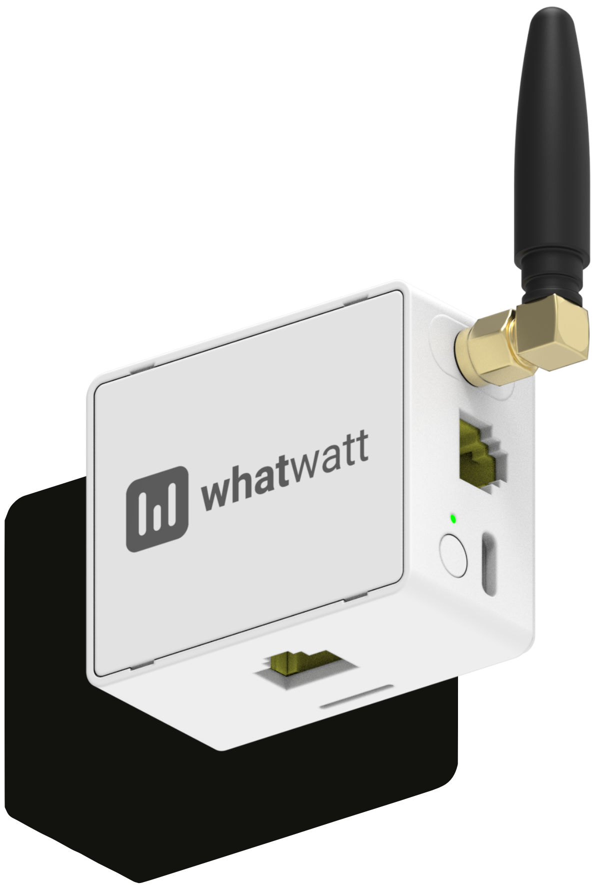
Local Integration via REST/MQTT



Important Note

Services like REST API and MQTT require a license.

Please visit www.whatwatt.ch/pricing for more information.

Version	2.0
Date	22/08/2025

whatwatt.ch
info@whatwatt.ch

whatwatt

Version	Date	Author	Changes
1.0	02.09.2024	SJ	Initial Version
1.1	05.09.2024	SJ	Corrections
1.2	17.01.2025	TK	Corrections
1.3	20.01.2025	TK	Corrections
1.4	20.01.2025	SJ	Update & Corrections
1.5	04.02.2025	SJ	Update & Corrections
1.6	17.03.2025	SJ	Update & Corrections
1.7	28.03.2025	SJ	Update & Corrections
1.8	03.06.2025	SJ	Update & Corrections – Actions
2.0	22.08.2025	SJ	Added „ Secure MQTT Integration via e.g. Mosquitto“

1. Introduction

The whatwatt Go device can be integrated into systems using two primary methods: the MQTT client connection and local REST API over HTTP. Each method offers unique advantages and is suited for different scenarios, particularly concerning network reliability, bandwidth usage, and ease of integration.

1.1. Local REST API over HTTP

The REST API provides a straightforward method for integrating the whatwatt Go device through standard web protocols. Key advantages include:

- Ease of integration with various systems due to its simplicity and compatibility with many development environments.
- Allows for direct implementation of CRUD operations (Create, Read, Update, Delete) on the data provided by the device.

1.2. MQTT Client Connection

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe messaging protocol. It is particularly beneficial for scenarios where bandwidth usage and network reliability are significant concerns. By connecting as an MQTT client, the whatwatt GO device can:

- Efficiently handle high volumes of data transmission with minimal overhead.
- Ensure low latency, making it ideal for real-time monitoring and control applications.
- Optimize the use of network resources, providing a reliable method for data communication.

1.3. Conclusion

In summary, the whatwatt Go device offers versatile integration options to cater to different needs. The MQTT client connection method is ideal for applications requiring real-time power usage monitoring and efficient network usage, while the REST API over HTTP offers a simple and widely compatible integration method. Both methods ensure that the device can be seamlessly incorporated into various systems, enhancing its functionality and utility.

2. General Information

2.1. Device connected via WiFi and powered by the meter

When powering the device via the meter interface only (especially via MBUS) and using WiFi, keep in mind the limited power resources of the device, do not call the HTTP API too often because in extreme cases the system will shut down for a certain period of time.

If you notice that the device turns off when working with the cloud via MQTT, disable unnecessary services and reduce the frequency of sending reports.

Also keep in mind that the distance from the WiFi router matters in regards to the energy consumption.

2.2. Finding a device on your local network

You can locate the device via mDNS.

The hostname of device is whatwatt-<last 6 hexadecimal upper-case letters of device identifier>.local

You need to search for the HTTP service: _http.tcp of name whatwatt Go ABCDEF (where ABCDEF is the last 6 hexadecimal upper-case letters of device identifier).

In TXT record for service, you can find the device id and device type. The device type for Whatwatt GO is equal to 100.

Example result

```
ens38 IPv4 Whatwatt Go 9F8124 WebUI    Web Site     local
hostname = [Whatwatt-9F8124.local]
address = [192.168.99.176]
port = [80]
txt = ["id=A842E39F8124" "type=100"]
```

2.3. Reading general system information

System information can be read using HTTP. To do this, make a GET request with the path api/v1/system.

Endpoint	api/v1/system
Method	GET
Response content type	application/json

Note – If you have set a password for the Web UI interface, then each HTTP request requires the credentials to be passed in the form of Basic-Authentication.

Example response

```
{
  "device":{
    "name":"",
    "id":"A842E39F8124",
    "model":"WW_Go_1.1",
    "firmware":"1.2.6",
    "upgrade_capable":true,
    "date":"2025-01-22",
    "time":"11:25:31",
    "time_since_boot":166,
    "last_reboot":{
      "date":"2025-01-22",
      "time":"11:22:45"
    },
    "plug":{
      "v_usb":5.25,
      "v_mbus":0,
      "v_p1":5.23,
      "v_scap":4.75
    }
  },
  "clouds":{
    "whatwatt":{
      "enabled":true,
      "connected":true
    },
    "mystrom":{
      "enabled":false,
      "status":"DISABLED"
    },
    "solar_manager":{
      "enabled":false,
      "connected":false
    },
    "mqtc":{
      "enabled":false,
      "connected":false
    }
  },
  "meter":{
    "status":"OK",
    "interface":"P1",
    "id":"20000741",
    "manufacturer":"Ensor",
    "type":"ESR51030712084367",
    "model":"1ESR0012084367",
    "protocol":"DSMR",
    "protocol_version":"5.0",
    "report_interval":0.998,
    "tariff":1,
    "date":"2025-01-22",
    "time":"11:25:31"
  },
  "wifi":{
    "ssid":"sjj",
    "bssid":"DC15C84FBAB6",
    "channel":13,
    "ht":"20",
    "rssi":-35,
    "signal":100,
    "auth_mode":"WPA2-WPA3",
    "pairwise_cipher":"CCMP",
    "group_cipher":"CCMP",
    "phy":"bgn",
    "wps":false,
    "country":"CH",
    "mac":"A842E39F8124",
    "ip":"192.168.99.176",
    "mask":"255.255.255.0",
    "gateway":"192.168.99.1",
    "dns":"0.0.0.0",
    "status":"OK",
    "mode":"STA"
  },
  "sd_card":{
    "installed":true,
    "type":"SDHC/SDXC",
    "size":7618,
    "speed":20
  }
}
```

JSON object fields description

Field	Type	Range	Description
.device.name	string	0..31	You can set device name used Web UI System > Name
.device.id	string	12 upper-case hexadecimal letters	The unique identifier of device
.device.model	string		The model of device
.device.firmware	string		Firmware version installed on device
.device.date	string	YYYY-MM-DD	System date in local time zone
.device.time	string	HH:MM:SS	System time in local time zone
.device.time_since_reboot	int		Seconds from last reboot
.device.last_reboot.date	string	YYYY-MM-DD	Date of last reboot in local time zone
.device.last_reboot.time	string	HH:MM:SS	Time of last reboot in local time zone
.device.plug.v_usb	double		
.device.plug.v_mbus	double		
.device.plug.v_p1	double		
.device.plug.v_scap	double		In the case of powering the device from the meter via the M-bus and many enabled services, check if this voltage does not drop below 4.5V. If the voltage keeps dropping since the device is turned on, reduce the number of services turned on or the report times.
.clouds.whatwatt.enabled	boolean		Status is the whatwatt cloud enabled
.clouds.whatwatt.connected	boolean		Status is there a connection to the cloud whatwatt
.clouds.mystrom.enabled	boolean		Status if myStrom Cloud is enabled
.clouds.mystrom.status	string	DISABLED DISCONNECTED WAITING TIME CONNECTING DO HANDSHAKE CONNECTED REGISTERED	myStrom Service Status
.clouds.solar_manager.enabled	boolean		Status if Solar Manager Cloud is enabled
.clouds.solar_manager.connected	boolean		Status: whether there is a connection to the Solar Manager cloud
.clouds.mqtc.enabled	boolean		Status if the local MQTT client is enabled
.clouds.mqtc.connected	boolean		Status if the local MQTT client is connected
.meter.status	string	NOT CONNECTED NO DATA RECOGNITION OK ENCRYPTION KEY KEY REQUIRED NOT RECOGNIZED	

Field	Type	Range	Description
.meter.interface	string	NONE P1 MBUS TTL MEP	Type of meter interface used, physical layer
.meter.id	string		Meter identifier
.meter.manufacturer	string		Meter manufacturer if specified
.meter.type	string		Meter type if specified
.meter.model	string		Meter model if specified
.meter.protocol	string	DSMR DLMS KMP MEP MBUS	Data protocol, logical layer
.meter.protocol_version	string		Meter protocol version if specified
.meter.report_interval	double		Meter report interval
.meter.tariff	uint	1, 2	Current tariff on meter if specified then value is different from zero
.meter.date	string		Date on meter in local time zone
.meter.time	string		Time on meter in local time zone
.wifi.ssid	string	1..32	The SSID (Service Set Identifier) is the name of a Wi-Fi network. It's the identifier that devices use to connect to the correct wireless network among multiple available networks.
.wifi.bssid	string	12 upper-case hexadecimal letters	The BSSID (Basic Service Set Identifier) is the MAC (Media Access Control) address of a wireless access point or router. It uniquely identifies each access point in a Wi-Fi network.
.wifi.channel	uint	1..13	A Wi-Fi channel is a specific frequency range within a Wi-Fi band that routers and devices use to communicate wirelessly.
.wifi.ht	string	20 40+ 40-	Wi-Fi HT (High Throughput) is a mode used in the Wi-Fi 802.11n standard that increases the network's data throughput. It uses MIMO (Multiple Input Multiple Output) technology to transmit multiple data streams simultaneously, enhancing network performance.
.wifi.rssi	int	dBm	Wi-Fi RSSI: RSSI (Received Signal Strength Indicator) measures the power level of a received signal. It's expressed in decibels (dBm), with higher values (closer to zero) indicating stronger signals. For example, -30 dBm is a very strong signal, while -90 dBm is very weak.
.wifi.signal	uint	0..100	Wi-Fi signal strength in precents
.wifi.auth_mode	string	open WEP WPA WPA2 WPA-WPA2 EAP WPA3 WPA2-WPA3 WAPI OWE WPA3-ENT	Wi-Fi auth_mode (authentication mode) determines how device authenticate on a Wi-Fi network.

Field	Type	Range	Description
.wifi.pairwise_cipher	string		The pairwise cipher in Wi-Fi security refers to the encryption method used to secure unicast (one-to-one) communication between a client device and an access point.
.wifi.group_cipher	string	none WEP40 WEP104 TKIP CCMP TKIP-CCMP AES-CMAC-128 SMS4 GCMP GCMP256 AES-GMAC-128 AES-GMAC-256 unknown	The group cipher in Wi-Fi security refers to the encryption method used to secure multicast and broadcast communications within a Wi-Fi network.
.wifi.phy	string	bgn	
.wifi.wps	string	true or false	
.wifi.country	string	2 characters	The Wi-Fi country code is a setting that determines the regulatory domain for a Wi-Fi device, such as a router or access point.
.wifi.mac	12 upper-case hexadecimal letters	12 upper-case hexadecimal letters	A MAC address (Media Access Control address) is a unique identifier assigned to a network interface controller (NIC) for use in communications within a network segment.
.wifi.ip	IPv4 string	ddD.ddD.ddD.ddD	An IPv4 address assigned to an interface is a unique identifier used to identify a device on a network.
.wifi.mask	IPv4 string	ddD.ddD.ddD.ddD	A netmask is a 32-bit binary mask used to divide an IP address into subnets and specify the network's available hosts.
.wifi.gateway	IPv4 string	ddD.ddD.ddD.ddD	A gateway IP address, also known as a default gateway, is an IP address that serves as an access point or "gateway" to other networks. It acts as a bridge between your local network and external networks, such as the internet.
.wifi.dns	IPv4 string	ddD.ddD.ddD.ddD	A DNS IP address refers to the IP address of a DNS (Domain Name System) server. The DNS server is responsible for translating human-readable domain names (like www.example.com) into machine-readable IP addresses (like 192.168.1.1).
.wifi.status	string	OK Error Disabled Disconnected	
.wifi.mode	string	STA	
.ethernet.mac	string	12 upper-case hexadecimal letters	
.ethernet.ip	IPv4 string	ddD.ddD.ddD.ddD	Same as in the case of Wi-Fi. Small d is optional digit, big D is always present digit
.ethernet.mask	IPv4 string	ddD.ddD.ddD.ddD	Same as in the case of Wi-Fi. Small d is optional digit, big D is always present digit
.ethernet.gateway	IPv4 string	ddD.ddD.ddD.ddD	Same as in the case of Wi-Fi. Small d is optional digit, big D is always present digit
.ethernet.dns	IPv4 string	ddD.ddD.ddD.ddD	Same as in the case of Wi-Fi. Small d is optional digit, big D is always present digit

Field	Type	Range	Description
.ethernet.status	string	Up Down	Ethernet interface status
.sd_card.installed	boolean		Is a microSD card installed in the system
.sd_card.type	string		Type of microSD card
.sd_card.size	uint		Logical size of the microSD card
.sd_card.speed	uint		Card bus frequency in MHz

3. Reading meter reports with REST API HTTP

3.1. Polling method

Reading measurements from the meter is possible by calling the API api/v1/report with the GET method.

Endpoint	api/v1/report
Method	GET
Response content type	application/json

Example response for Landis+Gyr E450

```
{
  "report":{
    "id":54143,
    "interval":3.055,
    "date_time":"2024-08-24T13:51:07Z",
    "instantaneous_power":{
      "active":{
        "positive":{
          "total":0.042
        },
        "negative":{
          "total":0
        }
      }
    },
    "energy":{
      "active":{
        "positive":{
          "total":47.251,
          "t1":33.388,
          "t2":14.642
        },
        "negative":{
          "total":8.965,
          "t1":7.868,
          "t2":1.097
        }
      },
      "reactive":{
        "imported":{
          "inductive":{
            "total":33.713,
            "t1":31.7,
            "t2":2.013
          },
          "capacitive":{
            "total":2.247,
            "t1":2.247,
            "t2":0
          }
        },
        "exported":{
          "inductive":{
            "total":3.555,
            "t1":2.727,
            "t2":0.828
          },
          "capacitive":{
            "total":30.562,
            "t1":4.852,
            "t2":26.951
          }
        }
      }
    }
  },
  "meter":{
    "status":"OK",
    "interface":"MBUS",
    "protocol":"DLMS",
    "id":"636192",
    "model":"636192"
  },
  "system":{
    "id":"A842E39F8124",
    "date_time":"2024-09-02T13:02:37Z",
    "boot_id":"99D2A6A4",
    "time_since_boot":253234
  }
}
```

Example response for Ensor eRS801

```
{
  "report":{
    "id":1810,
    "interval":1.001,
    "tariff":1,
    "date_time":"2025-01-22T11:54:02Z",
    "instantaneous_power":{
      "active":{
        "positive":{
          "total":0.037,
          "l1":0.037,
          "l2":0,
          "l3":0
        },
        "negative":{
          "total":0,
          "l1":0,
          "l2":0,
          "l3":0
        }
      }
    },
    "voltage":{
      "l1":234.34,
      "l2":36.19,
      "l3":36.54
    },
    "current":{
      "l1":0.22,
      "l2":0,
      "l3":0
    },
    "energy":{
      "active":{
        "positive":{
          "total":68.092,
          "t1":44.258,
          "t2":23.834
        },
        "negative":{
          "total":0,
          "t1":0,
          "t2":0
        }
      }
    },
    "conv_factor":1
  },
  "meter":{
    "status":"OK",
    "interface":"P1",
    "protocol":"DSMR",
    "protocol_version":"5.0",
    "logical_name":"ESR51030712084367",
    "id":"20000741",
    "model":"1ESR0012084367",
    "vendor":"Ensor",
    "prefix":"ESR"
  },
  "system":{
    "id":"A842E39F8124",
    "date_time":"2025-01-22T11:54:03Z",
    "boot_id":"B3E61904",
    "time_since_boot":1878
  }
}
```

The API returns an object containing three main sub-objects: report, meter, and system. The report object includes real-time measurements. The meter object provides details about the meter, and the system object describes the system.

Note –Fields within the report object may vary, and the presence of certain fields depends on the data sent by the meter.

Fields description

Field	Type	Unit	Description
.report.id	uint		Report incremental identifier
.report.interval	double	s	Report period
.report.tariff	uint		1 - t1, 2 - t2
.report.date_time	ISO8601		Time when report was generated in local time. The default time zone is European time zone. CET-1CEST,M3.5.0,M10.5.0/3
.report.instantaneous_power.active.positive.total	double	kW	Positive active instantaneous power (A+)
.report.instantaneous_power.active.positive.l1	double	kW	Positive active instantaneous power (A+) in phase L1
.report.instantaneous_power.active.positive.l2	double	kW	Positive active instantaneous power (A+) in phase L2
.report.instantaneous_power.active.positive.l3	double	kW	Positive active instantaneous power (A+) in phase L3
.report.instantaneous_power.active.negative.total	double	kW	Negative active instantaneous power (A-)
.report.instantaneous_power.active.negative.l1	double	kW	Negative active instantaneous power (A-) in phase L1
.report.instantaneous_power.active.negative.l2	double	kW	Negative active instantaneous power (A-) in phase L2
.report.instantaneous_power.active.negative.l3	double	kW	Negative active instantaneous power (A-) in phase L3
.report.instantaneous_power.reactive.positive.total	double	kvar	Positive reactive instantaneous power (Q+)
.report.instantaneous_power.reactive.positive.l1	double	kvar	Positive reactive instantaneous power (Q+) in phase L1
.report.instantaneous_power.reactive.positive.l1	double	kvar	Positive reactive instantaneous power (Q+) in phase L2

Field	Type	Unit	Description
.report.instantaneous_power.reactive.positive.l3	double	kvar	Positive reactive instantaneous power (Q+) in phase L3
.report.instantaneous_power.reactive.negative.total	double	kvar	Negative reactive instantaneous power (Q-)
.report.instantaneous_power.reactive.negative.l1	double	kvar	Negative reactive instantaneous power (Q-) in phase L1
.report.instantaneous_power.reactive.negative.l2	double	kvar	Negative reactive instantaneous power (Q-) in phase L2
.report.instantaneous_power.reactive.negative.l3	double	kvar	Negative reactive instantaneous power (Q-) in phase L1
.report.instantaneous_power.apparent.total	double	kVA	Apparent instantaneous power (S+)
.report.voltage.l1	double	V	Instantaneous voltage (U) in phase L1
.report.voltage.l2	double	V	Instantaneous voltage (U) in phase L2
.report.voltage.l3	double	V	Instantaneous voltage (U) in phase L3
.report.current.l1	double	A	Instantaneous current (I) in phase L1
.report.current.l2	double	A	Instantaneous current (I) in phase L2
.report.current.l3	double	A	Instantaneous current (I) in phase L3
.report.energy.active.positive.total	double	kWh	Positive active energy (A+) total
.report.energy.active.positive.t1	double	kWh	Positive active energy (A+) in tariff T1
.report.energy.active.positive.t2	double	kWh	Positive active energy (A+) in tariff T2
.report.energy.active.negative.total	double	kWh	Negative active energy (A-) total
.report.energy.active.negative.t1	double	kWh	Negative active energy (A-) in tariff T1
.report.energy.active.negative.t2	double	kWh	Negative active energy (A-) in tariff T2
.report.energy.rective.positive.total	double	kvarh	Positive reactive energy (Q+) total
.report.energy.rective.positive.t1	double	kvarh	Positive reactive energy (Q+) in tariff T1
.report.energy.rective.positive.t2	double	kvarh	Positive reactive energy (Q+) in tariff T2
.report.energy.rective.negative.total	double	kvarh	Negative reactive energy (Q-) total
.report.energy.rective.negative.t1	double	kvarh	Negative reactive energy (Q-) in tariff T1
.report.energy.rective.nagative.t2	double	kvarh	Negative reactive energy (Q-) in tariff T2
.report.energy.reactive.imported.inductive.total	double	kvarh	Imported inductive reactive energy in 1-st quadrant (Q1) total
.report.energy.reactive.imported.inductive.t1	double	kvarh	Imported inductive reactive energy in 1-st quadrant (Q1) in tariff T1
.report.energy.reactive.imported.inductive.t2	double	kvarh	Imported inductive reactive energy in 1-st quadrant (Q1) in tariff T2
.report.energy.reactive.imported.capacitive.total	double	kvarh	Imported capacitive reactive energy in 2-nd quadrant (Q2) total
.report.energy.reactive.imported.capacitive.t1	double	kvarh	Imported capacitive reactive energy in 2-nd quadr. (Q2) in tariff T1
.report.energy.reactive.imported.capacitive.t2	double	kvarh	Imported capacitive reactive energy in 2-nd quadr. (Q2) in tariff T2
.report.energy.reactive.exported.inductive.total	double	kvarh	Exported inductive reactive energy in 3-rd quadrant (Q3) total

Field	Type	Unit	Description
.report.energy.reactive.exported.inductive.t1	double	kvarh	Exported inductive reactive energy in 3-rd quadrant (Q3) in tariff T1
.report.energy.reactive.exported.inductive.t2	double	kvarh	Exported inductive reactive energy in 3-rd quadrant (Q3) in tariff T2
.report.energy.reactive.exported.capacitive.total	double	kvarh	Exported capacitive reactive energy in 4-th quadrant (Q4) total
.report.energy.reactive.exported.capacitive.t1	double	kvarh	Exported capacitive reactive energy in 4-th quadr. (Q4) in tariff T1
.report.energy.reactive.exported.capacitive.t2	double	kvarh	Exported capacitive reactive energy in 4-th quadr. (Q4) in tariff T2
.report.max_demand.active.positive.total	double	kW	Positive active maximum demand (A+) total
.report.max_demand.active.positive.t1	double	kW	Positive active maximum demand (A+) in tariff T1
.report.max_demand.active.positive.t2	double	kW	Positive active maximum demand (A+) in tariff T2
.report.max_demand.active.negative.total	double	kW	Negative active maximum demand (A-) total
.report.max_demand.active.negative.t1	double	kW	Negative active maximum demand (A-) in tariff T1
.report.max_demand.active.negative.t2	double	kW	Negative active maximum demand (A-) in tariff T2
.report.power_factor	double		Instantaneous power factor
.report.conv_factor	uint		Conversion coefficient, the value of the integer by which instantaneous power, energy, current, max demand will be multiplied
.meter.status	string	NOT CONNECTED NO DATA RECOGNITION OK ENCRYPTION KEY KEY REQUIRED NOT RECOGNIZED	Enumerated string
.meter.interface	string	P1 TTL MBUS	Physical interface
.meter.protocol	string	DSMR DLMS KMP MEP	Logical interface
.meter.protocol_version	string		Communication protocol version
.meter.logical_name	string		Meter logical name
.meter.id	string		Meter identifier, usually serial number
.meter.model	string		Meter model in case it could be identified
.meter.vendor	string		Meter supplier name, if identifiable
.meter.prefix	string		3 letter vendor prefixes
.system.id	string		whatwatt Go unique identifier

Field	Type	Unit	Description
.system.date_time	ISO8601		Local date time for time zone
.system.boot_id	string		Random string generated after each reboot
.system.time_since_boot	double	s	Time in second since boot

3.2. Streaming method

Endpoint	api/v1/live
Method	GET
Response content type	text/event-stream

To use this method, you must specify connection/keep-alive in the request.
Data will be sent at the same frequency as the meter sends reports.

Note – You can only establish one connection to this endpoint, if disconnected reconnect.

Example event stream data. Measurement data is sent in a live event.

```
event: live
data:
{"P_In":0.036,"P_Out":0,"P_P1_In":0,"P_P2_In":0,"P_P3_In":0,"P_P1_Out":0,"P_P2_Out"
:0,"P_P3_Out":0,"P_P_In":0,"P_P_In_T1":0,"P_P_In_T2":0,"I_P1":0,"I_P2":0,"I_P3":0,"
V_P1":0,"V_P2":0,"V_P3":0,"rP_In":0,"rP_Out":0,"PF":0,"E_In":47.251,"E_In_T1":33.38
8,"E_In_T2":14.668,"E_Out":8.965,"E_Out_T1":7.868,"E_Out_T2":1.097,"rE_In":0,"rE_Ou
t":0,"Date":"2024-08-24","Time":"14:34:00","Uptime":71.05}
```

In case of this API all fields are always sent, even if the meter does not send such value, then the field value is zero or empty string.

Fields in data object

Field	Type	Unit	Description
P_In	double	kW	Positive active instantaneous power (A+)
P_Out	double	kW	Negative active instantaneous power (A-)
P_P1_In	double	kW	Positive active instantaneous power (A+) in phase L1
P_P2_In	double	kW	Positive active instantaneous power (A+) in phase L2
P_P3_In	double	kW	Positive active instantaneous power (A+) in phase L3
P_P1_Out	double	kW	Negative active instantaneous power (A-) in phase L1
P_P2_Out	double	kW	Negative active instantaneous power (A-) in phase L2
P_P3_Out	double	kW	Negative active instantaneous power (A-) in phase L3
P_P_In	double	kW	Positive active maximum demand (A+) total
P_P_In_T1	double	kW	Positive active maximum demand (A+) in tariff T1
P_P_In_T2	double	kW	Positive active maximum demand (A+) in tariff T2

Field	Type	Unit	Description
I_P1	double	A	Instantaneous current (I) in phase L1
I_P2	double	A	Instantaneous current (I) in phase L2
I_P3	double	A	Instantaneous current (I) in phase L1
V_P1	double	V	Instantaneous voltage (U) in phase L1
V_P2	double	V	Instantaneous voltage (U) in phase L2
V_P3	double	V	Instantaneous voltage (U) in phase L3
rP_In	double	kvar	Positive reactive instantaneous power (Q+)
rP_Out	double	kvar	Negative reactive instantaneous power (Q-)
PF	double		Instantaneous power factor
E_In	double	kWh	Positive active energy (A+) total
E_In_T1	double	kWh	Positive active energy (A+) in tariff T1
E_In_T2	double	kWh	Positive active energy (A+) in tariff T2
E_Out	double	kWh	Negative active energy (A-) total
E_Out_T1	double	kWh	Negative active energy (A-) in tariff T1
E_Out_T2	double	kWh	Negative active energy (A-) in tariff T2
rE_In	double	kvarh	Positive reactive energy (Q+) total
rE_Out	double	kvarh	Negative reactive energy (Q-) total
Date	string	Y-m-d	State the date in the format year-month-day based on local time
Time	string	H:M:S	provide the time in local format as hour:minute:second
Uptime	double	hour	System uptime

4. Integration over MQTT client

The device can be connected to the MQTT broker using the built-in MQTT client. The MQTT client supports unencrypted and encrypted connections.

Note – whatwatt Go will only publish values if a meter is connected.

HTTP REST API to configure MQTT client (you can also do it from the device's WebUI)

Endpoint	api/v1/mqtt/settings
Method	GET, POST, PUT
Response content type	application/json

MQTT settings object description

Field	Type	Default	Range	Remarks
.enable	boolean	false		Enables or disables the MQTT client.
.url	string	empty string	0..127, should start with mqtt:// or mqtts://	mqtt - is for not encrypted TCP based connections mqtts - if for encrypted TLS based connections
.username	string	empty string	0..127	The username is a unique identifier for the client, allowing the broker to manage and control access levels.
.password	string	empty string	0..127	This field is not returned on read (GET), instead the password_len field is returned which contains the length of the password, zero if a password is not set.
.client_id	string	empty string	0..63	The client ID is a unique identifier assigned to each client connecting to the MQTT broker. It is used to identify the client and manage its connection state. The client ID must be unique for each client connected to the same broker; otherwise, the broker will disconnect the existing client with the same ID.
.skip_cn_check	boolean	false		Skips server certificate Common Name validation.
.publish.topic	string	empty string	0..127	An MQTT publish topic is a string that the MQTT client uses to identify where to send messages.
.publish.template	string	empty string	0..1023	The published payload template.
.broker.certificate	string	null		Custom broker certificate (in section 5.4 the filename is ca.crt). Should be specified in PEM format.
.client.certificate	string	null		Client certificate for TLS mutual authentication (in section 5.4 the filename is whatwatt.crt), not required if mutual authentication is not needed. Must be provided with client.key. Should be specified in PEM format.

Field	Type	Default	Range	Remarks
.client.key	string	null		Client private key for TLS mutual authentication (in section 5.4 the filename is whatwatt.key), not required if mutual authentication is not needed. This field is not returned on read (GET), instead the key_len field is returned which contains the length of the certificate, zero if a password is not set.
.client.key_password	string	null	0..255	Client key decryption password. Required only in case when client key is protected with password. This field is not returned on read (GET), instead the key_password_len field is returned which contains the length of the key password, zero if no password is set.

Note – The maximum size of the set JSON cannot exceed 8 kB.

Example response – read by GET method

```
{
  "enable":true,
  "url":"mqtts://akenza.io",
  "username":"akenza",
  "client_id":"whatwatt",
  "skip_cn_check":false,
  "publish":{
    "topic":"test",
    "template": "{\n\t\"P_In\": ${1_7_1},\n\t\"P_Out\": ${2_7_1},\n\t\"E_In\":
${1_8_0},\n\t\"E_Out\": ${2_8_0},\n\t\"Meter\": {\n\t\t\"DateTime\": \"$
{meter.date_time}\"\n\t},\n\t\"Sys\": {\n\t\t\"Id\": \"${sys.id}\"\n\t}\n}"
  },
  "password_len": 0
}
```

To set the value, send the same object as you received, add the password field if required. To replace the entire configuration use the POST method, to update the configuration, e.g. by sending one field, use the PUT method (then the password will not be deleted). If you want to delete any of the certificate, key, key_password fields set the field value to null in JSON.

It is also possible to set the period with which reports will be sent via MQTT, the default period is 30s but if the meter sends reports less often it will be longer. You can change this setting via WebUI in the **System > Interval to Systems** section.

4.1.1. Template Description

The message published by the client is defined using a template, the message format can be anything but always text.
You can embed variables in the template, these can both be measurement and system variables.
A variable in a template is embedded in a section starting with a dollar sign, followed by an opening curly brace, the variable name, and a closing curly brace: **${variable_name}.**

The available variables are predefined, at this time you cannot define your own.

Note – If a variable appears in the template that is not resolved, then the variable will not be replaced and the entire ${some_undefined_variable} string will be in the output message. Be careful when integrating with your system, it may happen that one of the variables is not sent by the meter, the variable will take the value null in case of a numeric variable or an empty string in case of a string variable. Some meters send values alternately and after power on the device, they will be determined only after some time, during this time some variables may not be correctly substituted.

Example template

```
{
  "P_In": 0,
  "P_Out": 4.286,
  "E_In": 87.104,
  "E_Out": 50.369,
  "Meter": {
    "DateTime": "2025-03-21T10:54:02Z"
  },
  "Sys": {
    "Id": "000000000000"
  }
}
```

Note – This is not valid JSON. The JSON will only be valid when all variables are resolved. Also note that some variables are always a number (written as text) and some are text, in the JSON payload you need to enclose text variables in quotes.

This template will generate a message in JSON format after substituting variables

```
{
  "sys_id":"A842E39F8124",
  "meter_id":"636192",
  "time":1725281386,
  "tariff":0,
  "power_in":1.1,
  "power_out":0,
  "energy_in":123.4,
  "energy_out":0
}
```

Possible variables

The first column Name usually refers to short OBIS form part C.D.E. Keep in mind that the meter does not send all fields. The value returned for a network interface depends on which one is connected.

Name	Type	Unit	Description
1_8_0	double	kWh	Positive active energy (A+) total
energy.in	double	kWh	Positive active energy (A+) total
1_8_1	double	kWh	Positive active energy (A+) in tariff T1
1_8_2	double	kWh	Positive active energy (A+) in tariff T2
2_8_0	double	kWh	Negative active energy (A-) total
energy.out	double	kWh	Negative active energy (A-) total
2_8_1	double	kWh	Negative active energy (A-) in tariff T1
2_8_2	double	kWh	Negative active energy (A-) in tariff T2
3_8_0	double	kvarh	Positive reactive energy (Q+) total
3_8_1	double	kvarh	Positive reactive energy (Q+) in tariff T1
3_8_2	double	kvarh	Positive reactive energy (Q+) in tariff T2
4_8_0	double	kvarh	Negative reactive energy (Q-) total
4_8_1	double	kvarh	Negative reactive energy (Q-) in tariff T1
4_8_2	double	kvarh	Negative reactive energy (Q-) in tariff T2
5_8_0	double	kvarh	Imported inductive reactive energy in 1-st quadrant (Q1) total
5_8_1	double	kvarh	Imported inductive reactive energy in 1-st quadrant (Q1) in tariff T1

Name	Type	Unit	Description
5_8_2	double	kvarh	Imported inductive reactive energy in 1-st quadrant (Q1) in tariff T2
6_8_0	double	kvarh	Imported capacitive reactive energy in 2-nd quadrant (Q2) total
6_8_1	double	kvarh	Imported capacitive reactive energy in 2-nd quadr. (Q2) in tariff T1
6_8_2	double	kvarh	Imported capacitive reactive energy in 2-nd quadr. (Q2) in tariff T2
7_8_0	double	kvarh	Exported inductive reactive energy in 3-rd quadrant (Q3) total
7_8_1	double	kvarh	Exported inductive reactive energy in 3-rd quadrant (Q3) in tariff T1
7_8_2	double	kvarh	Exported inductive reactive energy in 3-rd quadrant (Q3) in tariff T2
8_8_0	double	kvarh	Exported capacitive reactive energy in 4-th quadrant (Q4) total
8_8_1	double	kvarh	Exported capacitive reactive energy in 4-th quadr. (Q4) in tariff T1
8_8_2	double	kvarh	Exported capacitive reactive energy in 4-th quadr. (Q4) in tariff T2
1_6_0	double	kW	Positive active maximum demand (A+) total
1_6_1	double	kW	Positive active maximum demand (A+) in tariff T1
1_6_2	double	kW	Positive active maximum demand (A+) in tariff T2
2_6_0	double	kW	Negative active maximum demand (A-) total
2_6_1	double	kW	Negative active maximum demand (A-) in tariff T1
2_6_2	double	kW	Negative active maximum demand (A-) in tariff T2
1_7_0	double	kW	Positive active instantaneous power (A+)
power.in	double	kW	Positive active instantaneous power (A+)
21_7_0	double	kW	Positive active instantaneous power (A+) in phase L1
41_7_0	double	kW	Positive active instantaneous power (A+) in phase L2
61_7_0	double	kW	Positive active instantaneous power (A+) in phase L3
2_7_0	double	kW	Negative active instantaneous power (A-)
power.out	double	kW	Negative active instantaneous power (A-)
22_7_0	double	kW	Negative active instantaneous power (A-) in phase L1
42_7_0	double	kW	Negative active instantaneous power (A-) in phase L2
62_7_0	double	kW	Negative active instantaneous power (A-) in phase L3
3_7_0	double	kvar	Positive reactive instantaneous power (Q+)
23_7_0	double	kvar	Positive reactive instantaneous power (Q+) in phase L1
43_7_0	double	kvar	Positive reactive instantaneous power (Q+) in phase L2
63_7_0	double	kvar	Positive reactive instantaneous power (Q+) in phase L3
4_7_0	double	kvar	Negative reactive instantaneous power (Q-)
24_7_0	double	kvar	Negative reactive instantaneous power (Q-) in phase L1
44_7_0	double	kvar	Negative reactive instantaneous power (Q-) in phase L2
64_7_0	double	kvar	Negative reactive instantaneous power (Q-) in phase L3
9_7_0	double	kVA	Apparent instantaneous power (S+)
31_7_0	double	A	Instantaneous current (I) in phase L1
51_7_0	double	A	Instantaneous current (I) in phase L1

Name	Type	Unit	Description
71_7_0	double	A	Instantaneous current (I) in phase L3
s31_7_0	double	A	Signed instantaneous current (I) in phase L1
s51_7_0	double	A	Signed instantaneous current (I) in phase L2
s71_7_0	double	A	Signed instantaneous current (I) in phase L3
32_7_0	double	V	Instantaneous voltage (U) in phase L1
52_7_0	double	V	Instantaneous voltage (U) in phase L2
72_7_0	double	V	Instantaneous voltage (U) in phase L3
13_7_0	double		Instantaneous power factor
tariff	uint		1, 2
conv_factor	double		Conversion coefficient, the value of the integer by which instantaneous power, energy, current, max demand will be multiplied
timestamp	uint		UTC UNIX timestamp
meter.date_time	string	ISO8601	Report date time in local time
meter.id	string		Meter ID
meter.type	string		Meter type
meter.vendor	string		Meter vendor
meter.model	string		Model of meter
meter.interface	string	P1 TTL MBUS MEP	
meter.protocol	string	DSMR DLMS KMP MEP	
meter.protocol_ver	string		Meter protocol version
meter.status	string	NOT CONNECTED NO DATA RECOGNITION OK ENCRYPTION KEY KEY REQUIRED NOT RECOGNIZED	Status ENCRYPTION KEY meaning that something wrong is with encryption key, it can be incorrect or additional key is needed.
sys.name	string		Name of device (can be set WebUI)
sys.id	string		WhatWhat Go system identifier
sys.firmware	string		Firmware version
sys.date_time	string	ISO8601	System local time

Name	Type	Unit	Description
plug.interface	string	NONE P1 TTL MBUS MEP	Physical interface connected to the device
plug.voltage.usb	double	V	
plug.voltage.p1	double	V	
plug.voltage.mbus	double	V	
wifi.mode	string	off sta ap apsta nan	Wi-Fi operation mode
wifi.sta.status	string	disabled disconnected error ok	Connection status
wifi.sta.ssid	string		Name of AP network - name of network where device is connected
wifi.sta.bssid	string	MAC	MAC address of AP
wifi.sta.rssi	int	dBm	WiFi received signal strength indication
wifi.sta.channel	uint	1-13	Wi-Fi channel
wifi.sta.mac	string	MAC	MAC address of Wi-Fi interface
wifi.sta.ip	string	IPv4	IPv4 address assigned to Wi-Fi interface
wifi.sta.mask	string	IPv4	IPv4 netmask assigned to Wi-Fi interface
wifi.sta.gw	string	IPv4	IPv4 gateway address assigned to Wi-Fi interface
wifi.sta.dns	string	IPv4	IPv4 DNS server assigned to Wi-Fi interface
eth.state	string	up down	Status of Ethernet
eth.mac	string	MAC	MAC address of Ethernet interface
eth.ip	string	IPv4	IPv4 address assigned to Ethernet interface
eth.mask	string	IPv4	IPv4 netmask assigned to Ethernet interface
eth.gw	string	IPv4	IPv4 gateway address assigned to Ethernet interface
eth.dns	string	IPv4	IPv4 DNS server assigned to Ethernet interface

4.1.2. Reading variables locally

Endpoint	api/v1/variables
Method	GET
Response content type	application/json

Example result

```
[
  {
    "sys.name":""
  },
  {
    "sys.id":"A842E39F8124"
  },
  {
    "sys.firmware":"1.2.15"
  },
  {
    "sys.date_time":"2025-02-20T17:49:45Z"
  },
  {
    "1_8_0":"24.087"
  },
  {
    "energy.in":"24.087"
  },
  {
    "1_8_1":"null"
  },
  {
    "1_8_2":"null"
  },
  {
    "2_8_0":"0.004"
  },
  {
    "energy.out":"0.004"
  },
  {
    "2_8_1":"null"
  },
  {
    "2_8_2":"null"
  },
  {
    "3_8_0":"10.505"
  },
  {
    "3_8_1":"null"
  },
  {
    "3_8_2":"null"
  },
  {
    "4_8_0":"15.385"
  },
  {
    "4_8_1":"null"
  },
  {
    "4_8_2":"null"
  },
  {
    "5_8_0":"null"
  },
  {
    "5_8_1":"null"
  },
  {
    "5_8_2":"null"
  },
  {
    "6_8_0":"null"
  },
  {
    "6_8_1":"null"
  },
  {
    "6_8_2":"null"
  },
  {
    "7_8_0":"null"
  },
  {
    "7_8_1":"null"
  },
  {
    "7_8_2":"null"
  },
  {
    "8_8_0":"null"
  },
  {
    "8_8_1":"null"
  },
  {
    "8_8_2":"null"
  },
  {
    "1_6_0":"null"
  },
  {
    "1_6_1":"null"
  },
  {
    "1_6_2":"null"
  },
  {
    "2_6_0":"null"
  },
  {
    "2_6_1":"null"
  },
  {
    "2_6_2":"null"
  },
  {
    "1_7_0":"0.006"
  },
  {
    "power.in":"0.006"
  },
  {
    "21_7_0":"0.006"
  },
  {
    "41_7_0":"0"
  },
  {
    "61_7_0":"0"
  },
  {
    "2_7_0":"0"
  },
  {
    "power.out":"0"
  },
  {
    "22_7_0":"0"
  },
  {
    "42_7_0":"0"
  },
  {
    "62_7_0":"0"
  },
  {
    "3_7_0":"null"
  },
  {
    "23_7_0":"0"
  },
  {
    "43_7_0":"0"
  },
  {
    "63_7_0":"0"
  },
  {
    "4_7_0":"null"
  },
  {
    "24_7_0":"0.007"
  },
  {
    "44_7_0":"0"
  },
  {
    "64_7_0":"0"
  },
  {
    "9_7_0":"null"
  },
  {
    "31_7_0":"0.06"
  },
  {
    "51_7_0":"0"
  },
  {
    "71_7_0":"0"
  },
  {
    "32_7_0":"233"
  },
  {
    "52_7_0":"0"
  },
  {
    "72_7_0":"0"
  },
  {
    "13_7_0":"null"
  },
  {
    "tariff":"2"
  },
  {
    "conv_factor":"1"
  },
  {
    "timestamp":"1735542122"
  },
  {
    "meter.date_time":"2024-12-30T08:02:02Z"
  },
  {
    "meter.id":""
  },
  {
    "meter.type":"LGZ1030662444349"
  },
  {
    "meter.vendor":"Landis+Gyr"
  },
  {
    "meter.model":""
  },
  {
    "meter.interface":"MBUS"
  },
  {
    "meter.protocol":"DLMS"
  },
  {
    "meter.protocol_ver":""
  },
  {
    "meter.status":"OK"
  },
  {
    "plug.interface":"MBUS"
  },
  {
    "plug.voltage.usb":"5.272"
  },
  {
    "plug.voltage.p1":"2.844"
  },
  {
    "plug.voltage.mbus":"10.952"
  },
  {
    "wifi.mode":"sta"
  },
  {
    "wifi.sta.status":"ok"
  },
  {
    "wifi.sta.ssid":"sjj"
  },
  {
    "wifi.sta.bssid":"DC15C84FBAB6"
  },
  {
    "wifi.sta.rssi":"-30"
  },
  {
    "wifi.sta.channel":"13"
  },
  {
    "wifi.sta.mac":"A842E39F8124"
  },
  {
    "wifi.sta.ip":"192.168.99.176"
  },
  {
    "wifi.sta.mask":"255.255.255.0"
  },
  {
    "wifi.sta.gw":"192.168.99.1"
  },
  {
    "wifi.sta.dns":"0.0.0.0"
  }
]
```

5. Secure MQTT Integration

5.1. Scope

This document shows how to connect a WhatWatt Go energy meter to a local Mosquitto 2.0.21 broker using ECDSA certificates, mutual-TLS, and IP addresses (no hostname validation). All certificates are stored in /etc/mosquitto/certs/. Optional production-grade notes are included at the end.

5.2. Prerequisites

Component	Version / Notes
Lubuntu 25.04	fresh install, sudo user
Mosquitto	2.0.21 (apt repo)
OpenSSL	3.x (included)
WhatWatt Go	firmware ≥ 1.7.6, REST API enabled
Python	≥ 3.8 (for provisioning script)

5.3. Install Mosquitto & clients

```
sudo apt update
sudo apt install --yes mosquitto mosquitto-clients openssl
Mosquitto starts automatically; verify:
systemctl status mosquitto
```

5.4. Generate an ECDSA-P256 Certificate Chain

```
#Root CA
openssl ecparam -name prime256v1 -genkey -noout -out ca.key
openssl req -x509 -new -key ca.key -sha256 -days 3650 \
  -subj "/C=US/O=Lab/OU=IoT/CN=Lab ECC Root CA" \
  -out ca.crt

#Mosquitto server cert (CN may stay mqtt.lab.local)
openssl ecparam -name prime256v1 -genkey -noout -out server.key
openssl req -new -key server.key \
  -subj "/C=US/O=Lab/OU=IoT/CN=mqtt.lab.local" \
  -out server.csr
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key \
  -CAcreateserial -sha256 -days 365 -out server.crt

#Client cert for the meter
openssl ecparam -name prime256v1 -genkey -noout -out whatwatt.key
openssl req -new -key whatwatt.key \
  -subj "/C=US/O=Lab/OU=Metering/CN=whatwatt-001" \
  -out whatwatt.csr
openssl x509 -req -in whatwatt.csr -CA ca.crt -CAkey ca.key \
  -CAcreateserial -sha256 -days 365 -out whatwatt.crt
```

5.5. Deploy Certificates

```
sudo mkdir -p /etc/mosquitto/certs
sudo cp {ca.crt,server.crt,server.key} /etc/mosquitto/certs/
sudo chown root:mosquitto /etc/mosquitto/certs/{*.crt,*.key}
sudo chmod 640              /etc/mosquitto/certs/{*.crt,*.key}
```

5.6. Configure Mosquitto (/etc/mosquitto/conf.d/tls.conf)

```
listener 8883
protocol mqtt

cafile   /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile  /etc/mosquitto/certs/server.key

require_certificate true
use_identity_as_username true
allow_anonymous false
# No tls_version / ciphers override – broker auto-negotiates (TLS 1.3 preferred).
```

Restart and watch the log:

```
sudo systemctl restart mosquitto
journalctl -u mosquitto -f
```

5.7. Quick Local Subscription (CLI)

```
mosquitto_sub -h 127.0.0.1 -p 8883 -v \
-cafile ca.crt \
-cert   whatwatt.crt \
-key    whatwatt.key \
-insecure \
t 'lab/energy/#'
```

--insecure skips the CN check, matching the device behaviour.

5.8. Provision WhatWatt Go (JSON payload)

Sample file mqtt.json (≤ 8 kB):

```
{
  "enable": true,
  "url": "mqtts://192.168.99.186:8883",
  "skip_cn_check": true,
  "client_id": "whatwatt-001",
  "publish": {
    "topic": "lab/energy/whatwatt-001",
    "template": "{ \"P_In\": ${1_7_0}, \"P_Out\": ${2_7_0} }"
  },
  "broker": { "certificate": "-----BEGIN CERTIFICATE-----\n...ca...\n-----END
CERTIFICATE-----\n" },
  "client": {
    "certificate": "-----BEGIN CERTIFICATE-----\n...whatwatt...\n-----END
CERTIFICATE-----\n",
    "key": "-----BEGIN EC PRIVATE KEY-----\n...key...\n-----END EC PRIVATE KEY-----
\n"
  }
}
```

Upload:

```
curl -X POST http://192.168.99.176/api/v1/mqtt/settings \
     -H "Content-Type: application/json" \
     --data-binary @mqtt.json
```

5.9. Parameterised Provisioning Script (setup_mqtt.py)

```
#!/usr/bin/env python3
"""Configure a WhatWatt Go device over REST.
Example:
  python setup_mqtt.py --device 192.168.99.176 --broker 192.168.99.186 \
         --topic lab/energy/whatwatt-001 --id whatwatt-001
"""
import argparse, pathlib, requests, json, sys

def args():
    p = argparse.ArgumentParser()
    p.add_argument("--device", default="192.168.99.176")
    p.add_argument("--broker", default="192.168.99.186")
    p.add_argument("--port", type=int, default=8883)
    p.add_argument("--id",      default="whatwatt-001")
    p.add_argument("--topic",   default="lab/energy/whatwatt-001")
    p.add_argument("--template", default='{"P_In": ${1_7_0}, "P_Out": ${2_7_0}}')
    p.add_argument("--ca",    default="ca.crt")
    p.add_argument("--cert", default="whatwatt.crt")
    p.add_argument("--key",  default="whatwatt.key")
    return p.parse_args()

a = args()
api = f"http://{a.device}/api/v1/mqtt/settings"

data = {
    "enable": True,
    "url": f"mqtts://{a.broker}:{a.port}",
    "skip_cn_check": True,
    "client_id": a.id,
    "publish": {"topic": a.topic, "template": a.template},
    "broker": {"certificate": pathlib.Path(a.ca).read_text()},
    "client": {
        "certificate": pathlib.Path(a.cert).read_text(),
        "key": pathlib.Path(a.key).read_text()
    }
}
try:
    r = requests.post(api, json=data, timeout=10)
    r.raise_for_status()
    print("Success:", json.dumps(r.json(), indent=2))
except (requests.RequestException, OSError) as e:
    sys.exit(f"Error: {e}")

Mark executable (chmod +x setup_mqtt.py) and run with desired arguments.
```

5.10. Production Hardening

- Hostname validation – issue server cert with SAN `DNS:broker.example.com; remove skip_cn_check & --insecure.`
- Unique client certificates per meter → granular ACLs + easy revocation.
- Security Level 2 default is fine; you may limit ciphers to GCM/CHACHA suites only.
- Automate certificate rotation (yearly) via your PKI or tools like CFSSL.
- Restrict broker to trusted VLAN / firewall ingress to TCP 8883.

6. Settings

6.1. General convention for using local REST API

The local REST API allows you to read, overwrite, update, or delete values. Typically, the body is returned and processed in JSON format.

The activity is determined by the HTTP method used in the API request

- The **GET** method allows you to read the value.
- The **POST** method overwrites all values; if no value is sent in JSON, it defaults to the same value as if the device was restored to factory settings.
- The **PUT** method allows you to update selected fields; the JSON you upload should contain the fields you selected.
- The **DELETE** method deletes all configuration for the selected endpoint and is only available for selected endpoints.

In addition, the use of a local REST API follows the general conventions for the use of local APIs, which ensures compatibility and optimal communication performance between different components of the system.

Execution of each method returns an HTTP status code

- 200 success along with the response, usually in JSON format,
- 204 execution without a return message,
- 400 bad request, invalid parameters, ranges in JSON object,
- 401 unauthorized,
- 404 endpoint not found, disabled or unavailable,
- 500 internal problem of the device,
- 503 service unavailable.

6.2. Service management and basic settings

This endpoint enables management of services and basic system settings.

HTTP REST API to configure services (you can also do it from the device's WebUI):

Endpoint	api/v1/settings
Method	GET, POST, PUT
Response content type	application/json

Settings JSON object description

Field	Type	Default	Range	Remarks
.system.name	string	empty string	0..31	System name, device name, name given by the user
.system.host_name	string	whatwatt_XXXXXX, where X is the last 6 digits of the device ID	0..31	Hostname as seen on the network
.system.protection	boolean	false		Specifies whether the password for the HTTP server, and thus the REST API and local WebUI, should be required

Field	Type	Default	Range	Remarks
.system.password	string	empty string	0..31	HTTP server password. The password is not returned when reading this API
.services.cloud.what_watt	boolean	true		Enables/disables whatwatt (cloud connection)
.services.cloud.solar_manager	boolean	false		Enables/disables the Solar Manager service (cloud connection)
.services.cloud.mystrom	boolean	true		Enables/disables myStrom (cloud connection)
.services.local.solar_manager	boolean	true		Enables/disables the local Solar Manager integration API
.services.broadcast	boolean	true		Enables/disables mDNS broadcasting
.services.report_interval	uint	30		Frequency of reporting to the cloud, currently only applies to custom MQTT integration
.services.sd.frequency	uint	15	1..1440	Frequency in seconds of writing reports to the SD card
.services.sd.enable	boolean	false		Enable/disable saving reports to SD card
.services.modbus.enable	boolean	false		enables/disables the Modbus TCP Slave Server
.services.modbus.port	uint	502	1..65535	port on which the Modbus TCP Slave Server is to run

*Not described fields are not currently supported

Example response on GET request

```
{
  "system":{
    "name":"",
    "host_name":"whatwatt_9F8124",
    "protection":false
  },
  "services":{
    "cloud":{
      "what_watt":true,
      "solar_manager":false,
      "mystrom":false
    },
    "local":{
      "solar_manager":false
    },
    "broadcast":true,
    "other_energy_provider":false,
    "report_interval":30,
    "log":true,
    "meter_proxy":false,
    "sd":{
      "frequency":15,
      "enable":false
    }
  }
}
```

6.3. Meter communication settings

This endpoint allows you to set the parameters of communication with the meter. Mainly the settings of the serial port and the physical layer of communication.

HTTP REST API to configure meter port (you can also do it from the device's WebUI):

Endpoint	api/v1/meter/settings
Method	GET, POST, PUT
Response content type	application/json

Meter settings JSON object description

Field	Type	Default	Range	Remarks
.baudrate	uint	115200	300..115200	Data transfer rate over serial interface
.parity	string	none	none odd even	Parity control of data transmitted over the serial interface
.stop_bits	string	1	1 1.5 2	Number of stop bits in serial transmission
.encryption	boolean	false		Enable data encryption at the logical layer
.encryption_key	string	empty string	32..32 hexadecimal characters	Data encryption key. The key value is not returned when reading this API. If you want to reset the key, set the field to 32 zeros
.authentication_key	string	empty string	32..32 hexadecimal characters	The meter authorization key on whatwatt device is optional. The key is not returned when reading the API. If you want to reset it, set it to 32 zeros
.tx_invert	bool	false		Reverse the polarity of the transmission data line in the serial interface
.rx_invert	bool	false		Reverse the polarity of the data receiving line in the serial interface
.auto_baudrate	bool	true		Automatic serial port settings. The whatwatt device will try to determine the serial port settings such as baudrate, parity, stop_bits, tx_invert, rx_invert based on the electrical parameters of the serial interface
.if_type	string	auto	auto p1 mbus ttl mep	Physical interface of the meter connected to the whatwatt device. With the auto option selected, the whatwatt device will try to automatically determine the type of interface, and thus the serial port settings and the protocol used on the logical layer
.conv_factor	uint	1	1..10000	Conversion coefficient, the value of the integer by which instantaneous power, energy, current, max demand will be multiplied
.time_offset	int	0	int32	Meter time correction, if the meter has an incorrect time, it is possible to set an offset in seconds to correct the time in reports processed by the device
.sync_time_offset_with_ntp	bool	false		NTP time-based time_offset auto-following. The time offset value is automatically adjusted. Note that by disabling this option, the time_offset will not be restored to the previous value or zero and must be set separately

Field	Type	Default	Range	Remarks
.scalers_set	string	default lge570 custom		Allows you to select the scaling scheme for the values sent by the meter, this applies to COSEM class 3 objects. The set includes different scales for different logical names (OBIS). A default value will cause the device to attempt to automatically adjust the scale, in some cases it will default to scaling in the remaining cases. The lge570 value is a predefined scale set for the L&G E570. The custom set allows you to set your own scaling for individual values

Example response on GET request

```
{
  "baudrate":115200,
  "parity":"none",
  "stop_bits":"1",
  "buffer_size":64,
  "method":"Passive Push",
  "encryption":false,
  "rx_invert":false,
  "tx_invert":false,
  "auto_baudrate":true,
  "if_type":"auto",
  "conv_factor":1,
```

Setting	Expected value input	Conversion & Storage Rules
hex	A hexadecimal string whose length is an even number of characters.	The string is parsed byte-by-byte and the resulting raw bytes are written sequentially, starting at the base register address. Example: 0809A0B0
short	A 16-bit integer (signed or unsigned).	The integer is converted to big-endian byte order and stored in one register (16 bits). Example: 1234 or -1234
int	A 32-bit integer (signed or unsigned).	Converted to big-endian; the two most-significant bytes are written to the base register, the two least-significant bytes to the next register (address + 1).
float	A 32-bit floating-point number.	Encoded as IEEE-754, big-endian, and stored in two consecutive registers. Example: -1.23
double	A 64-bit floating-point number.	Encoded as IEEE-754, big-endian, and stored in four consecutive registers.

```
  "time_offset":0,
  "sync_time_offset_with_ntp":false,
  "scalers_set":"default"
```

Modbus Function Code	Guidance for type and value
5 – Write Single Coil	Use short with 0 / 1, or use hex with 0000 (OFF) or 00FF (ON). Only one register is written.
6 – Write Single Register	Same approach as Function 5: choose short (integer) or hex (exact two-byte value). Exactly one register is written.
15 – Write Multiple Coils	Use hex notation. The number of registers written depends on the length of the hex string: 4 hex characters = 1 register. Example: an 8-character string (A1B2C3D4) writes two registers.
16 – Write Multiple Registers	You may supply hex or any numeric type (short, int, long, float, double). The data are automatically expanded across as many consecutive registers as required by the chosen type.
double	A 64-bit floating-point number.

```
}
```

6.4. Reading of currently applied scaler values

This endpoint allows you to read individual scale values for specific register values that the meter transmits. This API uses a shortened convention of logical name/OBIS, which is in the format C.D.E.

The present value is the value that has been determined by the device itself at the detection stage, covered/overridden by a set of scalers selected by the scalers_set (note that the scalers_set does not have to cover all values).

Currently, the current value of scale factors can only be read for meters that exchange data in DLMS format.

Scale factors apply only to COSEM Class 3 objects.

Endpoint	api/v1/meter/scalers/current
Method	GET
Response content type	application/json
Avalliable since FW version	1.2.15

Current scalers JSON object description

Field	Type	Remarks
[].obis	string	The field value is a shorthand representation of the OBIS notation representing the C.D.E part. Only those OBIS codes that have been recognized by the device and belong to COSEM class 3 are included in the response.
[].scaler	int	The value of the field determines the currently applied scaling factor for the OBIS code. The scaled factor is expressed as a power of ten (10^scaler). For example, a value of -3 means that the output value from the meter is multiplied by 0.001, and a value of 3 means multiplication by 1000.

Example response on GET request

```
[
  {
    "obis":"1.8.0",
    "scaler":0
  },
  {
    "obis":"2.8.0",
    "scaler":0
  },
  {
    "obis":"3.8.0",
    "scaler":0
  },
  {
    "obis":"4.8.0",
    "scaler":0
  },
  {
    "obis":"1.7.0",
    "scaler":0
  },
  {
    "obis":"2.7.0",
    "scaler":0
  },
  {
    "obis":"32.7.0",
    "scaler":0
  },
  {
    "obis":"52.7.0",
    "scaler":0
  },
  {
    "obis":"72.7.0",
    "scaler":0
  },
  {
    "obis":"31.7.0",
    "scaler":-2
  },
  {
    "obis":"51.7.0",
    "scaler":-2
  },
  {
    "obis":"71.7.0",
    "scaler":-2
  },
  {
    "obis":"21.7.0",
    "scaler":-2
  },
  {
    "obis":"41.7.0",
    "scaler":-2
  },
  {
    "obis":"61.7.0",
    "scaler":-2
  },
  {
    "obis":"22.7.0",
    "scaler":-2
  },
  {
    "obis":"42.7.0",
    "scaler":-2
  },
  {
    "obis":"62.7.0",
    "scaler":-2
  },
  {
    "obis":"23.7.0",
    "scaler":-2
  },
  {
    "obis":"43.7.0",
    "scaler":-2
  },
  {
    "obis":"63.7.0",
    "scaler":-2
  },
  {
    "obis":"24.7.0",
    "scaler":-2
  },
  {
    "obis":"44.7.0",
    "scaler":-2
  },
  {
    "obis":"64.7.0",
    "scaler":-2
  }
]
```

6.5. Meter custom scalers settings

This endpoint allows for the setting of individual scaling values for specific register values transmitted by the meter. This API uses a shortened convention of logical name/OBIS, which is in the format C.D.E.

Currently, these settings operate exclusively for messages in the DLMS format.

Scale factors apply only to COSEM Class 3 objects.

HTTP REST API to configure individual scalers (you can also do it from the device's WebUI):

Endpoint	api/v1/meter/scalers/custom
Method	GET, POST
Response content type	application/json

Field	Type	Range	Remarks
[].obis	string	0..254.0..254.0..255	This field should be formatted to represent the C.D.E part of the logical OBIS code. For example, it can be 1.8.0. If we want to apply the scaler for all E values, such as in the case of tariffs, we can write 1.8.255. The value '255' indicates that the scaler will be applied to all E values. Only OBIS codes belonging to COSEM class 3 should be defined.
[].scaler	int	-6..6	This field should specify the scaler value, which is expressed in powers of ten. For example, a value of -3 will multiply the register value by 0.001, and a value of 3 will multiply it by 1000.

Example response on GET request

```
[
  {
    "obis":"1.7.0",
    "scaler":-1
  },
  {
    "obis":"2.7.0",
    "scaler":-1
  }
]
```

When performing the settings using the POST method, the same JSON format must be used.

6.6. Actions

This API provides a structured way to define and execute automated actions consisting of HTTP/ Modbus requests. The execution mechanism ensures asynchronous processing, with built-in mechanisms for monitoring execution status. The API is designed for automation, integration, and remote control of devices or services via HTTP/Modbus requests.

Note – This API is available since firmware version 1.6.1

6.6.1. Action Definition

Endpoint	api/v1/actions
Method	GET, POST, DELETE
Response content type	application/json

Methods

GET – Retrieve the definition of actions.

POST – Define new actions.

DELETE – Delete existing actions definition.

Object Description

Field	Type	Required	Range	Remarks
.const	object	No		Optional object for defining constants such as IP addresses. Constants should be defined as object fields and can be of type string, number, boolean, or null. Nested objects and arrays are not supported. Constants can be referenced in url and payload fields using ${constant_name} syntax.
.actions[].id	string	Yes	1..31	Unique action identifier, which may be a name rather than a numeric ID. Used when executing an action.
.actions[].enable	boolean	No		Defaults to true. Determines if an action is active and executable. This field appears in the return JSON only if it is false.
.actions[].requests[]	array of objects	Yes		List of requests (currently only HTTP/HTTPS requests are supported) that the action should perform.
.actions[].requests[].http.enable	boolean	No		Defaults to true. Specifies whether the request should be active within the action. This field appears in the return JSON only if it is false.
.actions[].requests[].http.url	string	Yes	1..255	URL for HTTP requests. Constants may be used.
.actions[].requests[].http.method	string	No	GET, POST, PUT, DELETE	HTTP method for the request. GET by default. This field appears in the return JSON only if it is different than GET.
.actions[].requests[].http.payload	string	No	0..1023	HTTP request payload. Used only for POST and PUT methods.
.actions[].requests[].http.headers.*	object of strings	No		Optional object of up to 4 HTTP headers. Each field name should corresponds to header name, and the field value to header value. Length of header name + header value shouldn't exceed 253 characters.
.actions[].requests[].http.timeout	floating	No	0.1..10	HTTP request timeout expressed in seconds. The default timeout is 4s. This field is optional and appears in the return JSON only if it is set.
.actions[].requests[].modbus.enable	boolean	No		Defaults to true. Specifies whether the request should be active within the action. This field appears in the return JSON only if it is false.
.actions[].requests[].modbus.host	string	Yes	4..63	The target device's hostname or IP address for the Modbus TCP connection. This identifies the device to communicate with.
.actions[].requests[].modbus.port	uint	No	1..65535	The TCP port on the target device for Modbus communication. The default Modbus TCP port is 502, but this can be set to a different port if the device uses one.
.actions[].requests[].modbus.unit_id	uint	No	0..255	The Modbus Unit Identifier (also known as Slave ID) of the target device. This is typically a number from 1 to 247 identifying the device on the Modbus network (use 1 if the device is alone or you are unsure). Default value is 0.

.actions[].requests[].modbus.func	unit	Yes	5, 6, 15, 16	The Modbus function code representing the operation to perform. 5 is to write a single coil, 6 is to write a single register, 15 is to write multiple coils, and 16 is to write multiple registers.
.actions[].requests[].modbus.address	uint	Yes	0..65535	The starting data address for the Modbus operation. This is the offset of the first coil or register to access. Modbus addresses are typically given in the range 0–65535 (where each range corresponds to coil, input, holding register, etc., depending on the function code). Note: The address is zero-based as per Modbus protocol (e.g., address 0 refers to the first coil/ register in the device's memory block for that function).
.actions[].requests[].modbus.type	string	No	hex short int long float double	Default hex. See table Modbus type field.
.actions[].requests[].modbus.value	string	Yes	1..492	The content of the value field must match the data type selected in the type attribute, see table Modbus value field.
.actions[].requests[].modbus.timeout	floating	No	0.1..10	Modbus request timeout expressed in seconds. The default timeout is 5s. This field is optional and appears in the return JSON only if it is set.

Modbus Type Field

Setting	Expected value input	Conversion & Storage Rules
hex	A hexadecimal string whose length is an even number of characters.	The string is parsed byte-by-byte and the resulting raw bytes are written sequentially, starting at the base register address. Example: 0809A0B0
short	A 16-bit integer (signed or unsigned).	The integer is converted to big-endian byte order and stored in one register (16 bits). Example: 1234 or -1234
int	A 32-bit integer (signed or unsigned).	Converted to big-endian; the two most-significant bytes are written to the base register, the two least-significant bytes to the next register (address + 1).
float	A 32-bit floating-point number.	Encoded as IEEE-754, big-endian, and stored in two consecutive registers. Example: -1.23
double	A 64-bit floating-point number.	Encoded as IEEE-754, big-endian, and stored in four consecutive registers.

Modbus Value Field

Modbus Function Code	Guidance for type and value
5 – Write Single Coil	Use short with 0 / 1, or use hex with 0000 (OFF) or 00FF (ON). Only one register is written.
6 – Write Single Register	Same approach as Function 5: choose short (integer) or hex (exact two-byte value). Exactly one register is written.
15 – Write Multiple Coils	Use hex notation. The number of registers written depends on the length of the hex string: 4 hex characters = 1 register. Example: an 8-character string (A1B2C3D4) writes two registers.

16 – Write Multiple Registers	You may supply hex or any numeric type (short, int, long, float, double). The data are automatically expanded across as many consecutive registers as required by the chosen type.

Example configuration

```
{
  "const":{
    "bulb":"192.168.99.101",
    "switch":"192.168.99.151",
    "bri":"20"
  },
  "actions":[
    {
      "id":"1",
      "requests":[
        {
          "http":{
            "url":"http://192.168.0.21/api/v1/device/self",
            "method":"POST",
            "payload":"action=toggle",
            "headers":{
              "Content-Type":"application/x-www-form-urlencoded"
            }
          }
        },
        {
          "http":{
            "url":"http://${switch}/toggle",
         }
        }
      ]
    },
    {
      "id":"2",
      "requests":[
        {
          "http":{
            "url":"http://${bulb}/light/0?turn=on&brightness=${bri}&temp=3000",
          }
        }
      ]
    },
    {
      "id":"3",
      "requests":[
        {
          "http":{
            "url":"http://${bulb}/light/0",
            "method":"POST",
            "payload":"turn=off",
            "timeout":2
          }
        }
      ]
    },
    {
      "id":"4",
      "requests":[
        {
          "modbus":{
            "host":"192.168.99.179",
            "port":1502,
            "unit_id":1,
            "func":16,
            "address":10,
            "type":"short",
            "value":"${bri}",
            "timeout":2
          }
        }
      ]
    }
  ]
}
```

Response Codes

- **200 OK** – Successful retrieval or modification.
- **404 Not Found** – Actions have not been set or have been deleted.
- **400 Bad Request** – Invalid request format.

Note – The action definition object size cannot exceed 8191 bytes.

6.6.2. Actions Execution

Execute a specified action.

Endpoint	api/v1/actions/call?id=<action_id>
Method	POST

Example Request

```
curl -i -X POST 'http://192.168.99.176/api/v1/actions/call?id=1'
```

Expected Response

```
HTTP/1.1 202 Accepted
Content-Length: 0
Location: /api/v1/actions/status?id=139
X-Content-Type-Options: nosniff
Cache-Control: no-store, no-cache
```

Response Codes

- **202 Accepted** – At least one request from the action definition has been queued for asynchronous execution.
- **400 Bad Request** – The action is disabled, or all action requests are disabled.
- **404 Not Found** – No action found for the specified ID.
- **409 Conflict** – The action is already executing and cannot be restarted until completion.
- **429 Too Many Requests** – Too many requests are pending execution (max: 20 total requests across all pending actions).
- **500 Internal Server Error** – An unspecified issue occurred.

6.6.3. Checking Action Execution Status

Retrieve execution status for specified request IDs.

Endpoint	api/v1/actions/status?id=<status_id>[,<status_id>]
Method	GET
Response content type	application/json

Response object

Field	Type	Remarks
.action_id	string	The action ID corresponding to this status entry.
.id	uint	Execution status ID (from the Location header returned in the action execution request).
.code	uint	The HTTP response code from the executed request. A value of ≤0 indicates a network or connection issue.
.exe_time	float	Time taken to execute the action, in seconds.

Modbus Status Codes

Code	Description
1	Success
-1	Generic issue
-2	Response packet is too long
-3	Response receive timeout
-4	Response receive error
-5	Invalid PDU length in response
-6	Memory error
-7	Request sent error
-8	Invalid response header
-9	Transaction ID mismatch
-10	Function mismatch
-11	Invalid exception frame
-12	Unit ID mismatch
-13	Response frame too short
-14	Response body mismatch
-15	Cannot process response
-16	Cannot resolve host
-17	Cannot connect
-19	Cannot create connection
-100	Exception base
-101	Exception: illegal function
-102	Exception: illegal data address
-103	Exception: illegal data value
-104	Exception: slave device failure
-105	Exception: acknowledge
-106	Exception: slave device busy
-107	Exception: negative acknowledge

-108	Exception: memory parity error
-110	Exception: gateway path unavailable
-111	Exception: gateway target device failed to respond

Example Request

```
curl -s 'http://192.168.99.176/api/v1/actions/status?id=139'
```

Example Response

```
[
  {
    "action_id": "1",
    "id": 139,
    "code": 200,
    "exe_time": 0.133
  }
]
```

Notes

- The Location header in the action execution response provides the execution status ID(s).
- Multiple status IDs can be checked simultaneously by separating them with commas.

6.7. Wi-Fi network setup

This endpoint allows you to configure a connection to a Wi-Fi network.
HTTP REST API to configure wifi settings (you can also do it from the device's WebUI):

Endpoint	api/v1/wifi/sta/settings
Method	GET, POST, PUT, DELETE
Response content type	application/json

Wifi sta settings JSON object description

Field	Type	Default	Range	Remarks
.enable	boolean	false		Enables Wi-Fi communication in station mode (the device connects to the Router or Access Point)
.name	string	empty string	1..32	The name of the Wi-Fi network you want the device to connect to (SSID). Field is required
.password	string	empty string	8..64	The password to the Wi-Fi network, if set, must be at least 8 characters long. It is possible to enter a password in the form of a hexadecimal notation of 64 characters. The field may remain late if the network is an open network (without a password)
.static_ip	bool	false		Applies to setting a static recipient instead of using a DHCP client. Remember that when setting static addressing, address collision can occur in the network, so you need to be sure that no other device has the same address or will not have it. DHCP servers on Routers are usually configured in such a way that they assign addresses in the range 192.168.X.100 to 192.168.X.200, so that static addresses for various devices are safely set below 100 or above 200.
.ip	string	0.0.0.0		String stored as an IPv4 address. This is a static IP address of the device in the network, it can have a value e.g. 192.168.X.201 where X is a subnet e.g. 0, 1, 2 etc.
.netmask	string	0.0.0.0		String stored as an IPv4 address. This is a subnet mask, it must be set to non-zero values if you set a static IP address. Typically, the mask value is 255.255.255.0

Field	Type	Default	Range	Remarks
.gateway	string	0.0.0.0		String stored as an IPv4 address. This is the gateway address, in other words, the address of the device (Router) to which the whatwatt device sends data to redirect it to the Internet. The value cannot be zero in the case of setting a static IP address. Typically, the gateway value is 192.168.X.1, where X replaces your individual gateway value
.dns	string	0.0.0.0		String stored as an IPv4 address. This is the address of the DNS server, this server is used to resolve names on the network to IP addresses, e.g. Example.com can be resolved to 1.2.3.4. Setting this value to nonzero is required if you set a static IP address. You can enter an external IP address here, such as 8.8.8.8 or the same address as in the gateway field, then the router will resolve the names
.max_power	float	17	0..21	The transmission power of the Wi-Fi radio on the whatwatt device expressed in dBm, you can increase or decrease this value. Please note that in the case of restricted power supply options, such as powering only from the M-bus interface without an external power supply, the amount of power may be insufficient. Resolution is 0.5

The DELETE method erases all Wi-Fi configuration and shuts down the client.

Example response on GET request

```
{
  "enable":true,
  "name":"sjj",
  "static_ip":false,
  "ip":"0.0.0.0",
  "netmask":"0.0.0.0",
  "gateway":"0.0.0.0",
  "dns":"0.0.0.0",
  "max_tx_power":0
}
```

6.8. Scan Wi-Fi networks

This endpoint allows you to search for nearby Wi-Fi networks. Scanning Wi-Fi networks over Ethernet is not recommended.

HTTP REST API to scanning Wi-Fi networks (you can also do it from the device's WebUI):

Endpoint	api/v1/wifi/scan
Method	GET
Response content type	application/json

Wifi scan JSON object description

Field	Type	Range	Remarks
[].ssid	string	1..32	The SSID (Service Set Identifier) is the name of a Wi-Fi network. It's the identifier that devices use to connect to the correct wireless network among multiple available networks.
[].bssid	string	12 hexadecimal characters	The BSSID (Basic Service Set Identifier) is the MAC (Media Access Control) address of a wireless access point or router. It uniquely identifies each access point in a Wi-Fi network.
[].channel	uint	1..13	A Wi-Fi channel is a specific frequency range within a Wi-Fi band that routers and devices use to communicate wirelessly.
[].ht	string	20 40+ 40-	Wi-Fi HT (High Throughput) is a mode used in the Wi-Fi 802.11n standard that increases the network's data throughput. It uses MIMO (Multiple Input Multiple Output) technology to transmit multiple data streams simultaneously, enhancing network performance.

Field	Type	Range	Remarks
[].rssi	int	-127..0	Wi-Fi RSSI: RSSI (Received Signal Strength Indicator) measures the power level of a received signal. It's expressed in decibels (dBm), with higher values (closer to zero) indicating stronger signals. For example, -30 dBm is a very strong signal, while -90 dBm is very weak.
[].signal	uint	0..100	Wi-Fi signal strength in precents
[].auth_mode	string	open WEP WPA WPA2 WPA-WPA2 EAP WPA3 WPA2-WPA3 WAPI OWE WPA3-ENT	Wi-Fi ***auth_mode*** (authentication mode) determines how device authenticate on a Wi-Fi network.
[].pairwise_cipher	string		The pairwise cipher in Wi-Fi security refers to the encryption method used to secure unicast (one-to-one) communication between a client device and an access point.
[].group_cipher	string	none WEP40 WEP104 TKIP CCMP TKIP-CCMP AES-CMAC-128 SMS4 GCMP GCMP256 AES-GMAC-128 AES-GMAC-256 unknown	The group cipher in Wi-Fi security refers to the encryption method used to secure multicast and broadcast communications within a Wi-Fi network.
[].phy	string	bgn	
[].wps	string	true or false	
[].country	string	2 characters	The Wi-Fi country code is a setting that determines the regulatory domain for a Wi-Fi device, such as a router or access point.

6.9. Starting WPS pairing

This endpoint allows you to start pairing using WPS. The same can be done by pressing a button on the device. Pairing will automatically turn off after 2 minutes if the pairing button is not pressed on the Access Point or Router, or if there is a problem. Wi-Fi pairing via WPS can be invoked in both client mode and whatwatt access point mode, WPS pairing can also be enabled when the device has Wi-Fi turned off. Successful pairing automatically configures the device to operate in station/client mode. A failed pairing reverts to the previous Wi-Fi settings if the device was already paired or turns off Wi-Fi if you start it from the access point mode of the whatwatt device. Enabling WPS pairing over Ethernet is not recommended.

Sent a POST request to this endpoint returns a 204 code with no message if successful.

HTTP REST API to start WPS pairing (you can also do it from the device's WebUI):

Endpoint	api/v1/wifi/wps
Method	POST

Wi-Fi Protected Setup (WPS) is a network security standard that facilitates the connection between a router and wireless devices. It simplifies the process of connecting to a secure wireless network by enabling users to press a physical button on the router to pair devices. The goal of WPS is to make it easier for non-technical users to connect devices to their Wi-Fi network without entering long passphrases.

6.10. Ethernet Configuration

This endpoint allows you to configure an Ethernet connection.

HTTP REST API to configure Ethernet settings (you can also do it from the device's WebUI)

Endpoint	api/v1/eth/settings
Method	GET, POST, PUT
Response content type	application/json

Ethernet settings JSON object description

Field	Type	Default	Range	Remarks
.enable	boolean	true		Enables Ethernet port
.static_ip	bool	false		Applies to setting a static addressing instead of using a DHCP client. Remember that when setting static addressing, address collision can occur in the network, so you need to be sure that no other device has the same address or will not have it. DHCP servers on Routers are usually configured in such a way that they assign addresses in the range 192.168.X.100 to 192.168.X.200, so that static addresses for various devices are safely set below 100 or above 200.
.ip	string	0.0.0.0		String stored as an IPv4 address. This is a static IP address of the device in the network, it can have a value e.g. 192.168.X.201 where X is a subnet e.g. 0, 1, 2 etc.
.netmask	string	0.0.0.0		String stored as an IPv4 address. This is a subnet mask, it must be set to non-zero values if you set a static IP address. Normally, the mask value is 255.255.255.0
.gateway	string	0.0.0.0		String stored as an IPv4 address. This is the gateway address, in other words, the address of the device (Router) to which the whatwatt device sends data to redirect it to the Internet. The value cannot be zero in the case of setting a static IP address. Normally, the gateway value is 192.168.X.1, where X replaces your individual gateway value
.dns	string	0.0.0.0		String stored as an IPv4 address. This is the address of the DNS server, this server is used to resolve names on the network to IP addresses, e.g. Example.com can be resolved to 1.2.3.4. Setting this value to nonzero is required if you set a static IP address. You can enter an external IP address here, such as 8.8.8.8 or the same address as in the gateway field, then the router will resolve the names

Example response on GET request

```
{
  "enable":true,
  "static_ip":false,
  "ip":"0.0.0.0",
  "netmask":"0.0.0.0",
  "gateway":"0.0.0.0",
  "dns":"0.0.0.0"
}
```

6.11. Restarting the device device

This endpoint allows you to reboot the device.

Performing a POST under this endpoint returns a 204 code with no message if successful (you can also do it from the device's WebUI)

HTTP REST API to reboot the device:

Endpoint	api/v1/reboot
Method	POST

6.12. Factory reset

This endpoint allows you to restore your device to factory settings. You can also do this with a push.

Performing a POST under this endpoint returns a 204 code with no message if successful (you can also do it from the device's WebUI).

HTTP REST API to factory reset the device:

Endpoint	api/v1/restore
Method	POST

6.13. SD card access

This endpoint permits the browsing and downloading of files from your SD card. If the card is not mounted, an error code 503 will be returned.

HTTP REST API endpoint

Base Path	/sdcard/
Method	GET, DELETE
Response content type	Varies depending on the requested resource (e.g., application/json for directory listing, text/csv when downloading a CSV file).

Directory Listing

Request `GET /sdcard/ or GET /sdcard/<directory_name>/`

Example JSON response

```
{
  "path": "/sdcard/",
  "files": [
    {
      "name": "20240929.CSV",
      "size": 1161385,
      "type": "file"
    },
    {
      "name": "SYSTEM~1",
      "size": 0,
      "type": "dir"
    },
    {
      "name": "NIHAO.TXT",
      "size": 13,
      "type": "file"
    }
  ]
}
```

Structure of the response fields:

- **path (string)** – the complete directory path for which the contents are being listed.

- **files[].name (string)** – name of the file or directory.
- **files[].size (uint)** – file size in bytes (directories are reported as size 0).
- **files[].type (string)** – indicates whether the item is a file or a dir.

File Download

Request `GET /sdcard/<filename.extension>`

If the path points to a specific file (e.g., 20240929.CSV), the server attempts to detect the MIME type based on the file extension. For example, a .csv file will typically be served with the MIME type text/csv.

Example (using curl) `curl -s http://<device_address>/sdcard/20241010.CSV`

Sample response (CSV snippet)

```
RID,TIME,MID,MSTAT,TARIFF,PF,EAP_T,EAP_T1,EAP_T2,EAN_T,EAN_T1,EAN_T2,...
1,"2024-10-10T05:01:36Z","220000399","OK",2,0.169,29.424,20.841,8.583,...
4,"2024-10-10T05:01:55Z","220000399","OK",2,0.23,29.424,20.841,8.583,...
...
```

- MIME type (Multipurpose Internet Mail Extensions) specifies the nature and format of the file, which helps browsers or other applications interpret it correctly.
- To download a file from the server and save it with the same name as in the URL, simply use the -O switch. Then cURL will download the file and automatically give it a name that matches the last element of the path in the URL.

  ```
  curl -O http://192.168.99.178/sdcard/20241010.CSV
  ```
- In this case, you don't need to specify the file name separately with the -o option; cURL will automatically create a file named 20241010.CSV.
- You can also **easily download** the file using a browser by entering the path in the address bar or by opening the page **`http://<device_ip>/sd.html`**

File Delete

Request for file deletion: DELETE /sdcard/[path/]<filename.extension>

Request for directory deletion: DELETE /sdcard/path/[path/]

- In the case of deleting directories, all files and subdirectories contained within will also be deleted.

  Example (using curl) `curl -i -X DELETE http://127.0.0.1:8080/sdcard/20250127.csv`
- Deletes file. If successful, returns HTTP code 204.

  Example (using curl) `curl -i -X DELETE http://127.0.0.1:8080/sdcard/some_directory/`
- Deletes directory. If successful, returns HTTP code 204.

Error Handling

- **503 (Service Unavailable)** – returned when the SD card is not mounted or otherwise unavailable.
- **400 (Bad Request)** – returned when required operation is invalid, for example delete of root directory.
- **404 (Not Found)** – returned when there no file or directory.

6.13.1.Usage Examples

Listing the Contents of the SD Card Root Directory

```
curl -s http://192.168.99.178/sdcard/
```

Sample response

```
{
  "path": "/sdcard/",
  "files": [
    {"name":"NIHAO.TXT","size":13,"type":"file"},
    {"name":"SYSTEM~1","size":0,"type":"dir"},
    {"name":"20240924.CSV","size":493074,"type":"file"},
    ...
  ]
}
```

Downloading a CSV File

```
curl -s http://192.168.99.178/sdcard/20241010.CSV
```

Sample response

```
RID,TIME,MID,MSTAT,TARIFF,PF,EAP_T,EAP_T1,EAP_T2,EAN_T,EAN_T1,EAN_T2,...
1,"2024-10-10T05:01:36Z","220000399","OK",2,0.169,29.424,20.841,8.583,...
...
```

or save the file to disk immediately

```
curl -O http://192.168.99.178/sdcard/20241010.CSV
```

Delete a CSV File

```
curl -i -X DELETE http://192.168.99.178/sdcard/20241010.CSV
```

Returns 204 in case of success.

Script to download CSV files from a date range

CSV files with reports on the card are saved in the YYYYMMDD.CSV format. By default, saving reports to the SD card is disabled, you can enable it in the System WebUI section, and you can also set the saving frequency (resolution) there.

Script arguments

- argument 1: device IP address
- argument 2: start date (YYYY-MM-DD)
- argument 3: end date (YYYY-MM-DD)

Script source code:

```
#!/usr/bin/env bash

if [ $# -ne 3 ]; then
  echo "Usage: $0 <device_ip> <start_date> <end_date>"
  exit 1
fi

DEVICE_IP="$1"
START_DATE="$2"
END_DATE="$3"

if ! date -d "$START_DATE" &>/dev/null; then
  echo "Invalid start date: $START_DATE"
  exit 1
fi

if ! date -d "$END_DATE" &>/dev/null; then
  echo "Invalid end date: $END_DATE"
  exit 1
fi
```

```
START_DATE_INT=$(date -d "$START_DATE" +%Y%m%d)
END_DATE_INT=$(date -d "$END_DATE" +%Y%m%d)

if [ $START_DATE_INT -gt $END_DATE_INT ]; then
  echo "Error: start date is later than end date."
  exit 1
fi

FILE_LIST_JSON=$(curl -s "http://$DEVICE_IP/sdcard/")

if [ -z "$FILE_LIST_JSON" ]; then
  echo "No response or SD card unavailable."
  exit 1
fi

CSV_FILES=$(echo "$FILE_LIST_JSON" | jq -r '.files[] | select(.type=="file")
| .name' | grep -E '^[0-9]{8}\.CSV$')

if [ -z "$CSV_FILES" ]; then
  echo "No CSV files found in YYYYMMDD.CSV format."
  exit 0
fi

for file in $CSV_FILES; do
  DATE_PART="${file%.*}"
  YEAR=${DATE_PART:0:4}
  MONTH=${DATE_PART:4:2}
  DAY=${DATE_PART:6:2}
  FILE_DATE_INT=$(( 10#$YEAR * 10000 + 10#$MONTH * 100 + 10#$DAY ))
  if [ $FILE_DATE_INT -ge $START_DATE_INT ] && [ $FILE_DATE_INT -le
$END_DATE_INT ]; then
    echo "Downloading: $file"
    curl -s -O "http://$DEVICE_IP/sdcard/$file" || echo "Error downloading $file"
  fi
done

echo "Done."
```

To make the script executable, you need to give it permissions:

```
chmod +x download_csv.sh
```

Usage (example):

```
./download_csv.sh 192.168.99.178 2024-09-01 2024-09-30
```

or

```
./download_csv.sh 192.168.99.178 20240901 20240930
```

Result:

```
Downloading: 20240924.CSV
Downloading: 20240925.CSV
Downloading: 20240928.CSV
Downloading: 20240929.CSV
Downloading: 20240930.CSV
Done.
```

Summary

The `/sdcard/` endpoint provides a straightforward way to access the SD card contents. When /sdcard/ is requested without specifying a file, it returns a list of directories and files. If the request specifies a particular file (for example, `/sdcard/YYYYMMDD.csv`), the file is returned in the detected MIME format. If the SD card is missing or not mounted, the server responds with a 503 error.

Following the guidelines and examples above should help integrate SD card file handling into client applications or scripts (e.g., using `curl`) with minimal effort.

6.14. Firmware Update

This endpoint allows you to update the firmware. The file should be sent in multipart/form-data format.

(The update can also be performed from the WebUI of the device).

Endpoint	load
Method	POST
Response content type	plain/text

The following is an example of the curl command to update the firmware:

```
curl -i -F file=@upgrade_file.bin http://192.168.1.101/load
```

Multipart/form-data is a media type used to encode the files and other form data when they are being uploaded via HTTP POST requests. This format splits the form data into multiple parts, each separated by a boundary, and encodes each part with its own content type and disposition metadata. The parts together form the payload of the HTTP request.

The origin of multipart/form-data can be traced back to RFC 2388, which was published in 1998. This specification was developed to address the limitations of traditional form submission formats, such as application/x-www-form-urlencoded, which struggled with handling binary data and complex file uploads effectively. By allowing each part of the form to be processed independently, multipart/form-data enabled more robust and flexible handling of diverse data types, facilitating the seamless upload of files and enhancing the web's interactive capabilities.

Appendix A

Understanding HTTP Requests, Methods, Response Codes, Body, and Path.

Introduction

The Hypertext Transfer Protocol (HTTP) is the foundation of any data exchange on the Web and a protocol used for transmitting hypermedia documents, such as HTML. It is designed to enable communications between clients and servers. This guide will delve into the various aspects of HTTP requests, methods, response codes, body, and path.

HTTP Requests

An HTTP request is a message sent by the client to initiate an action on the server. The request contains several key components, including the method, path, headers, and body. The request's purpose is to perform a specific action, such as retrieving data, submitting data, or deleting data on the server.

HTTP Methods

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. These methods are often referred to as HTTP verbs. Here are some of the most commonly used methods:

- GET: Requests data from a specified resource.
- POST: Submits data to be processed to a specified resource.
- PUT: Updates a current resource with new data.
- DELETE: Deletes the specified resource.

Each method defines a specific action that can be performed on the resource, and it must be used appropriately to ensure the correct operation of the API.

HTTP Response Codes

When a server receives and processes an HTTP request, it sends back a response. The response includes a status code, which indicates the result of the request. Here are some of the key status codes:

- **200: Success** – The request has succeeded, and the server returns the requested resource, usually in JSON format.
- **204: No Content** – The server successfully processed the request, but there is no content to return.
- **400: Bad Request** – The server could not understand the request due to invalid syntax or parameters.
- **401: Unauthorized** – The client must authenticate itself to get the requested response.
- **404: Not Found** – The server cannot find the requested resource; it may be disabled or unavailable.
- **500: Internal Server Error** – The server encountered an internal problem and could not complete the request.
- **503: Service Unavailable** – The server is not ready to handle the request, often due to maintenance or overload.

HTTP Request Body

The body of an HTTP request is used to send data to the server. This data is typically sent with POST or PUT requests and can be in various formats, such as JSON, XML, or form data. The body contains the payload that the client wants to send to the server for processing.

HTTP Path

The path is a part of the URL that identifies a specific resource on the server. It usually follows the domain name and defines the endpoint to which the request is being sent. For example, in the context provided, the path for the service management and basic settings endpoint is:

api/v1/settings

This path, combined with the appropriate HTTP method, allows the client to perform actions such as retrieving, updating, or deleting the resource related to system settings.

Conclusion

Understanding HTTP requests, methods, response codes, body, and path is essential for effectively working with web APIs. Each component plays a crucial role in ensuring seamless communication between the client and server, allowing for efficient data exchange and resource management. By mastering these elements, developers can create robust and reliable applications that leverage the power of HTTP.

To interact with web APIs effectively, the `curl` command-line tool is invaluable. It allows for the execution of HTTP requests directly from the terminal, providing a versatile and powerful means of engaging with endpoints such as those described.

Appendix B

Using curl Command Options

- -i: This option is used to include the HTTP response headers in the output. When making a request, it's often crucial to see the headers returned by the server, as they contain important information such as status codes and content types.

  ```
  curl -i [URL]
  ```

- -s: The `-s` option stands for "silent" mode. It suppresses progress meters and error messages, making the output cleaner and more readable, especially useful when processing the response in scripts.

  ```
  curl -s [URL]
  ```

- -d: This option is used to send data in a POST request. You'll typically use this option when you need to submit form data or JSON payloads to the server for processing. It's crucial in scenarios where the request body must be included.

  ```
  curl -X POST -d '{"key": "value"}' [URL]
  ```

- -X: The `-X` option allows you to specify the HTTP method to use for the request, such as GET, POST, PUT, DELETE, etc. This is essential for interacting with APIs that require specific methods to perform different actions.

  ```
  curl -X PUT -d '{"setting": "new value"}' [URL]
  ```

By combining these options, you can craft precise and powerful HTTP requests tailored to your needs. For example, to send a JSON payload with a POST request and include the response headers, you could use:

```
curl -i -X POST -d '{"setting": "new value"}' [URL]
```

Mastering the `curl` command and its options equips you with the ability to seamlessly communicate with web APIs, ensuring efficient and effective interactions with server resources.

Appendix C

Below is a script that allows you to flatten a JSON structure using the jq command. This powerful tool streamlines the process of manipulating JSON data, enabling you to transform nested structures into a more manageable and readable format. By leveraging jq, you can effectively query and reshape your JSON data with ease, making it an invaluable asset for developers and data analysts alike. Dive into the script and discover how jq can enhance your data processing capabilities, turning complex JSON hierarchies into simplified, flat structures.

```
#!/bin/bash

jq -r '
  def walk(path):
    if (type == "object") then
      to_entries[]
      | . as $entry
      | $entry.value
      | walk(path + [ $entry.key ])
    elif (type == "array") then
      to_entries[]
      | . as $entry
      | $entry.value
      | walk(path + [ "[\($entry.key)]" ])
    else
      "\(path | join("."))\t\(type)\t\(.)"
    end;
  walk([])
' | column -t
```

Thanks to this script, we will receive responses in the following format, for example api/v1/report:

```
curl -s http://192.168.X.X/api/v1/report | ./format_json.sh


report.id                                         number  74149
report.interval                                   number  5.435
report.tariff                                     number  2
report.date_time                                  string  2024-12-14T05:23:05Z
report.instantaneous_power.active.positive.total  number  0
report.instantaneous_power.active.positive.l1     number  0
report.instantaneous_power.active.positive.l2     number  0
report.instantaneous_power.active.positive.l3     number  0
report.instantaneous_power.active.negative.total  number  0
report.instantaneous_power.active.negative.l1     number  0
report.instantaneous_power.active.negative.l2     number  0
report.instantaneous_power.active.negative.l3     number  0
report.instantaneous_power.reactive.positive.l1   number  0
report.instantaneous_power.reactive.positive.l2   number  0
report.instantaneous_power.reactive.positive.l3   number  0
report.instantaneous_power.reactive.negative.l1   number  0
report.instantaneous_power.reactive.negative.l2   number  0
report.instantaneous_power.reactive.negative.l3   number  0
report.voltage.l1                                 number  234
report.voltage.l2                                 number  0
report.voltage.l3                                 number  0
report.current.l1                                 number  0
report.current.l2                                 number  0
report.current.l3                                 number  0
report.energy.active.positive.total               number  23.888
report.energy.active.positive.t1                  number  33.388
report.energy.active.positive.t2                  number  49.248
report.energy.active.negative.total               number  0.004
report.energy.active.negative.t1                  number  7.868
report.energy.active.negative.t2                  number  1.097
report.energy.reactive.positive.total             number  10.505
report.energy.reactive.negative.total             number  15.147
report.energy.reactive.imported.inductive.total   number  33.715
report.energy.reactive.imported.inductive.t1      number  31.7
report.energy.reactive.imported.inductive.t2      number  2.015
report.energy.reactive.imported.capacitive.total  number  2.247
report.energy.reactive.imported.capacitive.t1     number  2.247
report.energy.reactive.imported.capacitive.t2     number  0
report.energy.reactive.exported.inductive.total   number  3.555
report.energy.reactive.exported.inductive.t1      number  2.727
report.energy.reactive.exported.inductive.t2      number  0.828
report.energy.reactive.exported.capacitive.total  number  94.536
report.energy.reactive.exported.capacitive.t1     number  4.852
report.energy.reactive.exported.capacitive.t2     number  89.684
report.conv_factor                                number  1
meter.status                                      string  OK
meter.interface                                   string  MBUS
meter.protocol                                    string  DLMS
meter.logical_name                                string  LGZ1030662444349
meter.id                                          string  636192
meter.model                                       string  636192
meter.vendor                                      string  Landis+Gyr
meter.prefix                                      string  LGZ
system.id                                         string  ECC9FF5C7A68
system.date_time                                  string  2025-02-04T15:11:47Z
system.boot_id                                    string  F8CB5873
system.time_since_boot                            number  450742
```

Appendix D

The following table provides a description of selected OBIS codes.

Logical name (C.D.E) Short OBIS	**Description**	**Unit**
96.1.0	Meter identifier	
96.14.0	Current tariff	
13.7.0	Instantaneous power factor	
1.8.0	Positive active energy (A+) total	Wh
1.8.1	Positive active energy (A+) in tariff T1	Wh
1.8.2	Positive active energy (A+) in tariff T2	Wh
2.8.0	Negative active energy (A-) total	Wh
2.8.1	Negative active energy (A-) in tariff T1	Wh
2.8.2	Negative active energy (A-) in tariff T2	Wh
3.8.0	Positive reactive energy (Q+) total	varh
3.8.1	Positive reactive energy (Q+) in tariff T1	varh
3.8.2	Positive reactive energy (Q+) in tariff T2	varh
4.8.0	Negative reactive energy (Q-) total	varh
4.8.1	Negative reactive energy (Q-) in tariff T1	varh
4.8.2	Negative reactive energy (Q-) in tariff T2	varh
5.8.0	Imported inductive reactive energy in 1-st quadrant (Q1) total	varh
5.8.1	Imported inductive reactive energy in 1-st quadrant (Q1) in tariff T1	varh
5.8.2	Imported inductive reactive energy in 1-st quadrant (Q1) in tariff T2	varh
6.8.0	Imported capacitive reactive energy in 2-nd quadrant (Q2) total	varh
6.8.1	Imported capacitive reactive energy in 2-nd quadr. (Q2) in tariff T1	varh
6.8.2	Imported capacitive reactive energy in 2-nd quadr. (Q2) in tariff T2	varh
7.8.0	Exported inductive reactive energy in 3-rd quadrant (Q3) total	varh
7.8.1	Exported inductive reactive energy in 3-rd quadrant (Q3) in tariff T1	varh
7.8.2	Exported inductive reactive energy in 3-rd quadrant (Q3) in tariff T2	varh
8.8.0	Exported capacitive reactive energy in 4-th quadrant (Q4) total	varh
8.8.1	Exported capacitive reactive energy in 4-th quadr. (Q4) in tariff T1	varh
8.8.2	Exported capacitive reactive energy in 4-th quadr. (Q4) in tariff T2	varh
1.6.0	Positive active maximum demand (A+) total	Wh
1.6.1	Positive active maximum demand (A+) in tariff T1	Wh
1.6.2	Positive active maximum demand (A+) in tariff T2	Wh
2.6.0	Negative active maximum demand (A-) total	Wh
2.6.1	Negative active maximum demand (A-) in tariff T1	Wh
2.6.2	Negative active maximum demand (A-) in tariff T2	Wh
1.7.0	Positive active instantaneous power (A+)	W

Logical name (C.D.E) Short OBIS	Description	Unit
21.7.0	Positive active instantaneous power (A+) in phase L1	W
41.7.0	Positive active instantaneous power (A+) in phase L2	W
61.7.0	Positive active instantaneous power (A+) in phase L3	W
2.7.0	Negative active instantaneous power (A-)	W
22.7.0	Negative active instantaneous power (A-) in phase L1	W
42.7.0	Negative active instantaneous power (A-) in phase L2	W
62.7.0	Negative active instantaneous power (A-) in phase L3	W
3.7.0	Positive reactive instantaneous power (Q+)	var
23.7.0	Positive reactive instantaneous power (Q+) in phase L1	var
43.7.0	Positive reactive instantaneous power (Q+) in phase L2	var
63.7.0	Positive reactive instantaneous power (Q+) in phase L3	var
4.7.0	Negative reactive instantaneous power (Q-)	var
24.7.0	Negative reactive instantaneous power (Q-) in phase L1	var
44.7.0	Negative reactive instantaneous power (Q-) in phase L2	var
64.7.0	Negative reactive instantaneous power (Q-) in phase L3	var
9.7.0	Apparent instantaneous power (S+)	VA
32.7.0	Instantaneous voltage (U) in phase L1	V
52.7.0	Instantaneous voltage (U) in phase L2	V
72.7.0	Instantaneous voltage (U) in phase L3	V
31.7.0	Instantaneous current (I) in phase L1	A
51.7.0	Instantaneous current (I) in phase L2	A
71.7.0	Instantaneous current (I) in phase L3	A
42.0.0	Meter identifier	
96.1.1	Meter model	