EM630/EM640 HTTPS REST API

REST API version: 1.0.0.3

HTTPS REST API

PROTOCOL

Public version

Version 1.1

April 28th, 2026

1. Introduction

1.1 General Information about REST API

REST API is an application programming interface that allows communication between different software systems over the internet / LAN. It relies on **standard** HTTP methods, such as **GET** and **POST** that are widely used and easy to implement.

REST APIs can be used across different platforms and programming languages, making them ideal for web and mobile applications.

1.2 REST API on EM630/EM640

All EM630/640 models support the use of REST API, the user can access to the device and get different information packages (see table below) in a JSON file format.

Table 1.1 - Description of the available information package

Package name	Description
Variable list	**Info** package, which allows the user to get a list of the available variables.
Real time data	**Data** package, which allows the user to get some real time measurement from the instrument.
Device info	**Info** package, which allows the user to get information about the device.
Diagnostic and alarms	**Status** package, which allows the user to get information about the status of the alarm and the connection of the device.

IMPORTANT: If a valid SSL certificate is not loaded on the device, **SSL certificate verification must be disabled on the client side**, since the embedded SSL certificate is self-signed.

Service	Where to find it
REST API HTTPS service for EM630/EM640	**https://<ip_address>/api/<method>**

1.3 Available product versions

Part Number	Family	Sub Family	Note
EM630BAV53XE2XXX	EM630-AV-E2	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only 2 Ethernet - Empty Module - standard
EM640BAV23XE2XXX	EM640-AV-E2	X	EM640 – 480 VLL / 65 A Direct - Only 2 Ethernet - Empty Module - standard
EM630BMV53XE2XXX	EM630-MV-E2	X	EM630 – 480 VLL / 333 mV by Primary - Only 2 Ethernet - Empty Module - standard
EM630WAV53XXXS1X	EM630-AV-W	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - S1 Module - standard
EM630WAV53XXXS1PFA	EM630-AV-W	PFA	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - S1 Module - MID PFA
EM630WAV53XXXS1PFB	EM630-AV-W	PFB	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - S1 Module - MID PFB
EM630WAV53XXXS1PFC	EM630-AV-W	PFC	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - S1 Module - MID PFC
EM630WAV53XXXM1X	EM630-AV-W	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - M1 Module - standard
EM630WAV53XXXM1PFA	EM630-AV-W	PFA	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - M1 Module - MID PFA
EM630WAV53XXXM1PFB	EM630-AV-W	PFB	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - M1 Module - MID PFB
EM630WAV53XXXM1PFC	EM630-AV-W	PFC	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - M1 Module - MID PFC
EM630WAV53XXXO1X	EM630-AV-W	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - O1 Module - standard
EM630WAV53XXXO1PFA	EM630-AV-W	PFA	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - O1 Module - MID PFA
EM630WAV53XXXO1PFB	EM630-AV-W	PFB	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - O1 Module - MID PFB
EM630WAV53XXXO1PFC	EM630-AV-W	PFC	EM630 – 480 VLL / 5 A by CTV, 5 A CT - Only wireless - O1 Module - MID PFC
EM640WAV23XXXS1X	EM640-AV-W	X	EM640 – 480 VLL / 65 A Direct - Only wireless - S1 Module - standard
EM640WAV23XXXS1PFA	EM640-AV-W	PFA	EM640 – 480 VLL / 65 A Direct - Only wireless - S1 Module - MID PFA
EM640WAV23XXXS1PFB	EM640-AV-W	PFB	EM640 – 480 VLL / 65 A Direct - Only wireless - S1 Module - MID PFB
EM640WAV23XXXS1PFC	EM640-AV-W	PFC	EM640 – 480 VLL / 65 A Direct - Only wireless - S1 Module - MID PFC
EM640WAV23XXXM1X	EM640-AV-W	X	EM640 – 480 VLL / 6 5A Direct - Only wireless - M1 Module - standard
EM640WAV23XXXM1PFA	EM640-AV-W	PFA	EM640 – 480 VLL / 65 A Direct - Only wireless - M1 Module - MID PFA
EM640WAV23XXXM1PFB	EM640-AV-W	PFB	EM640 – 480 VLL / 65 A Direct - Only wireless - M1 Module - MID PFB
EM640WAV23XXXM1PFC	EM640-AV-W	PFC	EM640 – 480 VLL / 65 A Direct - Only wireless - M1 Module - MID PFC
EM640WAV23XXXO1X	EM640-AV-W	X	EM640 – 480 VLL / 65 A Direct - Only wireless - O1 Module - standard
EM640WAV23XXXO1PFA	EM640-AV-W	PFA	EM640 – 480 VLL / 65 A Direct - Only wireless - O1 Module - MID PFA
EM640WAV23XXXO1PFB	EM640-AV-W	PFB	EM640 – 480 VLL / 65 A Direct - Only wireless - O1 Module - MID PFB
EM640WAV23XXXO1PFC	EM640-AV-W	PFC	EM640 – 480 VLL / 65 A Direct - Only wireless - O1 Module - MID PFC
EM630WMV53XXXS1X	EM630-MV-W	X	EM630 – 480 VLL / 333 mV by Primary - Only wireless - S1 Module - standard
EM630WMV53XXXM1X	EM630-MV-W	X	EM630 – 480 VLL / 333 mV by Primary - Only wireless - M1 Module - standard
EM630WMV53XXXO1X	EM630-MV-W	X	EM630 – 480 VLL / 333 mV by Primary - Only wireless - O1 Module - standard

EM630WAV53XE2S1X	EM630-AV-W-E2	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - S1 Module - standard
EM630WAV53XE2S1PFA	EM630-AV-W-E2	PFA	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - S1 Module - MID PFA
EM630WAV53XE2S1PFB	EM630-AV-W-E2	PFB	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - S1 Module - MID PFB
EM630WAV53XE2S1PFC	EM630-AV-W-E2	PFC	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - S1 Module - MID PFC
EM630WAV53XE2M1X	EM630-AV-W-E2	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - M1 Module - standard
EM630WAV53XE2M1PFA	EM630-AV-W-E2	PFA	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - M1 Module - MID PFA
EM630WAV53XE2M1PFB	EM630-AV-W-E2	PFB	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - M1 Module - MID PFB
EM630WAV53XE2M1PFC	EM630-AV-W-E2	PFC	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - M1 Module - MID PFC
EM630WAV53XE2O1X	EM630-AV-W-E2	X	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - O1 Module - standard
EM630WAV53XE2O1PFA	EM630-AV-W-E2	PFA	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - O1 Module - MID PFA
EM630WAV53XE2O1PFB	EM630-AV-W-E2	PFB	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - O1 Module - MID PFB
EM630WAV53XE2O1PFC	EM630-AV-W-E2	PFC	EM630 – 480 VLL / 5 A by CTV, 5 A CT - 2 Ethernet + wireless - O1 Module - MID PFC
EM640WAV23XE2S1X	EM640-AV-W-E2	X	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - S1 Module - standard
EM640WAV23XE2S1PFA	EM640-AV-W-E2	PFA	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - S1 Module - MID PFA
EM640WAV23XE2S1PFB	EM640-AV-W-E2	PFB	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - S1 Module - MID PFB
EM640WAV23XE2S1PFC	EM640-AV-W-E2	PFC	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - S1 Module - MID PFC
EM640WAV23XE2M1X	EM640-AV-W-E2	X	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - M1 Module - standard
EM640WAV23XE2M1PFA	EM640-AV-W-E2	PFA	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - M1 Module - MID PFA
EM640WAV23XE2M1PFB	EM640-AV-W-E2	PFB	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - M1 Module - MID PFB
EM640WAV23XE2M1PFC	EM640-AV-W-E2	PFC	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - M1 Module - MID PFC
EM640WAV23XE2O1X	EM640-AV-W-E2	X	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - O1 Module - standard
EM640WAV23XE2O1PFA	EM640-AV-W-E2	PFA	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - O1 Module - MID PFA
EM640WAV23XE2O1PFB	EM640-AV-W-E2	PFB	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - O1 Module - MID PFB
EM640WAV23XE2O1PFC	EM640-AV-W-E2	PFC	EM640 – 480 VLL / 65 A Direct - 2 Ethernet + wireless - O1 Module - MID PFC
EM630WMV53XE2S1X	EM630-MV-W-E2	X	EM630 – 480 VLL / 333 mV by Primary - 2 Ethernet + wireless - S1 Module - standard
EM630WMV53XE2M1X	EM630-MV-W-E2	X	EM630 – 480 VLL / 333 mV by Primary - 2 Ethernet + wireless - M1 Module - standard
EM630WMV53XE2O1X	EM630-MV-W-E2	X	EM630 – 480 VLL / 333 mV by Primary - 2 Ethernet + wireless - O1 Module - standard

2. EM630/EM640 REST API service description

2.1 REST API parameters

The REST API settings parameters are available from the Web Server, accessing to the *Settings* -> *Communication* menu. All parameters are described in the following table.

Note: the REST API service is enabled as default.

Table 2.1 - Webserver parameters for REST API

Parameter	Description	Values	Default
Enable	Activation/deactivation of the REST API service	Enable Disable	Enable
Username	Username to access the REST API service	String (max 31 characters)	admin
Password	Password to access the REST API service	String (max 31 characters)	adminRestAPI1!
Bearer token time validity [minutes]	Time span in minutes for the validity of the Bearer token	1 - 65535	30

IMPORTANT: for security reasons, the default password must be changed before using the service.

2.2 REST API password management

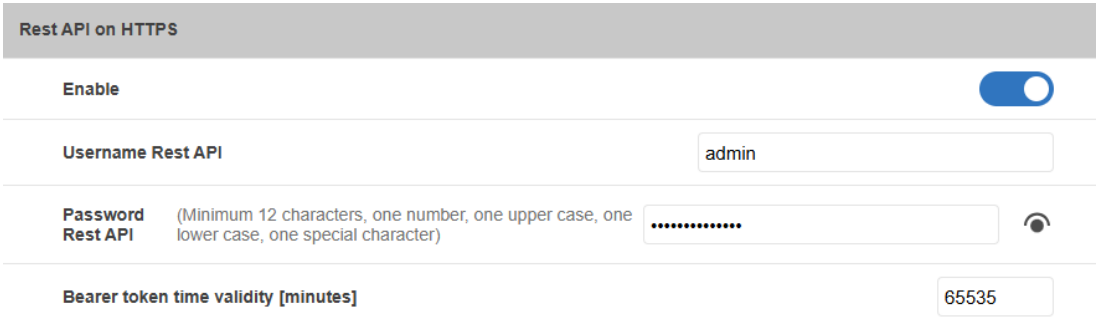
The REST API password must comply with the following requirements:

- Minimum length of **12 characters**
- At least **one numeric character**
- At least **one uppercase letter**
- At least **one lowercase letter**
- At least **one special character**

It is possible to modify the password in three different ways:

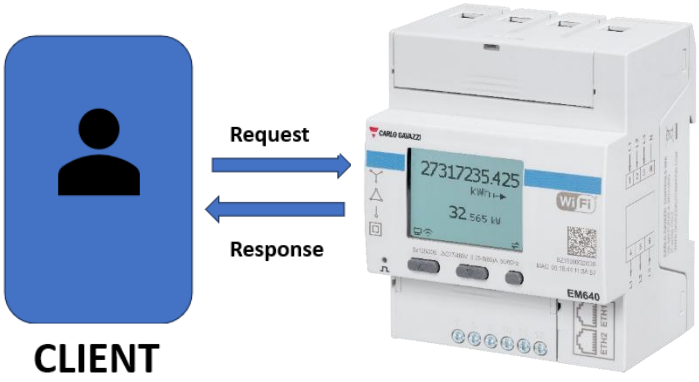
Table 2.2 - REST API password change

Option	How to manage it
Via web server	By connecting to the product web server
Via REST API	See Chapter 4
Via Modbus protocol	EM630_EM640_CPP.pdf



Picture 2.2 – Example of password change via web server

3. REST API communication procedure and structure



In this architecture, the REST API is embedded in the energy meter, which acts as the server and provides data in response to client requests.

3.1 Request structure

The request of the client has the following structure,

curl --location --request <http_method> https://<ip_address>/api/method/body

The table breaks it down and gives a full description of its element:

Table 3.1 - Description of the request structure

Parameter	Description	Values
curl	Command-line tool to transfer data to/from a server (HTTP, HTTPS, etc.)	-
--location	Tells curl to follow redirects (3xx responses)	-
--request <http_method>	The HTTP method use to communicate with the server	GET POST
https://<ip_address>	The IP address of the EM600 that the client is trying to reach to get information	-
api	Path of the service.	-
method	Specifies the information that the client wants to get from the server	See chapter **Information package**
body	Specification about the package of information that the client wants to receive	-

IMPORTANT: If a valid SSL certificate is not loaded on the device, **SSL certificate verification must be disabled on the client side**, since the embedded SSL certificate is self-signed. When using *curl*, this is done by adding the --insecure option to the request.

3.2 Response structure

If the request is successfully processed the server returns a JSON file, which is structured in two distinct parts:

Header: represents the top part of the file and gives some general information about the device, this section has fixed content irrespective of the request.

Table below provides a full description of the section header:

Table 3.2 - Description of the header content

Parameter	Description	Example
PN	Part number of the device	{ "PN": "EM630WAV53XE2S1X", "SN": "BA03500030011", "DeviceName": "EM600", "APIVer": "1.0.0.3", "FWmain": "1.0.0.3", "FWcommunication": "1.0.0.3", "FWmeasure": "1.0.0.3", "Timestamp": "2026-02-02T07:43:09+01:00" }
SN	Serial number of the device	
DeviceName	Name of the device	
APIVer	Version of the REST API	
FWmain	Main firmware version	
FWcommunication	Communication firmware version	
FWmeasure	Measurement firmware version	
Timestamp	Date and time according to ISO8601 format	

Object: represents the second part of the file, and its content depends on the status of the response;

- If **successful**, the file provides the *information package* required by the client in strings and/or arrays format (see chapters below for more detailed information).
- If **not successful,** the response contains an *error code* (see chapter **Error handling** for more detailed information).

For general information about JSON format check the link in the table below:

Information	Where to find it
JSON format	https://www.json.org/json-en.html

4. Information package

The information package should be specified in the request in the place of "**method**", immediately after the "**api**" block. Each package requires also the specification of a "**body**" parameter, which further describes the information a client wants to get.

The string below shows the general structure of a request down to the body block:

*curl --location --request <http_method> https://<ip_address>/**api**/**method**/ **body***

In the section below all the possible information that the client can get will be described in detail, for a general overview, the document will present specification about:

1. Bearer token
2. REST API password change
3. Variable list
4. Real-time data
5. Device info
6. Diagnostic and alarm
7. Error handling

For each method in the list, there will be a general description and examples.

4.1 POST requests

4.1.1 Generate a Bearer token: prerequisite for all the requests

*curl --location --request POST https://<ip_address>/api/**token**/ **body***

A **POST** request must be sent in order to get a Bearer Token in the JSON response that must be used for each **GET** request as Authorization type (i.e. **Header Authorization**). This token allows the client to authenticate to the device.

Request

In the **POST** request, REST API *username* and *password* are mandatory in multipart/form-data in the body of the *token* request.

Table 4.1.2-a Description of the request structure

Structure of the request	Example
Method: */token* **Body**: *--form "username=**<rest_API_username>**"* --form "password=**<rest_API_username>**" *[--insecure] (see note above)*	*curl --location --request POST* *https://192.168.66.146/api/token* *--form "username=admin"* *--form "password=* adminRestAPI1!*" [--insecure]*

IMPORTANT: This token must be used in each request to authenticate with the device.

Response

The response returns an object that contains strings specifying the access token and its expiration time (according to ISO8601 format).

Table 4.1.2-b Description of the response structure

Structure of the response	Example (only body)
Header **Body**: *{* *"access_token": <string>,* *"token_type": <string>,* *"expires": <string>* *}*	*{* *"access_token":* *"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDYXJsb0dhdmF6emkiLCJpYXQiOjE3MzE2NTY2NTYsImV4cCI6MTczMTY1NzI1Nn0.vFMcBHCnBkPGQnXgpBVSRL_DvmnK_RqdeB6T5mg3-ak",* *"token_type": "Bearer",* *"expires": " 2026-02-06T15:27:20+01:00"* *}*

4.1.2 REST API password change

*curl --location --request POST https://<ip_address>/api/**changePwd**/ **body***

A **POST** request must be sent in order to change the REST API password.

Request

In the **POST** request, REST API *username* and *password* are mandatory in multipart/form-data in the body of the *token* request.

Table 4.1.2-a Description of the request structure

Structure of the request	Example
Method: */changePwd* **Body**: *--form "RestAPINewPassword=adminRestAPI3!"* *[--insecure] (see note above)*	*curl --location --request POST* *https://192.168.66.146/api/changePwd* *--form "RestAPINewPassword=adminRestAPI3!" --insecure*

IMPORTANT: This token must be used in each request to authenticate with the device.

Response

The response returns an object that contains strings specifying the access token and its expiration time (according to ISO8601 format).

Table 4.1.2-b Description of the response structure

Structure of the response	Example (only body)
Header **Body**: { "access_token": <string>, "token_type": <string>, "expires": <string> }	{ "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDYXJsb0dhdmF6emkiLCJpYXQiOjE3MzE2NTY2NTYsImV4cCI6MTczMTY1NzI1Nn0.vFMcBHCnBkPGQnXgpBVSRL_DvmnK_RqdeB6T5mg3-ak", "token_type": "Bearer", "expires": " 2026-02-06T15:27:20+01:00" }

4.2 Variable list

*curl --location --request GET https://<ip_address>/api/**variableList**/**body***

A **GET** request must be sent in order to get the desired Variable List in the JSON response.

IMPORTANT: for a complete list and description of the available parameters see **Annex A: description of the available variables.**

Request

It is possible to get three different lists by typing the ***instantaneous***, ***counters*** and ***DMD*** in the body block of the request. Below a brief description of each of them.

4.2.1 Description of the response structure

Structure of the request	Example (of instantaneous)
Method: */variableList* **Body**: *<Instantaneous – counters – DMD>* **Authorization token:** --header "Authorization: Bearer <access_token_string>" *[--insecure] (see note above)*	*curl --location --request GET https://192.168.66.146/api/variableList/instantaneous\ --header "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDYXJsb0dhdmF6emkiLCJpYXQiOjE3MzE2NTg4MTIsImV4cCI6MTczMTY1OTQxMn0.B-LTSQxCghRGKXKm1uIAt6ebcSwVilmnDgmX3JlFblg" --insecure*

Response

The response returns an object that contains an array that specifies the variables, that are defined by three keys:

- **id**: the variable identifier**,**
- **name**: the variable name,
- **unit**: the variable engineering unit.

4.2.2 Description of the response structure

Structure of the response	Example (of instantaneous variable list)
Header **Body**: *{* *"variableList/instantaneous": [* *{* *"id": <string>,* *"name": <string>,* *"unit": <string>* *},* *……,* *]* *}*	<pre>"variableList/instantaneous": [{ "id": "0", "name": "VLN", "unit": "V" }, { "id": "1", "name": "VL1N", "unit": "V" }, { "id": "2", "name": "VL2N", "unit": "V" }, …] }</pre>

4.3 Real-time data

*curl --location --request GET https://<ip_address>/api/**realtimes**/**body?** **{ids}***

A **GET** request must be sent in order to get the desired real-time data in the JSON response.
Each parameter must be a number that defines the id of the variable whose user wants to get the real-time value.
IMPORTANT: This is the only method that can accept until maximum 32 parameters. To see the available parameters see **Annex A: description of the available variables** for specifications.

4.3.1 Real-time available information packaging

Method	Response	Format content
/realtimes/compact?{ids}	Returns the real-time values of the requested variables in a **compact** format	**“variable id”: “variable value”** "0": 0.0
/realtimes/standard?{ids}	Returns the real-time values of the requested variables in a **standard** format	**“variable id”: value** **“variable value”: value** "id": "3", "val": 0.0
/realtimes/extended?{ids}	Returns the real-time values of the requested variables in an **extended** format	**“variable id”: value** **“variable name”: value** **“variable value”: value** **“variable eng. unit”: value** "id": "1", "name": "VL1N", "val": 0.0, "unit": "V"
Parameter	**Descriprion**	**Note**
id	It is the variable identifier	get it using **/variableList** method (see paragraph **Variable list**)

IMPORTANT: The id must be taken from the variable list, see **Annex A: description of the available variables** for specifications.

Request

It is possible to get three different lists by typing the ***compact***, ***standard*** and ***extended*** in the body block of the request. The desired variable must be specified using its own ID. Below a brief description of the request.

4.3.2 Description of the request structure

Structure of the request	Example (of compact)
Method: */realtimes* **Body**: *<Compact – standard – extended>?{ids}* **Authorization token:** --header "Authorization: Bearer <access_token_string>" *[--insecure] (see note above)*	*curl --location --request GET https://192.168.66.146/api/* realtimes/compact?0,1,2,3\ *--header "Authorization: Bearer* *eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDYXJsb0dhdmF6emkiLCJpYXQiOjE3MzE2NTg4MTIsImV4cCI6MTczMTY1OTQxMn0.B-LTSQxCghRGKXKm1uIAt6ebcSwVilmnDgmX3JlFblg" --insecure*

Response

The response returns an object that specifies the variables, which format depends on the request; information are very similar through the three types of method, the last one has the largest size and needs more time to be created.

4.3.3 Description of the response structure

Structure of the response	Example (of instantaneous extended)
Header **Body**: *{* "realtime": [*{* "id": *<string>,* "name": *<string>,* "val": <value>, "unit": *<string>,* *},* *……,* *]* *}*	"realtimes": [{ "id": "0", "name": "VLN", "val": 0.0, "unit": "V" }, { "id": "1", "name": "VL1N", "val": 0.0, "unit": "V" }, { "id": "2", "name": "VL2N", "val": 0.0, "unit": "V" }, { "id": "3", "name": "VL3N", "val": 0.0, "unit": "V" }] }

4.4 Device info

*curl --location --request GET https://<ip_address>/api/**info***

A **GET** request must be sent in order to get the desired device info in the JSON response. This method does not accept parameters in the request, and it returns a JSON response that contains some device information, reported briefly in the table below:

Request

It is possible to get the device information by typing ***info*** in the body block of the request. All the information comes together. Below a brief description of each of them.

4.4.1 Description of the request structure

Structure of the request	Example
Method: */Info* **Body**: **Authorization token:** --header "Authorization: Bearer <access_token_string>" *[--insecure] (see note above)*	*curl --location --request GET https://192.168.66.146/api/info\ --header "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDYXJsb0dhdmF6emkiLCJpYXQiOjE3MzE2NTg4MTIsImV4cCI6MTczMTY1OTQxMn0.B-LTSQxCghRGKXKm1uIAt6ebcSwVilmnDgmX3JlFblg" --insecure*

Response

The response returns an object that contains an array that specifies the device info, that are defined by the couple:

- **key**: the variable name,
- **value**: a string or a number.

4.4.2 Description of the response structure

Structure of the response	Example
Header **Body**: { "key$_1$": *<string/value>,* ….. "key$_n$": *<string/value>,* }	{"MeasuringSys": "3Pn", "CT": 10, …. "ModbusTCPWiFiMode": "Read/Write", "WebServerEnabled": "On", "FreeAccessEnabled": "On" }

4.5 Diagnostic and alarm

*curl --location --request GET https://<ip_address>/api/**diagnostics***

A **GET** request must be sent in order to get the desired Variable List in the JSON response. This method does not accept parameters and its JSON response contains information about **diagnostics** and **alarms**.

Request

It is possible to get the device information by typing ***diagnostic*** in the body block of the request. Below a brief description of each of them.

4.5.1 Description of the request structure

Structure of the request	Example (of instantaneous)
Method: */diagnostic* **Body**: **Authorization token:** --header "Authorization: Bearer <access_token_string>" *[--insecure] (see note above)*	*curl --location --request GET https://192.168.66.146/api/diagnostic\ --header "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJDYXJsb0dhdmF6emkiLCJpYXQiOjE3MzE2NTg4MTIsImV4cCI6MTczMTY1OTQxMn0.B-LTSQxCghRGKXKm1uIAt6ebcSwVilmnDgmX3JlFblg" --insecure*

Response

The response returns an object that contains an array that specifies the variables, that are defined by the couple:

- **key**: the variable name,
- **value**: a string or a number.

4.5.2 Description of the response structure

Structure of the response	Example (of instantaneous variable list)
Header **Body**: { "key$_1$": *<string/value>,* ……. "key$_n$": *<string/value>,* }	{ "DeviceState": 0, "Alarm1Status": "Off", "Alarm2Status": "Off", "Alarm3Status": "Off", "Alarm4Status": "Off", "DIStatus": "Off", "DOStatus": "Off", "WebServerLoginStatus": "Not connected", "FreeAccessStatus": "Not connected" }

5. Error handling

This is a list of errors which can be returned if a REST API request failed, or if it is not correct:

Table 5.1 - Description of the error codes

Error code	Error message
0	Server internal error
1	HTTPS REST API service not enabled
2	Invalid Bearer token
3	Invalid method
4	Invalid URL query
5	Too many parameters
6	Variable ID not existing. Check variables list
7	Invalid ID format
8	Invalid URL: missing ID
9	Invalid Username and/or Password format
10	Wrong Username and/or Password
11	Rest API password is still the default one. It is mandatory to change it.
12	Rest API password is empty.
13	Rest API password is too long.
14	Parameter value is not valid.
15	New password does not meet constraints (only with /changePwd Rest API)
16	The request contains invalid characters (allowed ASCII range: 0x20-0x7E).

6. Annex A: description of the available variables

6.1 Variable list

The three tables below the parameters available in a response for device information:

Table 6.1 - Table of instantaneous variables:

Parameter ID	Parameter	Description	Note
0	VLN	Voltage L-N	-
1	VL1N	Voltage L1-N	-
2	VL2N	Voltage L2-N	-
3	VL3N	Voltage L3-N	-
4	VLL	Voltage L-L	-
5	VL1L2	Voltage L1-L2	-
6	VL2L3	Voltage L2-L 3	-
7	VL3L1	Voltage L3-L1	-
8	A	Current	-
9	AL1	Current A1	-
10	AL2	Current A2	-
11	AL3	Current A3	-
12	AN	Neutral current	-
13	kW	Active power	-
14	kWL1	Active power L1	-
15	kWL2	Active power L2	-
16	kWL3	Active power L3	-
17	kVA	Apparent power	-
18	kVAL1	Apparent power L1	-
29	kVAL2	Apparent power L2	-
20	kVAL3	Apparent power L3	-
21	kvar	Reactive power	-
22	kvarL1	Reactive power L1	-
23	kvarL2	Reactive power L2	-
24	kvarL3	Reactive power L3	-
25	PF	Power factor	-
26	PFL1	Power factor L1	-
27	PFL2	Power factor L2	-
28	PFL3	Power factor L3	-
29	PSeq	Phase Sequence	-
30	Hz	Frequency	-
31	THDAL1	THD Current L1	-
32	THDAL2	THD Current L2	-
33	THDAL3	THD Current L3	-
34	THDVL1N	THD Voltage L1-N	-
35	THDVL2N	THD Voltage L2-N	-
36	THDVL3N	THD Voltage L3-N	-
37	THDVL1L2	THD Voltage L-L	-
38	THDVL2L3	THD Voltage L-L	-
39	THDVL3L1	THD Voltage L-L	-

Table 6.2 - Table of counters variables:

Parameter ID	Parameter	Description	Note
40	kWh(+)TOT	Imported (+) Total Active power	-
41	kWh(+)PAR	Imported (+) partial Active power	-
42	kWh(-)TOT	Exported (-) Total Active power	-
43	kWh(-)PAR	Exported (-) partial Active power	-
44	kWh(+)T1	Imported (+) Total by tariff t1 Active power	-
45	kWh(+)T2	Imported (+) Total by tariff t2 Active power	-
46	kWh(+)L1	Imported (+) Total Active power	-
47	kWh(+)L2	Imported (+) Total Active power	-
48	kWh(+)L3	Imported (+) Total Active power	-
49	kWh(-)L1	Exported (-) Total	-
50	kWh(-)L2	Exported (-) Total Active power	-
51	kWh(-)L3	Exported (-) Total Active power	-
52	kWhQ1	Quadrant I Active power	-
53	kWhQ2	Quadrant II Active power	-
54	kWhQ3	Quadrant III Active power	-
55	kWhQ4	Quadrant IV Active power	-
56	kvarh(+)TOT	Imported (+) Total Reactive power	-
57	kvarh(+)PAR	Imported (+) partial Reactive power	-
58	kvarh(-)TOT	Exported (-) Total Reactive power	-
59	kvarh(-)PAR	Exported (-) partial Reactive power	-
60	kvarhQ1	Quadrant I Reactive power	-
61	kvarhQ2	Quadrant II Reactive power	-
62	kvarhQ3	Quadrant III Reactive power	-
63	kvarhQ4	Quadrant IV Reactive power	-
64	kVAhTOT	Total Apparent power	-
65	kVAhPAR	Partial Apparent power	-
66	kVAhQ1	Quadrant I Apparent power	-
67	kVAhQ2	Quadrant II Apparent power	-
68	kVAhQ3	Quadrant III Apparent power	-
69	kVAhQ4	Quadrant IV Apparent power	-
70	Hour(+)TOT	Total (kWh+) Hour counter	-
71	Hour(+)PAR	Partial (kWh+) Hour counter	-
72	Hour(-)TOT	Total (kWh-) Hour counter	-
73	Hour(-)PAR	Partial (kWh-) Hour counter	-
74	Hour	Total ON time	-

Table 6.3 - Table of DMD variables:

Parameter ID	Parameter	Description	Note
75	AL1DMD	Current phase 1 DMD	-
76	AL2DMD	Current phase 2 DMD	-
77	AL3DMD	Current phase 3 DMD	-
78	AL1DMDMAX	Current phase 1 maximum DMD	-
79	AL2DMDMAX	Current phase 2 maximum DMD	-
80	AL3DMDMAX	Current phase 3 maximum DMD	-
81	kWDMD	Active power DMD	-
82	kWL1DMD	Active power phase 1 DMD	-
83	kWL2DMD	Active power phase 2 DMD	-
84	kWL3DMD	Active power phase 3 DMD	-
85	kWDMDMAX	Active power maximum DMD	-
86	kWL1DMDMAX	Active power phase 1 maximum DMD	-
87	kWL2DMDMAX	Active power phase 2 maximum DMD	-
88	kWL3DMDMAX	Active power phase 3 maximum DMD	-
89	kVADMD	Apparent power DMD	-
90	kVADMDMAX	Apparent power maximum DMD	-

6.2 Device info

Table below shows the parameters available in a response for device information:

Table 6.4 - Description of the parameters available in the JSON file

Parameter	Description
MeasuringSys	Information about selected electrical system
CT	CT ratio value
DMDTimeInterval	Chosen time interval to calculate DMD values
mDNSEnabled	Activation/deactivation status of mDNS service
mDNSName	Name assigned to mDNS
EthernetMACAddress	MAC address of the Ethernet port
EthernetEnabled	Activation/deactivation status of the Ethernet interface
EthernetDHCPEnabled	Activation/deactivation status of the Ethernet DHCP option
EthernetIP	Ethernet IP address value
EthernetIPSubnetMask	Ethernet subnet mask
EthernetIPGateway	Ethernet gateway IP address
WiFiLANMACAddress	MAC address of the Wi-Fi LAN interface
WiFiLANEnabled	Activation/deactivation status of the Wi-Fi LAN interface
WiFiLANDHCPEnabled	Activation/deactivation status of the Wi-Fi LAN DHCP option
WiFiLANSSID	SSID set for the Wi-Fi LAN interface
WiFiLANIP	Wi-Fi LAN IP address
WiFiLANIPSubnetMask	Wi-Fi LAN subnet mask
WiFiLANIPGateway	Wi-Fi LAN gateway IP address
WiFi1to1MACAddress	MAC address of the Wi-Fi 1to1 interface
WiFi1to1Enabled	Activation/deactivation status of the Wi-Fi 1to1 interface
WiFi1to1SSID	SSID set for the Wi-Fi 1to1 interface
NTPServerEnabled	Activation/deactivation status of the NTP server
NTPServerConnection	Chosen NTP server connection type
NTPServerName	Set NTP server name
NTPServerIP	Set NTP server IP address
ModbusTCPEthPort	Set Modbus TCP Ethernet port
ModbusTCPEthMode	Set Modbus TCP Ethernet usage mode
ModbusTCPWiFiLANPort	Set Modbus TCP Wi-Fi LAN port
ModbusTCPWiFiLANMode	Set Modbus TCP Wi-Fi LAN usage mode
ModbusTCPWiFi1to1Port	Set Modbus TCP Wi-Fi 1to1 Ethernet port
ModbusTCPWiFi1to1Mode	Set Modbus TCP Wi-Fi 1to1 usage mode
WebServerEnabled	Activation/deactivation status of the Webserver
FreeAccessEnabled	Activation/deactivation status of the Free Access view.

7. Revision

Document revision	Date	REST API version	Note
Rev. 1.0	06/02/2026	1.0.0.2	First release
Rev. 1.1	30/04/2026	1.0.0.3	Added error code 16 in Chapter 5 “Error handling”.