webdyn
flexitron group

TITAN

Application Note 52

Autonomous Reading of IEC 60870-5-102 Electricity Meters with Transmission of Data to an MQTT Platform + CSD Data Calls

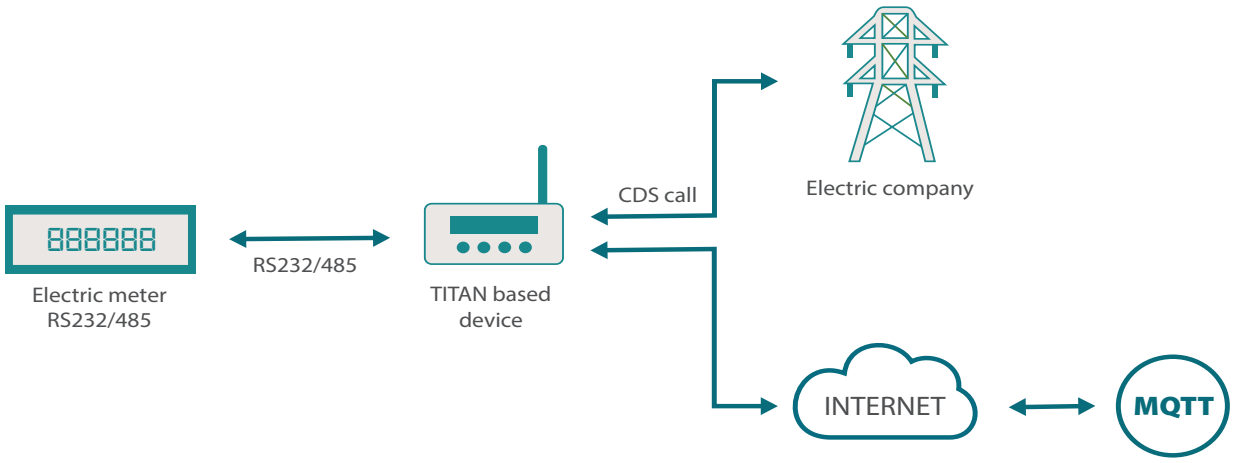
Reading of IEC 60870-5-102 Meters with Transmission of Data to an MQTT Platform + CSD Data Calls

1. Scenario Details

TITAN-based devices have all the typical functionalities of 4G/3G/2G routers, as well as a series of added features that make them one of the most feature-packed routers on the market. Some of the added features include the ability to read IEC 60870-5-102 meters autonomously, to periodically read the instantaneous values, to read the closing values each day and to store the data in the internal datalogger to send it to a platform (HTTP / HTTPS, MQTT / MQTTS, FTP), as long as there is 4G/3G/2G coverage. This feature can be combined with the reception of CSD data calls.

2. Description of the Scenario in the Example

- We have an Electricity Meter (IEC 60870-5-102) with an RS232 serial port (9600,8,N,1)
- The goal is to configure the TITAN-based device to autonomously read an IEC 60870-5-102 electricity meter every 15 minutes and extract the instantaneous values (absolute active energy, inductive absolute active energy, reactive active absolute energy, total active power, total reactive power, etc.) and sending said data from the Electricity Meter to an MQTT broker.
- The TITAN-based device also needs to be able to read the closing values of the electricity meter each day and to configure the number of days (31 days by default). Similarly, the values read must be sent to an MQTT broker each day.
- The TITAN-based device must also accept CSD data calls and create a transparent CSD-RS232 gateway so that the utility company (Enel, Iberdrola, etc.) can read the data from the Electricity Meter. The CSD call must take priority over autonomous reading of the instantaneous values.



3. Configuring the Associated Serial Port

In this example, the TITAN-based device's RS232 port will be used, this is the serial port through which the Electricity Meter is connected. To configure this, go to the "Serial Settings > Serial Port1-232" menu and enter the appropriate values. The appropriate values are those that coincide with the configuration of the Electricity Meter's serial port, which in the case of this example is 9600,8,N,1.

In this scenario we must also be able to read the Electricity Meter using a CSD call, so the "Allow incoming GSM Call" box must be checked.

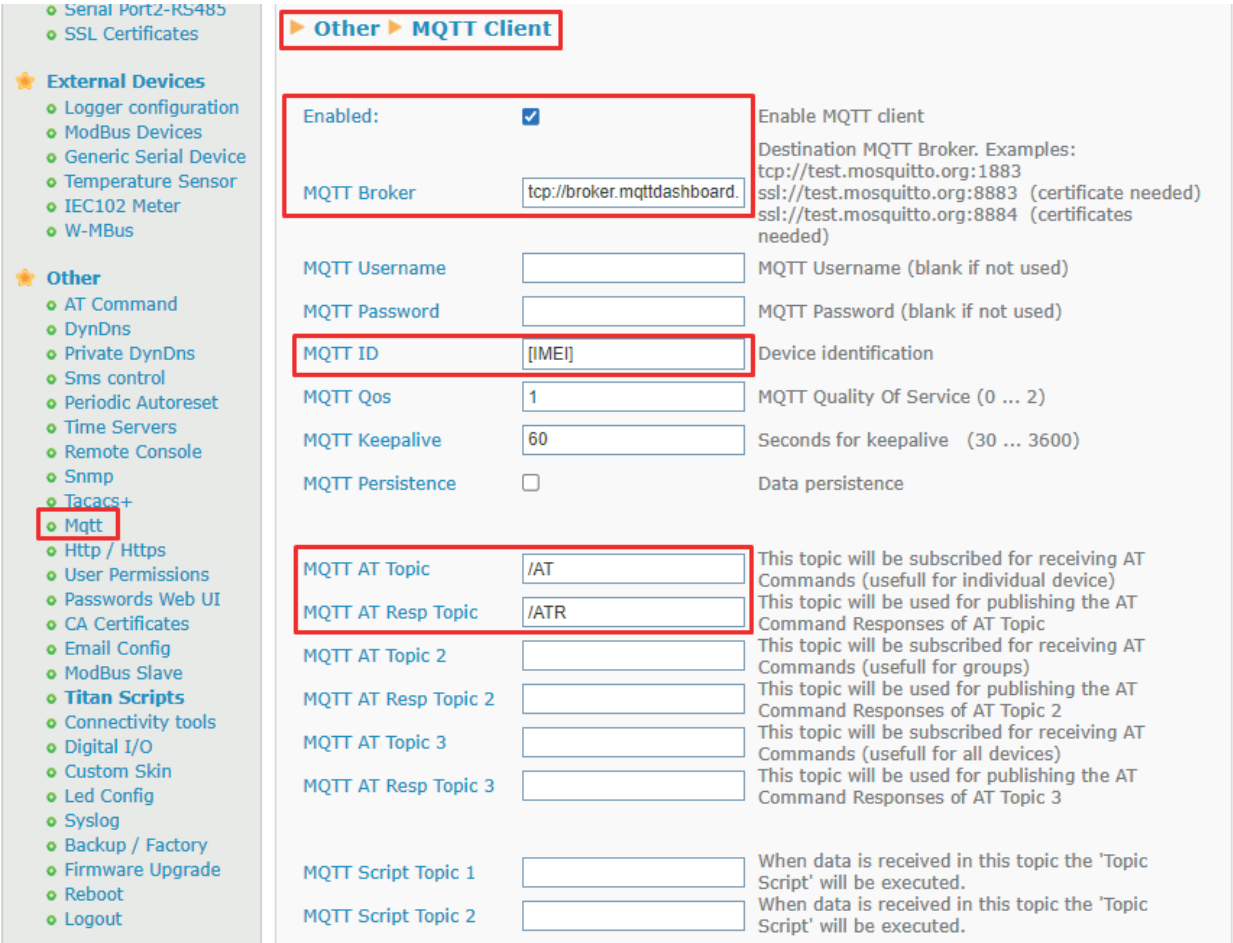
4. WAN Configuration

The TITAN-based device needs to be configured to have IP communication via GSM, as MQTT communications will require the corresponding interface in this scenario. To do this, go to the “Mobile > Basic Settings” menu and enable the WAN interface at the least, also specify the SIM card’s APN / username / password.

Given that we NEED to receive CSD calls, we must enter the value "2G" in the "Network Selection" field, otherwise CSD calls will probably not be able to be used (many GSM operators also allow TITAN-based devices to use AUTO mode (4G/2G), but before setting this mode we recommend consulting your telephone operator or performing a field test).

5. MQTT Configuration

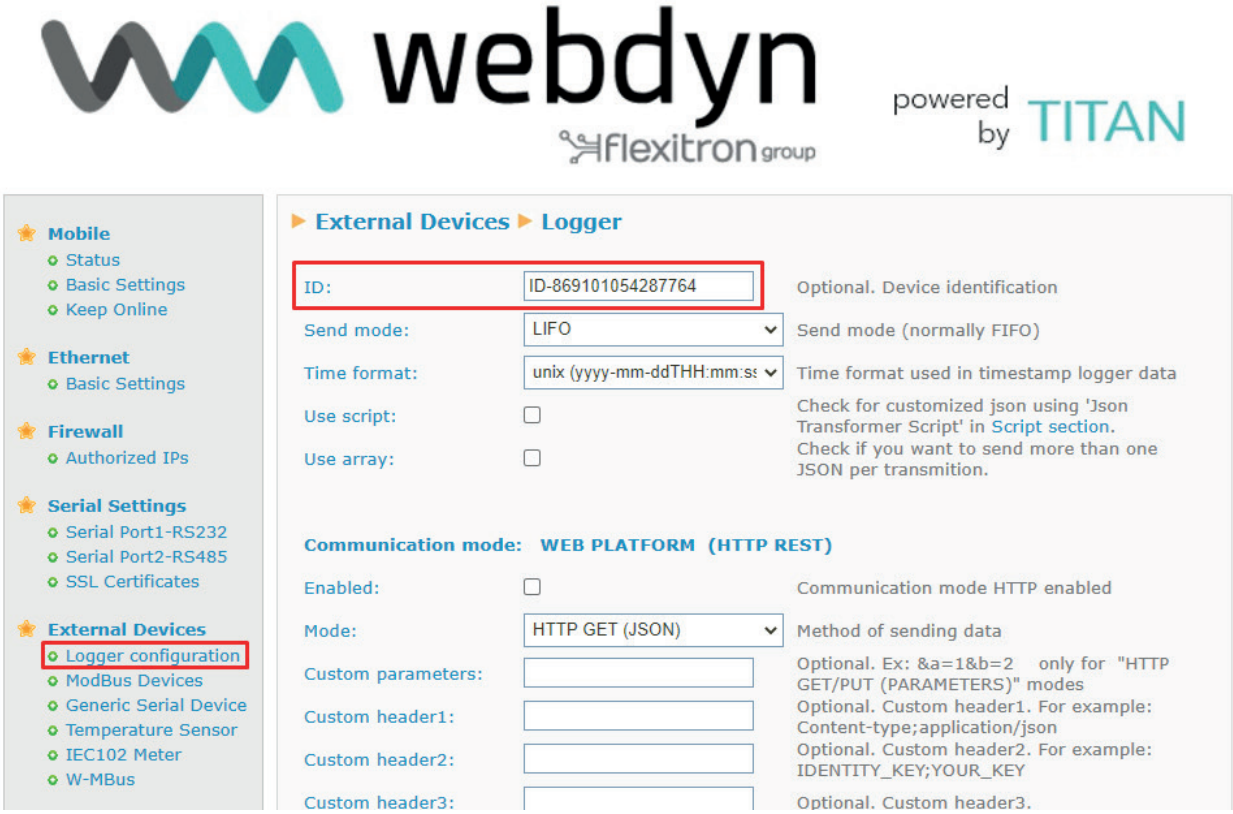
The TITAN-based device is going to send the data from the Electricity Meter to an MQTT broker, so the MQTT section of the device must also be configured. To do this, go to the “Other > Mqtt” menu and configure it appropriately. In this example, the HIVEMQ testing platform will be used. Enter “tcp://broker.mqttdashboard.com:1883” in the “MQTT Broker” field, and the device's IMEI in the “MQTT ID” field as an identifier, enter “[IMEI]”. We also need to be able to send AT commands to the TITAN-based device from the MQTT platform to perform maintenance tasks, configuration, device status readings, etc., in order to be able to send AT commands to the TITAN-based device remotely, the “MQTT AT Topic” and “MQTT AT Resp Topic” fields must be filled in. The AT commands should be sent to the first topic so that they are received and executed by the TITAN-based device. The TITAN-based device will send the responses to the executed AT commands to the second topic, this is the topic on the MQTT platform where we can get the responses to the executed AT commands. The following screenshot shows the settings required for this scenario.



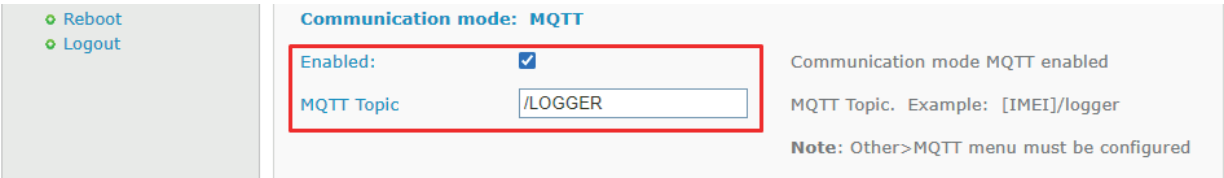
6. LOGGER Configuration

The next step is to configure the TITAN-based device's LOGGER. We must configure the device's internal memory (where it saves the data read from the Electricity Meter) and the method used to send the saved data to remote platforms (in this example, MQTT). The LOGGER is configured in the "Extenal Devices > Logger configuration" menu.

The optional ID field can be configured (with an arbitrary device identifier if you do not want to use the IMEI as such). The method for sending data to the platform can be configured as LIFO or FIFO, as applicable. The "Time format" field must be set to the standard UNIX format.



Also in the LOGGER configuration section, at the bottom of the page, the "Enabled" box must be checked to activate the MQTT delivery mode. The delivery topic (the MQTT topic to which the TITAN-based device will send the data) must be set to "/LOGGER" in this example.

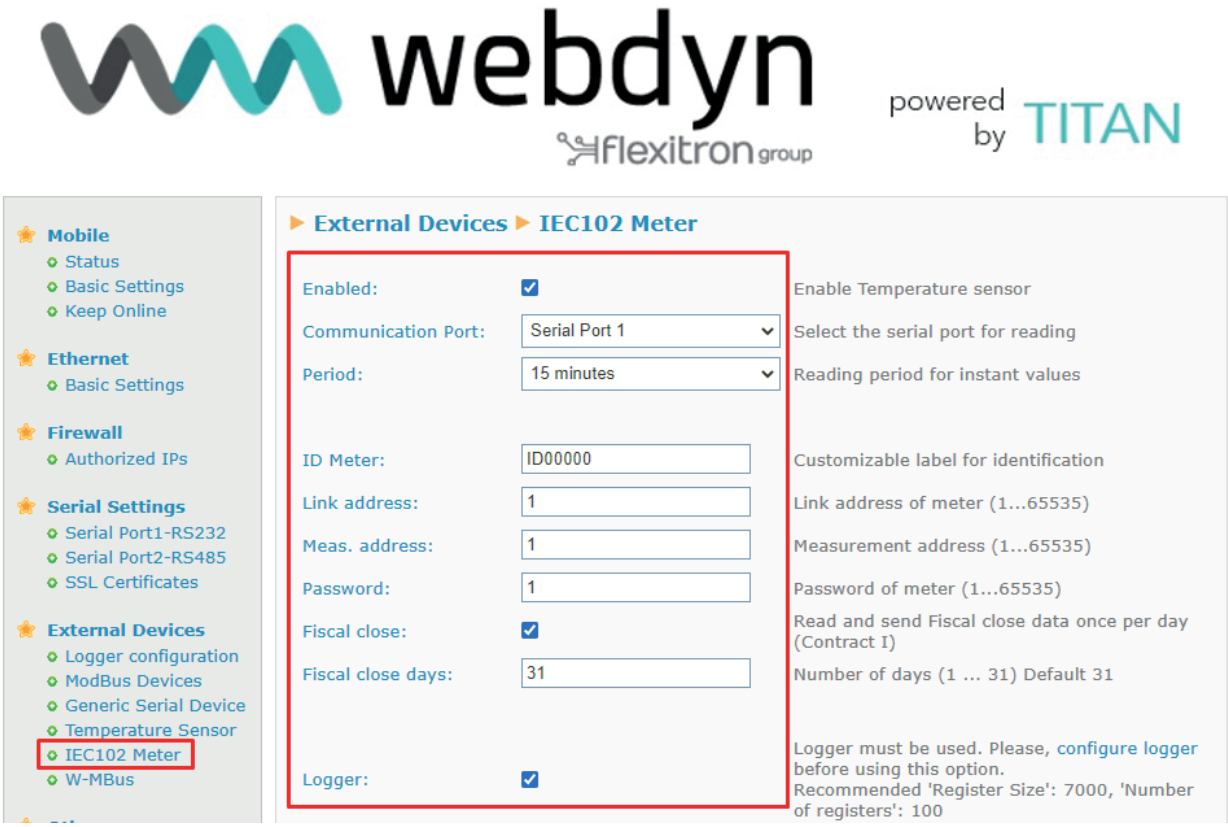


7. IEC-60870-5-102 Configuration

The last step is to configure the IEC-60870-5-102 protocol of the electricity meter. Go to the “External Devices > IEC102 Meter” menu. In this section we must enable the service and specify the serial port to use for the TITAN-based device (“Serial Port 1”), and the interval at which the instantaneous values are to be read, in this case 15 minutes.

In the following fields we enter a customizable value with the meter identifier (in this example ID00000), as well as the link address, the metering point address and the password. As the meter's pricing Information is also needed, the "Fiscal close" box must be checked and the number of days set to 31.

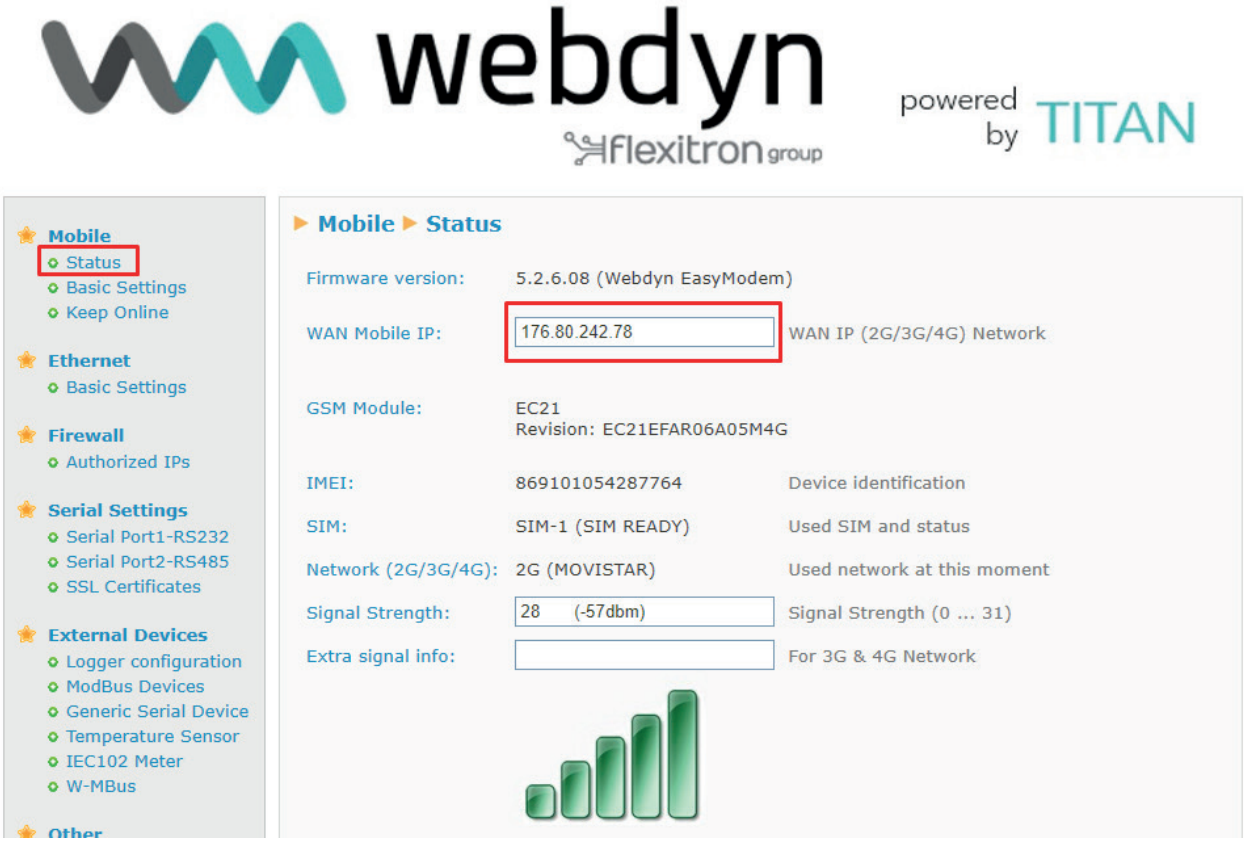
Lastly, in order for the data from the reading to be stored in the TITAN-based device’s LOGGER to be sent to the MQTT platform, the “Logger” box must be activated.



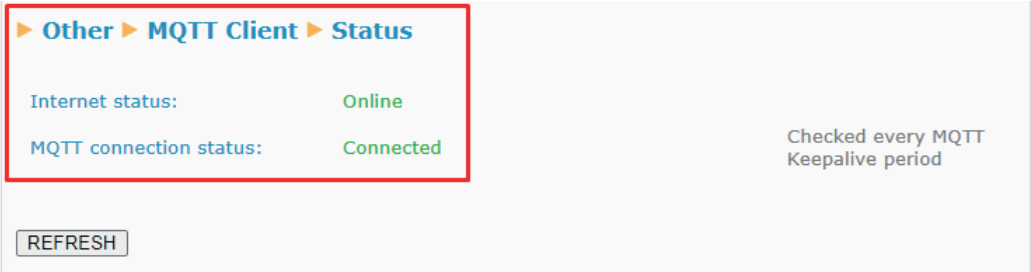
After pressing the “SAVE CONFIG” button to save the changes, we must REBOOT the TITAN-based device for the new configuration to take effect. It can be rebooted from the “Other > Reboot” menu.

8. Testing the Scenario

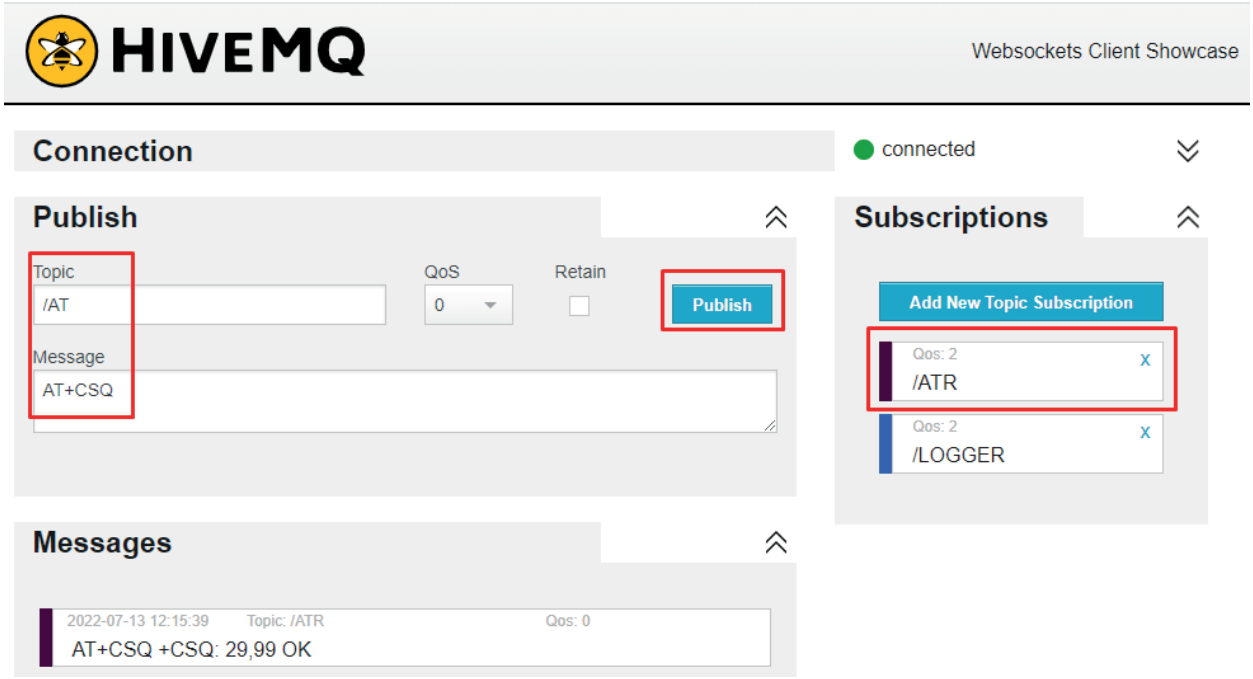
Lastly we need to check that the system is operating correctly. Once the TITAN-based device has been restarted, wait a few seconds and check that the device has obtained an IP address in the “Mobile>Status” menu.



Next, check that the connection with the MQTT broker is correct in the “Other>MQTT” menu.

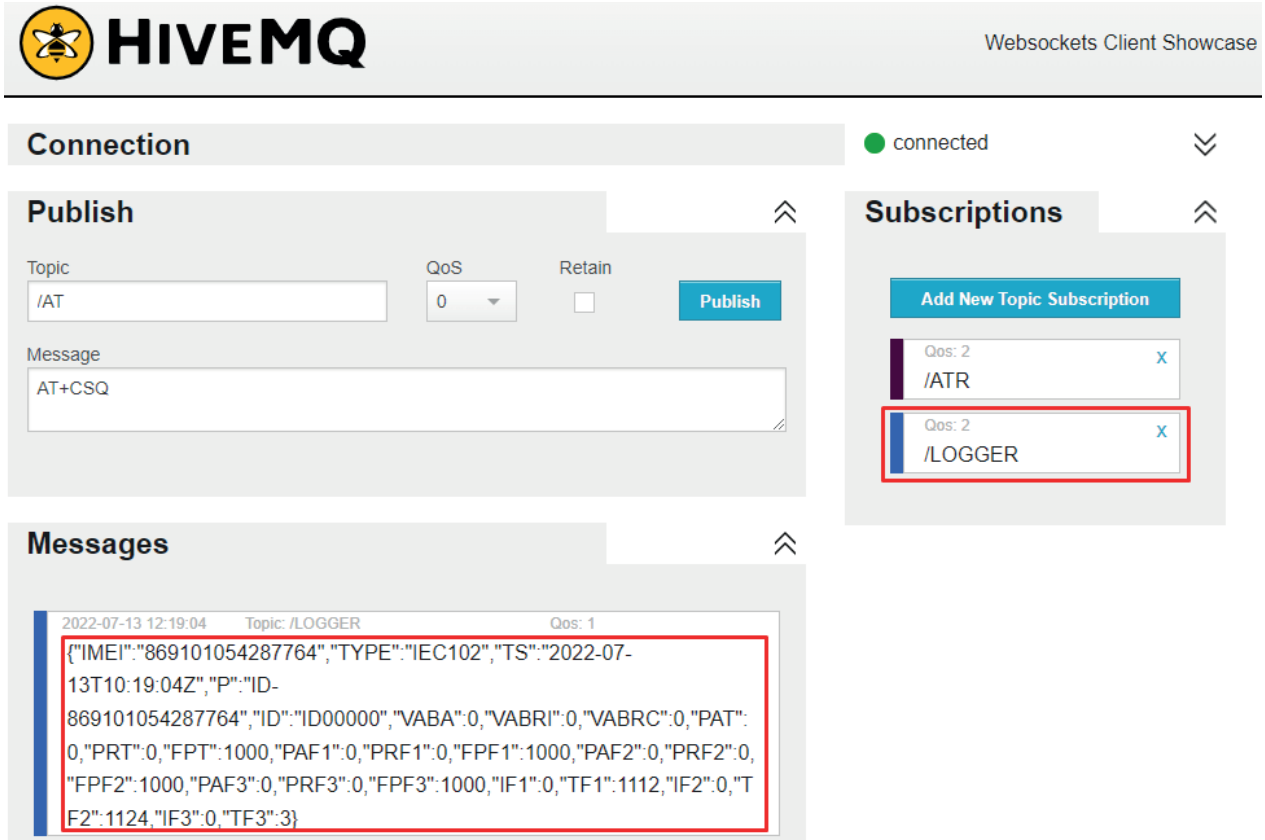


It can also check that the connection exists in the MQTT broker and is well established, by sending an AT command to the TITAN-based device via MQTT. As such, we must configure the /ATR (to receive responses from the executed AT commands) and /LOGGER topics to which the TITAN-based device will send the data read from the Electricity Meter.



Next we must check communications with the Electricity Meter. Go to the “External Devices>IEC102 Meter” menu. Towards the bottom of the screen, click on the “FORCE READ” button to force the device to read the instantaneous values from the Electricity Meter. The values read from the meter should then be displayed on the screen as shown in the following screenshot, indicating that the meter and the TITAN-based device’s serial port are configured correctly.

The data obtained by clicking on "FORCE READ" are NOT stored in the logger, as the "FORCE READ" button only forces a reading of the Electricity Meter's instantaneous values to check that the configuration is correct. Before the MQTT platform receives the instantaneous values, we will need to wait for the time interval configured in the TITAN-based device's "External Devices > IEC102 Meter" menu, which in this example is 15 minutes. Once this time has elapsed, the data sent by the TITAN-based device should be received on the MQTT platform, as shown in the following image.



Example of a JSON object containing the instantaneous values

{"IMEI":"869101054287764","TYPE":"IEC102","TS":
"2022-07-13T10:19:04Z","P":"ID-869101054287764","ID":"ID00000",
"VABA":0,"VABRI":0,"VABRC":0,"PAT":0,"PRT":0,"FPT":1000,
"PAF1":0,"PRF1":0,"FPF1":1000,"PAF2":0,"PRF2":0,"FPF2":1000,
"PAF3":0,"PRF3":0,"FPF3":1000,"IF1":0,"TF1":1112,"IF2":0,"TF2":
1124,"IF3":0,"TF3":3}

Where:

IMEI: ID number of the TITAN-based device.

TYPE: type of JSON. In this case the type is IEC102.

TS: timestamp of the time the data was collected.

P: Logger ID field.

ID: ID field of the IEC102 configuration in the TITAN-based device.

VABA: absolute active energy

VABRI: inductive absolute reactive energy

VABRC: inductive absolute reactive energy

PAT: total active power

PRT: total reactive power

FPT: total power factor

PAF1: phase I active power

PRF1: phase I reactive power

FPF1: phase I power factor

PAF2: phase II active power

PRF2: phase II reactive power

FPF2: phase II power factor

PAF3: phase III reactive power

PRF3: phase III reactive power

FPF3: phase III power factor

IF1: phase I intensity

TF1: phase I voltage

IF2: phase II intensity

TF2: phase II voltage

IF3: phase III intensity

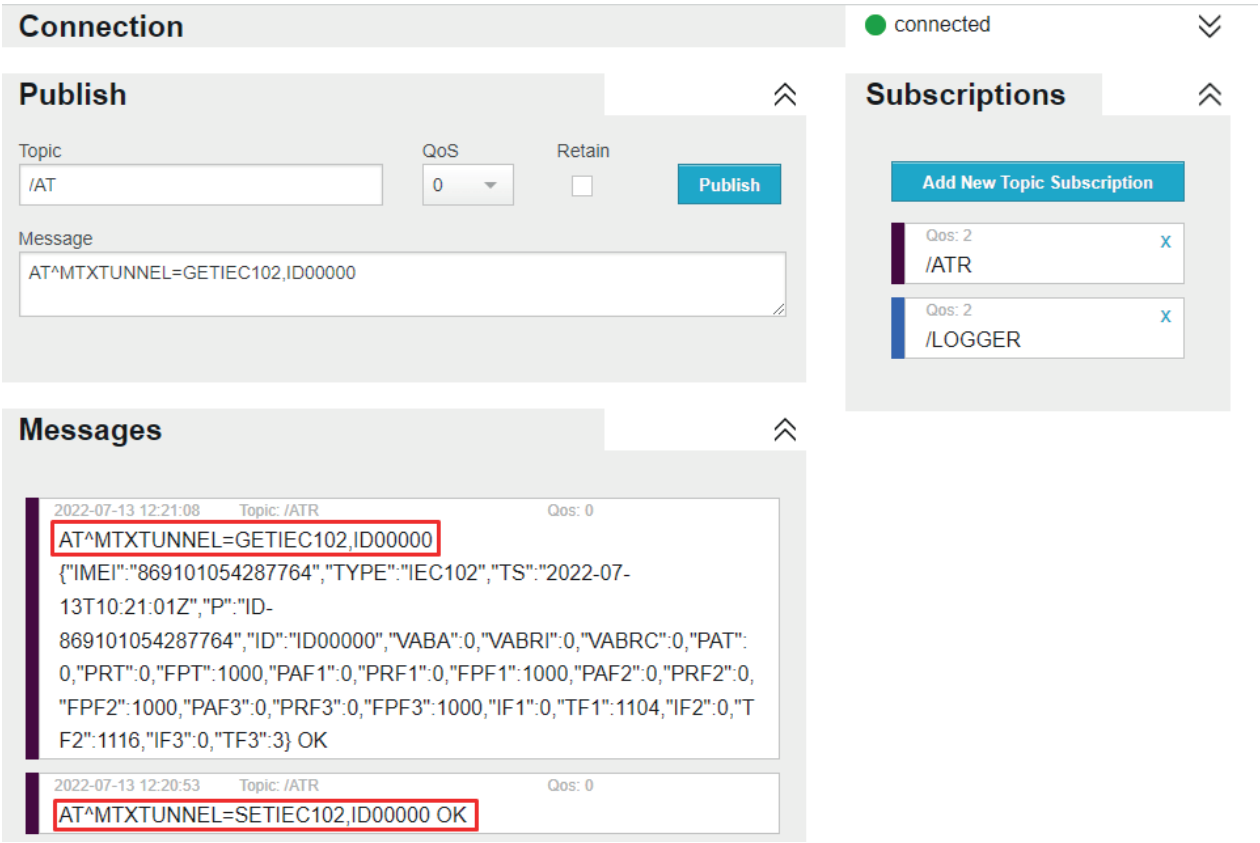
TF3: phase III voltage

Obtaining Real-Time Values from the Platform Without Waiting

The Electricity Meter's instantaneous values can be read from the MQTT platform at any time. To do this, simply execute the following AT command: AT^MTXTUNNEL=SETIEC102,ID00000 where ID00000 is the identifier of the meter specified in the TITAN-based device's "External devices >IEC102 Meter" menu.

A few seconds after the previous AT command was executed (the time required for the TITAN-based device to interrogate the Electricity Meter) we can obtain the instantaneous values via MQTT by executing the command: AT^MTXTUNNEL=GETIEC102,ID00000.

The following image shows a screenshot with both commands executed and their respective responses.



Reading the Electricity Meter's Closing Values

We can also read the Electricity Meter's closing values without having to wait for the day to change by executing a special AT command on the TITAN-based device:

AT^MTXTUNNEL=SETIEC102_CTAVM2,<IDMETER>,<horaIni>,<minutoIni>,<diaIni>,<mesIni>,<anoIni>,

<horaFin>,<minutoFin>,<diaFin>,<mesFin>,<anoFin>

An example of an AT command would be:

AT^MTXTUNNEL=SETIEC102_CTAVM2,ID00000,0,0,1,07,22,0,0,13,07,22

Similarly to the AT command for reading the instantaneous values, a few seconds after receiving the OK for this AT command, we can obtain the closing values from the Electricity Meter using the following AT command:

AT^MTXTUNNEL=GETIEC102_CTAVM2,ID00000

2022-07-13 12:23:10 Topic: /ATR Qos: 0

AT^MTXTUNNEL=GETIEC102_CTAVM2,ID00000

{"IMEI":"869101054287764","TYPE":"IEC102_CTAVM2","TS":"2022-07-13T10:23:05Z","P":"ID-869101054287764","ID":"ID00000","CTAVM2":[{"DO":20,"EaA":0,"EiA":0,"CA":2,"EaRi":0,"EiRi":0,"CRi":2,"EaRc":0,"EiRc":0,"CRc":2,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":128,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"},
{"DO":21,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":0,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"},
{"DO":22,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":0,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"},
{"DO":23,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":0,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"},
{"DO":24,"EaA":0,"EiA":0,"CA":2,"EaRi":0,"EiRi":0,"CRi":2,"EaRc":0,"EiRc":0,"CRc":2,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":2,"EPA":0,"CE":0,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"},
{"DO":25,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":0,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"},
{"DO":26,"EaA":0,"EiA":0,"CA":0,"EaRi":0,"EiRi":0,"CRi":0,"EaRc":0,"EiRc":0,"CRc":0,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":0,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"}]} OK

The JSON object containing the integrated totals returned by the TITAN-based device has the following structure:

{"IMEI":"869101054287764","TYPE":"IEC102_CTAVM2","TS":"2022-07-13T10:23:05Z","P":"ID-869101054287764","ID":"ID00000","CTAVM2":[{"DO":20,"EaA":0,"EiA":0,"CA":2,"EaRi":0,"EiRi":0,"CRi":2,"EaRc":0,"EiRc":0,"CRc":2,"R7":0,"C7":128,"R8":0,"C8":128,"MPA":0,"FMPA":"2022-07-11T13:49-1","CMA":0,"EPA":0,"CE":128,"DINI":"2022-06-22T15:14-1","DEND":"2022-07-11T13:49-1"}, ...

Where:

IMEI: ID number of the TITAN-based device.

TYPE: type of JSON. In this case the type is IEC102_CTAVM2.

TS: timestamp of the time the data was collected.

P: Logger ID field.

ID: ID field of the IEC102 configuration in the TITAN-based device.

CTAVM2: array containing the read data

The data array (CTAVM2 field) is made up of the following JSON objects:

DO: object address

EaA: absolute active energy

EiA: incremental active energy

AC: active energy qualifier

EaRi: inductive absolute reactive energy

EiRi: inductive incremental reactive energy

CRi: inductive reactive energy qualifier

EaRc: capacitive absolute reactive energy

EiRc: capacitive incremental reactive energy

CRc: capacitive reactive energy qualifier

R7: register 7 reserve

C7: register 7 reserve qualifier

R8: register 8 reserve

C8: register 8 reserve qualifier

MPA: maximum power

FMPA: date of the maximum

CMA: maximum qualifier

EPA: excesses of the powers

CE: excess qualifier

DINI: start of the period

DEND: end of the period