<hours>	Limits the time series for the query. - <start> is an ISO-8601 time value. - <duration> a number of 1 to 48 hours. - If <duration> is missing, the maximum period is assumed, i.e. full two days of energy forecast. - Time format encoding: "yyyyMMddTHHmmssTZD", "yyyy-MM-ddTHH:mm:ssTZD".

Example calls

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

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

Response objects

JSON object answer construction:

Type	Objects
	- pvSystemId (String) - data (Array) - logDateTime (DateTime) - logDuration (Integer - unit: seconds) - channels (Array) - ChannelName (String) - ChannelType (String) - Unit (String) - Value (Number)

Supported value channels:

Type	Channels	Remarks
PV system	- EnergyExpected	

Example responses

```json
{
  "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"
        }
      ]
    }
  ]
}
```

5.8 System messages calls

5.8.1 Messages: Get PV system messages

Use cases for web developers

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

Methods

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

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

Filters and parameters

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

Filter	Description
?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: / inverter / sensor / battery / smartmeter / ohmpilot / datalogger / evcharger

Example calls

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

Response objects

JSON object answer construction:

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

Example responses

```
[
  {
    "pvSystemId": "d587d328-3cac-4953-9caf-a4bb009def43",
    "deviceId": "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"
  }
]
```

5.8.2 Messages: Count PV system messages

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?from=<start>&to=<end>	Limits the time series for the query. - <start> and <end> are ISO-8601 time values. - If "to" is missing, then "to" is considered "now". ("from" must not be empty.) - Time format encoding: "yyyyMMddTHHmmssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?statetype=<type>	Filters by StateType, e.g. "Error", "Event".
?stateseverity=<level>	Filters by StateSeverity, i.e. "Error", "Warning", "Information".
?statecode=<code>	Filters by StateCode.
?type=<devicetype>	Type filter - one or more (comma separated, no spaces) types of devices whose messages should be shown: - inverter - sensor - battery - smartmeter - ohmpilot - datalogger - evcharger

Example calls

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

Response objects

JSON object answer construction:

Type	Objects
n/a	- count (Number)

Example responses

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

5.8.3 Messages: Get device system messages

Use cases for web developers

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

Methods

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

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

Filters and parameters

Filter	Description
?timezone=<"local", "zulu">	Specifies time format in response object: - zulu (default): returns time in UTC zulu time

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

Example calls

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

Response objects

JSON object answer construction:

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

Example responses

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

5.8.4 Messages: Count device system messages

Use cases for web developers

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

Methods

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

Filters and parameters

Filter	Description
?from=<start>&to=<end>	Limits the time series for the query. - <start> and <end> are ISO-8601 time values.

Filter	Description
	/ If "to" is missing, then "to" is considered "now". ("from" must not be empty.) / Time format encoding: "yyyyMMddTHHmmssTZD", "yyyy-MM-ddTHH:mm:ssTZD".
?statetype=<type>	Filters by StateType, e.g. "Error", "Event".
?stateseverity=<level>	Filters by StateSeverity, i.e. "Error", "Warning", "Information".
?statecode=<code>	Filters by StateCode.

Example calls

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

Response objects

JSON object answer construction:

Type	Objects
n/a	/ count (Number)

Example responses

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

6 APPENDIX

6.1 Response and error codes

6.1.1 HTML error codes

Used HTML response codes by SWQAPI:

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

6.1.2 Detailed error codes

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

Error code examples

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

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

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

List of detailed error codes

Fronius Response Code	Verbose Description	Comment
GENERAL (10xx)		
1001	Error while processing request.	General error for internal server.
1002	Requested resource not found.	
1003	No input set.	
1004	Input invalid.	The reason for the error is usually given.
1005	Invalid date and time format.	
1006	Invalid date format.	
1007	Invalid timezone parameter.	
1008	Invalid channels	List of invalid channels is appended at the end.
1009	Invalid language code	
1010	From date is after to date.	

Fronius Response Code	Verbose Description	Comment
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.
AUTHENTICATION (11xx)		
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.	
1114	Missing parameters: UserId, Password	Missing parameters could be UserId or Password
1115	Empty parameters: UserId, Password	Empty parameters could be UserId or Password

Fronius Response Code	Verbose Description	Comment
1116	Invalid scope. Token likely expired.	
1120	Refresh token invalid.	
1121	Refresh token expired.	
Metadata calls (30xx)		
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.
Power flow data (flowdata) calls (31xx)		
31xx		
Aggregation data (aggdata, aggrdata) calls (32xx)		
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.	
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.
Historical data (histdata) calls (33xx)		

Fronius Response Code	Verbose Description	Comment
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.
Messages calls (34xx)		
3401	Invalid state type.	
3402	Invalid state code.	
3403	Date range between from and to filters too big.	
3404	From date is required.	
Weather calls (35xx)		
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.)

6.2 Channels

6.2.1 Channel list

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
ApparentPower	Apparent power	inverter	VA	Apparent power (S)			x	
BattMode		battery		Battery operating state. State values are: / 0 = Disabled / 1 = Normal / 2 = ServiceMode / 3 = ChargeBoost / 4 = NearlyDepleted / 5 = SuspendedOnPurpose / 6 = Calibrate / 7 = GridSupport / 8 = DepletedRecovery / 9 = NonOperableTemperature / 10 = NonOperableVoltage / 11 = Preheating / 12 = Startup / 13 = AwakeButNonOperableTemperature / 14 = BatteryFull / 90 = ForcedStandby	x		x	
BattSOC	State of charge	battery, general	%	Battery state of charge	x		x	
CloudCover			%	Cloud coverage				x

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
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	
EVCMode		evcharger		Active charging mode of Wattpilot (refer to Wattpilot documentation for details): / EcoMode / NextTripMode / StandardMode / NoCar	x			

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
				- NotCharging - WaitingForPrice - FullyCharged - ConnectionLost - PvConnectionLost				
Earnings	Earning	general	<currency>	Energy fed into grid multipled by a tariff.		x		
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 Wattpilot			x	
EnergyConsumptionTotal	Consumption	general	Wh	Calculated consumed energy (EnergyPurchased + EnergySelfConsumption + EnergyBattDischarge)		x	x	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
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 + Wattpilot)		x		
EnergyEVCCharge		general	Wh	Energy flowing from generators to Wattpilot		x	x	
EnergyEVCChargeBatt		general	Wh	Energy flowing from battery to Wattpilot		x	x	
EnergyEVCChargeGrid		general	Wh	Energy flowing from grid to Wattpilot		x	x	
EnergyEVCChargeTotal		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	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weather
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	
EnergyOutput		general	Wh	Calculated total energy produced (NULL if smart meter is connected)		x	x	
EnergyProductionTotal	Production	general	Wh	Calculated energy flowing from generators to consumer, battery and grid (EnergySelfConsumption + EnergyBattCharge + EnergyFeedIn + EnergyOutput)		x	x	
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	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
EnergySelfConsumptionTotal	Own consumption	general	Wh	Calculated energy flowing from generators to consumers and battery (EnergySelfConsumption + EnergyBattCharge)		x	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	
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	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
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	
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^2	Insolation			x	
IsOnline		general	-	Online status of the PV system	x			

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
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	
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	

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
Power		general	W	Power measured by a submeter (requires a Smart Meter)	x			
PowerBattCharge		general	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			
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			

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
PowerOutput		general	W	Power generated by inverter (NULL if Smart Meter is connected)	x			
PowerPV		general	W	Power flowing from generators to the inverter	x			
PowerPurchase		general	W	Power flowing in from grid (requires a Smart Meter)	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	agg data, aggr data	hist data	weather
SavingsCO2	CO2 savings	general	kg	Equivalent of saved CO_2 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
Temp1, Temp2	Module temperature	sensor	°C	Temperature			x	
Temp (min, max)			°C	Temperature forecast				x

Channel	Channel name in Solar.web UI	Device type	Unit	Description	flow data	agg data, aggr data	hist data	weat her
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	
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

6.2.2 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^2
Percentage	%
Power	W
Precipitation (rain)	mm
Reactive Power	VAr
Symbol	Symbol
Temperature	°C
Time	Time
Velocity (wind speed)	m/s
Voltage	V

6.3 Meteorological weather symbols

6.3.1 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	🌦

Symbol description	Icon (example)
overcast with scattered rain showers	🌧
sunny with heavy rain showers	🌦
cloudy with heavy rain showers	🌦
overcast with heavy rain showers	🌧
sunny with thunderstorms	⛈
cloudy with thunderstorms	⛈
overcast with thunderstorms	🌩
sunny with strong thunderstorms	⛈
cloudy with strong thunderstorms	⛈
overcast with strong thunderstorms	🌩
light snowfall	🌨
snowfall	🌨
heavy snow	🌨
sunny with light snow showers	🌨
cloudy with light snow showers	🌨
overcast with light snow showers	🌨

Symbol description	Icon (example)
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	🌨
duststorm / sandstorm	🌬
drifting snow	🌬

Symbol description	Icon (example)
graupel	⚠🌨
fog patches	🌫
low clouds, sun	🌫☀
freezing fog	⚠🌫
sun and high clouds	🌤

6.4 Languages

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

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

6.5 Best practices and how-tos

6.5.1 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.

6.5.2 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
                },
...
```

6.5.3 Determine PV Energy and Load Energy

PV Energy = EnergySelfConsumption + EnergyFeedIn + EnergyBattCharge

Load Energy = EnergySelfConsumption + EnergyPurchased + EnergyBattDischarged

6.5.4 Determine if new systems were added to account

Method 1 - only systems are added:

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

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

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

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

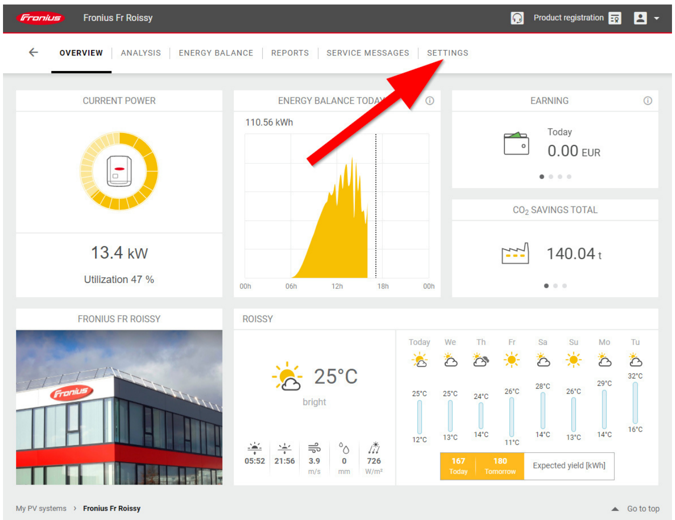
6.5.5 Grant permissions in Solar.web

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

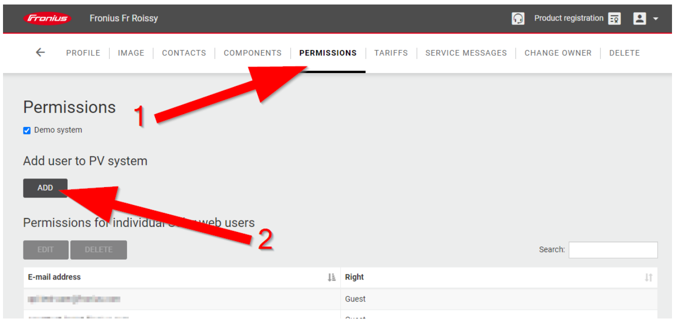
> ⓘ The API user has to take care of getting the permission from the owner (or a supervisor) of a PV system. Fronius does not grant anybody permission to a PV system that is not linked to the account (by ownership or permission).

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

2. Go to SETTINGS

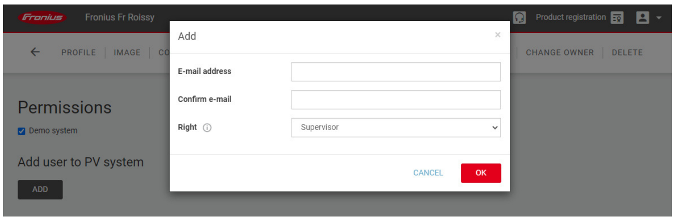


3. Go to PERMISSIONS (1) and click on ADD (2)



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



Add

E-mail address

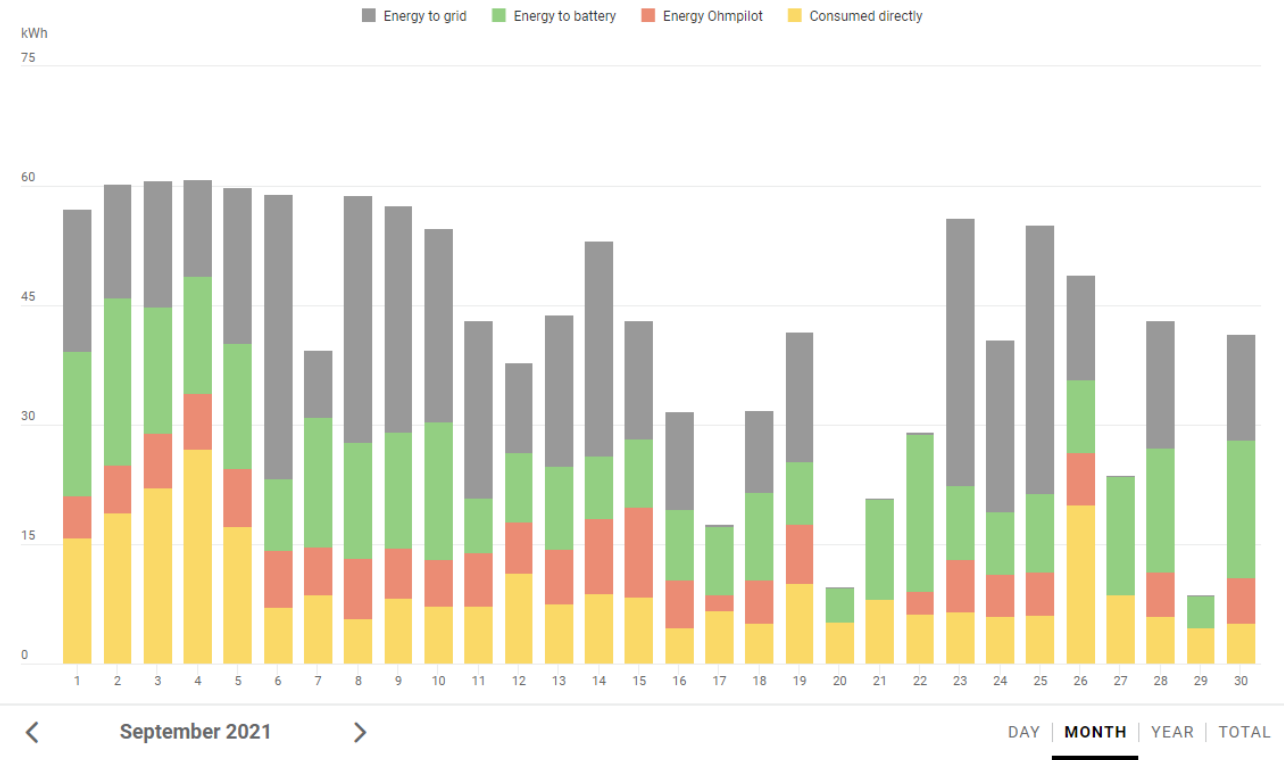
Confirm e-mail

Right: Supervisor

7 CHAPTER AGGREGATION

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:

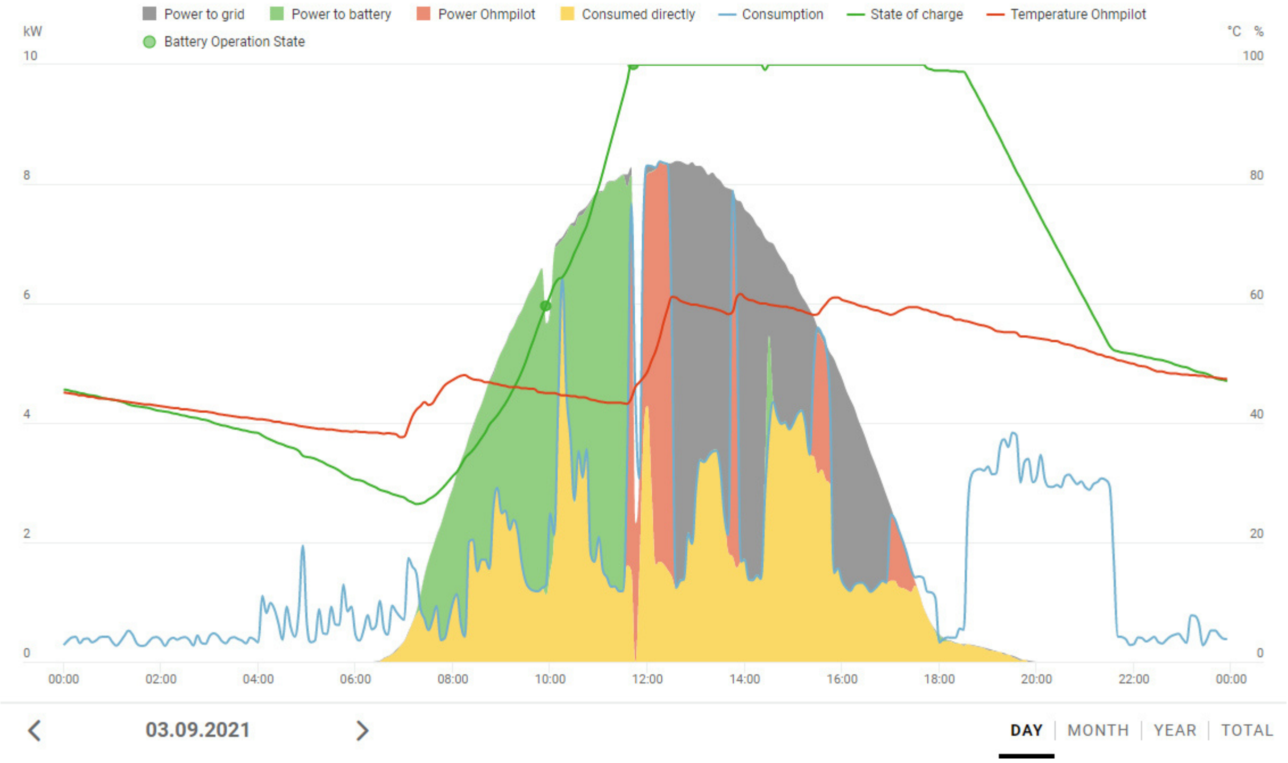


8 CHAPTER HISTORICAL

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:



9 CHAPTER INTRODUCTION

9.1 Introduction

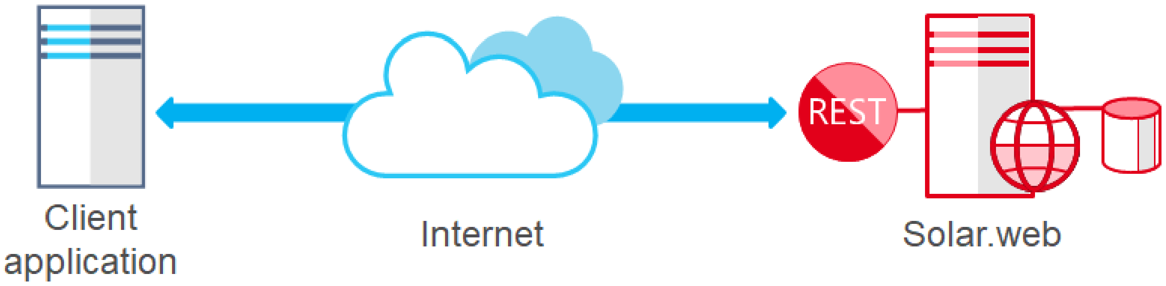
9.1.1 About Fronius Solar.web Query API

Fronius Solar.web

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

Fronius Solar.web Query API

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



9.1.2 Usage of the Fronius Solar.web Query API

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.

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.

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.

10 CHAPTER METADATA

This set contains methods to retrieve

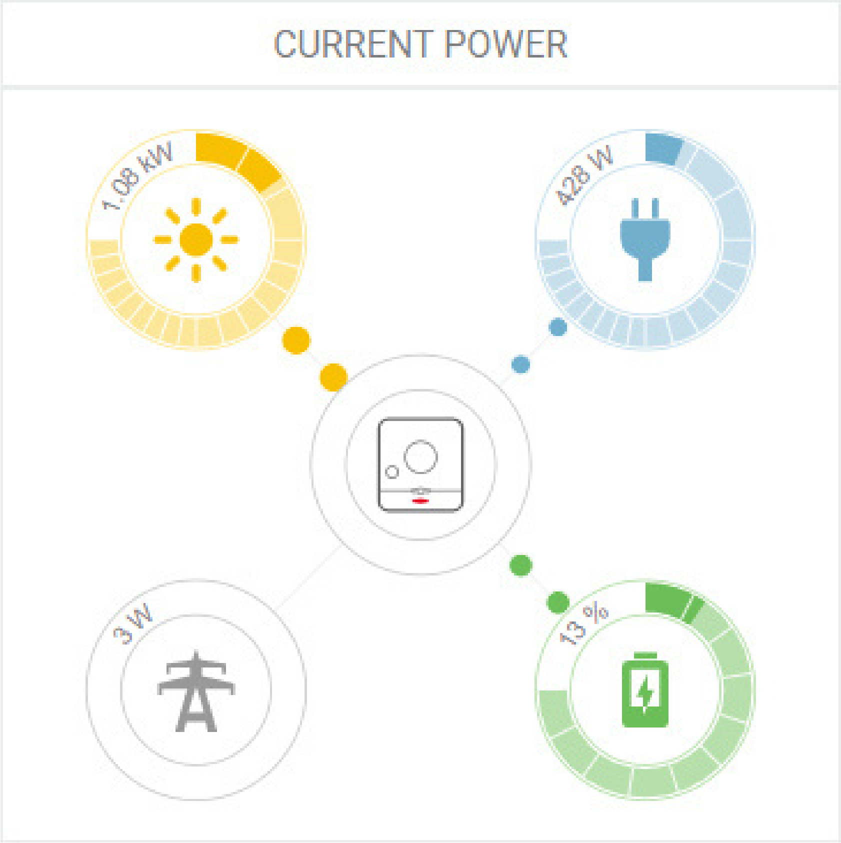
- the number of PV systems linked to the user's account,
- a list of all PV system IDs,
- detailed information about the PV systems,
- the number of devices of a given PV system,
- a list of all devices of a given PV system and
- detailed information about the devices of a PV system

11 CHAPTER POWERFLOW

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.

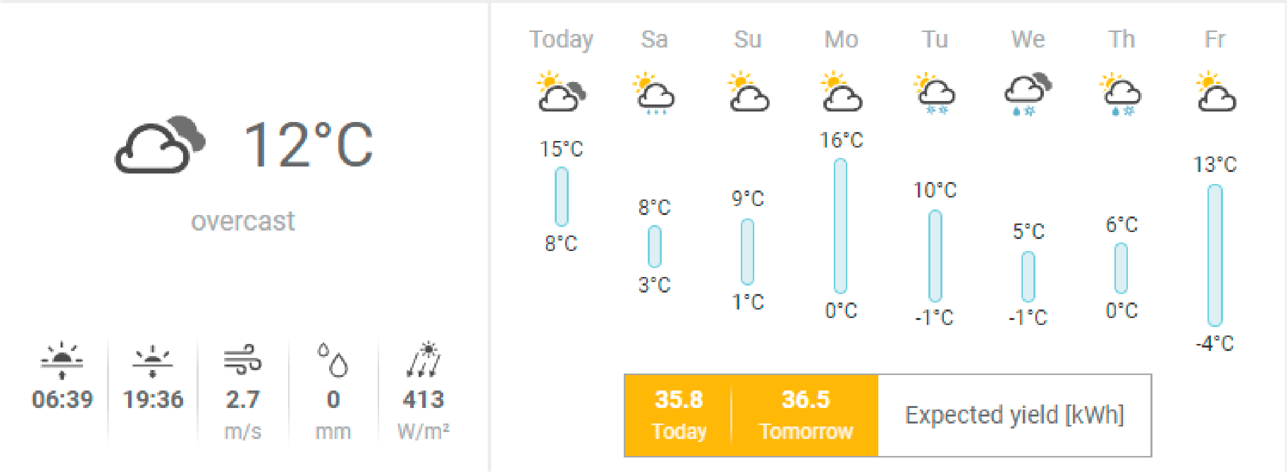
Power flows can be shown like in this diagram:



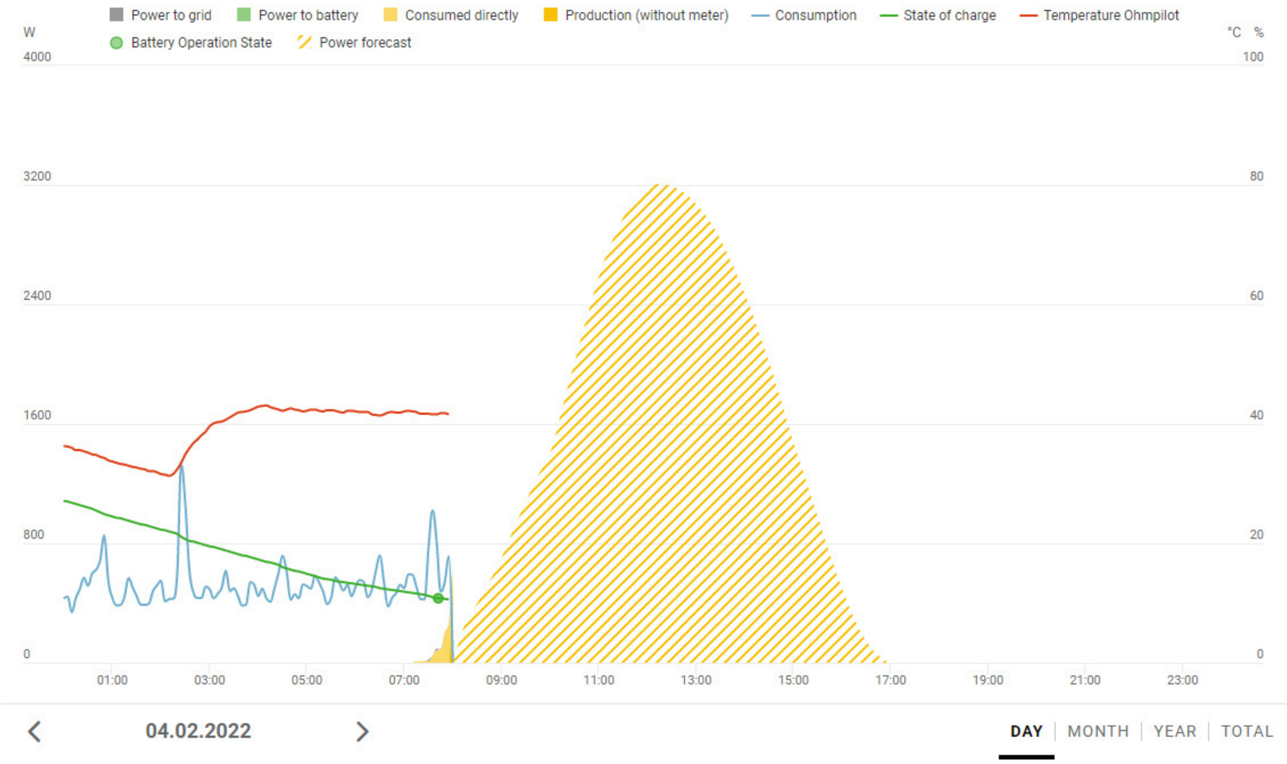
12 CHAPTER WEATHER

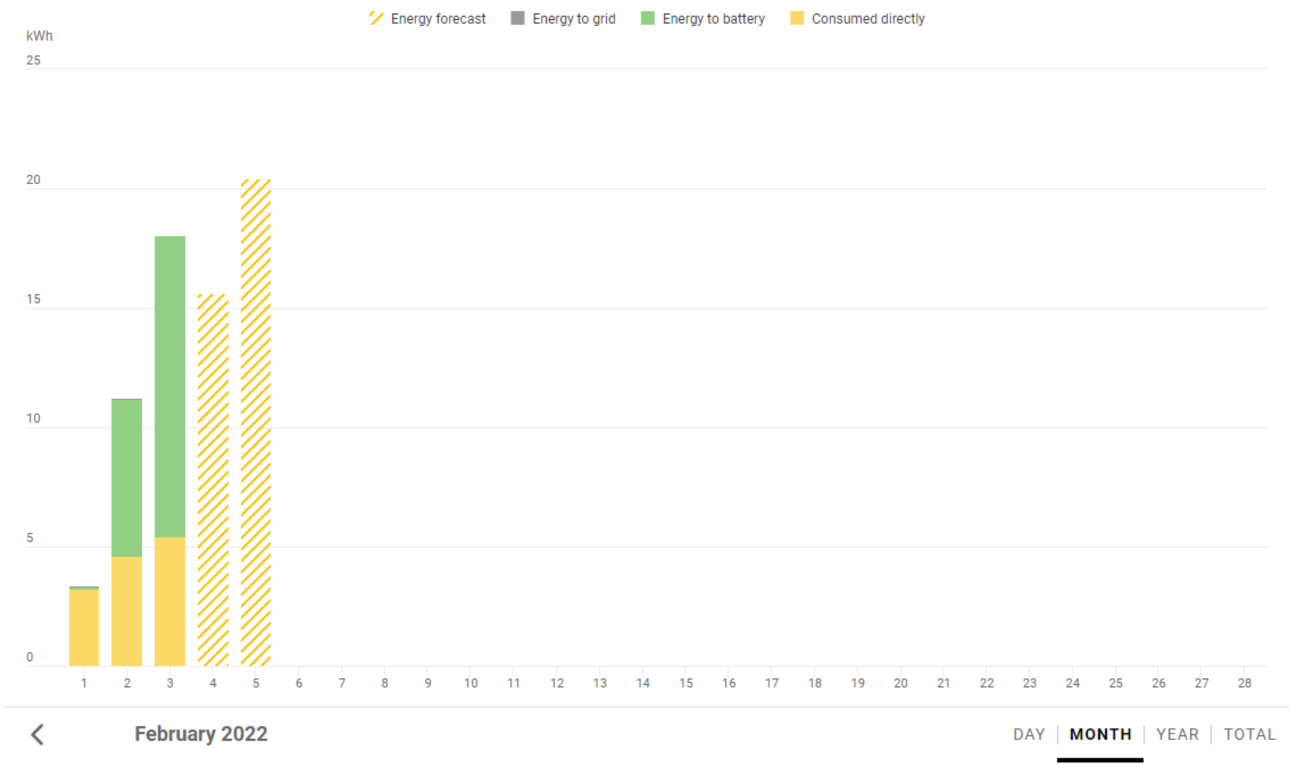
This set provides weather and energy forecast information for PV systems.

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):





12.1 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).

13 SWQAPI MANUAL VERSIONING

WORKING INSTRUCTIONS

Every time when you change something in the documentation, please document your changes in the "Description" column. You do not need to update the version information or modified date (this will be done when the documentation is exported). Example:

Version	Modified	Description	Modified by / Comments
		/ Updated camelCase notation in examples for system messages.	Thomas Hüttner

WORKING INSTRUCTIONS

Every time when you create a final version for export, you have to define a modification date and a version number. Example:

Version	Modified	Description	Modified by / Comments
47.0	Apr 21, 2021	/ Updated camelCase notation in examples for system messages.	Thomas Hüttner

Version	Modified	Description	Modified by / Comments
50.0	Aug 01, 2023	/ Added channels for Submeters. / Corrected "PowerPurchased" channel to "PowerPurchase". / Updated EnergyExported and EnergyImported channels for secondary meters / Pilot example added / Updated data plan details	TH, PW
49.0	Oct 12, 2022	/ Updated sections Metadata calls, Aggregation calls, Historical data calls, Realtime data calls, System messages calls: / Added Wattpilot data. / Updated some example responses. / Minor corrections in section Weather data calls. / Updated General information section. / Updated Supporting UIs section. / Updated channel list: / Added Wattpilot data / Removed reference to outdated Aggregation calls	PW

Version	Modified	Description	Modified by / Comments
48.0	Dec 09, 2021	/ Added notes to GenerateJwt call. / Added status code information for maintenance windows. / Updated device metadata: / Provided lists for Smart Meter locations and categories. / Provided list for sensor types. / Updated device metadata: / Added missing datalogger IDs. / Improved examples, e.g. a GEN24 PV system. / Added revoke JWT call. / Updated metadata calls with meteo filter. / Clarifications in error tables. / Removed "PowerBattDischarge" channel from channel list, and updated description of "PowerBattCharge". / Clarified positive/negative values for power flow data. / Clarified 3301 error code for historical data. / Added datalogger and Ohmpilot examples to device metadata. / Removed Fronius SSO login method.	TH
47.0	Apr 21, 2021	/ Updated camelCase notation in examples for system messages.	TH
46.0	Apr 02, 2021	/ Changed supported channel information for aggregated and historical data requests. / Updated Solar.web screenshots, error code lists and channel list.	PW
45.0	Feb 02, 2021	/ Version history added. / Added additional information about Solar.web Premium to chapter "User impersonation". / Updated chapter "Determine power values from energy values" in Appendix.	PW