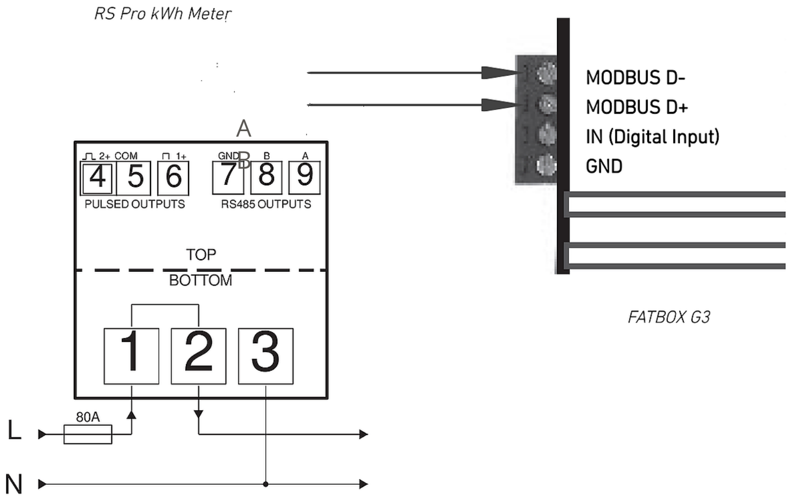
Connecting a RS Pro kWh energy meter to AWS IoT platform with the FATBOX G3 Gateway

Cloud monitoring energy usage via Modbus RTU

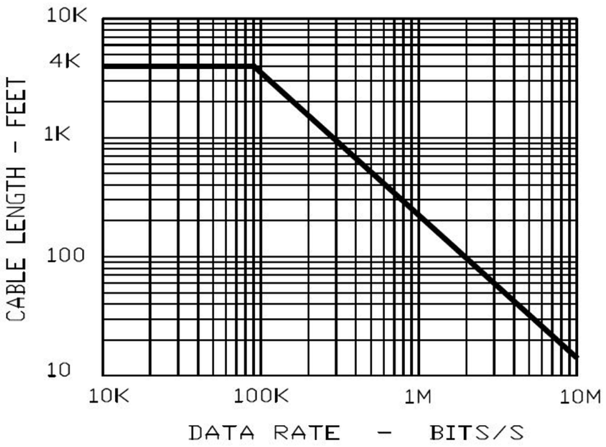
- In this tutorial, we will set up a RS Pro 45A kWh Energy Meter as a **Modbus RTU slave** and interface to our FATBOX G3 gateway **Modbus master**.
- The gateway is able to support up to 32 Modbus devices, including meters and remote I/O terminal. The standard industrial Modbus/RTU protocol runs on Serial RS-485 interface, providing a robust & reliable interface.
- We will also look at **connecting the G3 IOT Gateway to AWS IoT**. Supported clients are AWS IoT, 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



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.

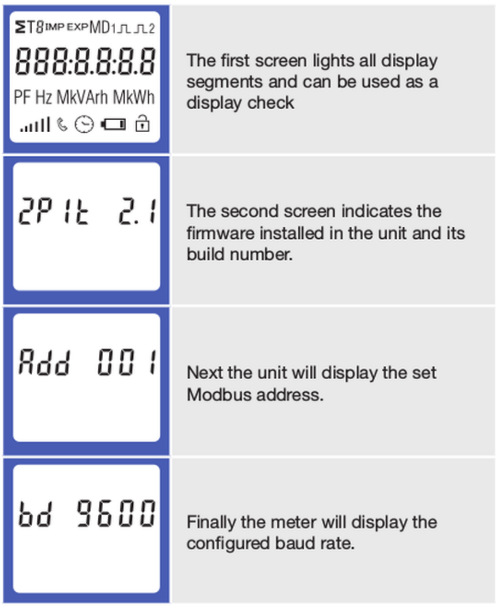


DATA SIGNALLING RATE VESUS CABLE LENGTH FOR BALANCED INTERFACE USING 24 AWG TWISTED PAIR CABLE

Image courtesy of eng-tip.com

RS Pro Meter Configuration

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

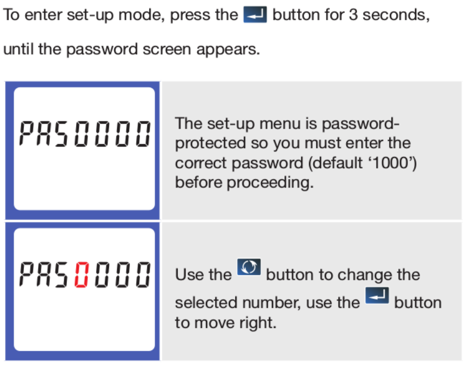


The first screen lights all display segments and can be used as a display check

The second screen indicates the firmware installed in the unit and its build number.

Next the unit will display the set Modbus address.

Finally the meter will display the configured baud rate.



To enter set-up mode, press the ↵ button for 3 seconds, until the password screen appears.

The set-up menu is password-protected so you must enter the correct password (default '1000') before proceeding.

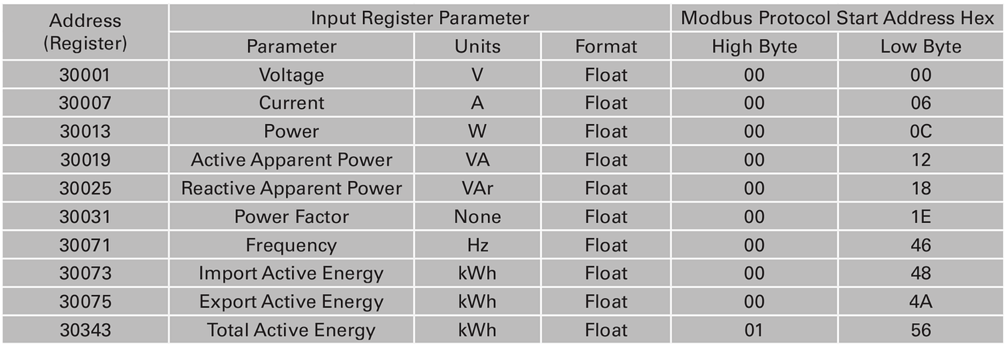
Use the ⟳ button to change the selected number, use the ↵ button to move right.

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 device configuration file called iotasset 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 setup this device configuration file, we will be referencing the Modbus register table from the product user manual (see sample below):



Address (Register)	Input Register Parameter			Modbus Protocol Start Address Hex	
	Parameter	Units	Format	High Byte	Low Byte
30001	Voltage	V	Float	00	00
30007	Current	A	Float	00	06
30013	Power	W	Float	00	0C
30019	Active Apparent Power	VA	Float	00	12
30025	Reactive Apparent Power	VAr	Float	00	18
30031	Power Factor	None	Float	00	1E
30071	Frequency	Hz	Float	00	46
30073	Import Active Energy	kWh	Float	00	48
30075	Export Active Energy	kWh	Float	00	4A
30343	Total Active Energy	kWh	Float	01	56

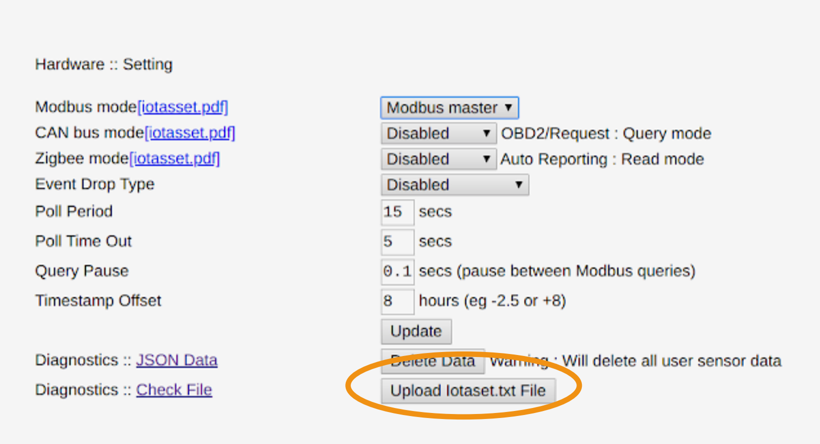
And here is a sample of the iotasset. In the top example, we are configuring to read Current (A) register from meter Modbus address 1, register 30007 and name data field name, as "M1_Current".

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 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

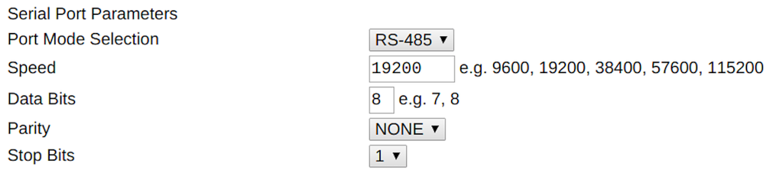


Hardware :: Setting

Modbus mode[iotasset.pdf]	Modbus master ▾
CAN bus mode[iotasset.pdf]	Disabled ▾ OBD2/Request : Query mode
Zigbee mode[iotasset.pdf]	Disabled ▾ Auto Reporting : Read mode
Event Drop Type	Disabled ▾
Poll Period	15 secs
Poll Time Out	5 secs
Query Pause	0.1 secs (pause between Modbus queries)
Timestamp Offset	8 hours (eg -2.5 or +8)
	Update
Diagnostics :: JSON Data	Delete Data Warning : Will delete all user sensor data
Diagnostics :: Check File	Upload Iotaset.txt File

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.

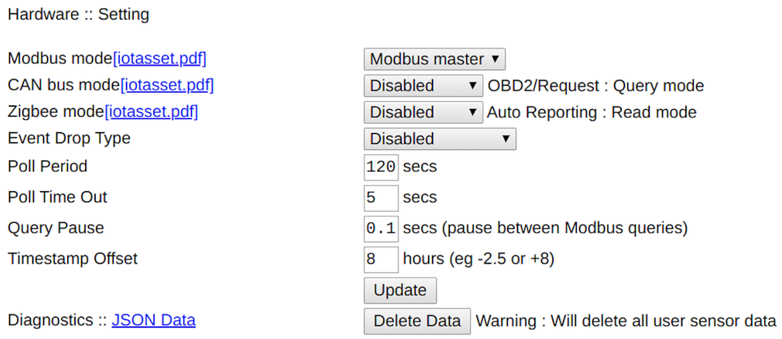
We will now configure the gateway's serial port to operate as required for the attached Modbus devices. Go to the *Port Settings* tab and enter the following settings.



Serial Port Parameters

Port Mode Selection	RS-485 ▾
Speed	19200 e.g. 9600, 19200, 38400, 57600, 115200
Data Bits	8 e.g. 7, 8
Parity	NONE ▾
Stop Bits	1 ▾

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).



Hardware :: Setting

Modbus mode[iotasset.pdf]	Modbus master ▾
CAN bus mode[iotasset.pdf]	Disabled ▾ OBD2/Request : Query mode
Zigbee mode[iotasset.pdf]	Disabled ▾ Auto Reporting : Read mode
Event Drop Type	Disabled ▾
Poll Period	120 secs
Poll Time Out	5 secs
Query Pause	0.1 secs (pause between Modbus queries)
Timestamp Offset	8 hours (eg -2.5 or +8)
	Update
Diagnostics :: JSON Data	Delete Data Warning : Will delete all user sensor data

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

Connect to a Cloud Service

Now that the connectivity between the RS Pro Meter and the G3 gateway has been set, we will then look at getting the data onto a suitably designed dashboard for monitoring on a custom cloud platform like Azure IoT, AWS IoT, Google Compute *or* an end solution like Ubidots.

For this project, we are going to showcase using AWS IoT cloud platform.

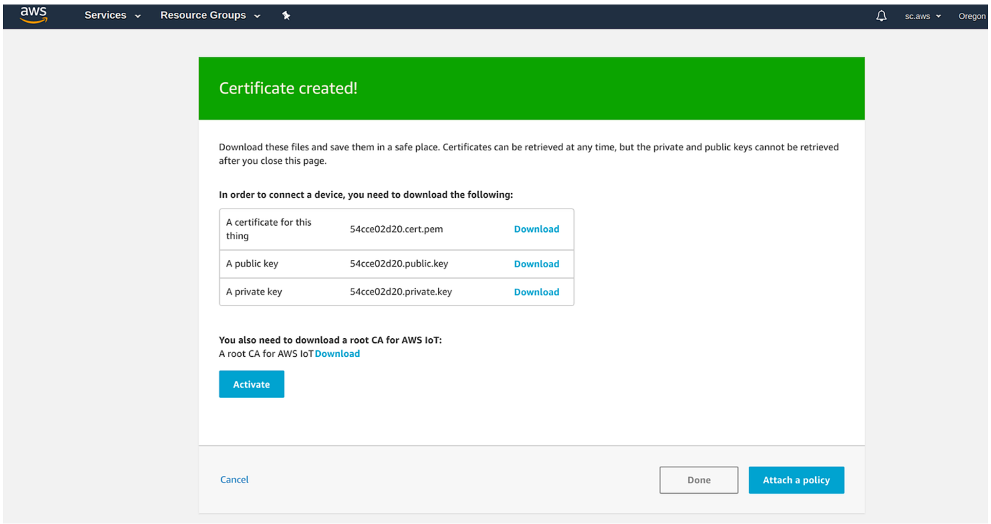
Contents

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



Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	54cce02d20.cert.pem	Download
A public key	54cce02d20.public.key	Download
A private key	54cce02d20.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT Download

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:

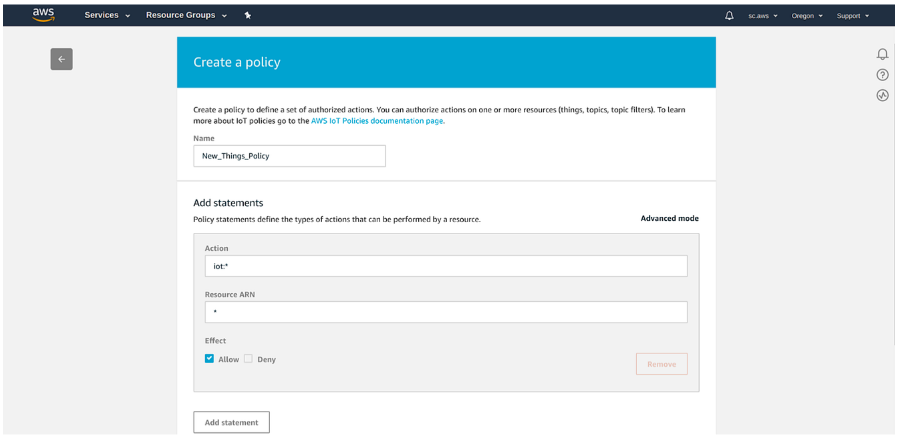
"**certificate.pem.crt**"
"**private.pem.key**"

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



Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: New_Things_Policy

Add statements

Policy statements define the types of actions that can be performed by a resource.

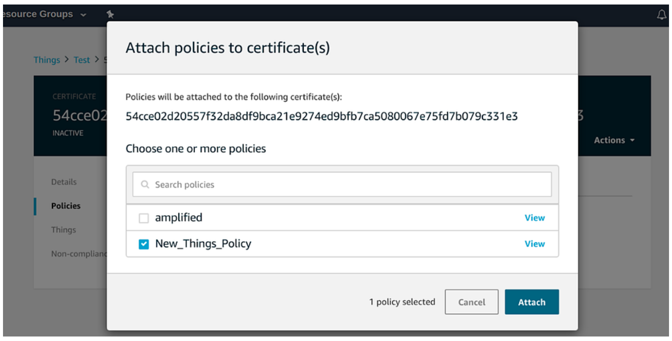
Action: iot:*

Resource ARN: *

Effect: ☑ Allow ☐ Deny

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

Under <Actions>, choose to Attach a Policy:



Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
54cce02d20557f32da8df9bca21e9274ed9bfb7ca5080067e75fd7b079c331e3

Choose one or more policies

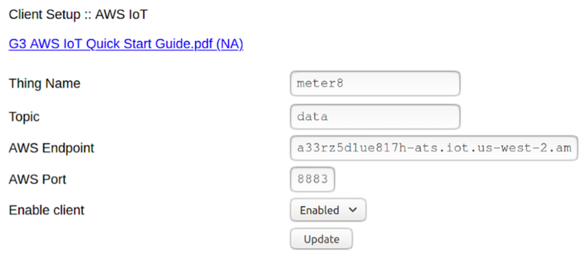
Search policies

□ amplified View
☑ New_Things_Policy View

1 policy selected Cancel Attach

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.

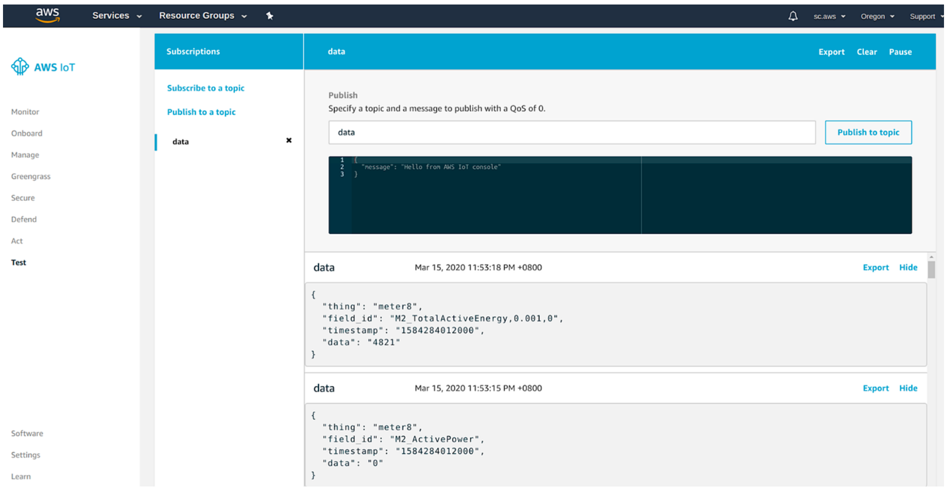


Client Setup :: AWS IoT

G3 AWS IoT Quick Start Guide.pdf (NA)

Thing Name	meter8
Topic	data
AWS Endpoint	a33rz5d1ue817h-ats.iot.us-west-2.am
AWS Port	8883
Enable client	Enabled

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



Publish
Specify a topic and a message to publish with a QoS of 0.

data

```
{
  "message": "Hello from AWS IoT console"
}
```

data Mar 15, 2020 11:53:18 PM +0800

```
{
  "thing": "meter8",
  "field_id": "M2_TotalActiveEnergy,0.001,0",
  "timestamp": "1584284012000",
  "data": "4821"
}
```

data Mar 15, 2020 11:53:15 PM +0800

```
{
  "thing": "meter8",
  "field_id": "M2_ActivePower",
  "timestamp": "1584284012000",
  "data": "0"
}
```

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.