TECHNICAL DOCUMENT

Wiotys – SoData API usage example

TRN 23 127373

Table of content

1 Prerequisite

This document is intended for developers who want to use the Wiotys SoData API in their application.

The SoData API provides access to the entire SoLive Pro objects (like SoLive Pro Administration do), as well as to the devices data.

SoData can be used with any programming language that allows Rest APIs calls. For the examples in this document, we will use the Python language.

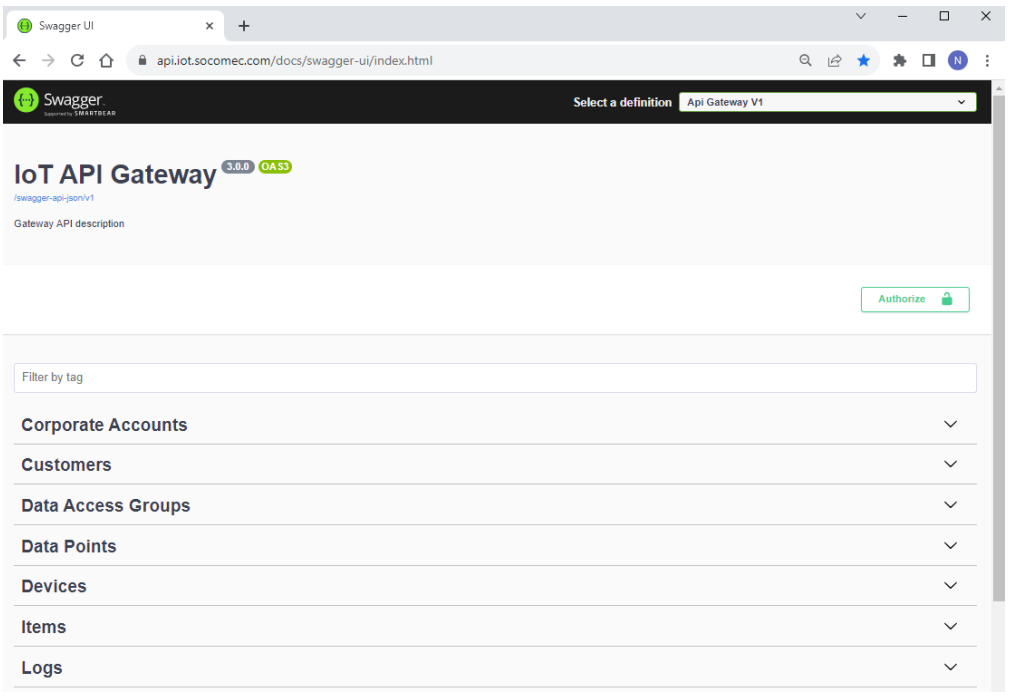
To get the most from this document, the reader should:

- Know what a Rest API is and have basic knowledge on OpenAPI (Swagger).
- Know the architecture of a Wiotys application (administration and SoLive Pro).
    - Cf. TRN 23 127385 - Wiotys - Administration - UPS.pdf
    - Cf. TRN 23 127386 - Wiotys - Administration - Hxx Dxx Mxx.pdf
    - Cf. TRN 23 127387 - Wiotys - Administration - LoRaWan.pdf
- Have notions in Python language for the examples.

2 SoData UI Swagger

The Swagger (Open API) interface can be used to test the API. Access to the SoData API can be done on Wiotys environments.

Swagger UI URL	https://api.iot.socomec.com/docs/swagger-ui/index.html



The SoData API provides access to the various objects of a Wiotys application, objects managed by the administration application, as well as to the data of the devices.

3 Wiotys application organization (reminder)

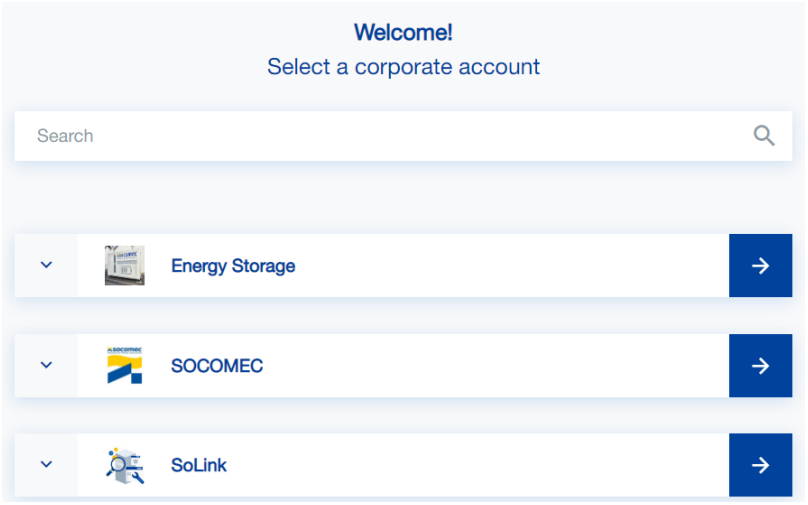
3.1 Application general organization

All Wiotys application start with a corporate account (ca).

Within this corporate account, we can manage application objects (customers, sites, devices…):

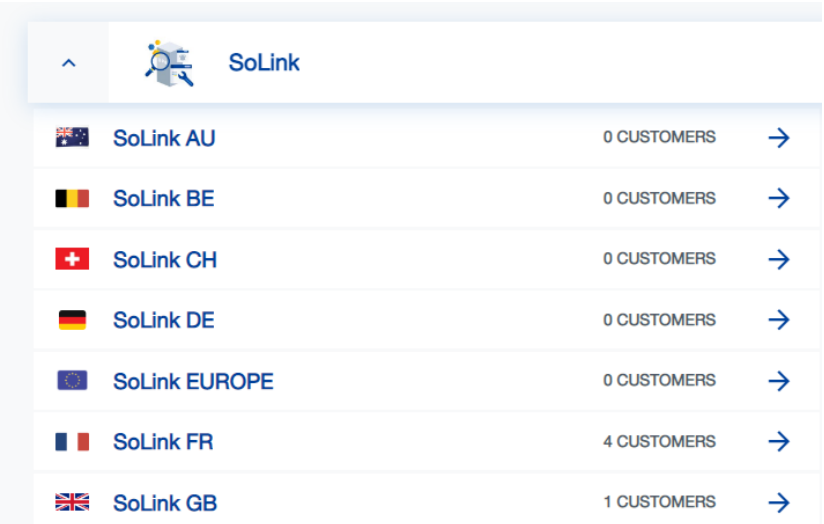
- ***The functional rights on these objects depend on the rights granted to the connected user.***
- ***Access rights to device's data are granted through data access groups (DAG).***

Example of corporate accounts (a user can have access to multiple corporate accounts):

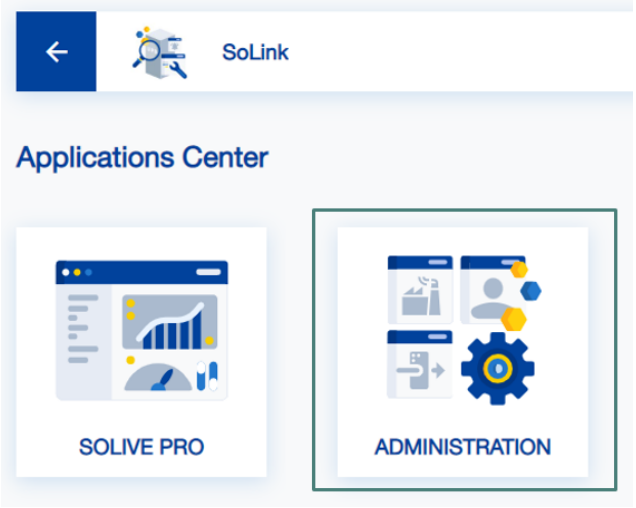


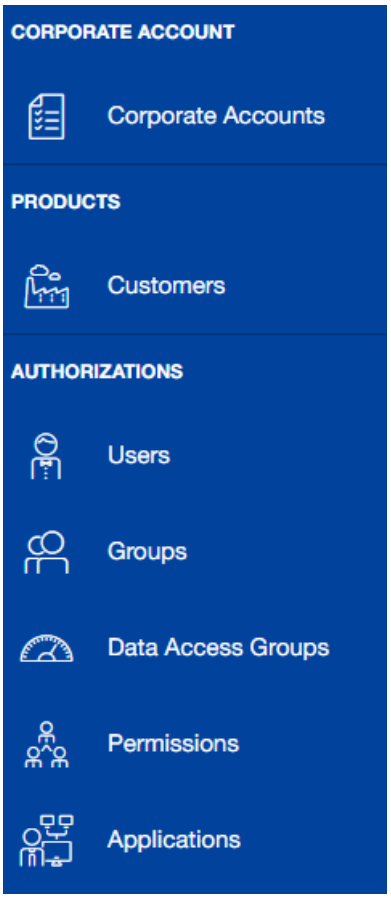
After logging in, a user has access to one or more corporate accounts. Within each corporate account, through the DAGs, he has access to different customers and their devices.

DAG (group of customers and devices) examples for the SoLink corporate account:



If the user connected has administrations rights, he can have access to all of these objects (screenshot of the Wiotys administration menu):





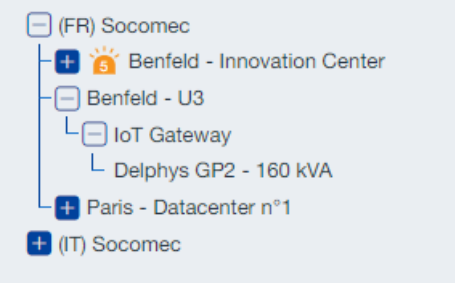
Wiotys Organization	Description	SoData API (group)
Corporate accounts	List and properties of Corporate accounts.	Corporates Account
Customers	List and properties of Customers.	Customers
Data Access Groups	List and properties of Data Access Groups (DAG). A DAG belongs to a corporate account.	Data Access Groups

3.2 Products hierarchy

Products hierarchy is composed by:

- Customer(s)
  - Site(s)
    - Device(s)

Example:



- (FR) Socomec is a Customer
  - Benfeld – U3 is a site of *(FR) Socomec*
    - IoT Gateway is a device of *Benfeld – U3* (a particular device because this is an IoT Gateway)
      - Delphys GP2 – 160 kVA is a device (an UPS) linked to the *IoT Gateway*

Note: in most case, a device (an UPS, an Energy meter, an Energy Stocker,…) is linked to an IoT Gateway (which is also a device), but it is not mandatory.

Wiotys Products hierarchy	Description	SoData API (group)
Customers	List and properties of Customers. A customer belongs to a corporate account.	Customers
Sites	List and properties of Sites. A site belongs to a customer.	Sites
Devices	List and properties of Devices. A device belongs to a Site.	Devices

3.3 Device structure

A Socomec device has a specific structure:

- Device
  - Item(s) (with indexes from 0 to ***index***-1, if we have multiple identical Items)
    - Variable(s) (link to a ***service***)

Example:

The device “Delphys GP2 – 160 kVA” (an UPS) has two different items:

- Global (Item *Global* with index *0*)
  - The *Global* item has some variables (classified by their type: Alarm, Measure,…):
    - For example: *Ups output global load rate*, link to the service *IUpsOutput_Lr*

- Unit 1 (Item *Unit* with index *0*, if we had a second unit, it would still have as item *Unit*, but as index the value *1*)

A ***service*** is used to categorize a variable, with a generic description (Positive active power for example), and a unit (like W, A, V,…). A service has a name (IUPSOutput_Lr), and a number (140100 for IUPSOutput_Lr, called serviceId).

Wiotys Devices	Description	SoData API (group)
Devices	List and properties of Devices. A device belongs to a customer site.	Devices
Items	List and properties of Items. An item is part of a device.	Items
Variables	List and properties of Variables. A variable belongs to an item.	Variables

3.3.1 Variables mapping config

The "mapping config" allows us to associate a "text" reference to a variable.

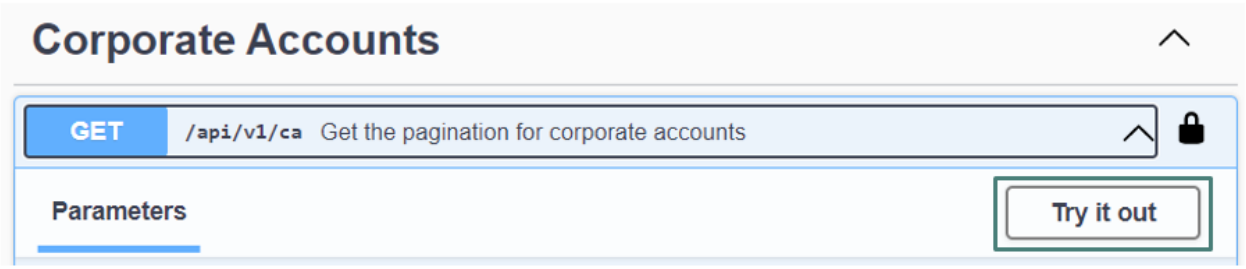
The mapping config syntax is:

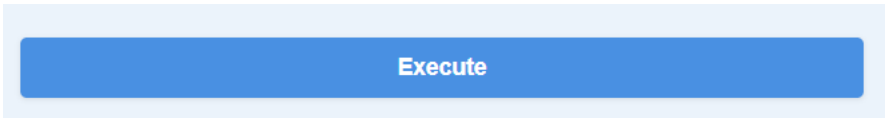
- **Item|Index|Service**
  - **Item**: the item name, like Global, Load, …
  - **Index**: the index of the item, starting from 0, if we have multiple same items.
  - **Service**: the service number (call also serviceId).

- Example:
  - **Global|0|140100** is the “Ups Output global load rate” for one UPS

4 Swagger (OpenAPI) tips

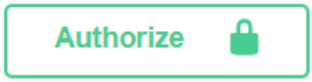
We can use Swagger to test the API, with the **Try it out** button, and **Execute** command:



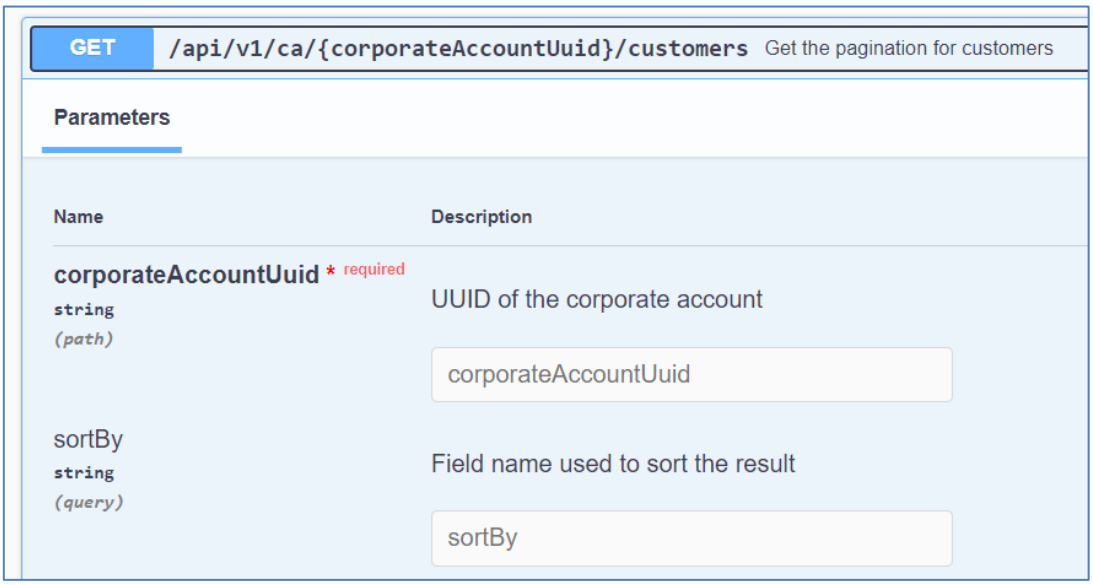


But before executing the request, we have to:

- Obtain the necessary authorizations to use the API (Authorize button)
  - We'll see it in details in the next chapters



- Fill in the parameters of each request:
  - Some of them are mandatory (* required), and others are optional (with or without default value)
    - In this example, **corporateAccountUuid** is required,
    - **sortBy** is optional

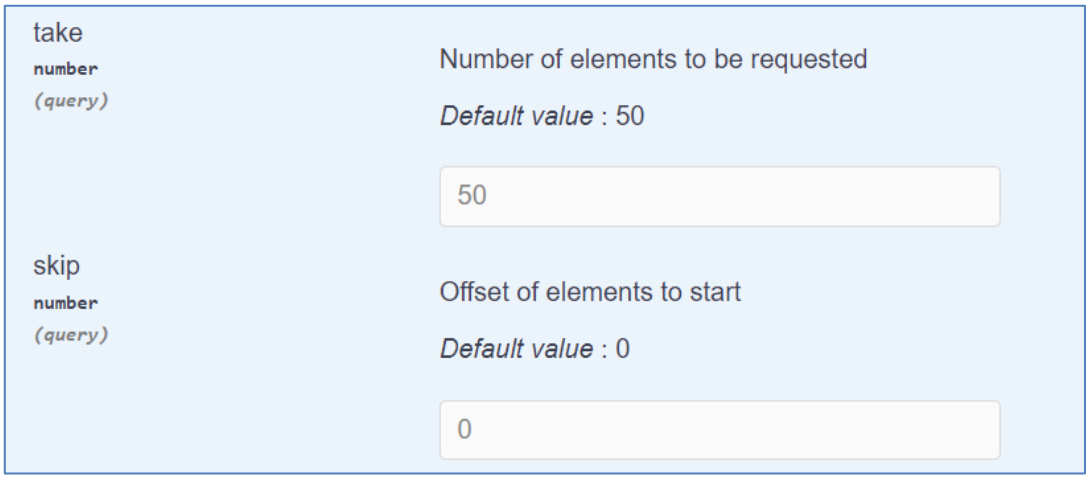


4.1 Request pagination

The majority of queries are paginated, i.e. they will only return a subset of the results, if the number of results exceeds the number of items requested.

Pagination is managed with the two parameters:

- take
- skip



You must manage this paging within your program.

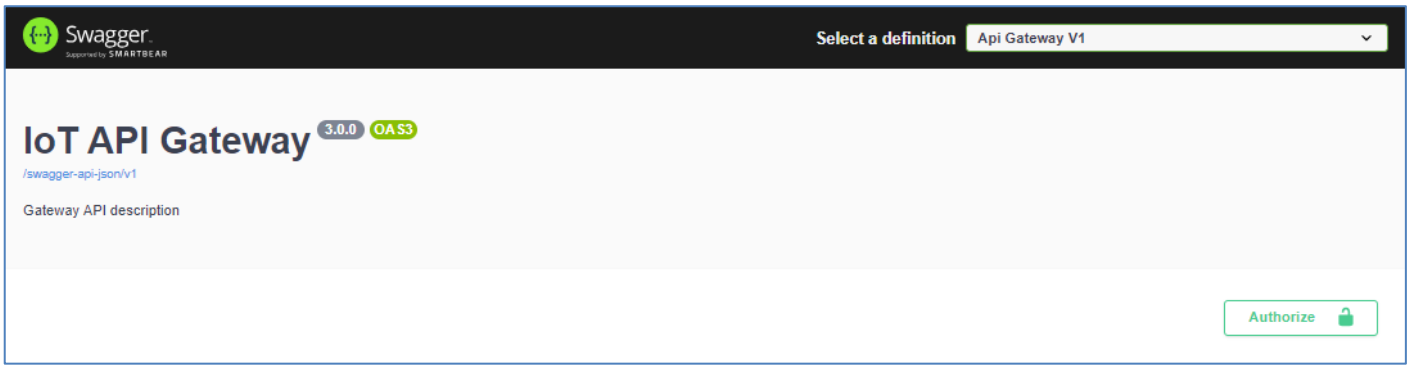
5 API description

5.1 Security

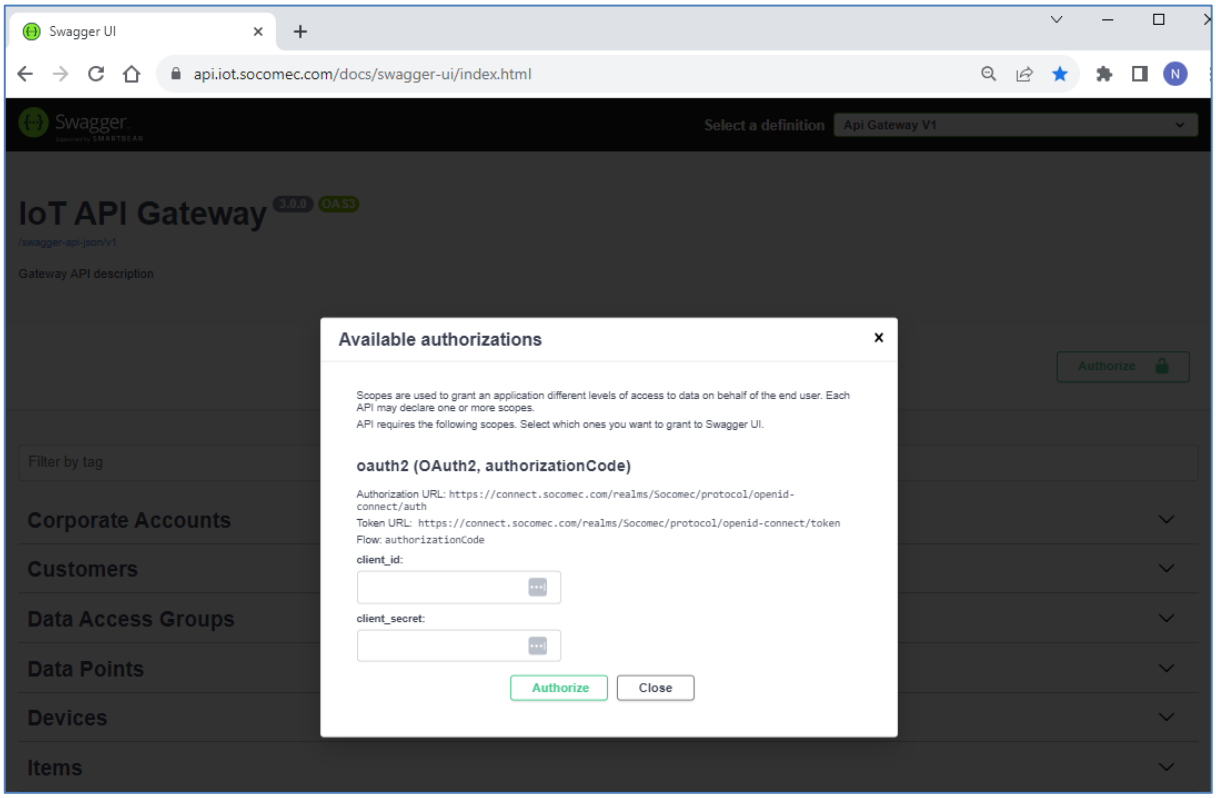
Access to the SoData API is secure, whether via Swagger, or via a programming language.

The API user must also identify himself, with his SoLive Pro login (email) and password, which makes it possible to determine his application rights and his access to devices data.

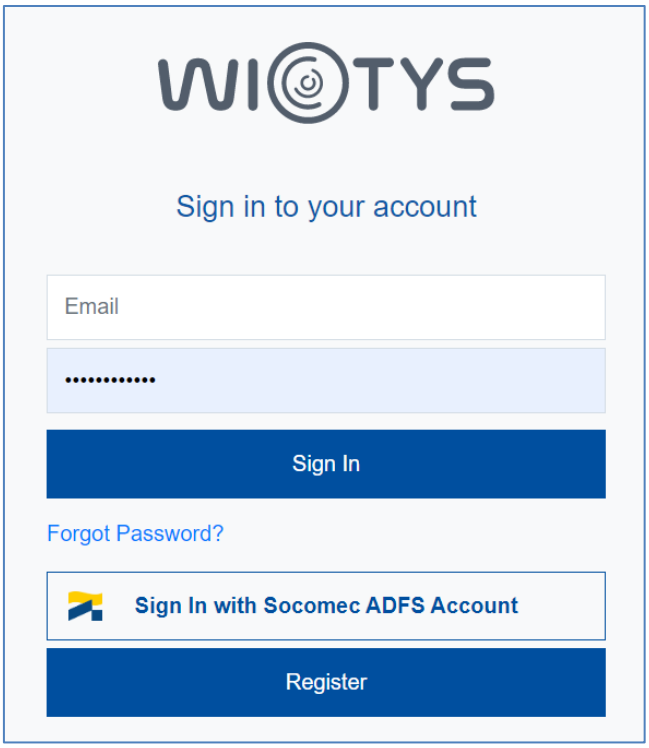
To activate authorizations from Swagger, click on the Authorize button:



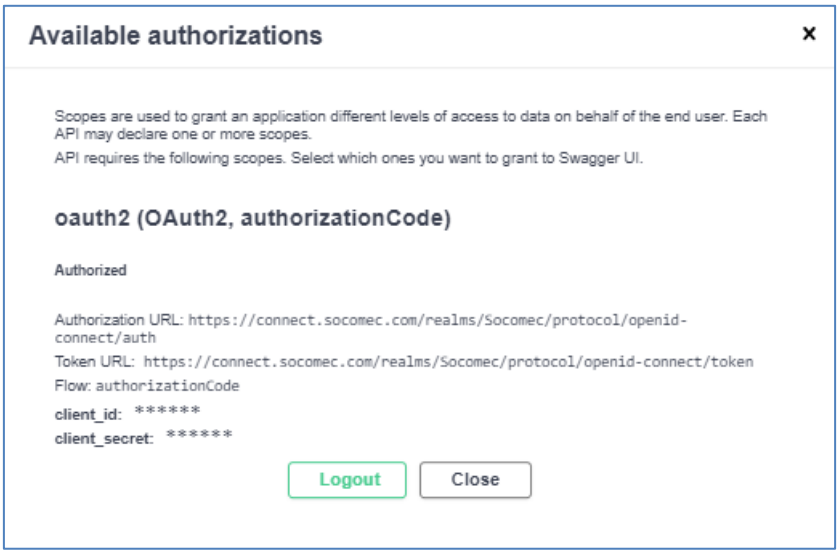
The following popup is displayed, you must click the Authorize button again to be redirected to the Wiotys login page:



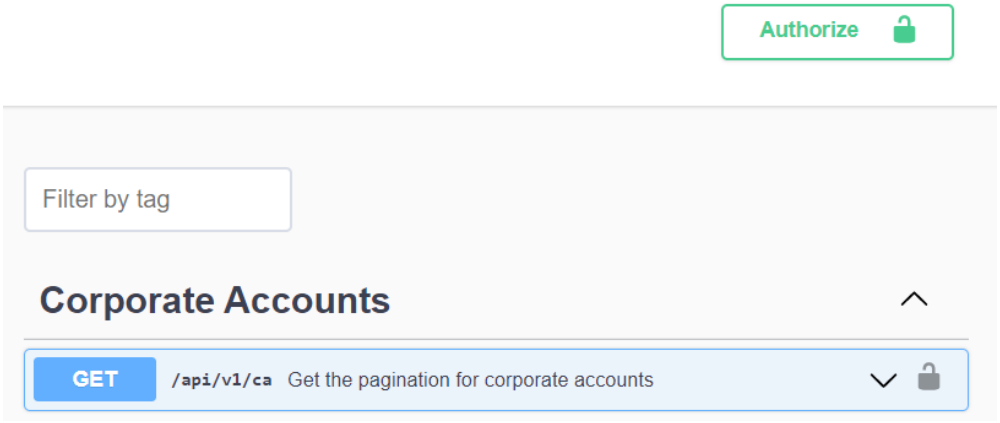
From the Wiotys login page, enter your SoLive Pro user login (Email) and password (or click on Sign in with Socomec ADF Account for Socomec users):



After Sign In, the next popup is displayed, click on **Close**:



We now have access to the API (padlocks are unlocked):



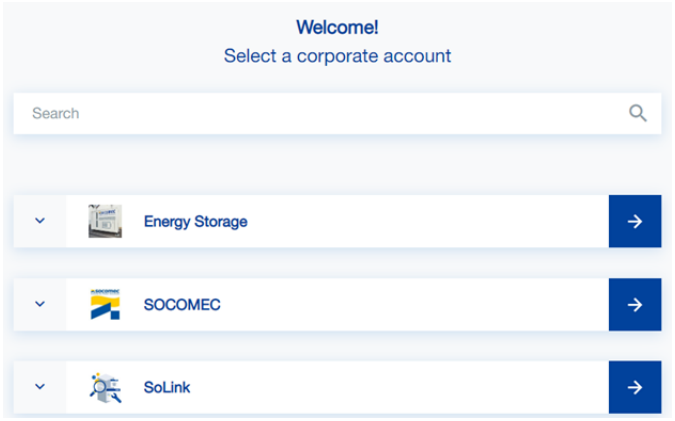
Authorizations are not permanent (in time), it is necessary to reconnect regularly.

5.2 Objects uuid

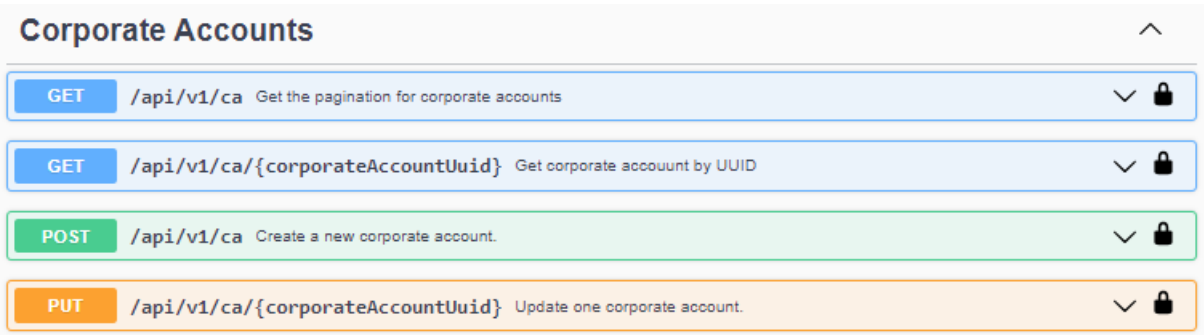
All objects accessible from the SoData API have a unique identifier (**uuid**). These identifiers are necessary to be able to make filtered requests: access to all the sites of a customer for example, by specifying the uuid of the customer.

5.3 Corporates Accounts (ca)

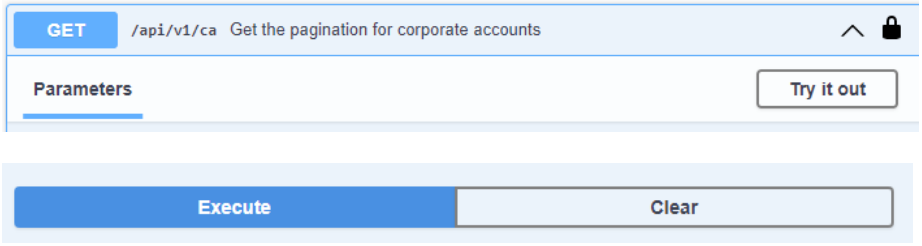
5.3.1 Wiotys corporate accounts example



5.3.2 SoData Corporate Accounts methods



5.3.3 Get all corporate accounts

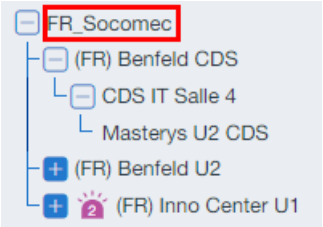
Action	Description
Prerequisite	Valid authentication (Authorize)
Swagger try out	No parameter is required, so you can click on **Try it out**, and then directly on **Execute**: 
Request URL	https://api.iot.socomec.com/api/v1/ca?take=50&skip=0

Response body	{
	"total": 1,
	"results": [
	{
	"defaultCustomerUuid": "",
	"**uuid**": "3756c84b-8697-446f-9ca3-931b3681f908",
	"name": "**SoLink**",
	"activationType": "MAINTENANCE",
	"emailTo": null,
	"emailCc": null,
	"canPublish": false,
	"autoTemplateSynchro": true,
	"isMultiCustomerAccount": true,
	"active": true,
	"movingAuthorization": "CONSERVATION",
	"iconUuid": "fd2f2e54-02fe-4ad2-9ccc-7a6c360dfd76",
	"userLinks": [
	{
	"uuid": "8092cecc-fdba-4cce-9637-c0ab62dd620e",
	"status": "VALIDATED",
	"isSupport": false,
	"corporateAccountId": 3,
	"userId": 4
	}
	],
	"isMaster": false
	}
	],
	"take": 50
	}

uuid	Memorize the corporate account **uuid** for next requests, for the **SoLink** ca: 3756c84b-8697-446f-9ca3-931b3681f908

5.4 Customers

5.4.1 Wiotys customers example

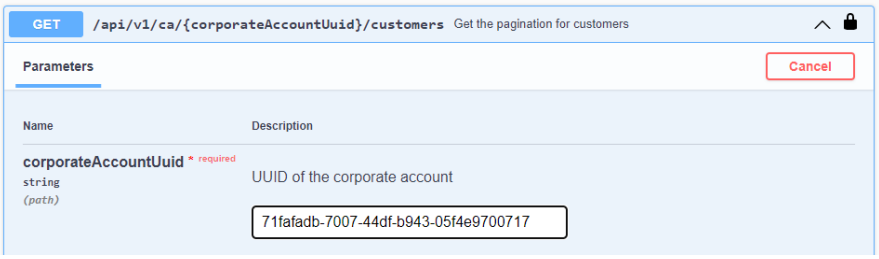


5.4.2 SoData Customers methods

Customers

DELETE	/api/v1/ca/{corporateAccountUuid}/customers/{customerUuid}	Delete a customer
GET	/api/v1/ca/{corporateAccountUuid}/customers	Get the pagination for customers
GET	/api/v1/ca/{corporateAccountUuid}/customers/{customerUuid}	Get customer by UUID
GET	/api/v1/ca/{corporateAccountUuid}/customers/{customerUuid}/icon	Get customer icon image
POST	/api/v1/ca/{corporateAccountUuid}/customers	Create a new customer
PUT	/api/v1/ca/{corporateAccountUuid}/customers/{customerUuid}	Update a customer
PUT	/api/v1/ca/{corporateAccountUuid}/customers/{customerUuid}/icon	Update a customer icon

5.4.3 Get all customers for the SoLink ca

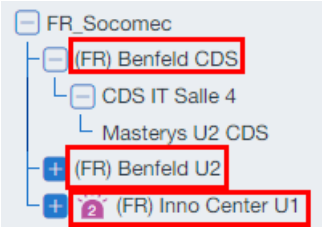
Action	Description
Prerequisite	Valid authentication (Authorize) Uuid of the SoLink ca: 3756c84b-8697-446f-9ca3-931b3681f908
Swagger try out	Enter the SoLink ca uuid: 
Request URL	https://api.iot.socomec.com/api/v1/ca/3756c84b-8697-446f-9ca3-931b3681f908/customers?take=50&skip=0&detailsLevel=ALL

Response body	{
	"total": 1,
	"results": [
	{
	"**uuid**": "751f149b-0fca-4f39-b0c7-6f9921852dc9",
	"emailTo": "frederic.morice@socomec.com",
	"emailCc": "nicolas.vogt@socomec.com",
	"technicalName": "Socomec",
	"name": "FR_Socomec",
	"reference": null,
	"description": "All Socomec UPS in production",
	"visibility": null,
	"corporateAccountId": 3,
	"additionalInformation": "",
	"movingAuthorization": "INHERITED",
	"corporateAccountUuid": "3756c84b-8697-446f-9ca3-931b3681f908",
	"iconUuid": "716bf4fb-e2c6-4e4c-987d-b1a42b91c78a",
	"companyAddress": {
	"uuid": "11b0243e-edae-4aa8-a047-106ff7bf3f46",
	"useParent": false,
	"street": "1 Rue de Westhouse",
	"postCode": "67235",
	"city": "Benfeld",
	"country": "France",
	"state": ""
	},
	"contact": {
	"uuid": "497ee681-6699-4fb5-83c2-9ab19aab8743",
	"useParent": false,
	"name": "Frédéric MORICE",
	"phone": "",
	"mobile": "",
	"email": "frederic.morice@socomec.com",
	"emailCC": ""
	},
	"sitesQuantity": 3
	}],
	"take": 50
	}

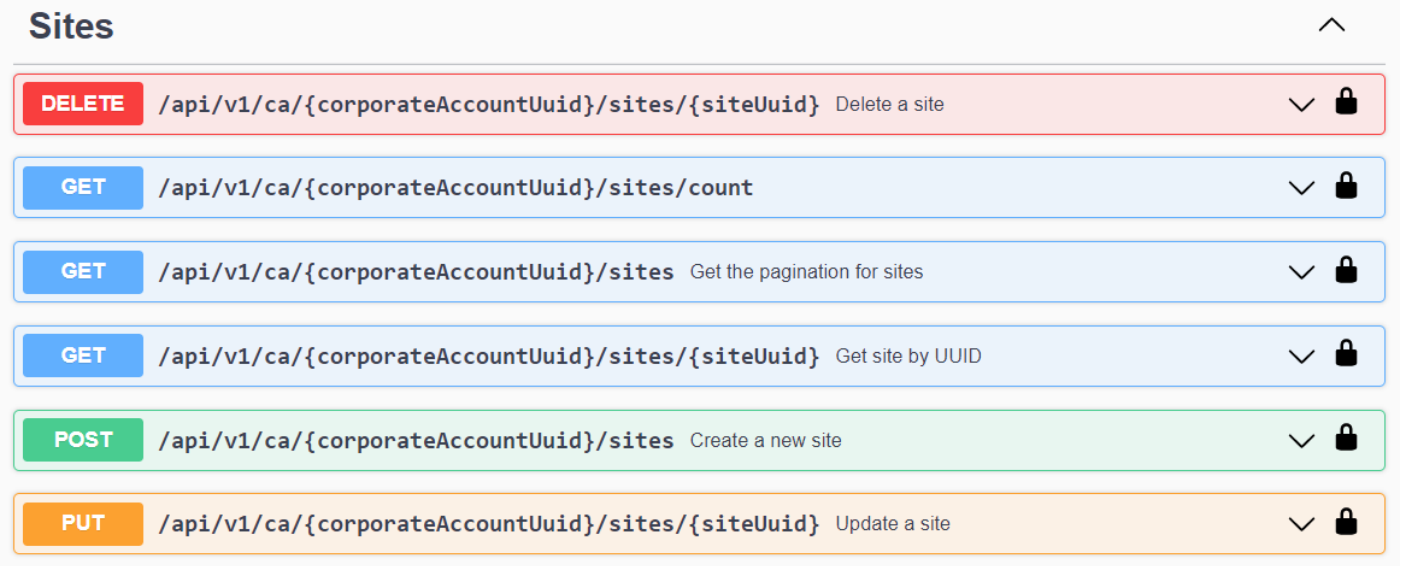
uuid	Memorize the customer **uuid** for next requests, for **FR_Socomec**: 751f149b-0fca-4f39-b0c7-6f9921852dc9

5.5 Sites

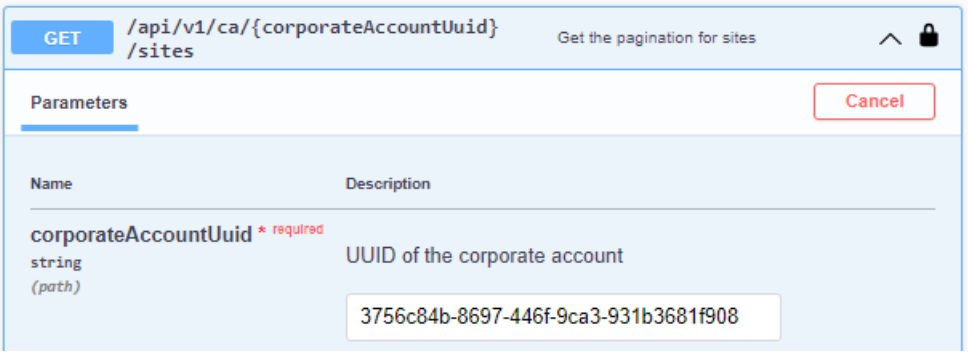
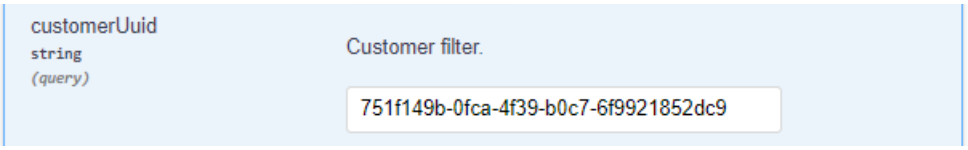
5.5.1 Wiotys sites example



5.5.2 SoData Sites methods



5.5.3 Get all site for the customer FR_Socomec

Action	Description
Prerequisite	Valid authentication (Authorize) Uuid of the **SoLink** ca: 3756c84b-8697-446f-9ca3-931b3681f908 Uuid of the customer **FR_Socomec:** 751f149b-0fca-4f39-b0c7-6f9921852dc9
Swagger try out	Enter the **SoLink** ca uuid:  Enter the customer **(FR) Socomec** uuid: 
Request URL	https://api.iot.socomec.com/api/v1/ca/3756c84b-8697-446f-9ca3-931b3681f908/sites?take=50&skip=0&customerUuid=751f149b-0fca-4f39-b0c7-6f9921852dc9

Response body (partially)	{
	"total": 3,
	"results": [
	{
	"**uuid**": "3fd80117-0674-4f30-bbff-9ef7f467a450",
	"emailTo": null,
	"emailCc": null,
	"technicalName": "FR_Benfeld_CDS",
	"language": null,
	"name": "**(FR) Benfeld CDS**",
	"description": "",
	"configurationOptions": {
	...
	},
	"activationKey": "03a9c0a4-cea0-411d-95e1-de1f76849bea",
	"visibility": null,
	"customerId": 7,
	"autoGatewaysRegistration": true,
	"autoProductsRegistration": true,
	"timeZone": "Europe/Paris",
	"additionalInformation": "",
	"movingAuthorization": "INHERITED",
	"customerUuid": "751f149b-0fca-4f39-b0c7-6f9921852dc9",
	"devicesQuantity": 2
	},...
	],
	"take": 50
	}

uuid	Memorize the site **uuid** for next requests, for **(FR) Benfeld CDS**: 3fd80117-0674-4f30-bbff-9ef7f467a450

5.6 Devices

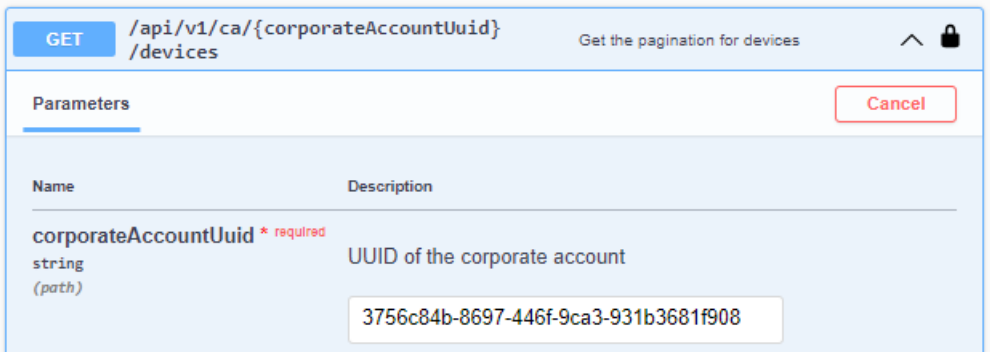
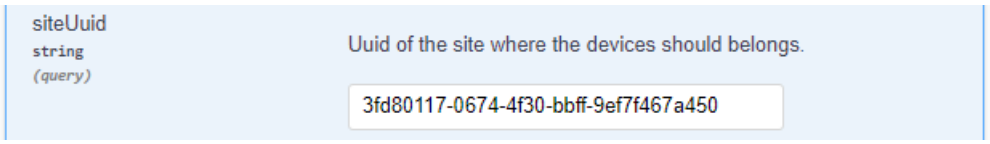
5.6.1 Wiotys devices example

5.6.2 SoData Devices methods

Devices

DELETE	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}	Delete a device
DELETE	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/share/{siteUuid}	Cancel sharing
GET	/api/v1/ca/{corporateAccountUuid}/devices	Get the pagination for devices
GET	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/gateway	Get parent gateway from device UUID
GET	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}	Get device by UUID
GET	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/icon	Get device icon image
POST	/api/v1/ca/{corporateAccountUuid}/devices	Create a new device
POST	/api/v1/ca/{corporateAccountUuid}/devices/pre-create	Pre-create a new device
POST	/api/v1/ca/{corporateAccountUuid}/devices/link	Link a new device
PUT	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}	Update a device
PUT	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/icon	Update a device icon
PUT	/api/v1/ca/{corporateAccountUuid}/devices/repair-devices	Repair all devices
PUT	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/activate	Activate a device
PUT	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/deactivate	Deactivate a device
PUT	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/push-profile	Push the profile
PUT	/api/v1/ca/{corporateAccountUuid}/devices/{deviceUuid}/repair	Repair a device

5.6.3 Get all devices for the site (FR) Benfeld CDS

Action	Description
Prerequisite	Valid authentication (Authorize) Uuid of the **SoLink** ca: 3756c84b-8697-446f-9ca3-931b3681f908 Uuid of the site **(FR) Benfeld CDS:** 3fd80117-0674-4f30-bbff-9ef7f467a450
Swagger try out	Enter the **SoLink** ca uuid:  Enter the site **(FR) Benfeld CDS** uuid: 
Request URL	https://api.iot.socomec.com/api/v1/ca/3756c84b-8697-446f-9ca3-931b3681f908/devices?take=50&skip=0&siteUuid=3fd80117-0674-4f30-bbff-9ef7f467a450&withContact=true&withAddress=true

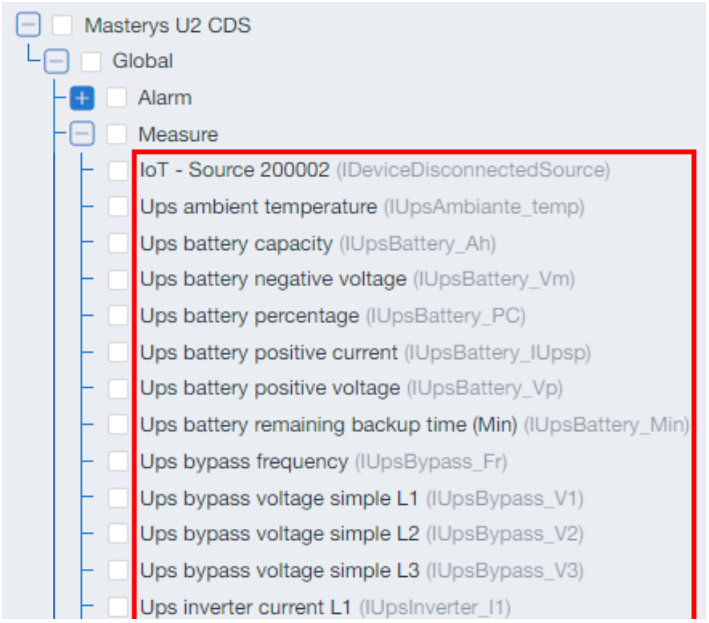
Response body (partially)	{
	"total": 2,
	"results": [
	{
	"uuid": "11d46e9a-c5cf-415c-b1a4-5b3259a9f4ce",
	"emailTo": "support@agoracalyce.com;socomec-infra@its-future.com",
	"emailCc": "frederic.morice@socomec.com",
	"technicalName": "MASTERIS_U2_CDS",
	"mappingMode": "SERVICE_ID",
	"acquisitionId": null,
	"description": null,
	"serialNumber": "P486271001",
	"custom": {
	"socomec": {
	"deviceId": "P486271001",
	"nomPowerKVA": "20000",
	"nomPowerKW": "0",
	"compatibilityVersion": "1"
	}
	},
	"configurationOptions": {
	...
	},
	"frequencyStatus": 60,
	"frequencyStream": 60,
	"status": "ACCEPTED",
	"creationType": "CREATED",
	"connectionStatus": null,

<table>
<tr><td></td><td><pre>
"visibility": null,
"longitude": 7.59328,
"latitude": 48.3729,
"deviceTypeUuid": "df14206d-7b64-11eb-b6f0-aa0fc901eabd",
"deviceTypeReferenceId": "9204",
"manufacturer": "Socomec",
"location": null,
"contractNumber": "Contract XXX",
"modelName": "Masterys",
"profileIsUpToDate": true,
"createNoCommunicationAlarm": true,
"noCommunicationPeriod": 60,
"tags": null,
"locked": false,
"additionalInformation": "",
"gatewayId": 161,
"configurationKey": "SOCOMEC_DEVICE_CONNECT_001",
"storageName": "DataHub",
"netId": null,
"movingAuthorization": "INHERITED",
"siteUuid": "3fd80117-0674-4f30-bbff-9ef7f467a450",
"gatewayUuid": "c591c2ed-749f-421b-8948-1396749ea7b2",
"iconUuid": "1b51c828-14c2-4489-8d9c-8f20a925f2fb",
"contact": {
"uuid": "61bce9de-2556-4b9e-9fd0-9715aa7793e7",
"useParent": true,
"name": "Frédéric MORICE",
"phone": "",
"mobile": "",
"email": "frederic.morice@socomec.com",
"emailCC": ""
},
"address": {
"uuid": "a805d87e-4b5b-4a87-9042-6b8b3fd68d26",
"useParent": false,
"street": "1 Rue de Westhouse",
"postCode": "67230",
"city": "Benfeld",
"country": "FRANCE",
"state": ""
},
"name": "Masterys U2 CDS",
"timeZone": "Europe/Paris",
"createdAt": "2021-03-23T14:52:40.552Z",
"owned": true
},...
],
"take": 50
}
</pre></td></tr>
<tr><td>uuid</td><td>Memorize the device uuid for next requests, for Masterys U2 CDS:
11d46e9a-c5cf-415c-b1a4-5b3259a9f4ce</td></tr>
</table>

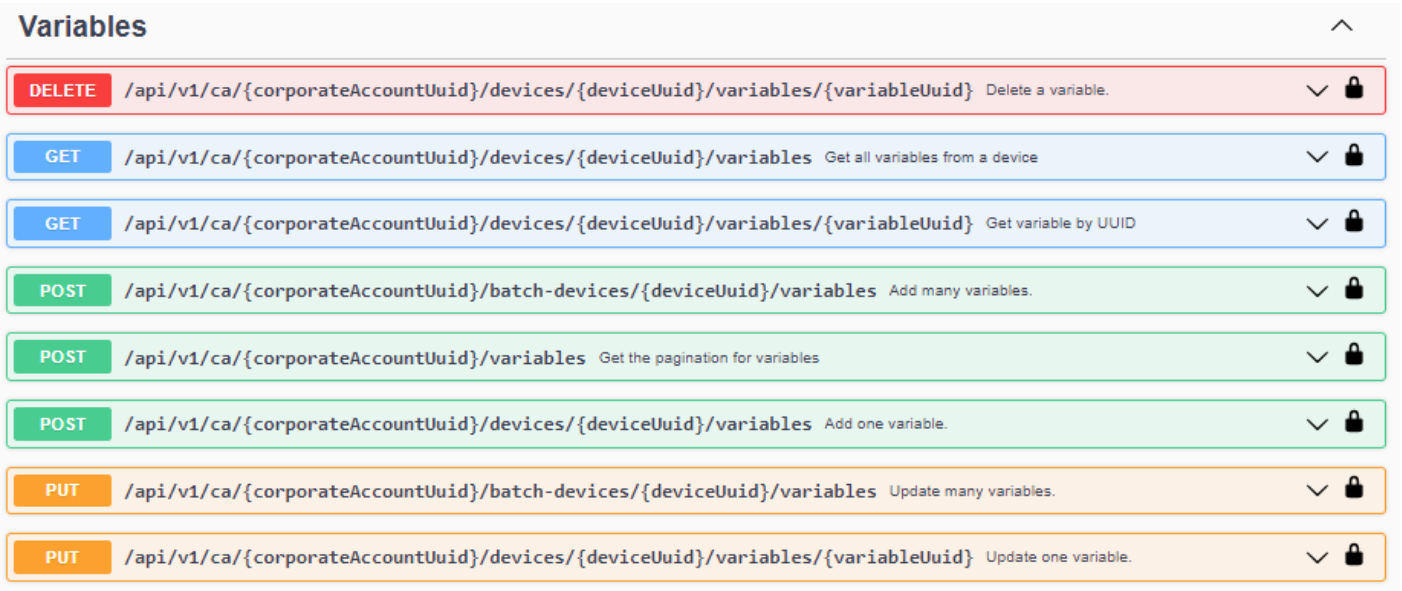
5.7 Variables

We skip the example to access to devices items, we directly test the variables request.

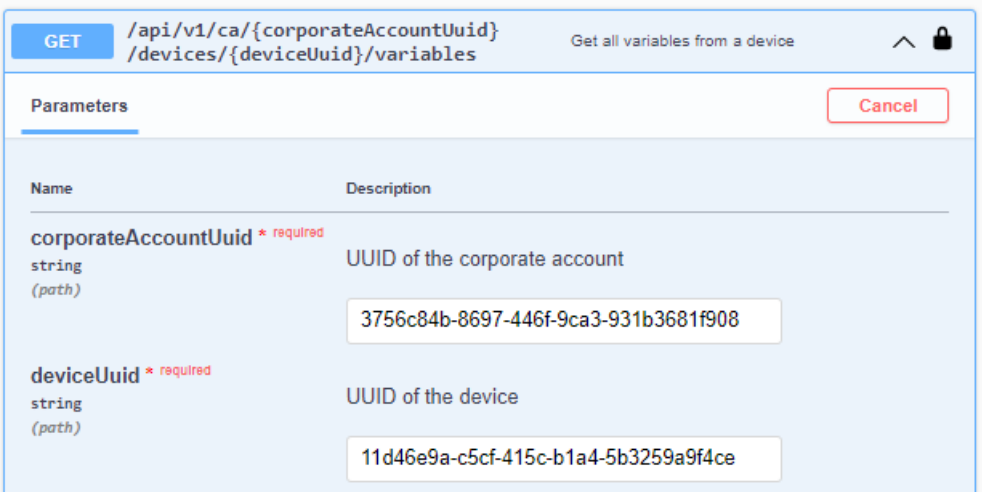
5.7.1 Wiotys devices variables example



5.7.2 SoData Variables methods



5.7.3 Get all variables for the device Masterys U2 CDS

Action	Description
Prerequisite	Valid authentication (Authorize) Uuid of the **SoLink** ca: 3756c84b-8697-446f-9ca3-931b3681f908 Uuid of the device **Masterys U2 CDS**: 11d46e9a-c5cf-415c-b1a4-5b3259a9f4ce
Swagger try out	Enter the **SoLink** ca uuid and the device **Masterys U2 CDS** uuid: 
Request URL	https://api.iot.socomec.com/api/v1/ca/3756c84b-8697-446f-9ca3-931b3681f908/devices/11d46e9a-c5cf-415c-b1a4-5b3259a9f4ce/variables
Response body (partially)	{ "results": [{ "uuid": "1e46bb11-361b-4112-b3ed-ff179f192aac", "deviceUuid": "11d46e9a-c5cf-415c-b1a4-5b3259a9f4ce", "deviceSerialNumber": "P486271001", "deviceTypeUuid": "df14206d-7b64-11eb-b6f0-aa0fc901eabd", "label": "MASTERIS_U2_CDS", "compatibilityVersion": null, "itemId": 293, "itemUuid": "670cd150-6421-40f9-a7d2-5840b91f081a", "itemIndex": "0", "itemCustom": null, "itemType": "Global", "serviceId": "141935", "serviceUuid": "be2f9626-51b4-4857-910b-294a4917446d", "serviceName": "IUpsStatistic_OverLoadMore5s_Delta", "coef": null, "offset": null, "baseCoef": null, "baseOffset": null, "serviceType": "Measure", "serviceGroup": "Output", "iotGroup": null, "unit": null, "severityName": null, "severityLevel": null, "manufacturer": null, "compatibilityMin": null, "compatibilityMax": null,

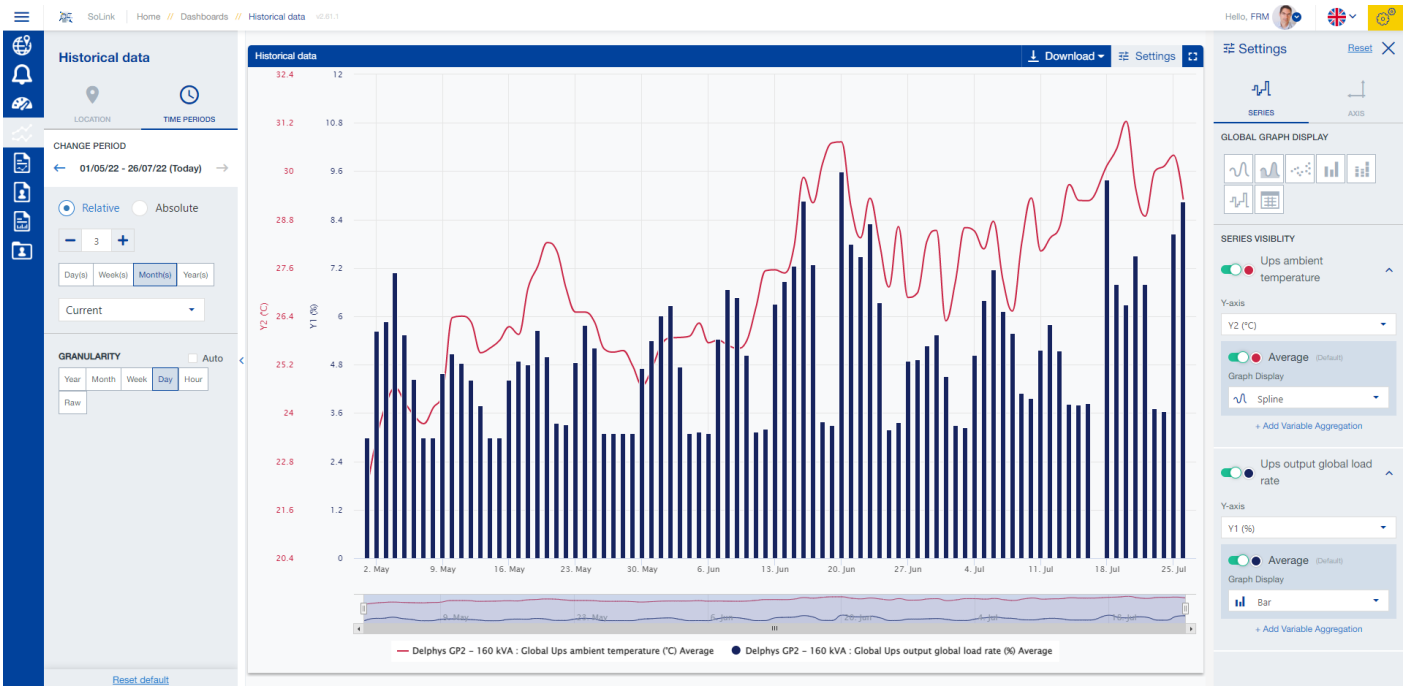
	"serviceTranslations": {
	"en": {
	"title": "Ups overload more than 5s Delta"
	},
	"fr": {
	"title": "Surcharge ASI supérieure à 5s (delta)",
	"shortName": "Surcharge ASI supérieure à 5s (delta)"
	}
	},
	"itemServiceTranslations": null,
	"reason": {
	"fromTemplate": true,
	"reasons": [
	{
	"value": "IS_DEFAULT"
	}
	]
	},
	"source": "PROFILE",
	"timestampId": null,
	"pollingTime": 3600,
	"sendOnUpdate": false,
	"isComputed": true,
	"configurationComputed": null,
	"tags": [],
	"overrideFields": null,
	"active": true,
	"streamRef": "Global|0|141935",
	"datahubId": 3214069
	},
	...
	],
	"total": 141
	}

uuid	Memorize the variables **uuid** for next requests, if needed.

5.8 Data (historical data)

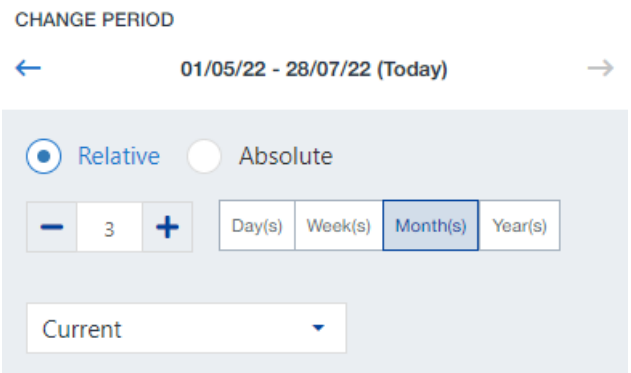
5.8.1 Wiotys devices data

From the Historical Data menu of SoLive Pro, it is possible to have access to the device data, example:

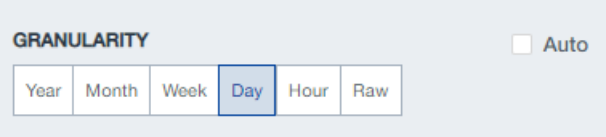


Over a period, the data can be displayed raw (raw value), or aggregated over a time step (granularity: hour, day,…).

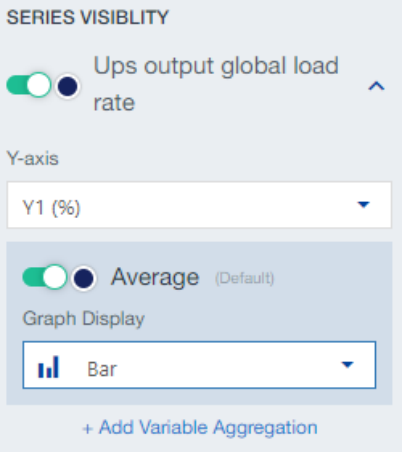
- The graphical interface allows you to choose a period:



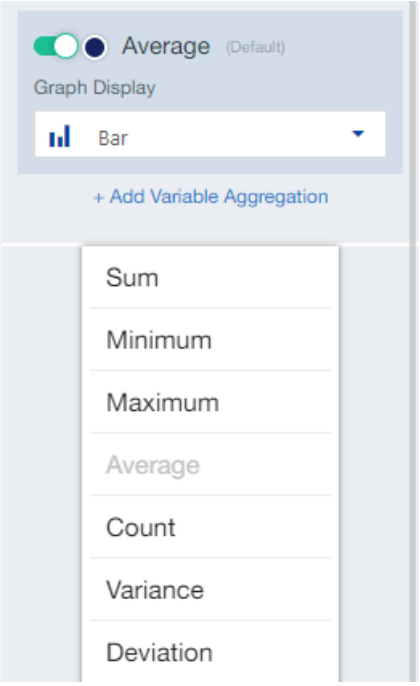
- Also a granularity (time step), from raw value (no time step) to year:



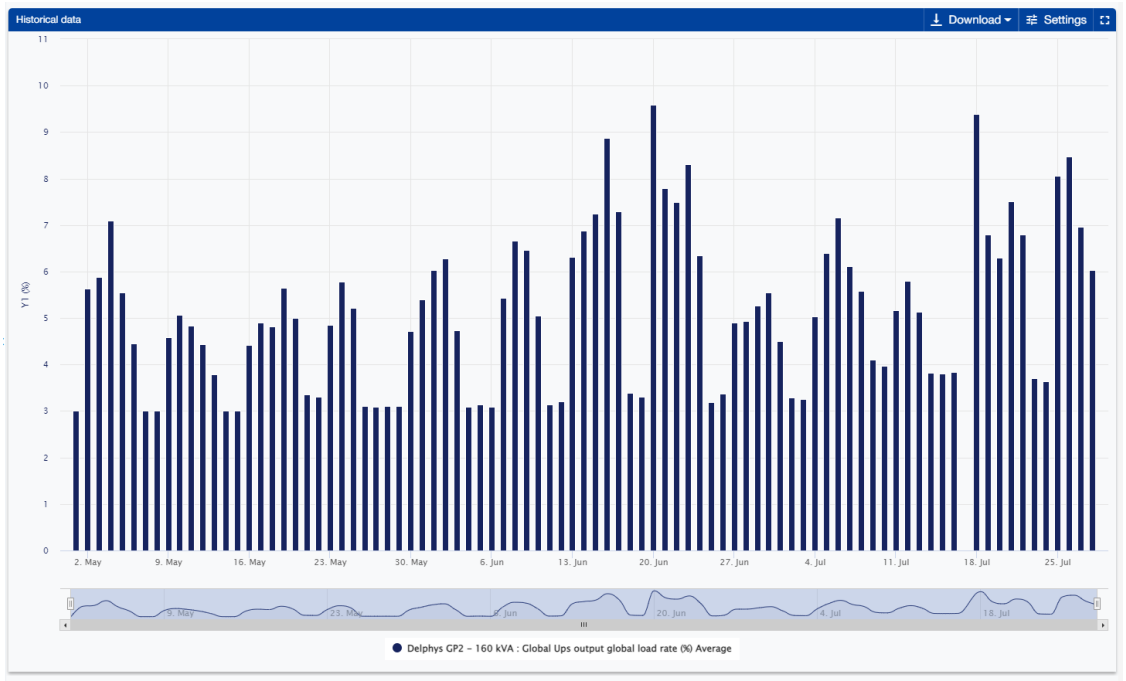
- And aggregations for the time step (if not raw), like Average:



- Available aggregations are:



- And the result is a daily average for the last 3 months:

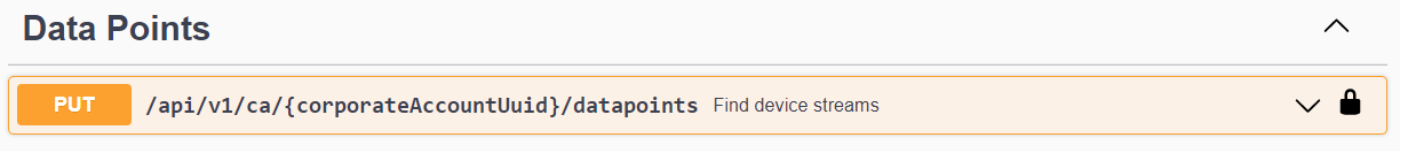


Historical data	Download	Settings	
Date (01/05/22 - 28/07/22)	Delphys GP2 - 160 kVA : Global Ups output global load rate %		
	Average		
01/05/22	3.00		
02/05/22	5.64		
03/05/22	5.88		
04/05/22	7.09		
05/05/22	5.55		
06/05/22	4.45		
07/05/22	3.00		
08/05/22	3.00		
09/05/22	4.58		
10/05/22	5.07		
11/05/22	4.84		
12/05/22	4.43		

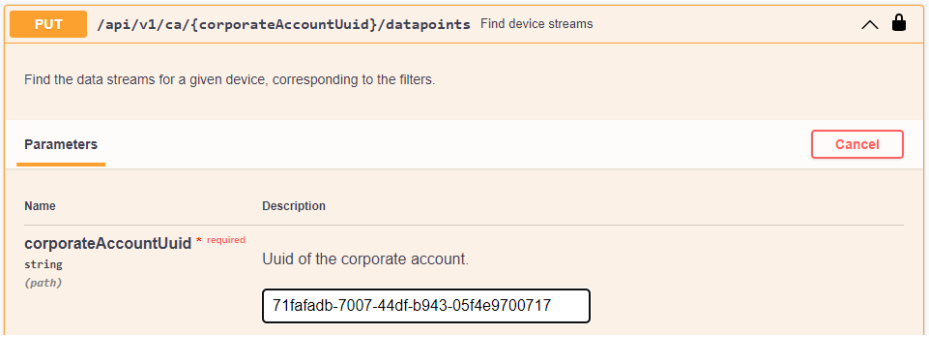
We can request the same data (raw values or aggregated values) with SoData API.

5.8.2 SoData data points methods

NB : this calls will change in v3.1



5.8.3 Get data for the device Delphys GP2 - 160 kVA

Action	Description
Prerequisite	Valid authentication (Authorize) Uuid of the **SoLink** ca: 71fafadb-7007-44df-b943-05f4e9700717 Uuid of the device **Delphys GP2 - 160 kVA**: 239b264b-fd7b-40ad-9a18-32f75d69db29 Uuid or mapping config of the variables
Swagger try out	Enter the **SoLink** ca uuid:  Enter the Request body (for this method, PUT, which is not a GET, we don't have query parameters, but a Request body to fill): Request body structure: 

	Example (for raw value and service 140100 (Ups global load rate)):
	
	{
	"from": "2022-05-18T00:00:00Z",
	"to": "2022-05-19T00:00:00Z",
	"timezone": "UTC",
	"useDeviceLocalTimezone": false,
	"granularity": "raw",
	"series": [
	{
	"deviceUuid": "239b264b-fd7b-40ad-9a18-32f75d69db29",
	"itemType": "Global",
	"itemIndex": "0",
	"serviceId": "140100"
	}
	]
	}
	
	Note: for raw "granularity", there is no "aggregation". We'll see it in more details below.

Request URL	https://api.iot.socomec.com/api/v1/ca/71fafadb-7007-44df-b943-05f4e9700717/datapoints Note: request body is not a parameter of the request
Response body (partially)	{ "from": "2022-05-18T00:00:00Z", "to": "2022-05-19T00:00:00Z", "timezone": "UTC", "useDeviceLocalTimezone": false, "granularity": "raw", "series": [{ "itemType": "Global", "itemIndex": "0", "serviceId": "140100", "deviceUuid": "239b264b-fd7b-40ad-9a18-32f75d69db29", "dataType": "number", "deviceTimeZone": "UTC", "data": [{ "at": "2022-05-18T00:05:05Z", "value": 4 }, { "at": "2022-05-18T00:10:14Z", "value": 4 }, { "at": "2022-05-18T00:15:23Z", "value": 4 }, { "at": "2022-05-18T00:20:37Z", "value": 4 },...

Request body details

The structure of the request body is:

```
{
  "from": "string",
  "to": "string",
  "timezone": "string",
  "useDeviceLocalTimezone": true,
  "granularity": "string",
  "series": [
    {
      "deviceUuid": "string",
      "itemType": "string",
      "itemIndex": "string",
      "serviceId": "string",
      "aggregation": "string"
    }
  ]
}
```

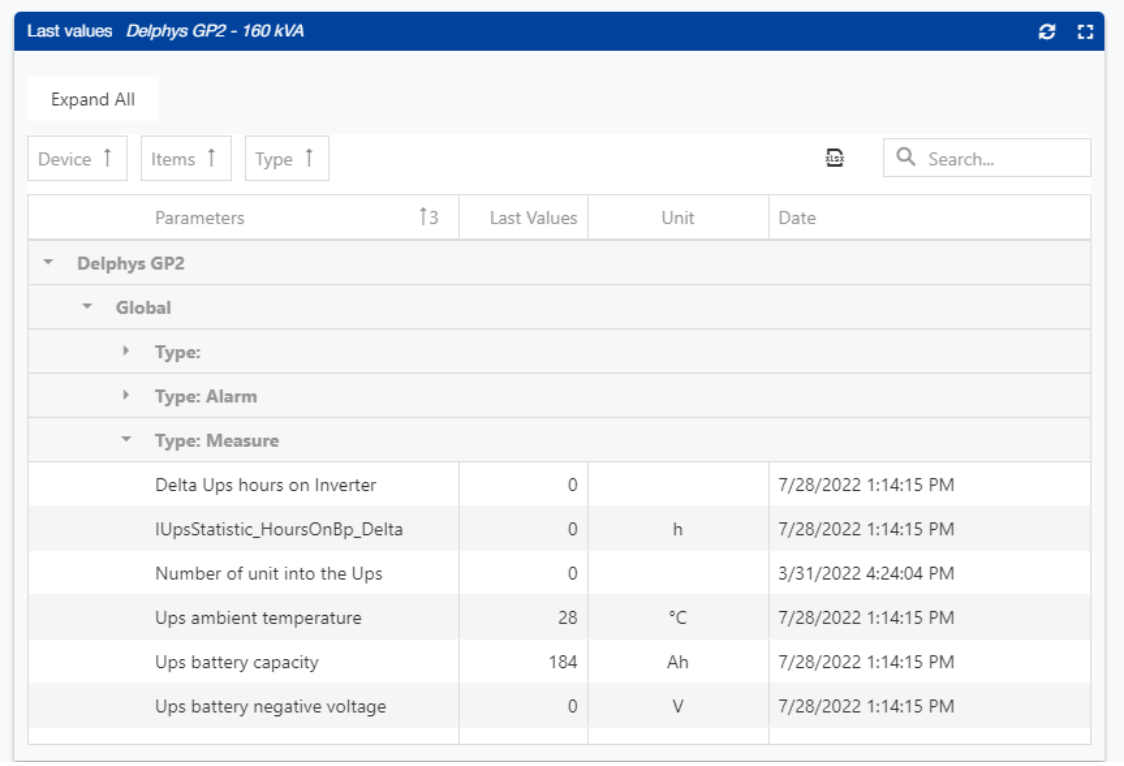
Property	Description
from	Date in ISO 8601 format which corresponds to the date from which we want data (inclusive).
to	Date in ISO 8601 format that corresponds to the date until which we want data (exclusive).
timezone	The timezone used for the from and to fields. **UTC** for Socomec devices.
useDeviceLocalTimezone	Indicates if you want to use the timezone as defined by the product. Set to **false** to have UTC dates.
granularity	The granularity of our data: • raw (raw value: no granularity) • minute (useful only if data acquisition is less than 1 minute) • hour • day • week • month • year
series	List of series (variables). Allows you to indicate which variables you want the data from, each series is a JSON object with properties.

Series properties	Description
deviceUuid	The device uuid.
itemType	The item of the variable: Global, Load, …
itemIndex	The item index of the variable: starting from 0
serviceId	The serviceId of the variable: 140100 for example (Ups output global load rate).
aggregation	The aggregation to apply on data for each granularity (time step) during all the period: from… to… For “aggregation”, we can use (no aggregation for “raw”): • avg • min • max • count • sum • var (Variance) • stdev (Deviation)

5.9 Data (Last values)

5.9.1 Wiotys devices data (Last values)

From the Last values menu of SoLive Pro, it is possible to have access to the device last data:



5.9.2 SoData data points methods (Last values)

5.9.3 Get last values for the device Delphys GP2 - 160 kVA

Action	Description
Prerequisite	Valid authentication (Authorize) Uuid of the **SoLink** ca: 71fafadb-7007-44df-b943-05f4e9700717 Uuid or the device **Delphys GP2 - 160 kVA:** 239b264b-fd7b-40ad-9a18-32f75d69db29
Swagger try out	Enter the **SoLink** ca uuid and device uuid: 
Request URL	https://api.iot.socomec.com/api/v1/ca/71fafadb-7007-44df-b943-05f4e9700717/devices/239b264b-fd7b-40ad-9a18-32f75d69db29/last-values
Response body (partially)	{ "devices": [{ "deviceUuid": "239b264b-fd7b-40ad-9a18-32f75d69db29", "values": [{ "serviceId": "140108", "itemType": "Global", "itemIndex": "0", "deviceUuid": "239b264b-fd7b-40ad-9a18-32f75d69db29", "value": "18", "date": "2022-07-28T11:19:25.000Z", "title": "Ups output current L3", "unit": "AMPERE", "deviceName": "Delphys GP2 - 160 kVA", "itemName": "Global", "serviceGroup": "Output", "serviceType": "Measure" }, ...

6 API Authentication

In order to use the API, you must first authenticate yourself; this is done by recovering a token from the user's identifiers.

Caution has replaced the correct values in the different examples!

To authenticate ourselves, we will use the Keycloak service.

6.1 Keycloack

Keycloak is an Open Source Identity and Access Management solution for modern Applications and Services.

6.2 Obtaining the token

To do this, we go through the keycloak to which we must send a POST request to https://keycloak.example.com/auth/realms/master/protocol/openid-connect/token. The request will be of type application/x-www-form-urlencoded and we will specify as information in the request:

- grant_type: the authentication type, which has the value password
- client_id: the client id configured in keycloak
- client_secret: the client secret generated by keycloak
- username: the username we want to use to make our calls (SoLive Pro user name: email)
- password: user's password (SoLive Pro pwd)

In the form of a cURL request it gives:

```
curl --location --request POST
'https://keycloak.example.com/auth/realms/master/protocol/openid-connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'client_id=abcdefgh' \
--data-urlencode 'client_secret=xxxxx-yyyyy-zzzzzz' \
--data-urlencode 'username=admin@example.com' \
--data-urlencode 'password=Strong_Password'
```

Which gives a JSON response that looks like this:

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6...",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token":
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI3YWJjODFjOC1lYzlmLTRhMGUtY...
",
  "token_type": "Bearer",
  "not-before-policy": 0,
  "session_state": "e041fd13-159d-4f4e-ad62-4d8633ad6ad1",
  "scope": "email profile"
}
```

6.3 Use

We see that we are recovering a Bearer type token, so we will be able to use it by adding our token to our headers in the access_token field. Which gives: Authorization: Bearer <token> (replace <token> with the value of the token in the JSON).

6.4 Token refresh

We can see that we have other information that will be useful, we know in particular thanks to the expires_in field that the token expires in 300 seconds (which gives 5 minutes). Or after 5 minutes you will have to use the refresh token (in refresh_token) which expires after 1800 seconds (30 minutes). The request is the same as for obtaining a token, except that the grant_type must have the value refresh_token and the username and password fields are replaced by the refresh_token field. Which in cURL gives:

```
curl --location --request POST
'https://keycloak.example.com/auth/realms/master/protocol/openid-connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
--data-urlencode 'client_id= abcdefgh' \
--data-urlencode 'client_secret=1cf53390-ffac-4b81-9096-97826bbd8660' \
--data-urlencode 'refresh_token=eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lk...'
```

7 Python example

You will find below a python script that allows to authenticate to Keycloak and to recover the access token in order to be able to make requests on our services.

The script automatically handles refreshing the token when it expires.

7.1 Full Python example

Global variables to set:

SODATA_API_URL	https://api.iot.socomec.com/
KEYCLOAK_AUTH_URL	https://connect.socomec.com/
CLIENT_ID	The client ID for SoData API you have receive from Socomec. Replace ***my_sodata_api_clientId*** with your client ID.
CLIENT_SECRET	The client secret for SoData API you have receive from Socomec. Replace ***my_sodata_api_client_secret*** with your client secret.
USER_USERNAME	Your SoLive Pro user email. Replace ***my_solivepro_email*** with your SoLive Pro user email.
USER_PASSWORD	Your SoLive Pro user password. Replace ***my_solivepro_pwd*** with your SoLive Pro user password.

Example description:

- Authentication
- Retrieve all available CorporateAccounts (ca)
- Find ca 'SoLink'
- Find device by serial number. Loop because of paging
    - find device with SN 1D15350041 (Delphys DGP 2 from (FR) Socomec/Benfeld – U3)
- Get data for this device
    - From 2021-09-20T07:47:26.724Z
    - To 2021-09-21T14:47:26.724Z
    - For variable: Global|0|140100 (Ups output global load rate)

```python
from datetime import datetime, timedelta
from keycloak import KeycloakOpenID
import requests
import json

# SoData API URL
SODATA_API_URL = "https://api.iot.socomec.com/"

# Link to Keycloak authentication API.
KEYCLOAK_AUTH_URL = "https://connect.socomec.com/"

# Client Informations.
# Generated by KeyCloak.
REALM_NAME = "Socomec"
CLIENT_ID = "my_sodata_api_clientId"
CLIENT_SECRET = "my_sodata_api_client_secret"

# User information.
# Those are your personnal logins.
USER_USERNAME = "my_wiotys_email"
USER_PASSWORD = "my_wiotys_pwd"

class Authentication:
    """
    Wrapper to manage the KeyCloak authentication.
    If required, the token is automatically refreshed.

    """
    def __init__(self, username="", password=""):
        # Configure the Keycloak client.
        self.keycloak_openid = KeycloakOpenID(
            server_url=KEYCLOAK_AUTH_URL,
            client_id=CLIENT_ID,
            realm_name=REALM_NAME,
            client_secret_key=CLIENT_SECRET,
        )

        # Authenticate and retrieve the tokens.
        self.token = self.keycloak_openid.token(username, password)

        # Save the expiration date to know when to request a refresh token.
        delta_secs = self.token["expires_in"] - 30
        self.expire_at = datetime.now() + timedelta(delta_secs)

    @property
    def access_token(self):
        """
        The access token. Automatically updated.

        When requesting the API, it should be placed in the Authorization header as a bearer.
        """

        # Verify the token validity
        if self.expire_at <= datetime.now():

            # Token is expired. Refreshing it.
            self.token = self.keycloak_openid.refresh_token(self.token["refresh_token"])
            delta_secs = self.token["expires_in"] - 30
            self.expire_at = datetime.now() + timedelta(delta_secs)

        # The access token.
        return self.token["access_token"]


def main():
    """
    Example of how to use the wrapper.
    """

    # Instantiate the wrapper.
    auth = Authentication(username=USER_USERNAME, password=USER_PASSWORD)

    # In this example, we retrieve all available CorporateAccounts.
    resp = requests.get(
        f"{SODATA_API_URL}api/v1/ca",
        headers={
            "Authorization": f"Bearer {auth.access_token}" # The header with the token.
        },
    )
    resp_json=resp.json()
```

```
    # find ca 'SoLink' (don't change the ca name in Wiotys)
    ca_uuid = ''
    if ( 'results' in resp_json):
        tab_ca = resp_json['results']

        for i in range(len(tab_ca)):
            if (tab_ca[i]['name'] == "SoLink"):
                ca_uuid = tab_ca[i]['uuid']

    # find device by sn. Loop because of paging
    if (ca_uuid != ''):
        print(ca_uuid)
        skip=0
        take=25
        total = take

        # find device with SN 1D15350041
        device_uuid = ''
        device_loop = True
        while (total > skip and device_loop):
            resp = requests.get(
                f"{SODATA_API_URL}api/v1/ca/{ca_uuid}/devices",
                params={'take': take,
                        'skip': skip,
                       },
                headers={
                    "Authorization": f"Bearer {auth.access_token}" # The header with the token.
                },
            )
            resp_json = resp.json()

            if ( 'results' in resp_json):
                tab_device = resp_json['results']

            for i in range(len(tab_device)):
                if (tab_device[i]['serialNumber'] == "1D15350041"):
                    device_uuid = tab_device[i]['uuid']
                    device_loop = False
                    break

            skip = skip + take
            total = resp_json['total']

    if (device_uuid != ''):
        # read datapoints
        put_body = {
                    "from": "2021-09-20T07:47:26.724Z",
                    "to": "2021-09-21T14:47:26.724Z",
                    "timezone": "UTC",
                    "useDeviceLocalTimezone": True,
                    "granularity": "raw",
                    "series": [
                        {
                        "deviceUuid" :device_uuid,
                        "itemType": "Global",
                        "itemIndex": "0",
                        "serviceId": "140100"
                        }
                    ]
                }
        put_body_data= json.dumps(put_body)

        resp = requests.put(
                f"{SODATA_API_URL}api/v1/ca/{ca_uuid}/datapoints",
                data=put_body_data,
                headers={
                    "Authorization": f"Bearer {auth.access_token}",
                    "Content-Type": "application/json; charset=utf-8" # The header with the token.
                },
            )
        resp_json = resp.json()
        # Display the result.
        print(resp_json)

if __name__ == "__main__":
    main()
```

Result (partially):


```
{
    'from': '2021-09-20T07:47:26.724Z',
    'to': '2021-09-21T14:47:26.724Z',
    'timezone': 'UTC',
    'useDeviceLocalTimezone': True,
    'granularity': 'raw',
    'series': [{
                'itemType': 'Global',
                'itemIndex': '0',
                'serviceId': '140100',
                'deviceUuid': '93d36cdf-70f2-441d-9560-4ca3d5498cd3',
                'dataType': 'number',
                'deviceTimeZone': 'Europe/Paris',
                'data': [{
                            'at': '2021-09-20T05:48:04Z',
                            'value': 16
                        }, {
                            'at': '2021-09-20T05:50:04Z',
                            'value': 16
                        }, {
                            'at': '2021-09-20T05:52:07Z',
                            'value': 16
                        }, {
```

7.1.1 Raw data request body

For raw data, request body has to be (no “aggregation” property):

```
put_body = {
                    "from": "2021-09-20T07:47:26.724Z",
                    "to": "2021-09-21T14:47:26.724Z",
                    "timezone": "UTC",
                    "useDeviceLocalTimezone": True,
                    "granularity": "raw",
                    "series": [
                        {
                        "deviceUuid" :device_uuid,
                        "itemType": "Global",
                        "itemIndex": "0",
                        "serviceId": "140100"
                        }
                    ]
                }
```

7.1.2 Aggregated data request body

For aggregated data, request body has to be (depending on the granularity (time step) and aggregation chosen: daily average in this example):

```
put_body = {
                    "from": "2021-09-20T07:47:26.724Z",
                    "to": "2021-09-21T14:47:26.724Z",
                    "timezone": "UTC",
                    "useDeviceLocalTimezone": True,
                    "granularity": "day",
                    "series": [
                        {
                         "deviceUuid" :device_uuid,
                         "itemType": "Global",
                         "itemIndex": "0",
                         "serviceId": "140100",
                              "aggregation": "avg",
                          }
                    ]
                }
```

8 Annex

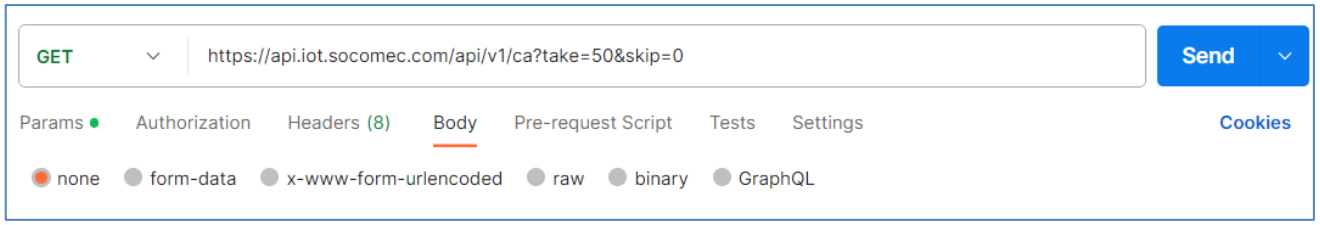
8.1 Postman

Swagger is a very simple tool to test the SoData API, but you can also use Postman:

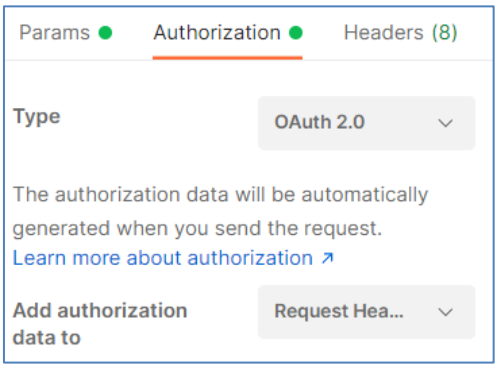
- https://www.postman.com/

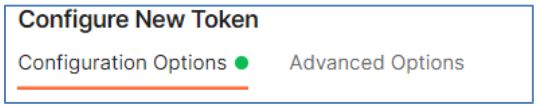
Create a new GET request:

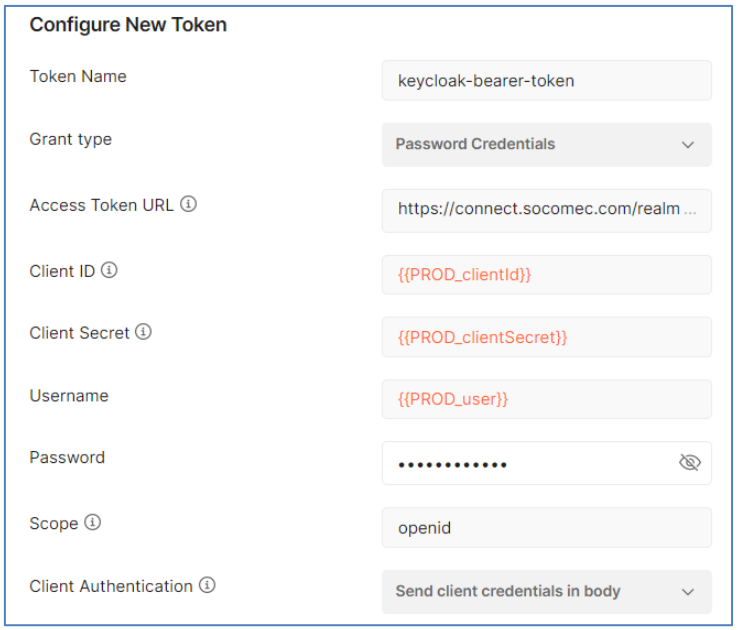
- https://api.iot.socomec.com/api/v1/ca?take=50&skip=0



Configure a new Token (click on Authorization):

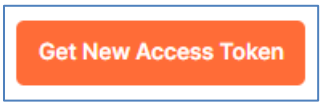




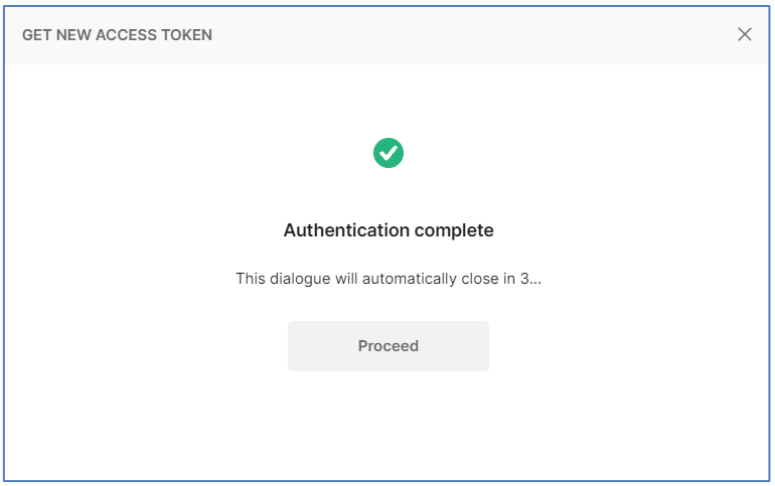


Token configuration options	
Token Name	keycloak-bearer-token
Access Token URL	https://connect.socomec.com/realms/Socomec/protocol/openid-connect/token
Client ID	Your SoData client ID
Client Secret	Your SoData client secret
Username	Your SoLive Pro Username
Password	Your SoLive Pro Password
Scope	openid

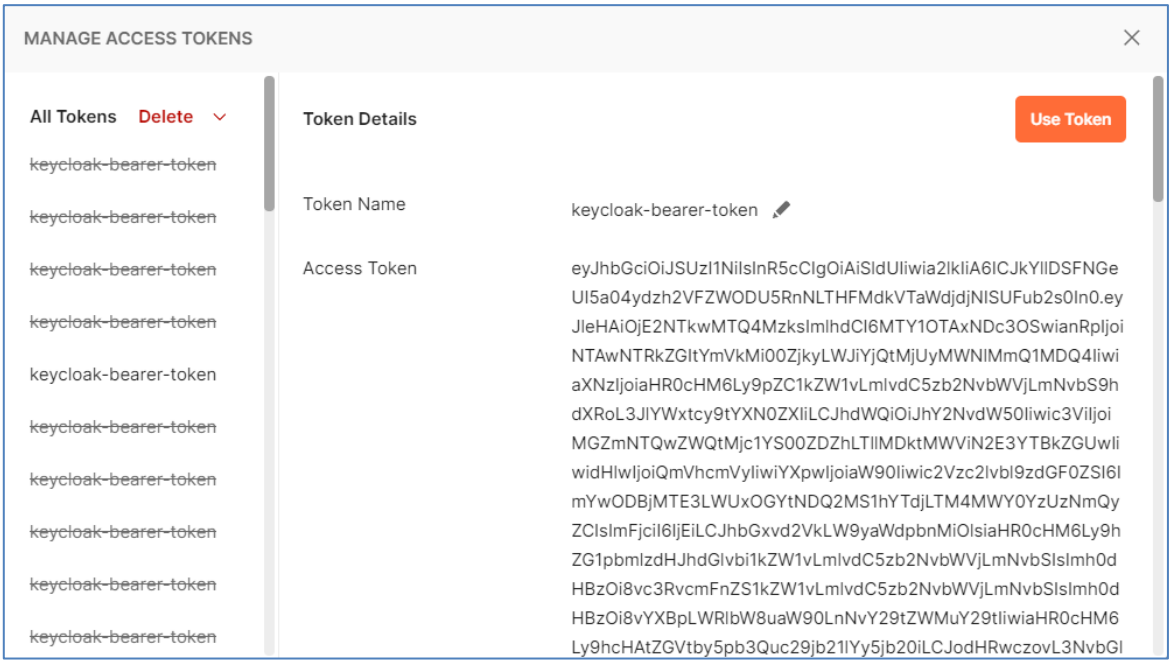
Click on Get New Access Token:



The popup is displayed:

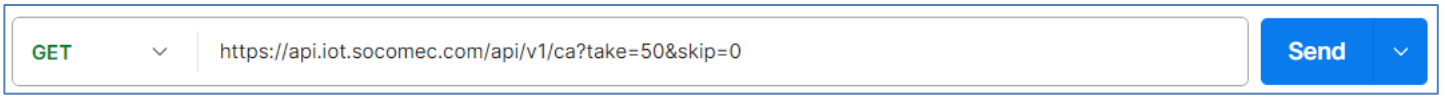


Click on Proceed, the popup is displayed:

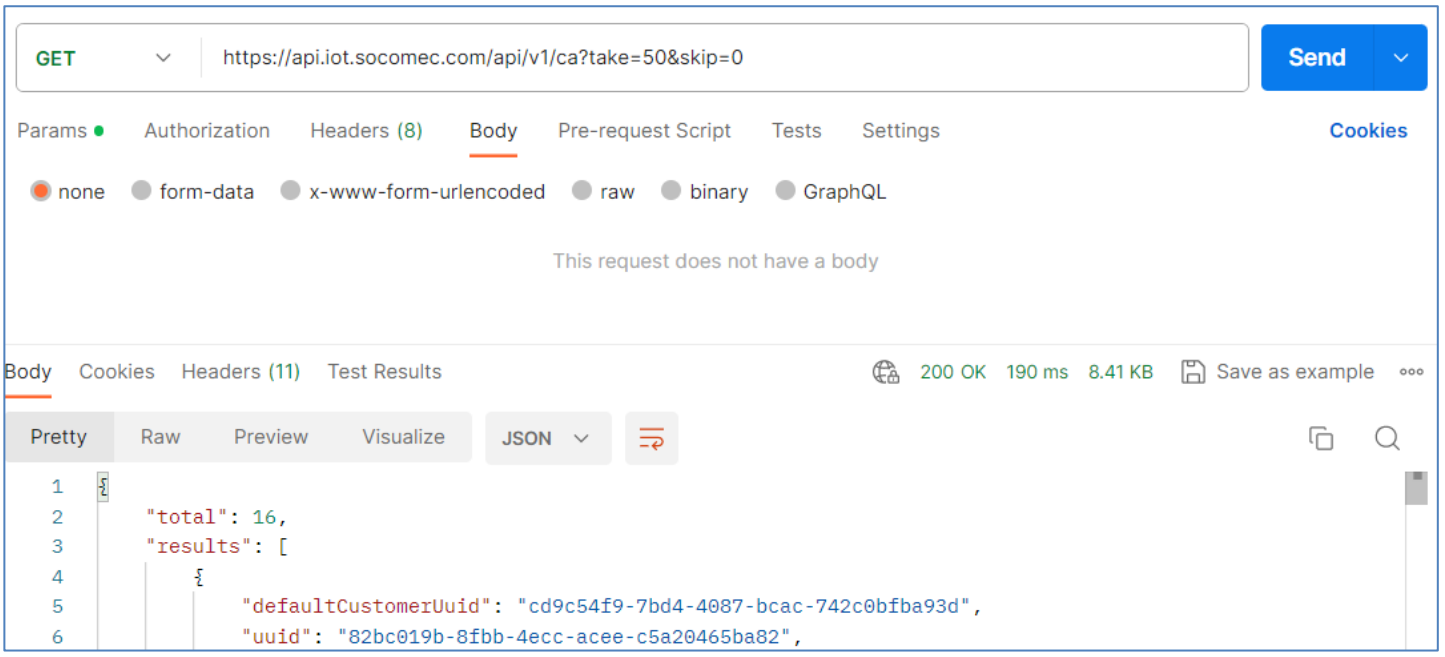


Click on Use Token:

Click on Send:



Look at the response:



Socomec, l'innovation au service de votre performance énergétique

1 constructeur indépendant

3 900 collaborateurs dans le monde

8 % du CA consacrés au R&D

400 experts dédiés aux services

L'expert de votre énergie

COUPURE

MESURE

CONVERSION D'ÉNERGIE

STOCKAGE D'ÉNERGIE

SERVICES EXPERTS

Le spécialiste d'applications critiques

- Contrôle, commande des installations électriques BT.
- Sécurité des personnes et des biens.
- Mesure des paramètres électriques.
- Gestion de l'énergie.
- Qualité de l'énergie.
- Disponibilité de l'énergie.
- Stockage de l'énergie.
- Prévention et intervention.
- Mesure et analyse.
- Optimisation.
- Conseil, déploiement et formation.

Une présence mondiale

12 sites industriels

- France (x3)
- Italie (x2)
- Tunisie
- Inde
- Chine (x2)
- USA (x2)
- Canada

30 filiales et implantations commerciales

- Afrique du Sud • Algérie • Allemagne • Australie
- Autriche • Belgique • Canada • Chine • Côte d'Ivoire
- Dubaï (Emirats Arabes Unis) • Espagne • France • Inde
- Indonésie • Italie • Pays-Bas • Pologne • Portugal
- Roumanie • Royaume-Uni • Serbie • Singapour • Slovénie
- Suède • Suisse • Thaïlande • Tunisie • Turquie • USA

80 pays où la marque est distribuée

SIÈGE SOCIAL

GROUPE SOCOMEC

SAS SOCOMEC au capital de 10 582 640 €
R.C.S. Strasbourg B 548 500 149
B.P. 60010 - 1, rue de Westhouse - F-67235 Benfeld Cedex
Tél. 03 88 57 41 41 - Fax 03 88 57 78 78

VOTRE CONTACT

www.socomec.fr

100 years OF SHARED ENERGY

socomec Innovative Power Solutions

TRN 23 127373 - Wiotys - SoData API usage example Produit par : Socomec