
# M4M Teltonika Integration

Prepared For: UP

Prepared By: Heinrich de Lange

# Executive Summary

This document serves as a guide to integrate Modbus meters, specifically ABB M4M30 meters with Teltonika RUT956 or TRB145 units via MQTT.

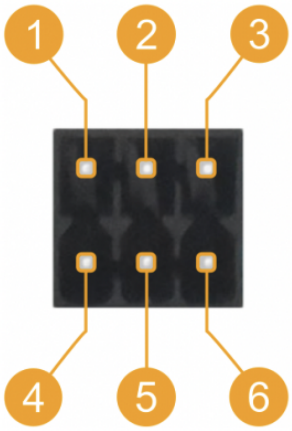
Source code for this project can be found here:
https://github.com/heinrich321/abb-teltonika

# Table of Contents

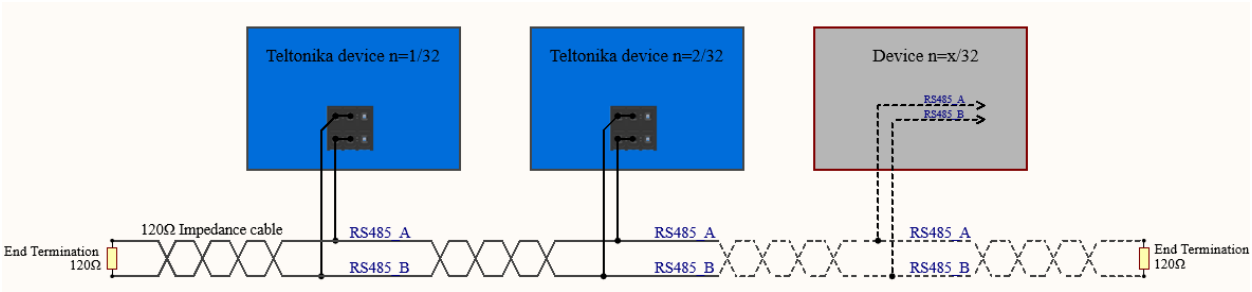
# Wiring

The configuration used is **not** full duplex and therefore the same 2 wires are used as transmit and receive wires. This requires the Negative Driver & Receiver to be bridged as well as the Positive Driver and Receiver.

| RS485 connector pinout | | |
|---|---|---|
| Pin | Name | Description |
| 1 | D_N | Driver negative signal |
| 2 | R_N | Receiver negative signal |
| 3 | GND | Device ground |
| 4 | D_P | Driver positive signal |
| 5 | R_P | Receiver positive signal |
| 6 | NC | Power input 9-30 VDC |




Bridging RS485 connector pins 1 & 2 as well as pin 4 & 5 will result in the below configuration:



# ABB M4M30 Configuration

Ensure that the communication settings on the M4M30 device align with the Teltonika settings. Navigate to the touchscreen interface to configure these parameters. For this integration guide, pay attention to the following crucial settings and their corresponding values:

- Baud Rate: 38400
- Parity: Even
- Bits: 8
- Stop Bit: 1

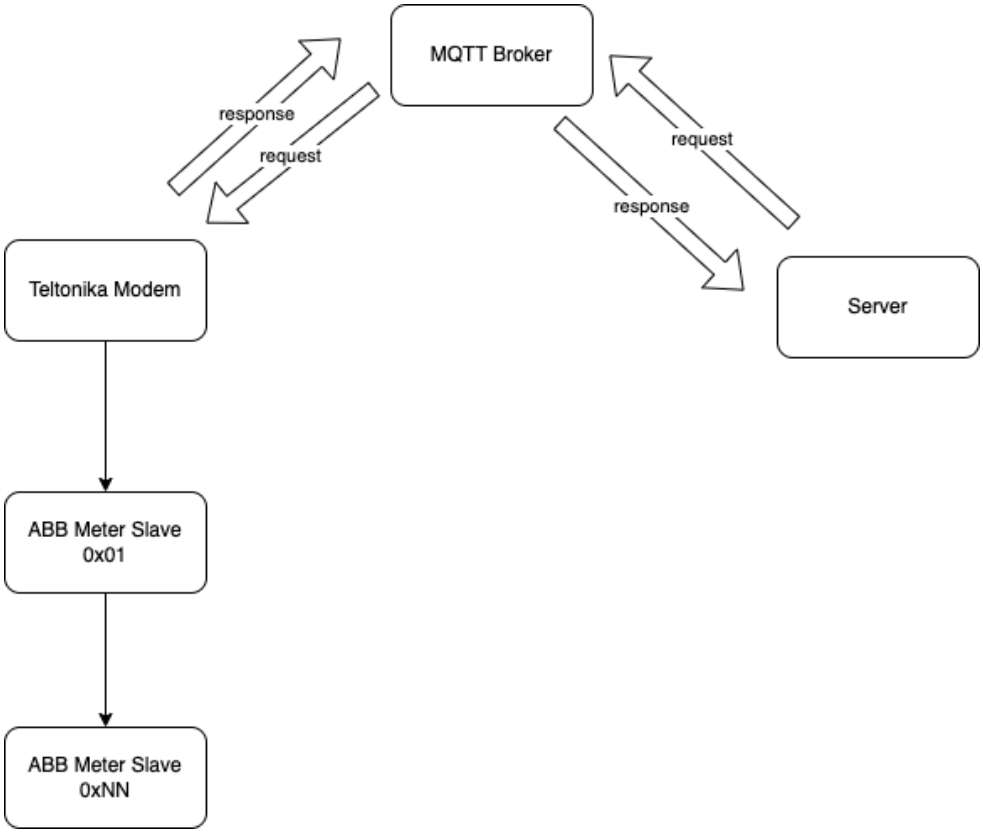
# Communication options

There are two main ways of configuring communication between an RS485 Modbus RTU device and an MQTT client. The first option is the “Gateway” configuration. This configuration allows Modbus commands to be sent by an external MQTT client and is then executed by the Teltonika device and the response is sent back.

The second option is to use the data-to-server functionality in combination with a running Modbus Serial Client on the Teltonika device. In this configuration the Teltonika device periodically reads all of the registers configured from the slave devices and then periodically publishes this data to an MQTT topic for MQTT clients to consume.

# Gateway Configuration

## Overview




The gateway configuration creates a bridge over MQTT between the server and the Modbus devices. Requests are sent to the /request topic and responses are sent back on the /response topic. These topics are configurable.

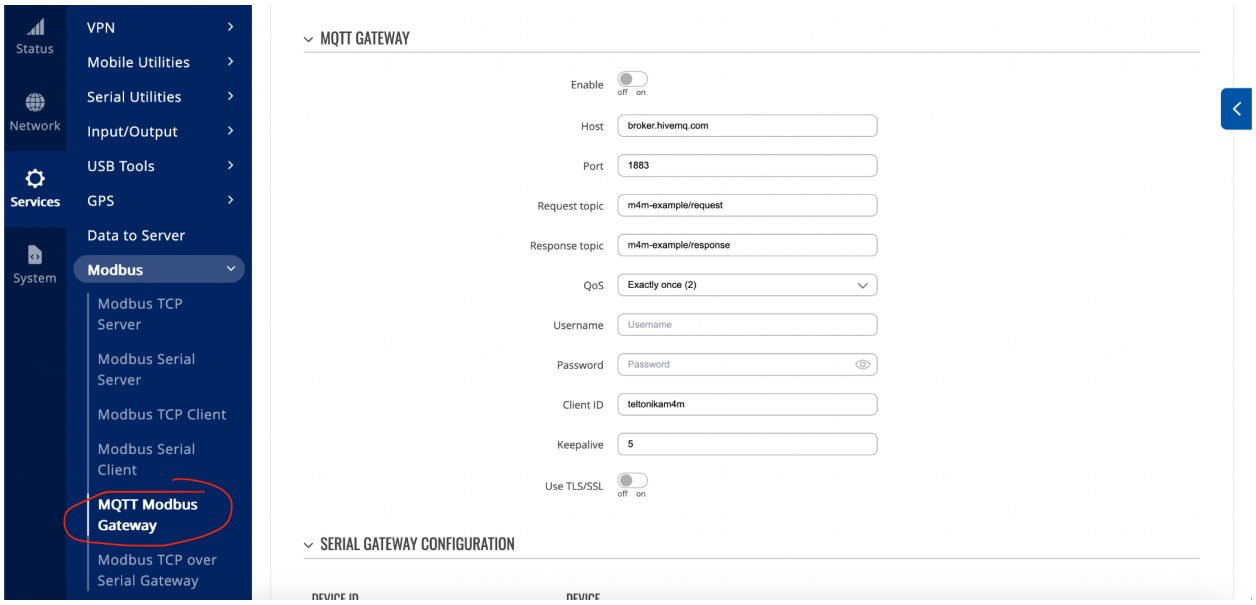
This configuration can be used if it is not known what parameters should be read in advance and transfers the logic for reading & decoding the Modbus packets to the server side.

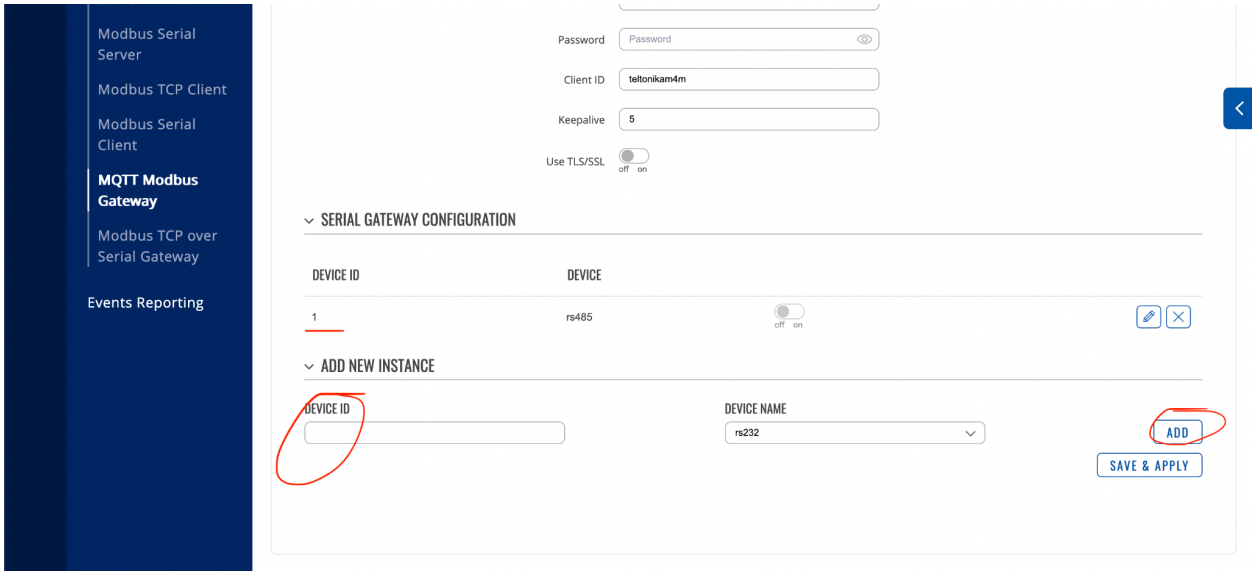
An example of this implementation can be found here : https://github.com/heinrich321/abb-teltonika

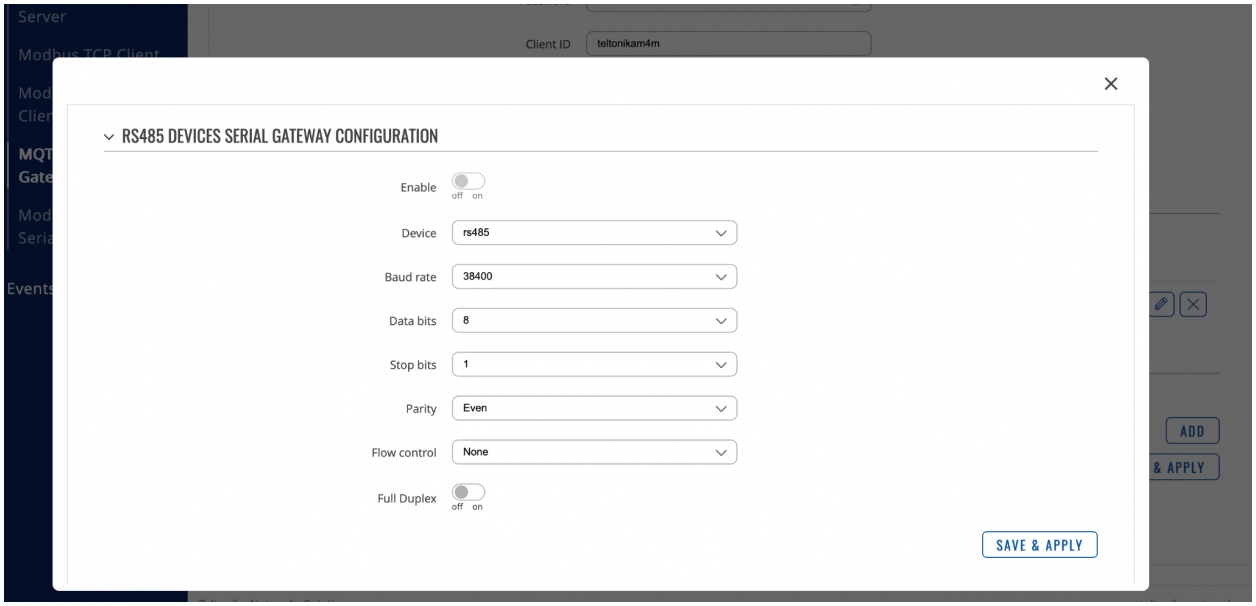
The gateway_client.py example will poll all the registers configured in m4m30.py at a set interval over MQTT and decode the responses.

OCTOCO CONSULTING (PTY) LTD.
VAT REG. NUMBER: 4110293455

# Teltonika Gateway Configuration

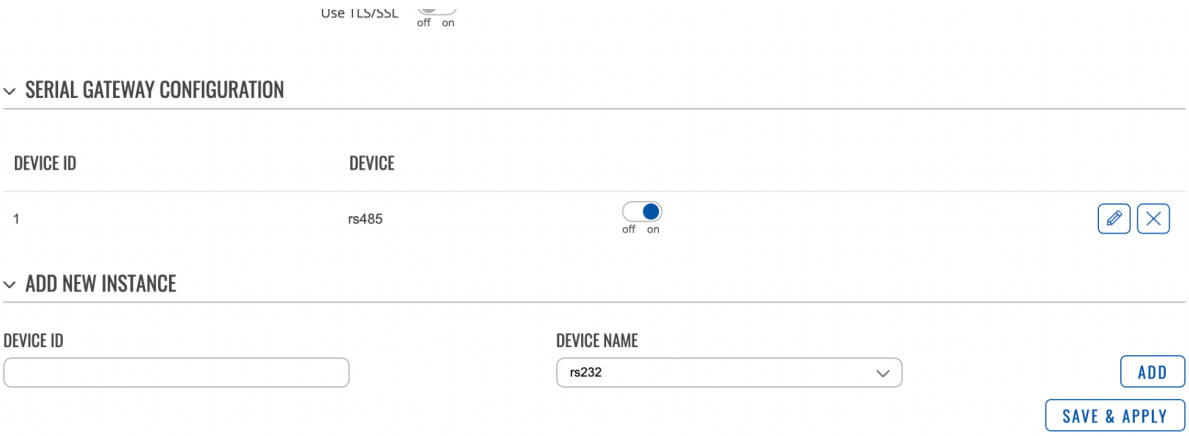





## MQTT GATEWAY

| | |
|---|---|
| Enable | off on |
| Host | broker.hivemq.com |
| Port | 1883 |
| Request topic | m4m-example/request |
| Response topic | m4m-example/response |
| QoS | Exactly once (2) |
| Username | Username |
| Password | Password |
| Client ID | teltonikam4m |
| Keepalive | 5 |
| Use TLS/SSL | off on |




## Command Structure

Command Structure: `1 <COOKIE> <SERIAL_DEVICE_ID> <TIMEOUT> <SLAVE_ID> <MODBUS_FUNCTION> <FIRST_REGISTER> <REGISTER_COUNT>`

| | |
|---|---|
| 1. Format version | 1 |
| 2. Cookie | from 0 to $2^{64}$ -1 |
| 3. Serial device ID | a string used to identify a serial device. Must match with Device ID field in MQTT Gateway page Serial gateway configuration section |
| 4. Timeout | timeout for Modbus connection, in seconds. Range [1..999]. |
| 5. Slave ID | Indicates to which slave request is sent |
| 6. Modbus function | Modbus task type that will be executed. Possible values are:<br>• 1 - read coils;<br>• 2 - read input coils;<br>• 3 - read holding registers;<br>• 4 - read input registers;<br>• 5 - set single coil;<br>• 6 - write to a single holding register;<br>• 15 - set multiple coils;<br>• 16 - write to multiple holding registers. |

| | |
|---|---|
| 7. First register | number (not address) of the first register/coil/input (in range [1..65536]) from which the registers/coils/inputs will be read/written to. |
| 8. Registry count | 1 - coil count (in range [1..2000]); must not exceed the boundary (first coil number + coil count <= 65537);<br>2 - input count (in range [1..2000]); must not exceed the boundary (first input number + input count <= 65537);<br>3 - holding register count (in range [0..125]); must not exceed the boundary (first register number + holding register count <= 65537);<br>4 - input register count (in range [0..125]); must not exceed the boundary (first register number + input register count <= 65537);<br>5 - coil value (in range [0..1]);<br>6 - holding register value (in range [0..65535]);<br>15 - coil count (in range [1..1968]); must not exceed the boundary (first coil number + coil count <= 65537); and coil values separated with commas, without spaces (e.g., 1,2,3,654,21,789); there must be exactly as many values as specified (with coil count); each value must be in the range of [0..1]. |

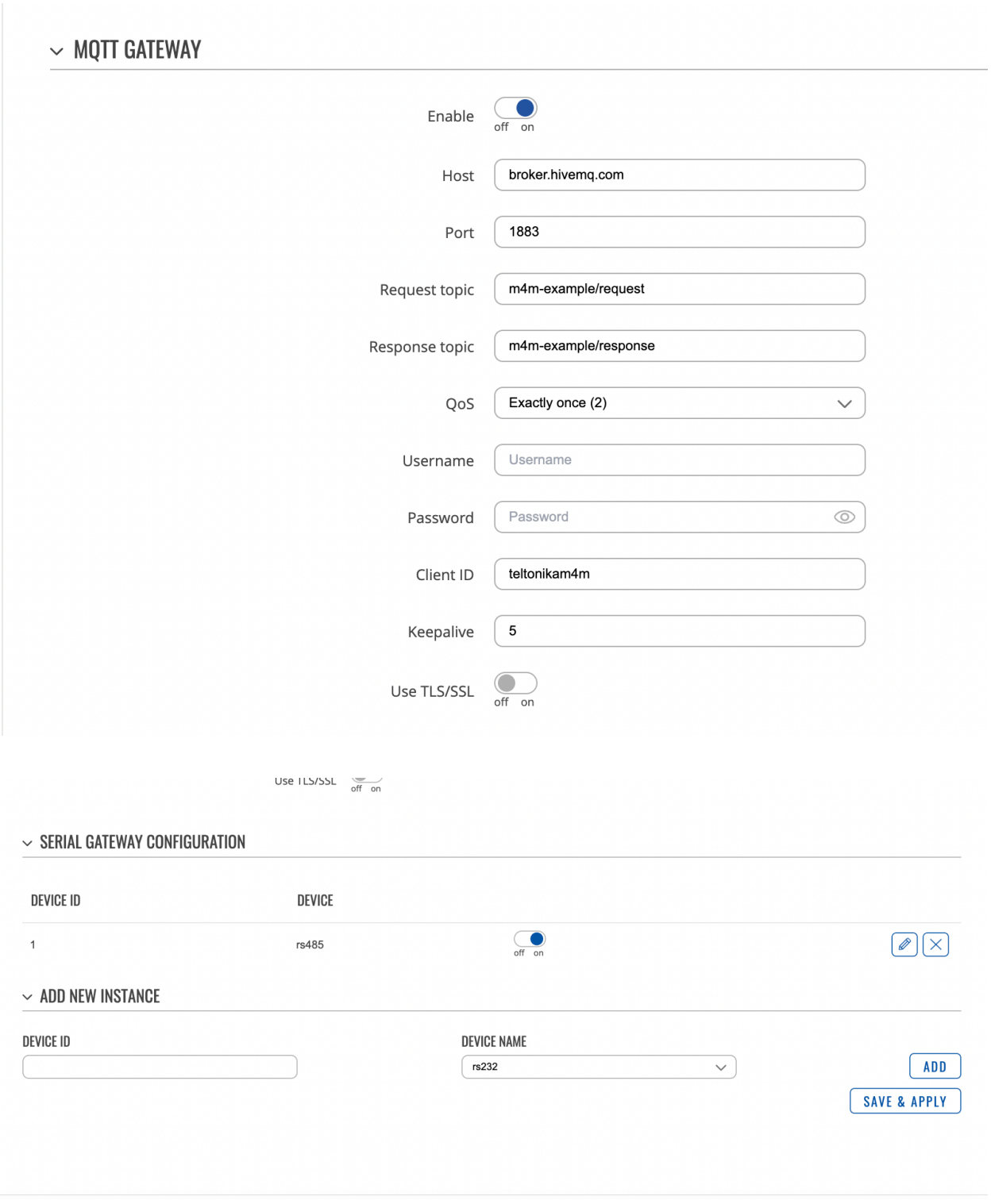
## Example

This example reads the frequency register:

| Parameter | Unit | Resolution | Data Type | Register (Hex) | Register (Dec) | Nr of quantities | Size | Nr of registers |
|---|---|---|---|---|---|---|---|---|
| Frequency | Hz | 0.01 | Unsigned | 5B32 | 23346 | 1 | 1 | 1 |

Example Request:
**Topic**: m4m-example/request
***Payload**: 1 1 1 1 1 3 23347 1*

Example Response:
**Topic**: m4m-example/response
**Response Payload**: 1 OK 5010

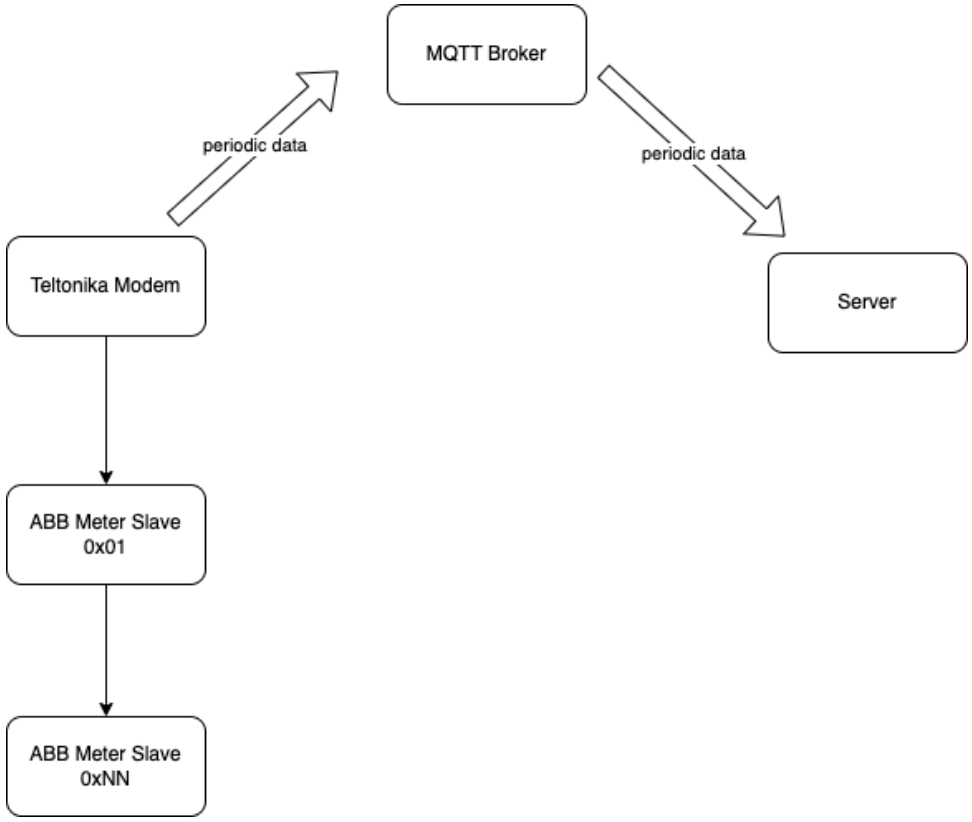

MQTT GATEWAY
Enable
off on
Host
broker.hivemq.com
Port
1883
Request topic
m4m-example/request
Response topic
m4m-example/response
QoS
Exactly once (2)
Username
Username
Password
Password
Client ID
teltonikam4m
Keepalive
5
Use TLS/SSL
off on
Use TLS/SSL
off on
SERIAL GATEWAY CONFIGURATION
DEVICE ID
DEVICE
1
rs485
off on
ADD NEW INSTANCE
DEVICE ID
DEVICE NAME
rs232
ADD
SAVE & APPLY

OCTOCO CONSULTING (PTY) LTD.
VAT REG. NUMBER: 4110293455

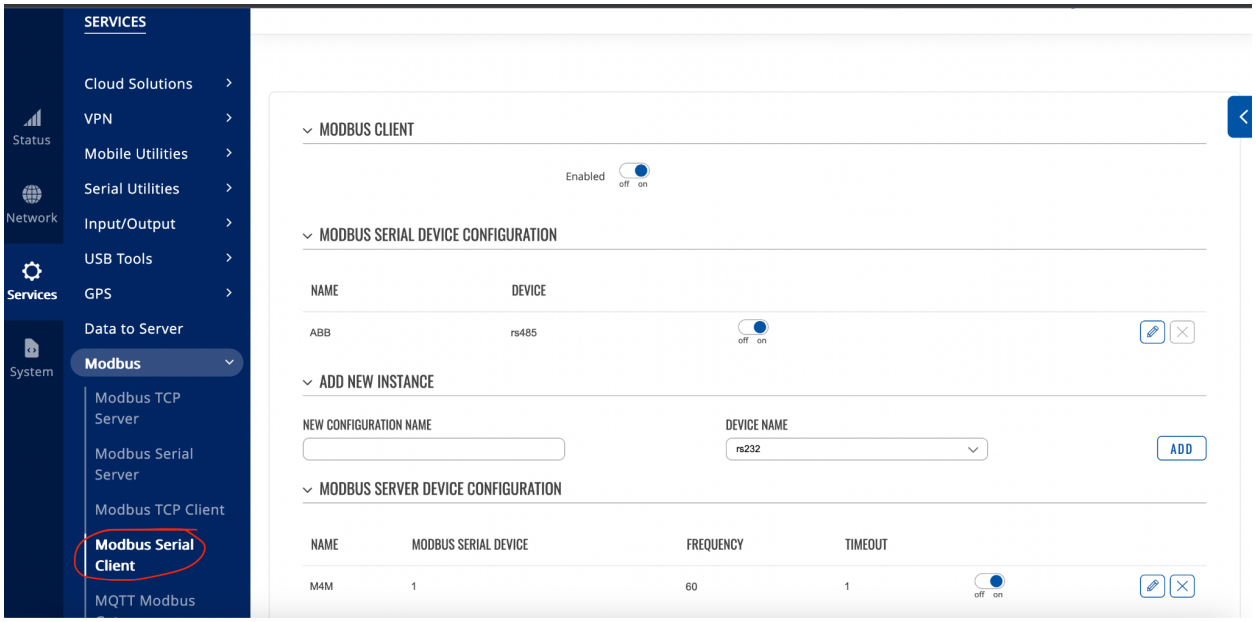
# Data to Server

## Data Flow

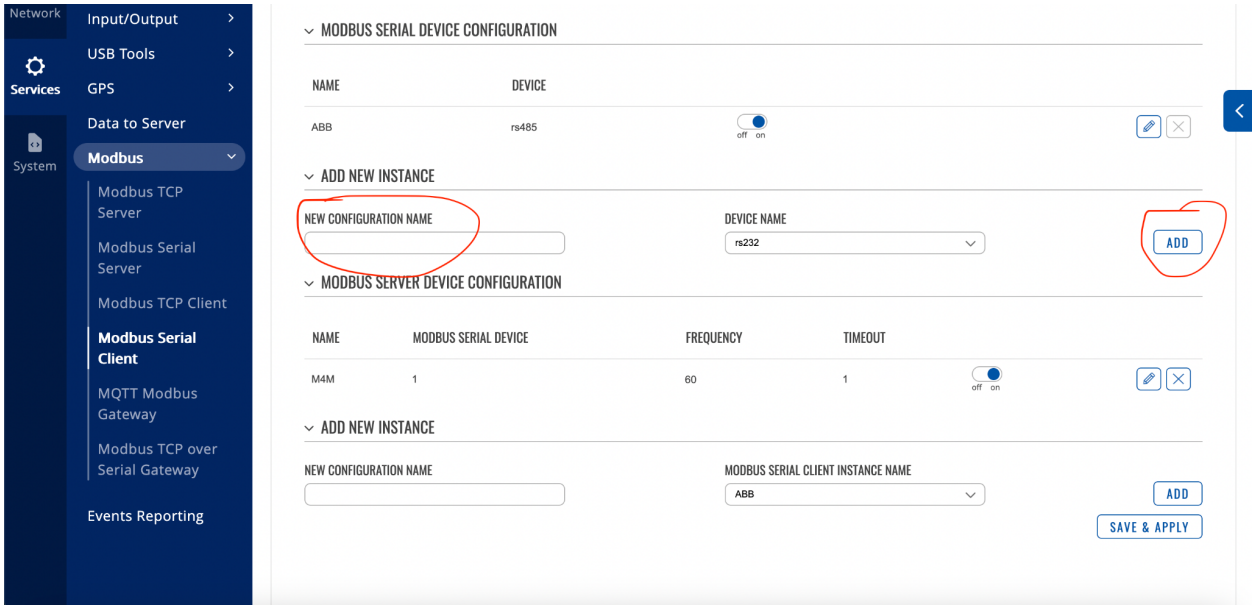
The data to the server sends all data accumulated over a period of time to the MQTT topic at a set interval.



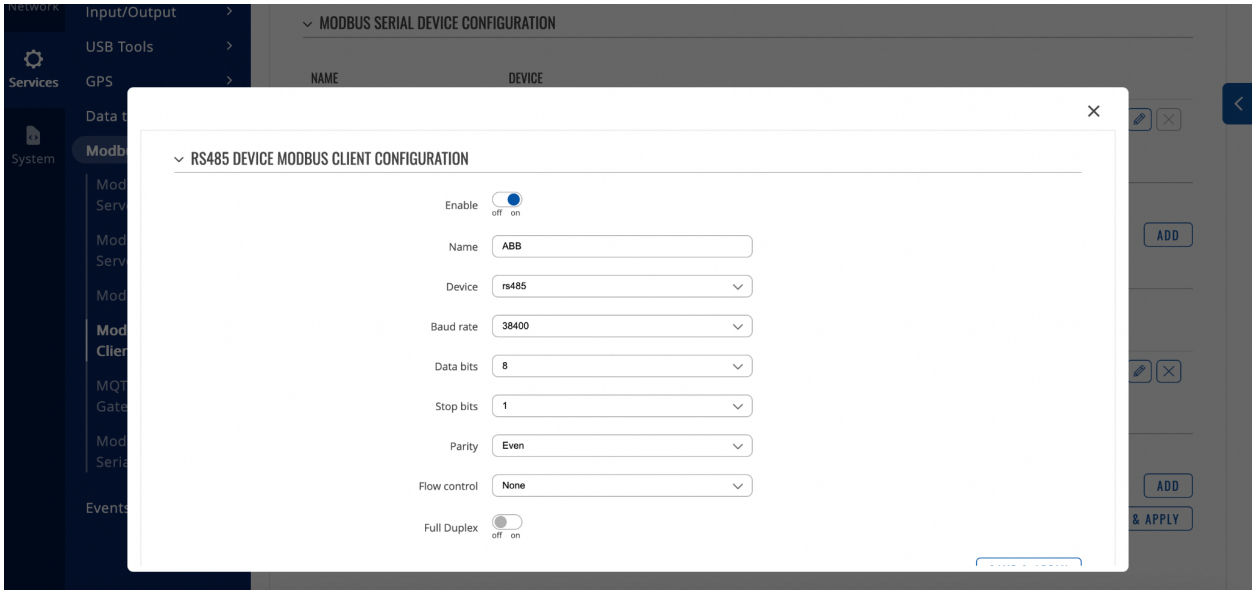

Navigate to Modbus Serial Client tab



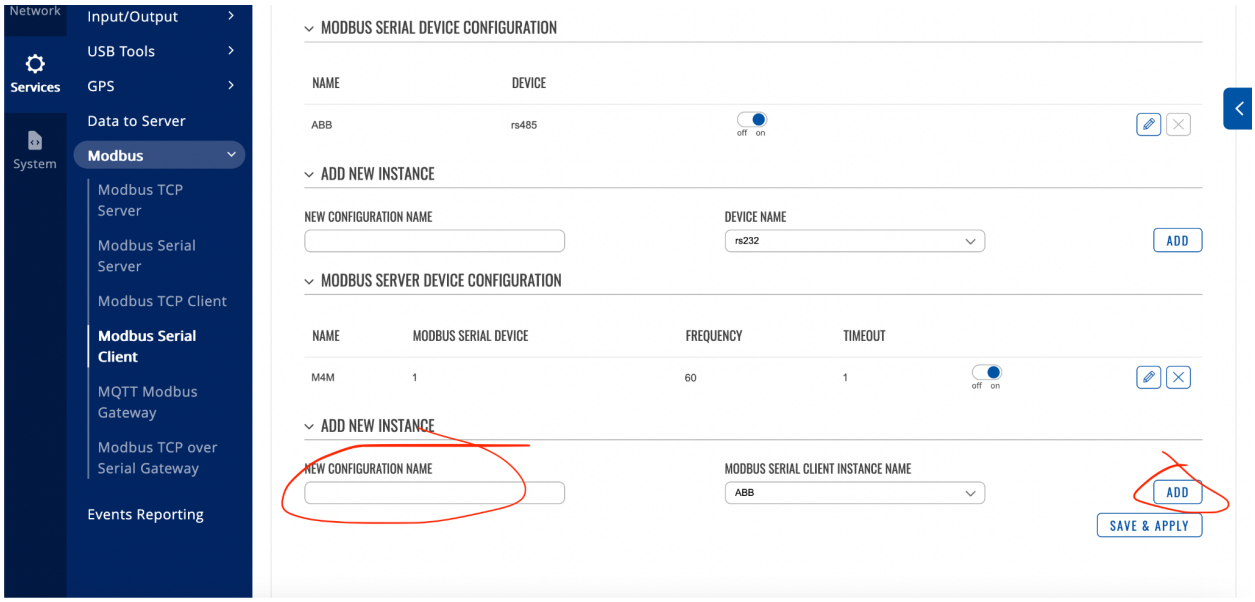

Add an Modbus client instance containing the settings for the slave devices



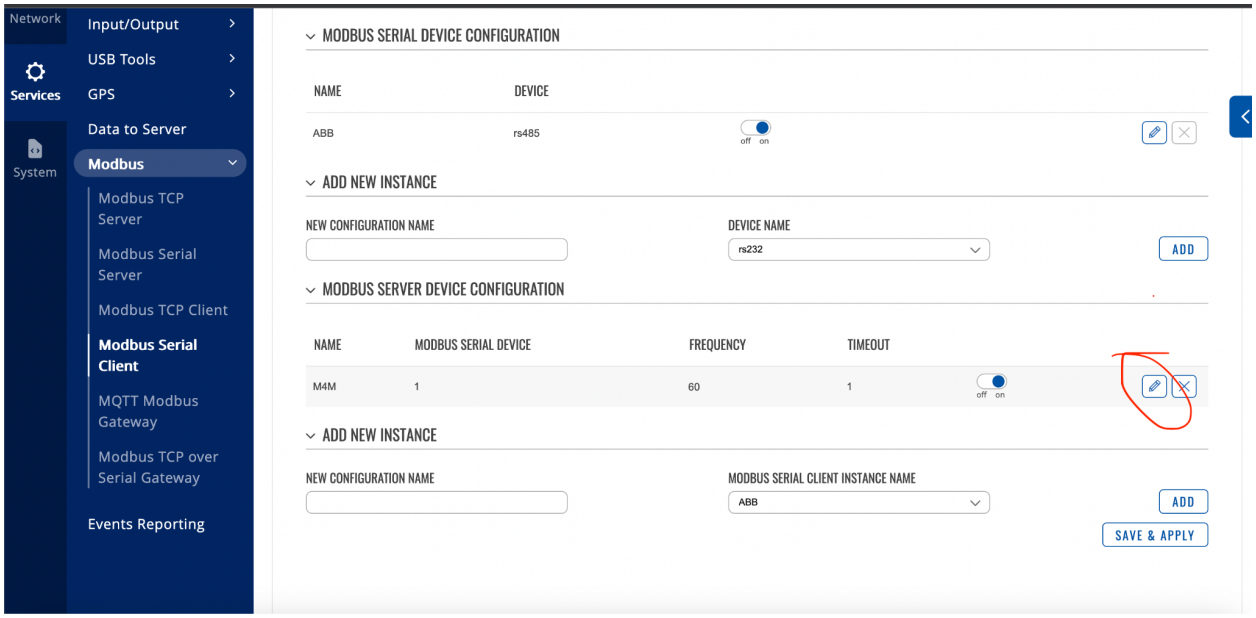

The configuration used for this specific ABB M4M Meter is the following:



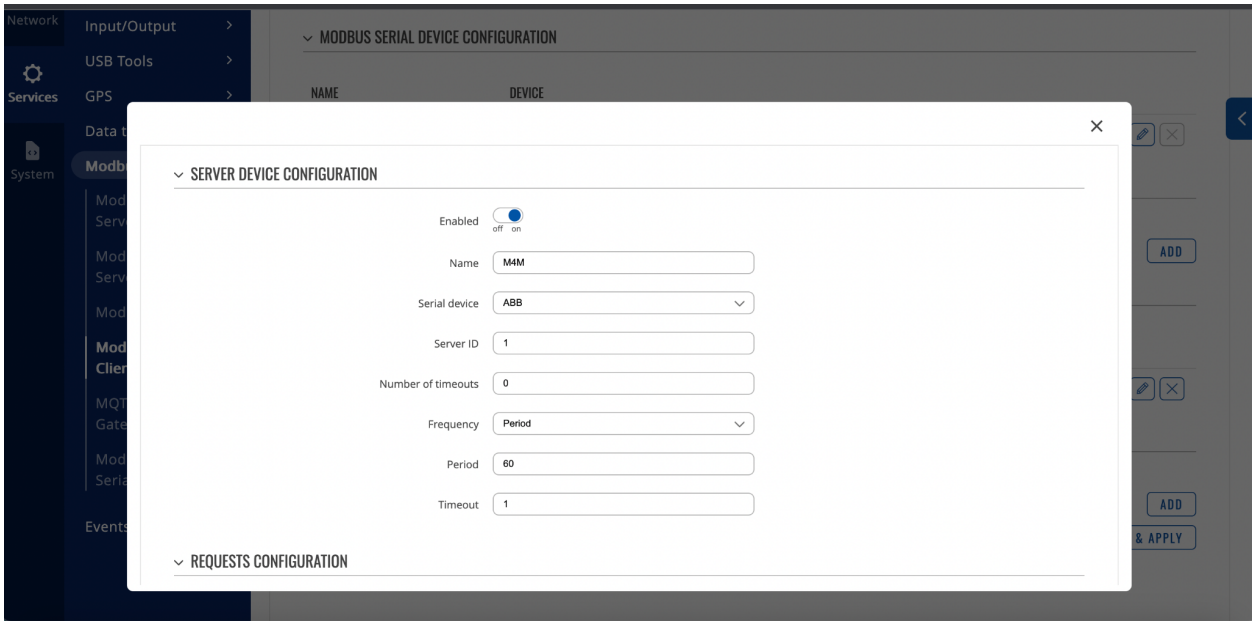

Add an instance. This will be done for every slave device connected to the bus



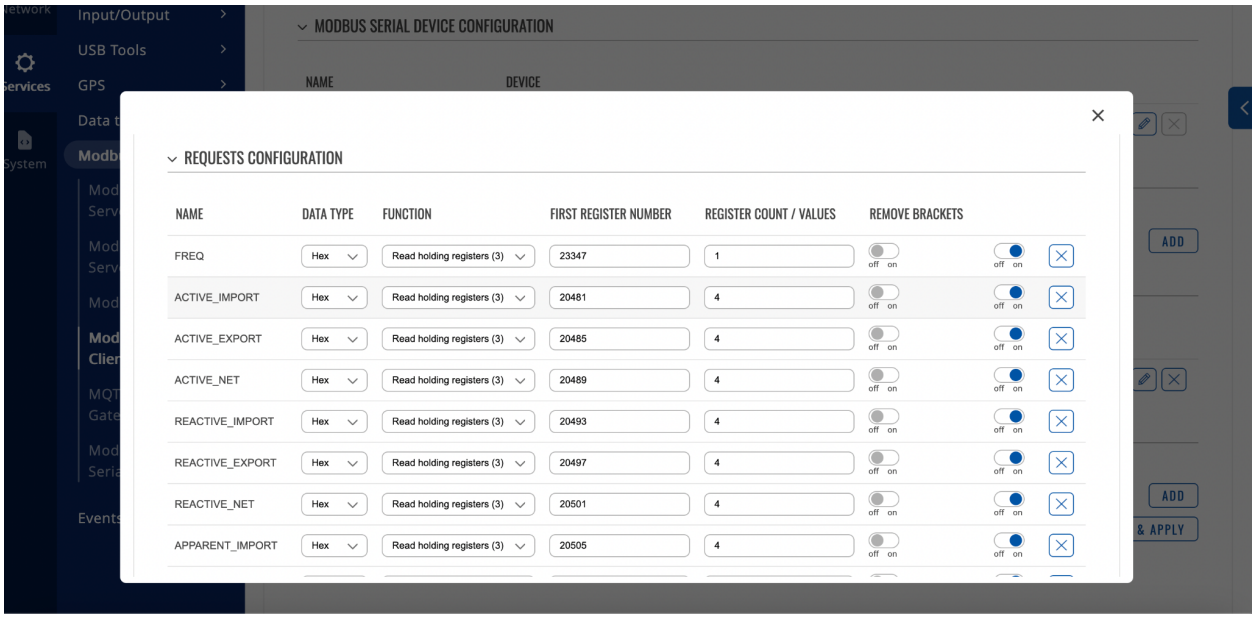

Edit the configuration for the slave device



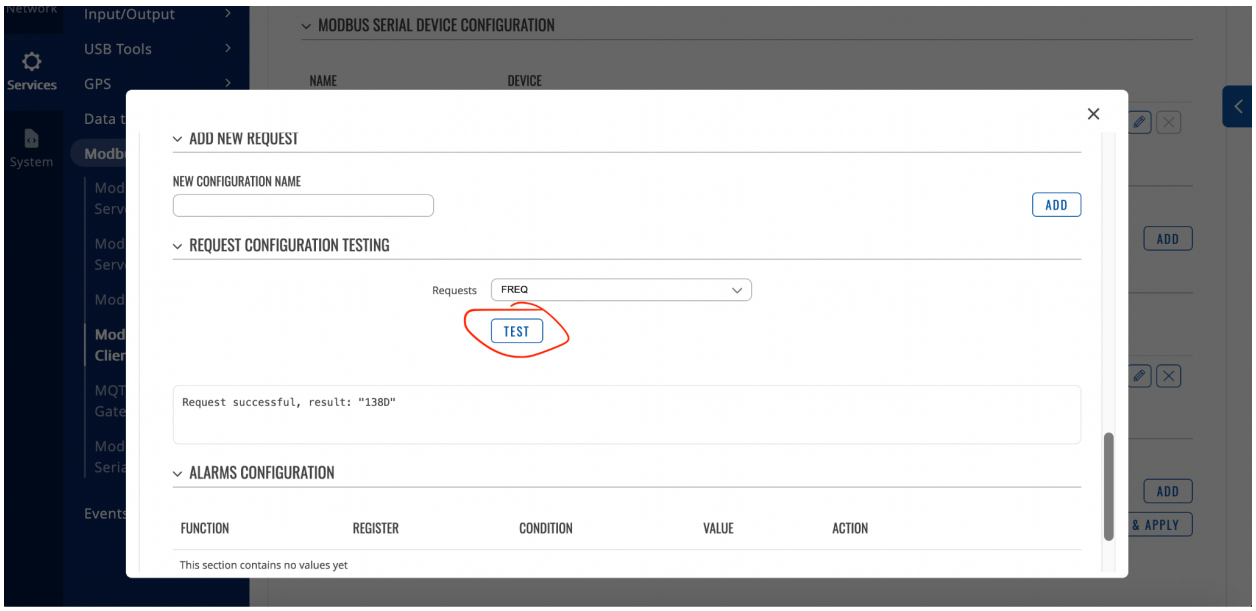

Configure the address of the slave, frequency of reads and timeouts



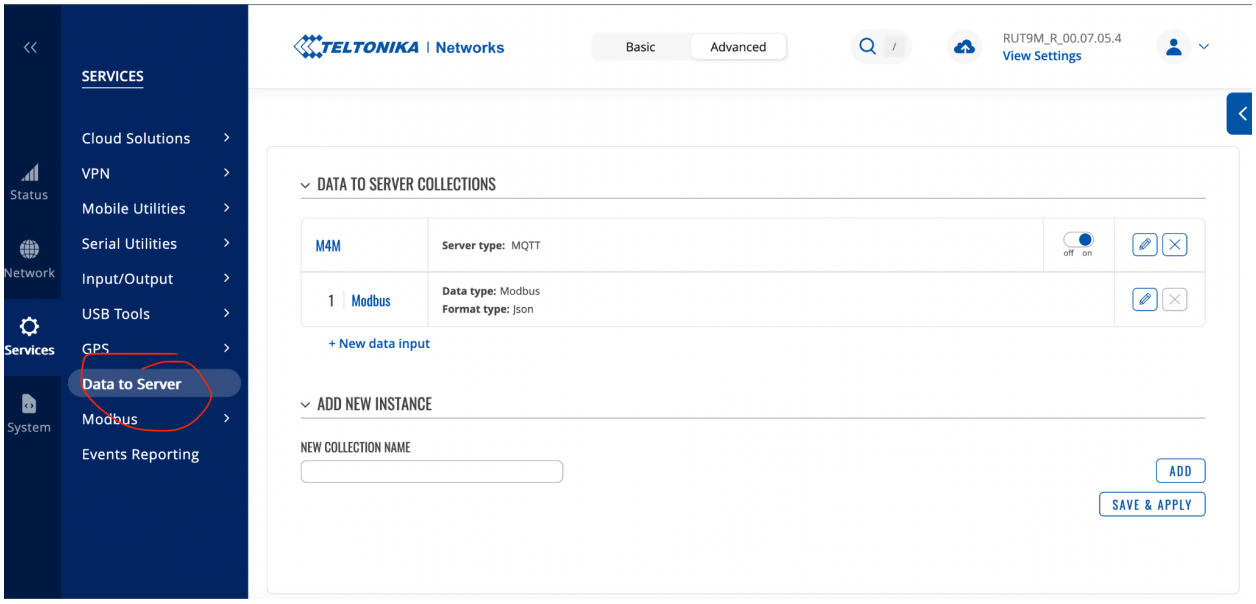
Configure the registers to be read. Pick the appropriate data type depending if you are doing server side decoding or decoding on the Teltonika device as described in the following sections.



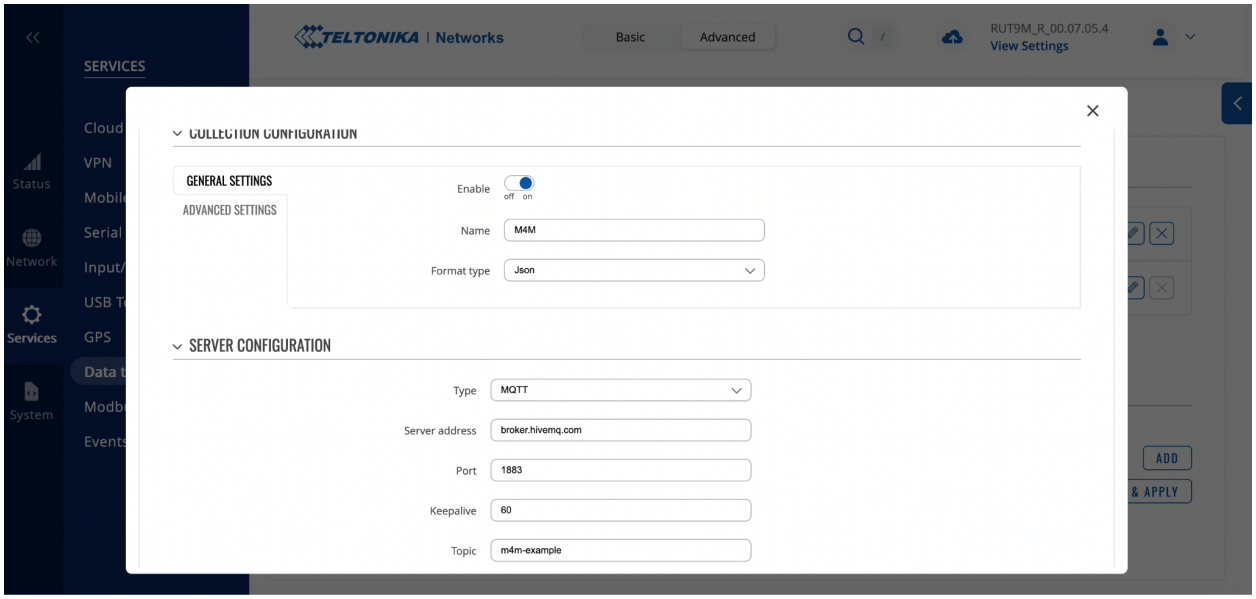

Test your configuration to make sure you are getting the appropriate response from the slave.



Next navigate to the Data to Server tab to configure the data transport settings over MQTT.



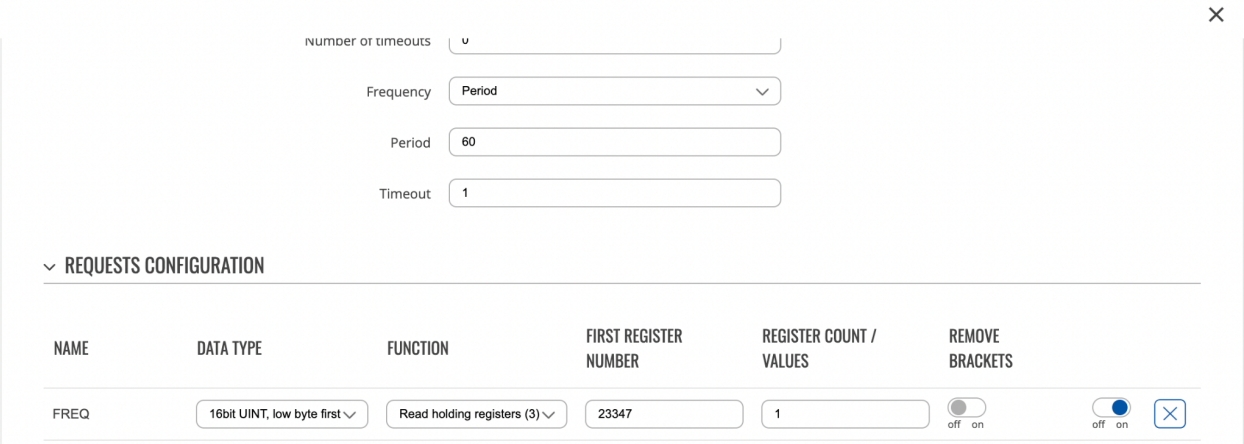

Add a new instance and configure the MQTT topic and frequency that data should be sent to the specific topic.



# Data to Server (MQTT - Decoded)

## Overview

The data to server MQTT method using pre-decoded as seen in the screenshot below:




Uses functionality on the Teltonika modem to decode the Modbus response. This however has limitations such as only supporting up to 32-bit numbers. The ABB M4M meter utilizes 64-bit numbers for multiple of the parameters and as such will limit functionality. It can still be used but will need additional post-processing since the data will come through in an array as shown in the example.

## Example MQTT Messages

### Topic

m4m-example

### Payload

{"Modbus":{"timestamp":1702826365,"date":"17/12/2023 15:19:25","bdate":null,"server_id":1,"bserver_id":1,"addr":23317,"baddr":null,"full_addr":"423317","size":5,"data":"[0,0]","raw_data":null,"server_name":"M4M","name":"CURRENT_L3"}}

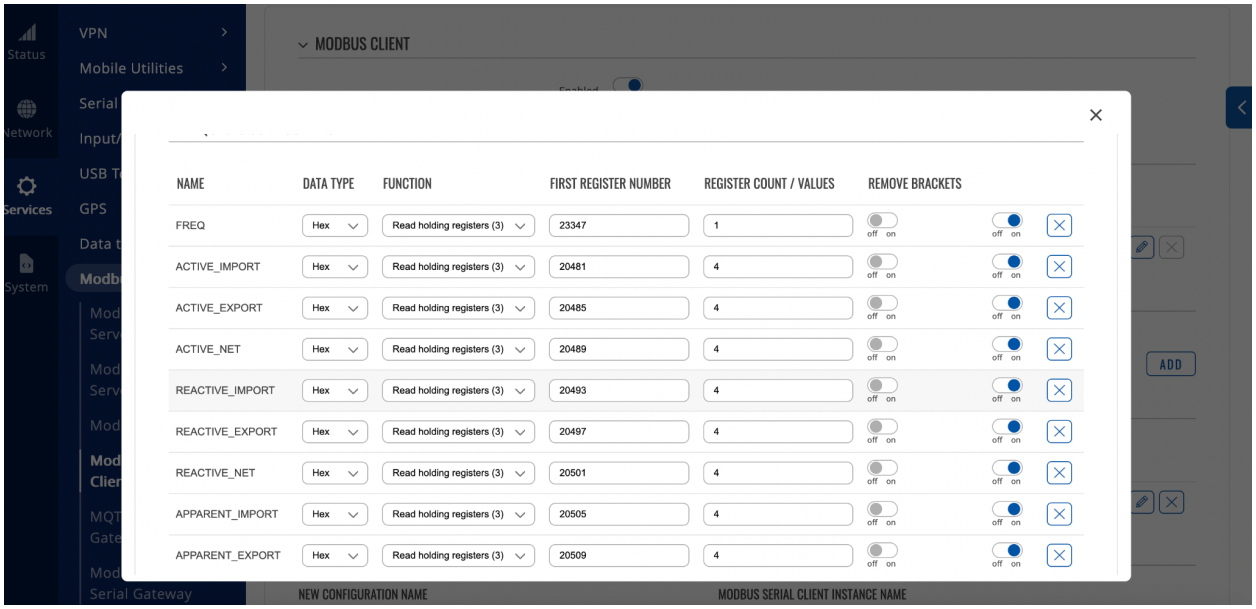
# Data to Server (MQTT - Hex)

## Overview

The hex data-type functionality on the Teltonika device leaves the Modbus response raw hex format and allows for more fine-grained control over post-processing.

An example of this implementation can be found here : https://github.com/heinrich321/abb-teltonika

The modbus_client.py example subscribes to the data-to-server topic and parses the incoming hex data according to the register map as set out in m4m30.py.




## Topic

m4m-example

## Payload

{"Modbus":{"timestamp":1702826551,"date":"17/12/2023 15:22:31","bdate":null,"server_id":1,"bserver_id":1,"addr":20513,"baddr":null,"full_add

r":"420513","size":18,"data":"\"000000000003BCD1\"","raw_data":null,"server_nam
e":"M4M","name":"APPARENT_NET"}}