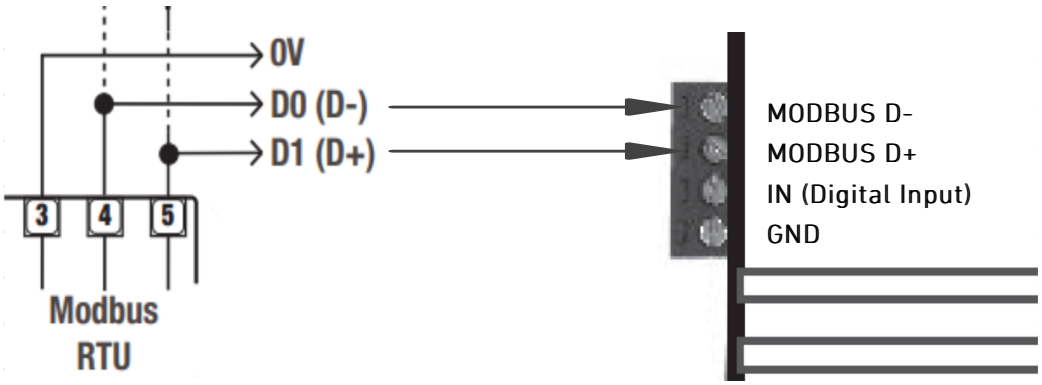
Connecting a Schneider energy meter to an AWS IoT Service with the FATBOX G3 Gateway

Cloud monitoring energy usage via Modbus RTU

- In this tutorial, we will set up a Schneider iEM2150 Energy Meter as a **Modbus RTU slave** and interface to our FATBOX G3 gateway **Modbus master**.
- The gateway is able to support up to 32 Modbus devices, including meters and remote I/O terminal. The standard industrial Modbus/RTU protocol runs on Serial RS-485 interface, providing a robust & reliable interface.
- We will also look at **connecting the G3 IOT Gateway to an AWS IoT Service**. Other supported clients are Azure IoT Hub and Ubidots.
- Users have the option of backhaul via either wired Ethernet, WIFI or cellular (4G/3G) for remote sites/redundancy.

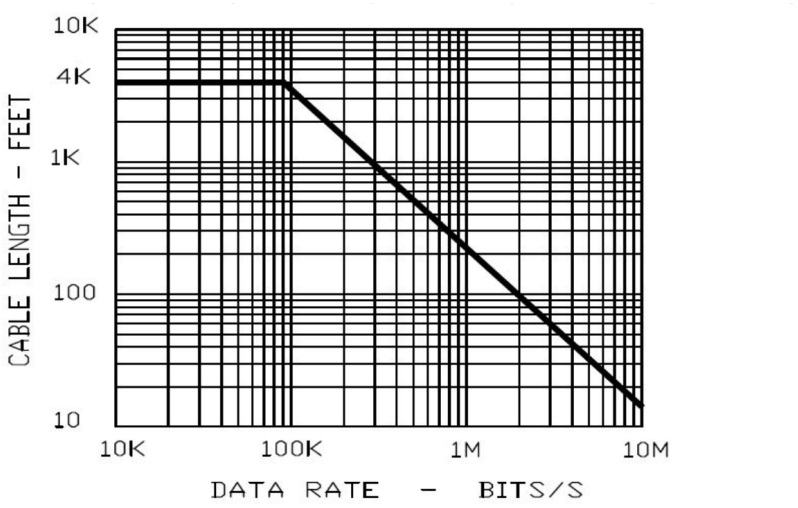
Hardware Wiring



Schneider iEM2150 *FATBOX G3*

Above shows the Serial RS-485 connection usually implemented using shielded twisted pair cable.

Note maximum cable length depends on baudrate and cable quality, e.g. using 24AWG shielded twisted pair, about 100Kbit/sec at 1000m.

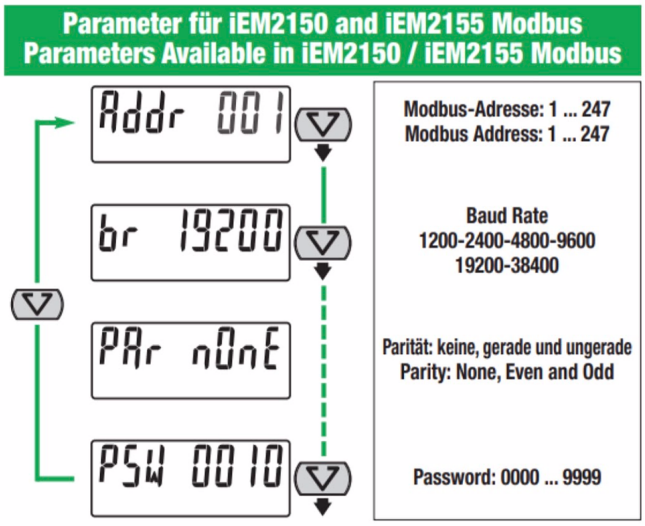


DATA SIGNALLING RATE VESUS CABLE LENGTH FOR BALANCED INTERFACE USING 24 AWG TWISTED PAIR CABLE

Image courtesy of eng-tip.com

Schneider Meter Configuration

Use the front display/button to set the serial parameters as below.



All slave devices connected on the RS-485 network must each have a unique address ranging from 1 to 247. Also, the baudrate and parity for all slave devices and master must be the same.

Connect Meter to Gateway

Our gateway uses a configuration file (\user\iotasset.txt) to map the required Modbus registers to be read during each polling cycle. This flexibility allows wide compatibility with most Modbus RTU devices from different manufacturers.

To write this iotasset.txt file we will be referencing the Modbus register table from the product user manual (see sample below):

Meter Data

Current, voltage, power, power factor and frequency

Register Address	Action (R/W/WC)	Size	Type	Units	Description
Current					
3000	R	2	Float32	A	Current
Voltage					
3028	R	2	Float32	V	Voltage
Power					
3054	R	2	Float32	kW	Active Power
3068	R	2	Float32	kVAR	Reactive Power
3076	R	2	Float32	kVA	Apparent Power

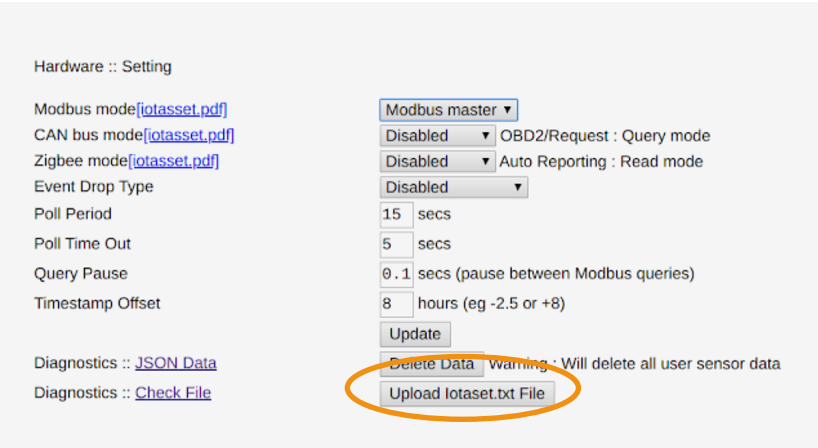
And here is a sample of the iotasset.txt. In the top example, we are configuring to read Current (A) register frm meter Modbus address 1, register 3000 (Schneider mandates a -1 offset) and name data field name, as "M1_Current".

The gateway is able to apply a customer multiplier (x0.1 in this case) and offset (0 in this case) to match the requirements of their cloud service.

```
MBM_START

TYPE,R
ADDR,1
MBFC,3
REGS,2999,2,FLOAT32ABCD,0.1,0
Key,M1_Current
Unit,A

TYPE,R
ADDR,1
MBFC,3
REGS,3027,2,FLOAT32ABCD
Key,M1_Voltage
Unit,V

TYPE,R
ADDR,1
MBFC,3
REGS,3053,2,FLOAT32ABCD
Key,M1_ActivePower
Unit,kW

MBM_STOP
```

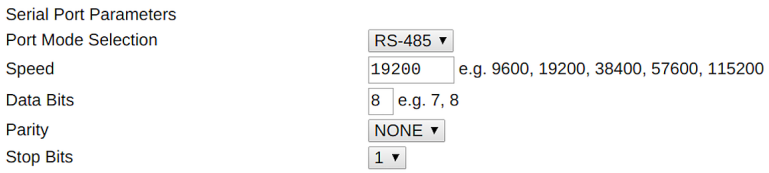
Once you have configured your iotasset.txt file the new device configurations can be updated to your FATBOX gateway securely over the air. The gateway HTTPS connected webconsole must be accessible, either via local network (LAN or WLAN) and over cellular network with assigned public IP address (If you do not have an Internet connection, you can follow the alternative steps here).

Log into your web console and go to the <IOT Hardware> tab, click on 'UPDATE FIRMWARE' in the Firmware Update section

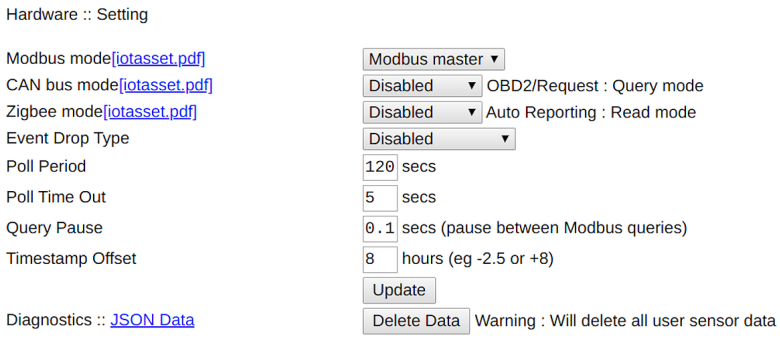


In the new window, click on 'CHOOSE FILE' and select from your local folder the updated iotasset.txt file then click on 'UPLOAD FIRMWARE FILE'. After the upload is successful you will need to close the page and log in again for security purpose.

We will now configure the gateway's serial port to operate as required for the attached Modbus devices. Go to the *Port Settings* tab and enter the following settings.



Then, go to the *IOT Hardware* menu to set the Hardware interface to "Modbus master" and register the polling configuration as required by the user (Note that all other interfaces like CAN bus and Zigbee can also run concurrently).



After these settings are updated, **REBOOT** the gateway. The user can then check the Modbus sensor data collected (JSON Data) or delete for testing.

Connecting to an AWS IoT Service

Now that the connectivity between the Schneider meter and the G3 gateway has been set, we will then look at getting the data onto a designated cloud service. We support an open customer software and 3rd party API (e.g. REST API for HTTP, HTTPS and MQTT) integration to connect to a chosen cloud data or dashboard service (or to use for on board data processing).

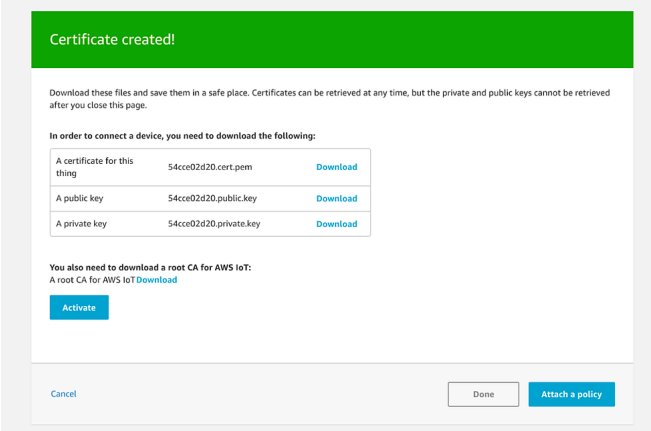
Direct deployment on Azure, AWS IoT or Ubidots can also be done as these IoT clients have been integrated on the G3 Gateway. For this project, we are going to showcase using AWS IoT (note that an AWS account is required to continue this tutorial).

1. Create a new AWS IoT Thing

First log into your AWS IoT Management Console and create a Thing:
AWS IoT > Manage > Thing > Create

Next go to:
Secure > Certificates
Then download the 'Certificate' & 'Private Key File'.

Ensure the Certificate is "Active", otherwise activate it under ACTIONS in
Things > Your Certificate > Security



2. Update your G3 IoT Gateway

Create a new local folder and name it "AWS".
Save the downloaded certificate files into this folder and rename them as the following:

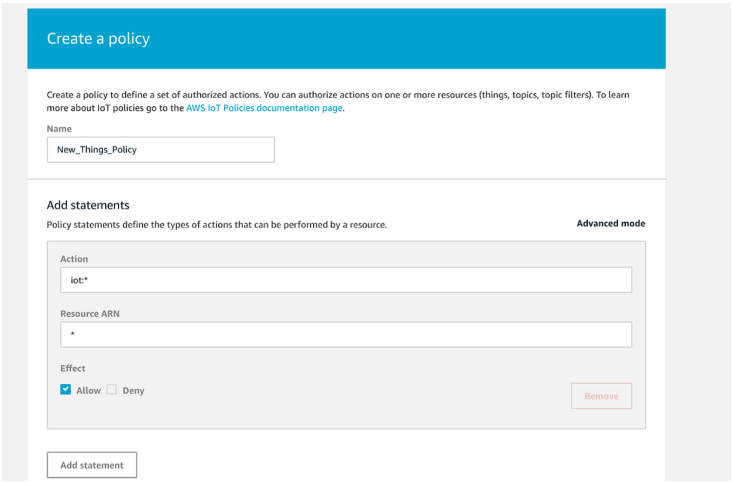
"**certificate.pem.crt**"
"**private.pem.key**"

Next zip the folder (ensure that you zip the entire folder and not just the files inside).

Then log into your FATBOX G3 web console and go to the <Management> tab.
Patch the zipped folder to the gateway using the UPDATE FIRMWARE button.

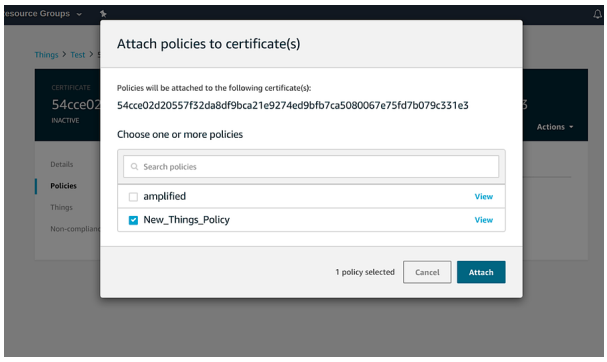
3. Create a Security Policy for your Thing

In your AWS IoT Management Console go to:
Secure > Policies > Create



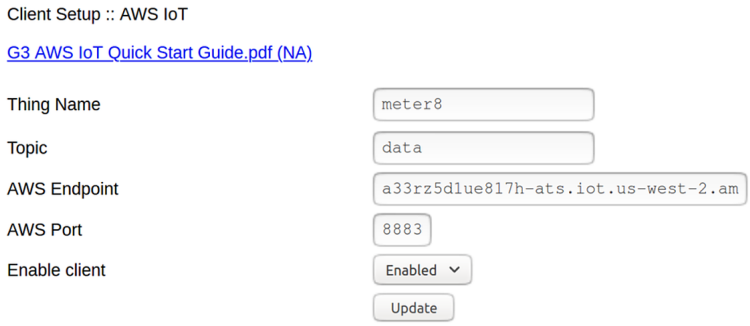
At <Things>, select your new Thing then click on
Security > Certificates > Policies

Under <Actions>, choose to Attach a Policy:

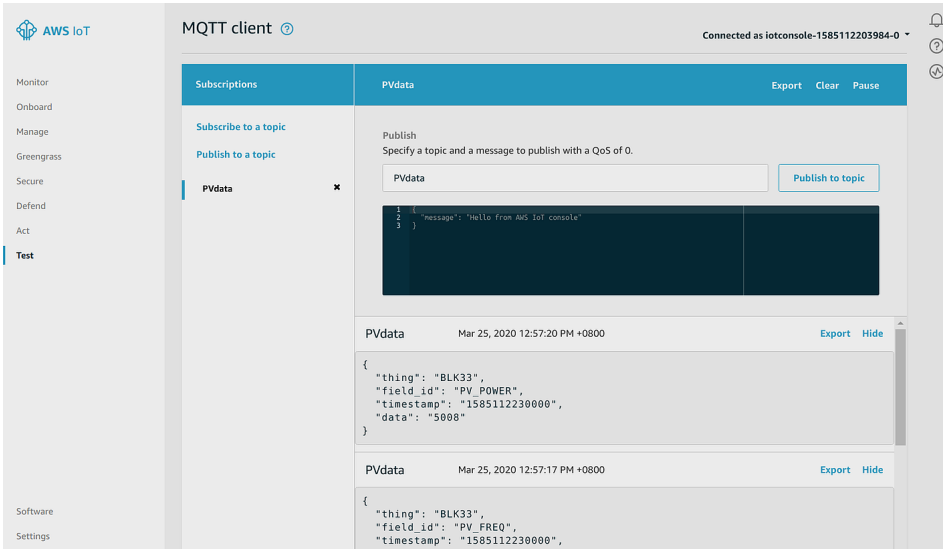


4. Configure your G3 IoT Gateway to send data to your AWS Account

Now, you are ready to configure your gateway to send AWS IoT endpoint to feed data to your AWS applications. In the FATBOX G3 web configuration menu, go to the <IoT Client> tab and configure your AWS client settings according as per your AWS end point and Thing settings. Then REBOOT your gateway.



Next go back to go your AWS IoT console and subscribe to the Topic to "Test" that data is being received.



Congratulations! You have succesfully sent your Modbus or CAN bus data to your AWS IoT endpoint and ready to, for example, push the data to a S3 bucket using a Rule in 'Act > Rules'.

The FATBOX G3 AWS IoT client side is built using AWS IoT Device SDK for Python and users are free to install, modify our device client codes for enhanced edge capabilities or other required functionalities.

See Also:: Our guides below for connecting to other supported IoT platforms includes Azure IoT and Ubidots. Here is a sample dashboard from Ubidots.