SmartServer IoT LoRa to LON and BACnet Gateway

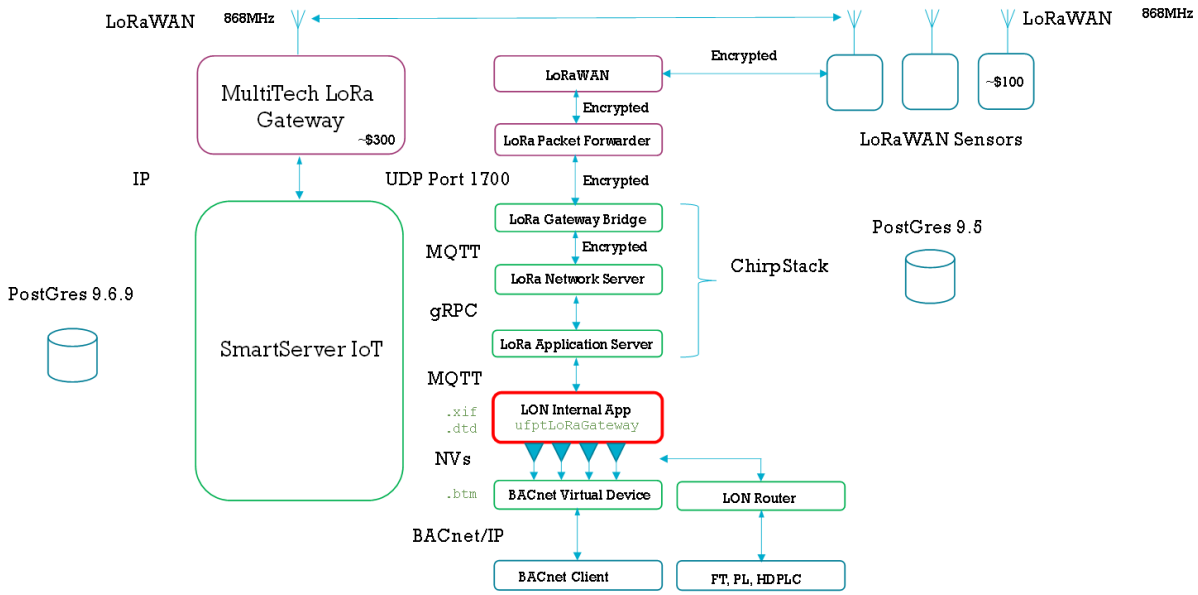
Contents

SmartServer IoT LoRa to LON and BACnet Gateway 1
Introduction 1
Bill of Materials 2
Cloning the GitHub Repository 2
Connect MultiTech Gateway and SmartServer IoT 3
Configure Multitech Gateway as Packet Forwarder 3
Update the SmartServer IoT 5
Installation of ChirpStack LoRaWAN Network Server Components on the SmartServer IoT 5
Configuring the LoRaWAN Network Server 16
Updating the LON Application's Gateway Interface 30
Backing up the Installation 31

Introduction

The example worked project allows data from LoRa sensors to be shared with both LON and BACnet networks. It demonstrates the open and extensible nature of the SmartServer IoT.

Data from NetVox R712 and Elsys ERS Co2 sensors are sent to a ChirpStack LoRa Gateway Bridge installed on the SmartServer IoT via a MultiTech LoRa gateway, which is acting in packet forward mode. Data from the Gateway Bridge is forwarded to the ChirpStack LoRa Network Server using MQTT. Data from the LoRa Network Server is passed to the ChirpStack LoRa application server using gRPC and then made available from the Application Server via MQTT to a LON Internal Application written in Node.js. The SmartServer IoT's BACnet Server virtualises the LON Internal Application and its associated datapoints such that they are visible to a BACnet network.



Example Configuration

Bill of Materials

For the worked example, you will need to purchase the following items:

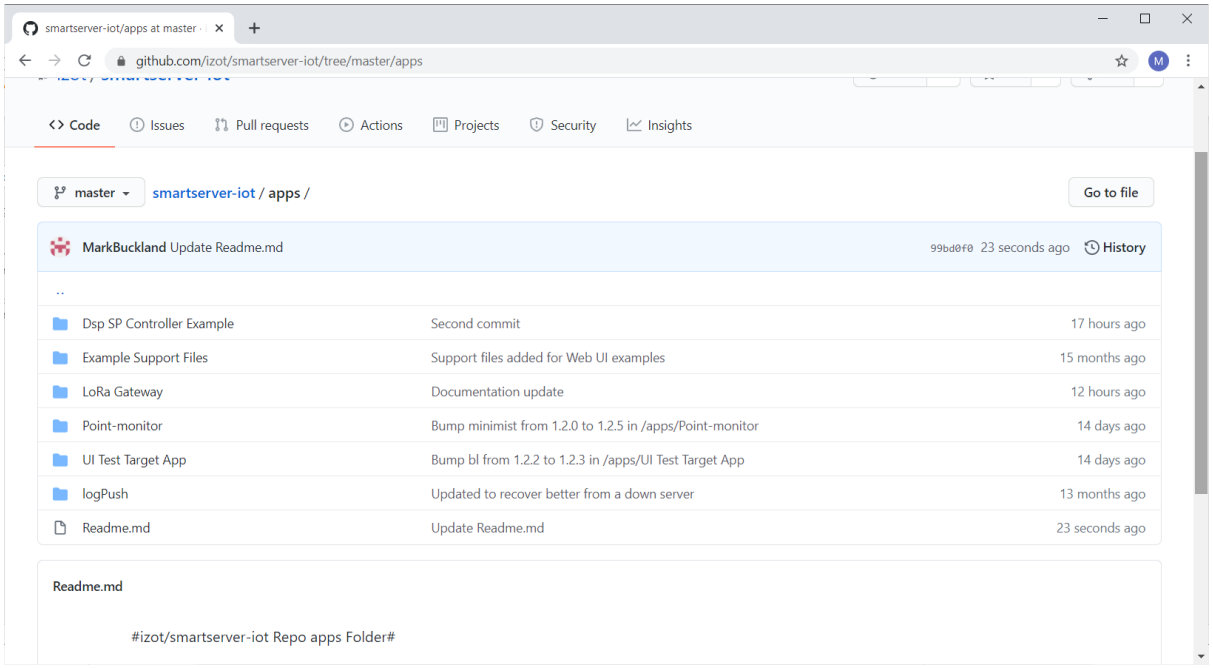
Quantity	Description	Model Number
1	SmartServer IoT Pro (or SmartServer IoT Pro EX)	72201R-240
1	Netvox Outdoor Temperature and Humidity Sensor	R712
1	Elsys ERS CO2	ERSCO2
1	MultiTech Conduit Access Point Ethernet Indoor Gateway with External Antenna	MTCAP-868-041A
2	Ethernet patch leads	
1	24VDC 10W Power Supply	

The SmartServer IoT can be purchased directly from Dialog, LoRa sensors and gateways can be purchased from alliot.co.uk or a local supplier.

You can modify the example to support different sensor types up to around 500 datapoints, by following Updating the Gateway Interface below. Increasing the datapoint count will increase the time to instantiate the device. The example does not support writing to LoRa sensors from the SmartServer IoT, but there is nothing to preclude this.

Cloning the GitHub Repository

1. Clone the repository at https://github.com/izot/smartserver-iot either by downloading a .zip file or using GitHub Desktop.



Connect MultiTech Gateway and SmartServer IoT

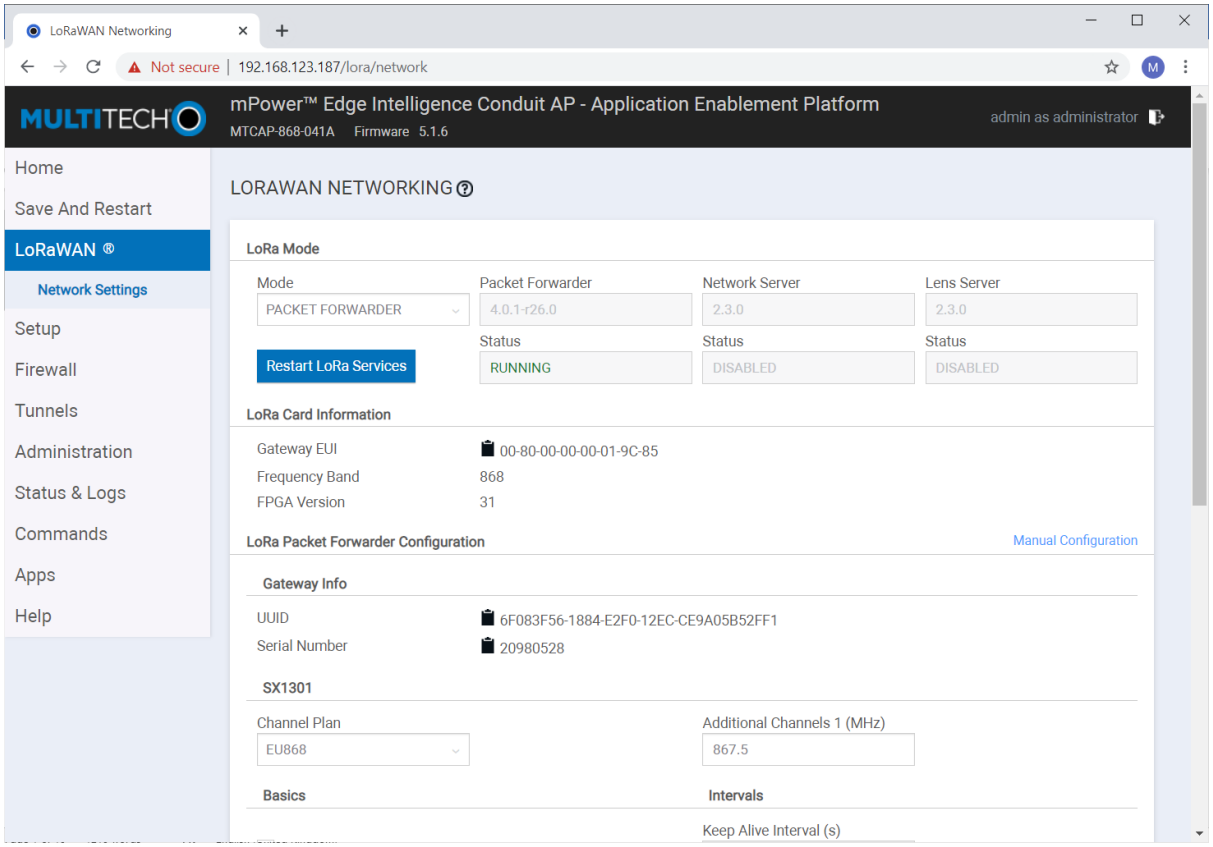
1. Connect the MultiTech Gateway and the SmartServer IoT using Ethernet patch leads to the same network with a DHCP server running.

   MultiTech Gateways are available with cellular modems, which with a fixed IP SIM card would allow remote connectivity between the gateway and SmartServer IoT behind a firewall with a fixed external IP address, or with its own cellular modem with a fixed IP address SIM card.

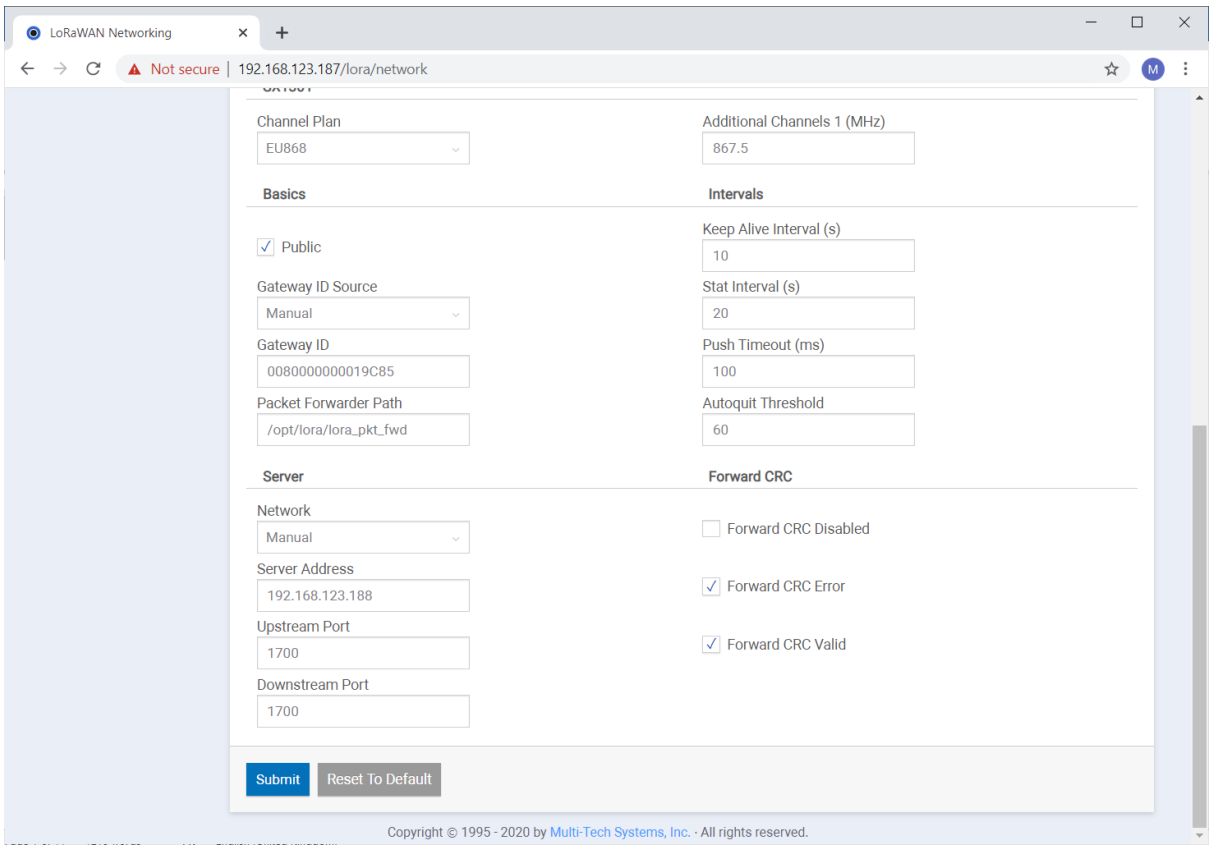
   See http://docs.adestotech.com/display/PortSSIoT/Step+5+-+%28Optional%29+Secure+Your+SmartServer for more information on using the SmartServer IoT and Network Address Translation.

Configure Multitech Gateway as Packet Forwarder

1. After selecting LoRaWAN from the left-hand menu, change the mode to PACKET-FORWARDER, as shown below:



2. Adjust the server address to that of the SmartServer and the uplink and downlink ports to 1700 as shown below:



3. Save and restart.

Update the SmartServer IoT

1. To ensure a known starting point, install the 3.00 build of the SmartServer IoT from: http://docs.adestotech.com/display/PortSSIoT/SmartServer+IoT+Release+Notes#SmartServerIoTReleaseNotes-CurrentRelease. Use the flash drive re-image.sh method. Please note that a re-image will remove the ChirpStack installation and an upgrade may remove the custom firewall settings as will apollo-reset normal.

Installation of ChirpStack LoRaWAN Network Server Components on the SmartServer IoT

1. Overview

The required ChirpStack LoRaWAN Network Server installation consists of three components:

- ChirpStack Gateway Bridge
- ChirpStack Network Server
- ChirpStack Application Server

Each component needs to be installed separately after some pre-requisite setup

2. Open Firewall Ports

Using putty.exe or a similar terminal emulator, connect to the console port of the SmartServer IoT to your PC using a USB Type A to Micro B cable. The serial configuration is 115200, 8, 1, N.

Using the following commands, open port 1700 and 8080:

```
$ sudo ufw allow 1700

$ sudo ufw allow 8080
```

Check the firewall status using the following command:

```
$ sudo ufw status
To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
80                         ALLOW       Anywhere
443                        ALLOW       Anywhere
5353                       ALLOW       Anywhere
2541/udp                   ALLOW       Anywhere
8883                       ALLOW       Anywhere
1883                       ALLOW       Anywhere
47808/udp                  ALLOW       Anywhere
47809/udp                  ALLOW       Anywhere
41797/udp                  ALLOW       Anywhere
1700                       ALLOW       Anywhere
8080                       ALLOW       Anywhere
1628/tcp                   ALLOW       Anywhere
```

```
1629/tcp                   ALLOW       Anywhere
```

3. Install Postgresql 9.5

Postgresql 9.6.9 is already installed in SmartServer IoT; however, some required extensions cannot be installed with 9.6.9 such as the postgresql-contrib extension.

From the console connection, run the following commands:

```
$ sudo apt-get update
$ sudo apt-get install postgresql-9.5
```

The default port number that Postgresql 9.6.9 uses is 5432, therefore Postgres 9.5 installation sets the port number to 5433 during installation.

Run the following command to configure Postgresql 9.5:

```
$ sudo -u postgres /usr/lib/postgresql/9.5/bin/psql -p 5433
```

Check the database is working using the following command:

```
$ pg_lsclusters
```

Example output is shown below:

```
Ver Cluster Port Status Owner    Data directory              Log file
9.5 main    5433 online postgres /var/lib/postgresql/9.5/main
/var/log/postgresql/postgresql-9.5-main.log
```

4. Setup ChirpStack software repository

ChirpStack provides a repository that is compatible with the Ubuntu apt package system.

Make sure that both dirmngr and apt-transport-https are installed using the following commands:

```
$ sudo apt install apt-transport-https dirmngr
```

Set up the key for this new repository using the following command:

```
$ sudo apt-key adv --keyserver
[keyserver.ubuntu.com](<http://keyserver.ubuntu.com/>) --recv-keys
1CE2AFD36DBCCA00
```

Add the repository to the repository list by creating a new file using the following command:

```
$ sudo echo "deb <https://artifacts.chirpstack.io/packages/3.x/deb> stable
main" | sudo tee /etc/apt/sources.list.d/chirpstack.list
```

Update the apt package cache using the following command:

```
$ sudo apt update
```

5. Install LoRa Gateway Bridge

Install the package using the following command:

```
$ sudo apt install chirpstack-gateway-bridge
```

The configuration file is located at/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml

The default log level is 4. However, level 2 is recommended to avoid excessive logging and disk usage.

Using Nano or another suitable editor after logging in as root (after having set the root password), add the following line

```
# debug=5, info=4, warning=3, error=2, fatal=1, panic=0
log_level = 2
```

Start the ChirpStack Gateway Bridge service:

```
Start lora-gateway-bridge using the following command:
$ sudo systemctl start chirpstack-gateway-bridge

Enable the chirpstack-gateway-bridge to start on bootup using the following
command:
$ sudo systemctl enable chirpstack-gateway-bridge
```

How to (re)start and stop the Chirpstack Gateway Bridge

```
$ sudo systemctl [start|stop|restart|status] chirpstack-gateway-bridge
```

Check if the installation works and logs correctly using the following command:

```
$ sudo journalctl -u chirpstack-gateway-bridge -f -n 50
```

Example output from the command above:

```
Apr 09 09:37:04 NanoPi-R1 lora-gateway-bridge[17552]: time="2019-04-
09T09:37:04Z" level=info msg="gateway: rxpk packet received"
addr="192.168.100.101:38361" data=QCzOABqA6YPeoz9zLx+kvjCuC1t41sNnz7Se
mac=00956901000000b8
Apr 09 09:37:04 NanoPi-R1 lora-gateway-bridge[17552]: time="2019-04-
09T09:37:04Z" level=info msg="backend: publishing packet" qos=0
topic=gateway/00956901000000b8/rx
Apr 09 09:37:11 NanoPi-R1 lora-gateway-bridge[17552]: time="2019-04-
09T09:37:11Z" level=info msg="gateway: rxpk packet received"
addr="192.168.100.101:38361" data=QCzOABqA6oPeK+Zw01ISAmgfjwVpSub9ysN3
mac=00956901000000b8
Apr 09 09:37:11 NanoPi-R1 lora-gateway-bridge[17552]: time="2019-04-
09T09:37:11Z" level=info msg="backend: publishing packet" qos=0
topic=gateway/00956901000000b8/rx
^[[BApr 09 09:37:25 NanoPi-R1 lora-gateway-bridge[17552]: time="2019-04-
09T09:37:25Z" level=info msg="gateway: rxpk packet received"
addr="192.168.100.101:38361" data=QCzOABqA64PebX+CVHhlsQwEc0F/3YylsVuS
mac=00956901000000b8
```

```
Apr 09 09:37:25 NanoPi-R1 lora-gateway-bridge[17552]: time="2019-04-
09T09:37:25Z" level=info msg="backend: publishing packet" qos=0
topic=gateway/00956901000000b8/rx
```

Check Gateway Data

Type the following command to ensure that data is being received (note that some of the data is encrypted):

```
$ mosquitto_sub -t "gateway/#" -v
```

Example data:

```
gateway/0080000000019c85/event/up
@N(▒▒|▒▒▒▒▒$EF▒c▒b▒7`▒▒▒▒        }
4/5<
▒▒▒▒(▒▒▒▒▒▒▒▒▒▒1@8zy$▒▒▒▒▒c▒B▒▒▒▒▒g▒▒C▒
gateway/0080000000019c85/event/stats
▒▒▒▒r▒(0J192.168.123.187Z▒▒▒▒<G▒▒▒▒▒▒▒▒
PuTTYgateway/0080000000019c85/command/config
▒▒▒▒05b193966-160d-4a60-bfb6-5954eebb243e-r160407183▒▒▒▒
}
▒鄞
}
▒▒▒▒▒▒
}

gateway/0080000000019c85/event/stats
▒▒▒▒r▒J192.168.123.187Z▒$"▒▒▒▒G▒▒▒▒\▒QB4
PuTTYgateway/0080000000019c85/command/config
▒▒▒▒05b193966-160d-4a60-bfb6-5954eebb243e-r160407183▒▒▒▒
}
▒鄞
}
▒▒▒▒▒▒
```

6. Install LoRa Network Server

Creating a user and database

The ChirpStack Network Server needs its own database, create a new database, start the PostgreSQL prompt as the postgres user with the following command:

```
$ sudo -u postgres /usr/lib/postgresql/9.5/bin/psql -p 5433
```

Within the Postgresql prompt, enter the following queries:

```
create role loraserver_ns with login password 'dbpassword';
create role chirpstack_ns with login password 'dbpassword';
create database loraserver_ns with owner loraserver_ns;
create database chirpstack_ns with owner chirpstack_ns;
```

Exit the Posgressql prompt by typing the following:

```
\q
```

Verify if the user and database have been setup correctly using the following command:

```
$ psql -h localhost -U chirpstack_ns -W chirpstack_ns -p 5433

Password for user chirpstack_ns:
psql (9.5.23)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384,
bits: 256, compression: off)
Type "help" for help.

chirpstack_ns=>
```

Install ChirpStack Network Server using the following command:

```
$ sudo apt-get install chirpstack-network-server
```

After installation, modify the configuration file /etc/chirpstack-network-server/chirpstack-network-server.toml using Nano or a similar editor

You will need to elevate to superuser using the following command:

```
$ su
```

In the [postgresql] section add/change the following:

```
log_level = 2 #add

dsn="postgres://chirpstack_ns:dbpassword@localhost:5433/chirpstack_ns?sslmo
de=disable"
```

Save the changes to the file and exit the editor, then logout of su using the following command:

```
# exit
```

Start the network server using the following command:

```
$ sudo systemctl start chirpstack-network-server
```

Check the status using the following command:

```
$ sudo systemctl status chirpstack-network-server
```

Example output is shown below:

```
● chirpstack-network-server.service - ChirpStack Network
Server

   Loaded: loaded (/lib/systemd/system/chirpstack-network-
server.service; enabled; vendor preset: enabled)

   Active: active (running) since Thu 2020-11-12 11:15:52 GMT;
5 days ago

     Docs: https://www.chirpstack.io/

 Main PID: 838 (chirpstack-netw)

   CGroup: /system.slice/chirpstack-network-server.service

           └─838 /usr/bin/chirpstack-network-server



Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="gateway/mqtt: uplink frame received" gat

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="uplink: frame(s) collected" ctx_id=a920e

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info msg="sent
uplink meta-data to network-control

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="device gateway rx-info meta-data saved"

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="device-session saved" ctx_id=a920ed21-41
```

```
Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="finished client unary call" ctx_id=a920e

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="gateway/mqtt: gateway stats packet recei

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="gateway updated" ctx_id=e5fb5511-fee2-41

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="gateway/mqtt: publishing gateway command

Nov 17 12:20:04 smartserver-17q5zwu chirpstack-network-
server[838]: time="2020-11-17T12:20:04Z" level=info
msg="finished client unary call" ctx_id=e5fb5

lines 1-18/18 (END)
```

Check logging using the following command:

```
$ sudo journalctl -u chirpstack-network-server -f -n 50
```

Example output:

```
-- Logs begin at Sun 2020-11-15 13:46:10 GMT. --
Nov 17 10:39:05 smartserver-17q5zwu chirpstack-network-server[838]:
time="2020-11-17T10:39:05Z" level=info msg="gateway/mqtt: gateway stats
packet received" gateway_id=0080000000019c85 stats_id=bc71bae2-31e8-4901-
a4da-f15286b76739
Nov 17 10:39:05 smartserver-17q5zwu chirpstack-network-server[838]:
time="2020-11-17T10:39:05Z" level=info msg="gateway updated"
ctx_id=bc71bae2-31e8-4901-a4da-f15286b76739 gateway_id=0080000000019c85
Nov 17 10:39:05 smartserver-17q5zwu chirpstack-network-server[838]:
time="2020-11-17T10:39:05Z" level=info msg="gateway/mqtt: publishing
gateway command" command=config gateway_id=0080000000019c85 qos=0
topic=gateway/0080000000019c85/command/config
Nov 17 10:39:05 smartserver-17q5zwu chirpstack-network-server[838]:
time="2020-11-17T10:39:05Z" level=info msg="finished client unary call"
ctx_id=bc71bae2-31e8-4901-a4da-f15286b76739 grpc.code=OK
grpc.ctx_id=c754a33b-5fc5-49ce-8e93-840805d6a323 grpc.duration=20.738234ms
grpc.method=HandleGatewayStats grpc.service=as.ApplicationServerService
span.kind=client system=grpc
```

7. Install the LoRa Application Server

Create a user and database using the following commands:

```
$ sudo -u postgres /usr/lib/postgresql/9.5/bin/psql -p 5433
```

Within the PostgreSQL prompt, enter the following queries:

```
create role chirpstack_as with login password 'dbpassword';
create database chirpstack_as with owner chirpstack_as;
\c chirpstack_as
create extension pg_trgm;
create extension hstore;

\q
```

To verify if the user and database have been setup correctly, try to connect to it:

```
$ psql -h localhost -U chirpstack_as -W chirpstack_as -p 5433
Password for user chirpstack_as:
psql (9.6.9, server 9.5.23)
Type "help" for help.

chirpstack_as=>
```

Exit postgresql using the following command:

```
\q
```

8. Install ChirpStack Application Server

```
$ sudo apt-get install chirpstack-application-server
```

After installation, modify the configuration file /etc/chirpstack-application-server/chirpstack-application-server.toml using a suitable editor such as nano.

Elevate to superuser using the following command:

```
$ su
```

You must change the application_server.external_api.jwt_secret.

First change the logging level to 2.

```
# debug=5, info=4, warning=3, error=2, fatal=1, panic=0
log_level = 2 #add
```

Given you used the password dbpassword when creating the PostgreSQL database, you need to change the config variable postgresql.dsn in the [postgressql] section as follows:

```
[postgresql]

dsn="postgres://chirpstack_as:dbpassword@localhost:5433/chirpstack_as?sslmo
de=disable"
```

In the [application_server.external_api]section change jwt_secret as below:

```
[application_server.external_api]
  jwt_secret="verysecret"

Write the changes to the file, close the editor and exit superuser using
the exit command:

# exit
```

Start the Chirpstack application server using the following command:

```
$ sudo systemctl start chirpstack-application-server
```

Monitor the status using the following command:

```
$ sudo systemctl status chirpstack-application-server
```

The output should be similar to the following:

```
● chirpstack-application-server.service - ChirpStack Application Server
   Loaded: loaded (/lib/systemd/system/chirpstack-application-
server.service; en
   Active: active (running) since Thu 2020-11-12 11:15:52 GMT; 5 days ago
     Docs: https://www.chirpstack.io/
 Main PID: 822 (chirpstack-appl)
   CGroup: /system.slice/chirpstack-application-server.service
           └─822 /usr/bin/chirpstack-application-server
```

```
Nov 17 11:54:44 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:54:44 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:54:44 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="20
lines 1-18/18 (END)
● chirpstack-application-server.service - ChirpStack Application Server
   Loaded: loaded (/lib/systemd/system/chirpstack-application-
server.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-11-12 11:15:52 GMT; 5 days ago
     Docs: https://www.chirpstack.io/
 Main PID: 822 (chirpstack-appl)
   CGroup: /system.slice/chirpstack-application-server.service
           └─822 /usr/bin/chirpstack-application-server

Nov 17 11:54:44 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:54:44Z" level=info msg="gateway updated" ctx_i
Nov 17 11:54:44 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:54:44Z" level=info msg="metrics saved" aggrega
Nov 17 11:54:44 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:54:44Z" level=info msg="finished unary call wi
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="device last-seen and d
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="finished unary call wi
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="integration/logger: lo
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="integration/mqtt: publ
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="gateway updated" ctx_i
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="metrics saved" aggrega
Nov 17 11:55:04 smartserver-17q5zwu chirpstack-application-server[822]:
time="2020-11-17T11:55:04Z" level=info msg="finished unary call wi
~
```

Check the logs using the following command:

```
$ sudo journalctl -f -n 100 -u chirpstack-application-server
```

You should see something similar to the following:

```
● chirpstack-application-server.service - ChirpStack Application
Server

   Loaded: loaded (/lib/systemd/system/chirpstack-application-
server.service; enabled; vendor preset: enabled)

   Active: active (running) since Thu 2020-11-12 11:15:52 GMT; 5
days ago

     Docs: https://www.chirpstack.io/

 Main PID: 822 (chirpstack-appl)

   CGroup: /system.slice/chirpstack-application-server.service

           └─822 /usr/bin/chirpstack-application-server


Nov 17 11:55:51 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:55:51Z" level=info msg="device
last-seen and d

Nov 17 11:55:51 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:55:51Z" level=info msg="finished
unary call wi

Nov 17 11:55:51 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:55:51Z" level=info
msg="integration/logger: lo

Nov 17 11:55:51 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:55:51Z" level=info
msg="integration/mqtt: publ

Nov 17 11:56:04 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:56:04Z" level=info msg="gateway
updated" ctx_i

Nov 17 11:56:04 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:56:04Z" level=info msg="metrics
saved" aggrega

Nov 17 11:56:04 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:56:04Z" level=info msg="finished
unary call wi

Nov 17 11:56:24 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:56:24Z" level=info msg="gateway
updated" ctx_i

Nov 17 11:56:24 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:56:24Z" level=info msg="metrics
saved" aggrega

Nov 17 11:56:24 smartserver-17q5zwu chirpstack-application-
server[822]: time="2020-11-17T11:56:24Z" level=info msg="finished
unary call wi
```

```
lines 1-18/18 (END)
```

Configuring the LoRaWAN Network Server

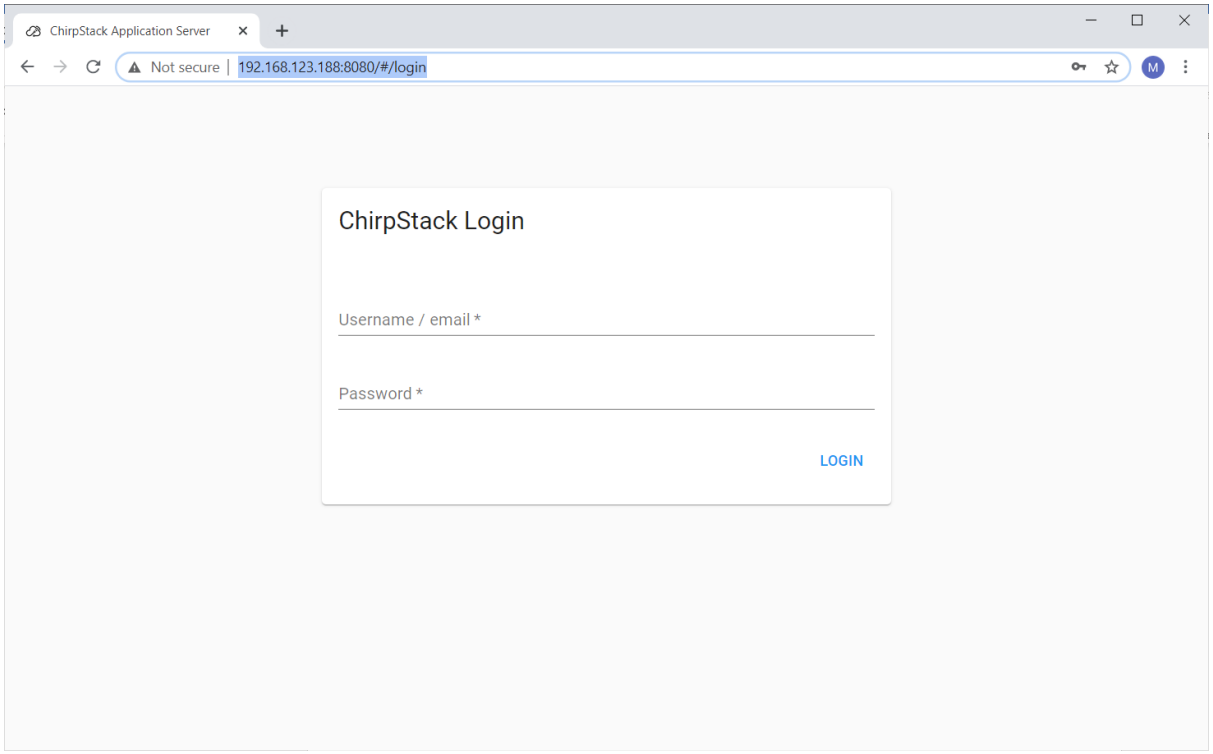
1. Login

http://<SmartServer IPV4 Address>:8080/#/login

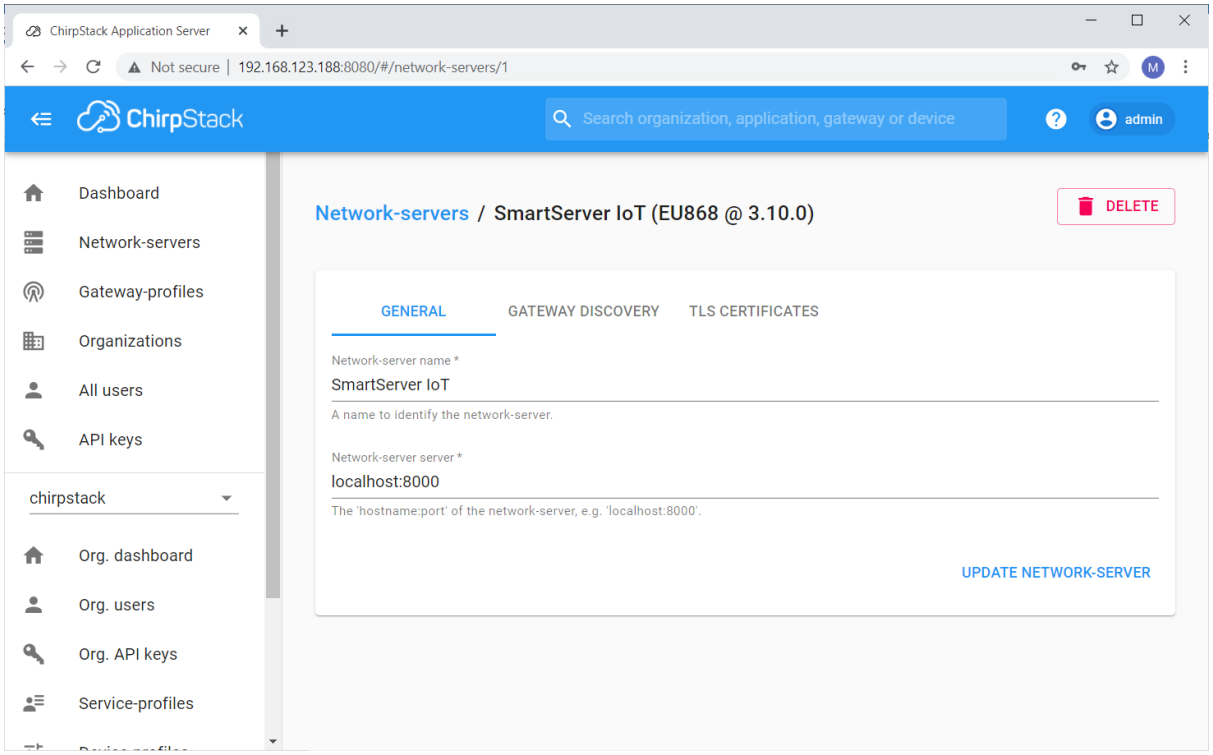
Default credentials are:

Username: admin

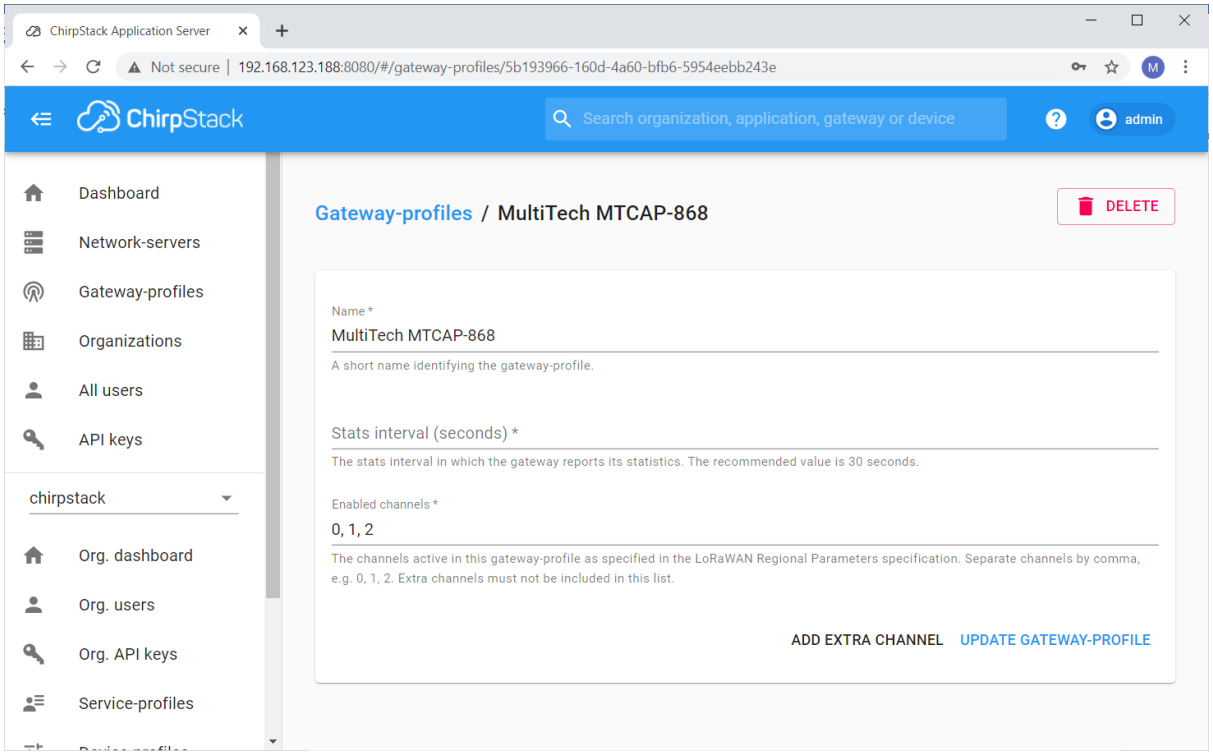
Password: admin



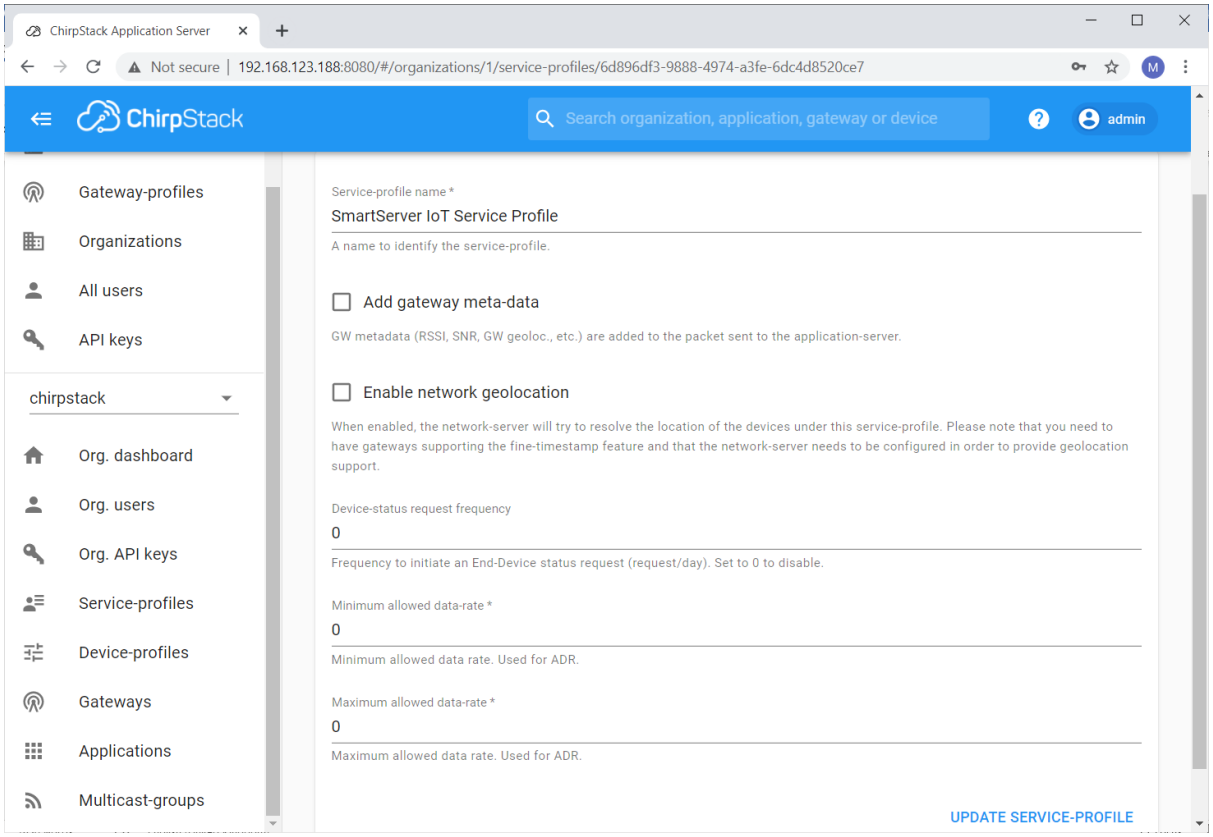
2. Add a Network Server as shown below:



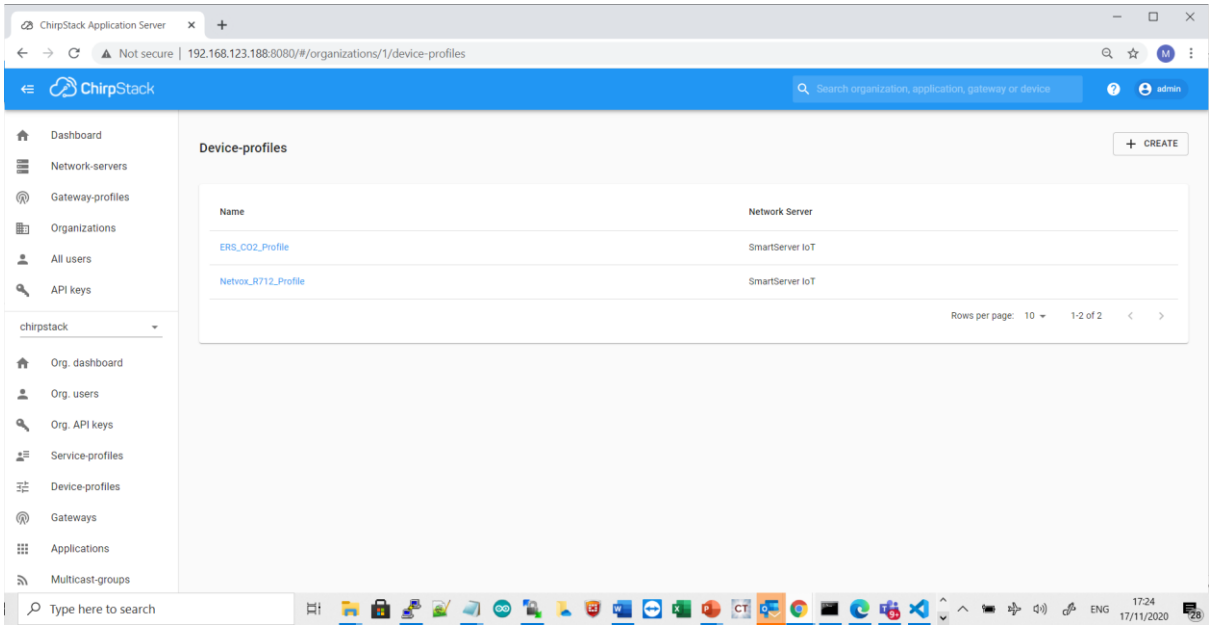
3. Create a Gateway Profile as shown below:



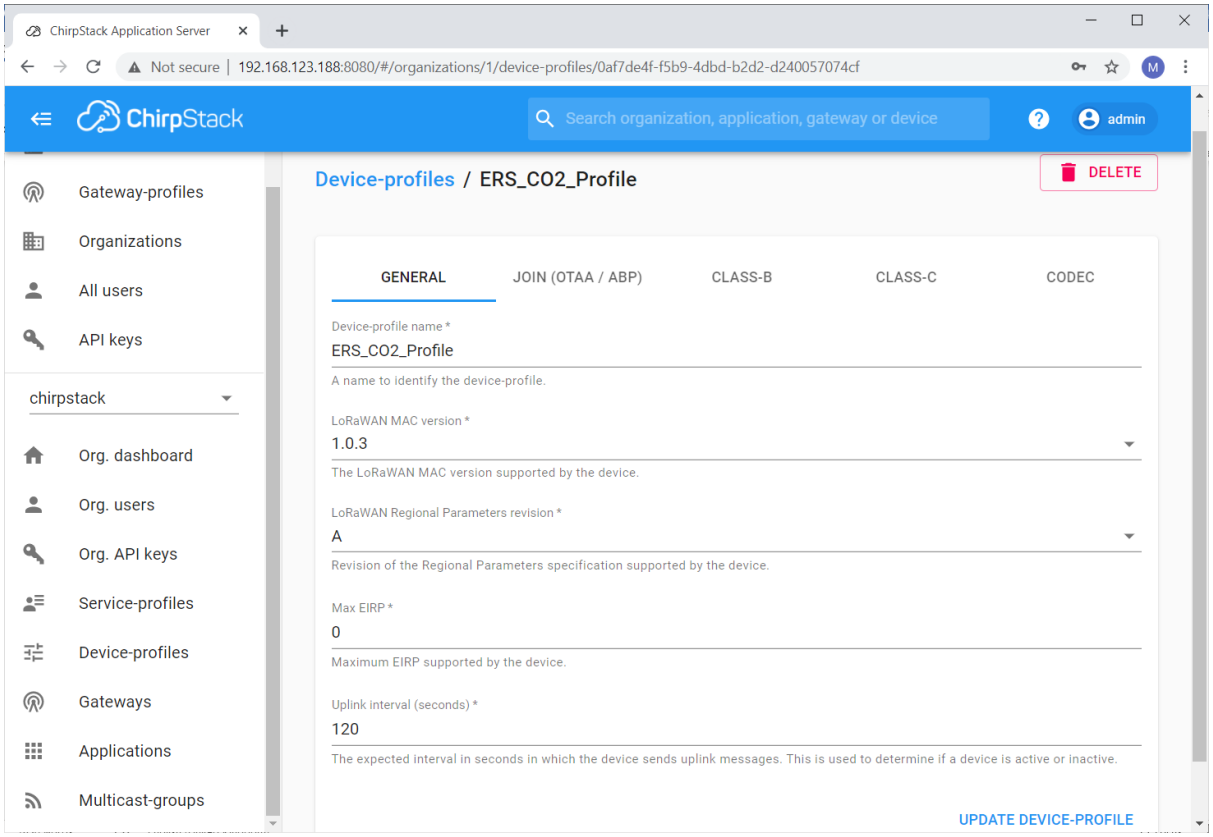
4. Create a Service Profile, as shown below:



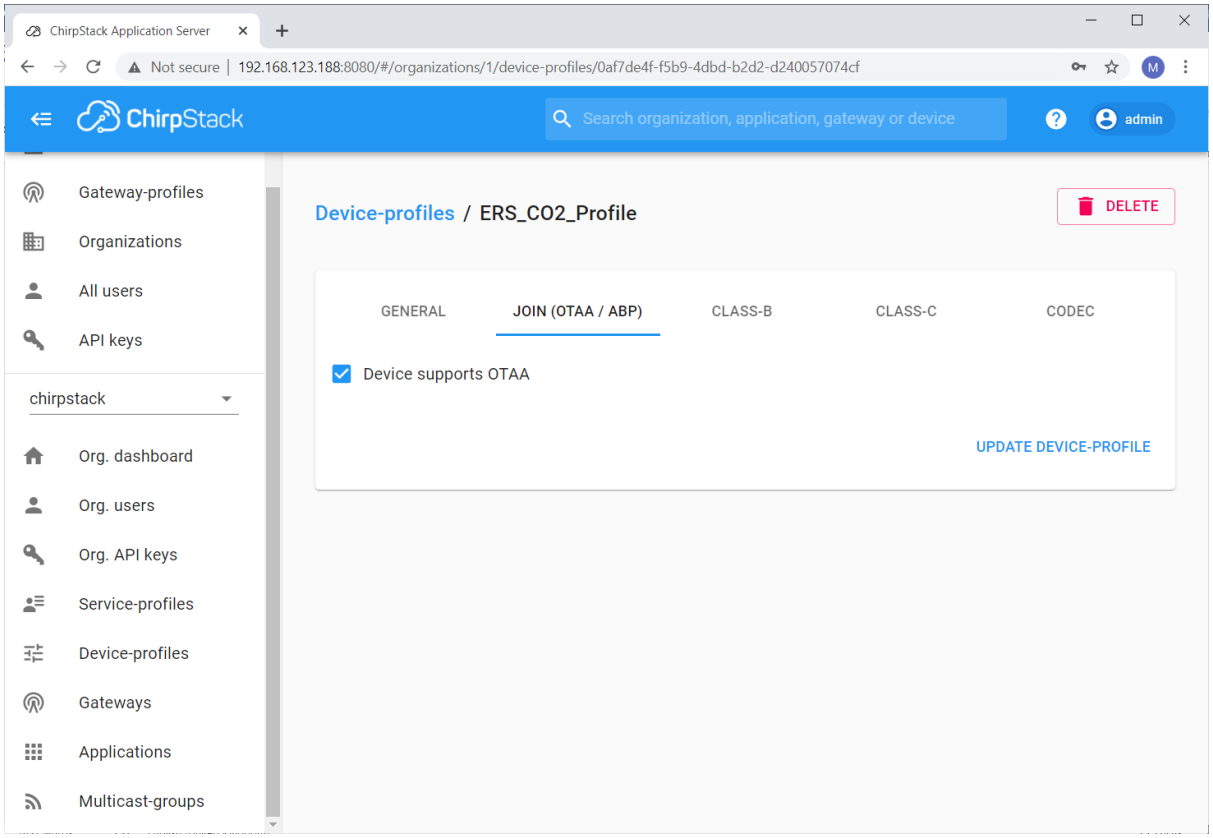
5. Create a Device Profile for the ERS CO2 sensor, as shown below:



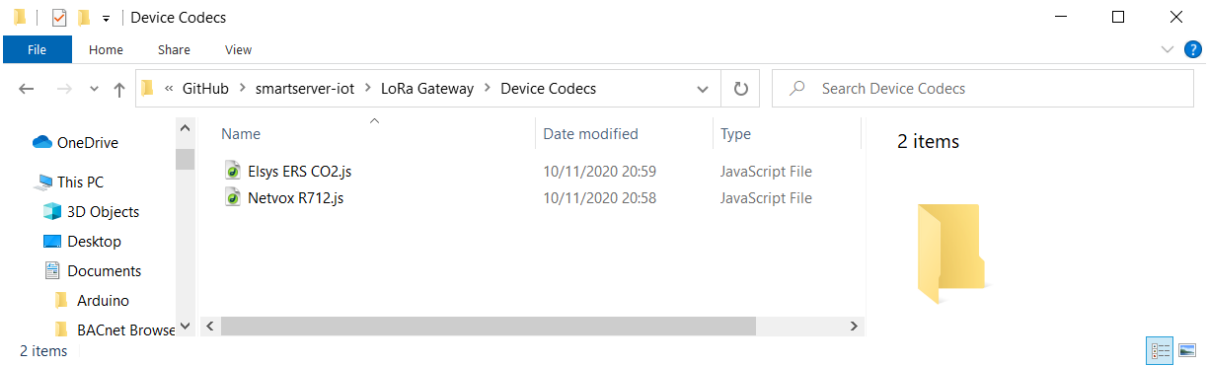
Add data in the general tab, as shown below:

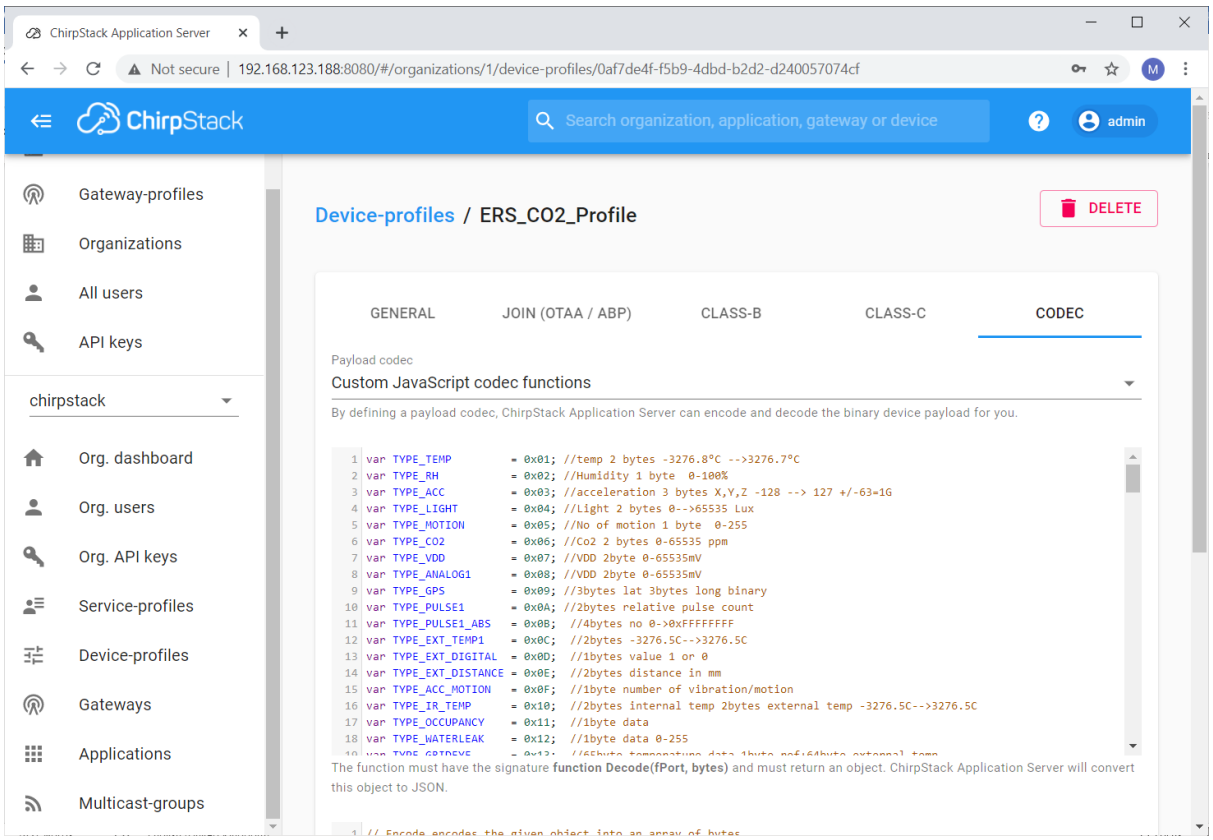


Enable Over The Air Activation in the JOIN (OTTA/ABP) tab, as shown below:

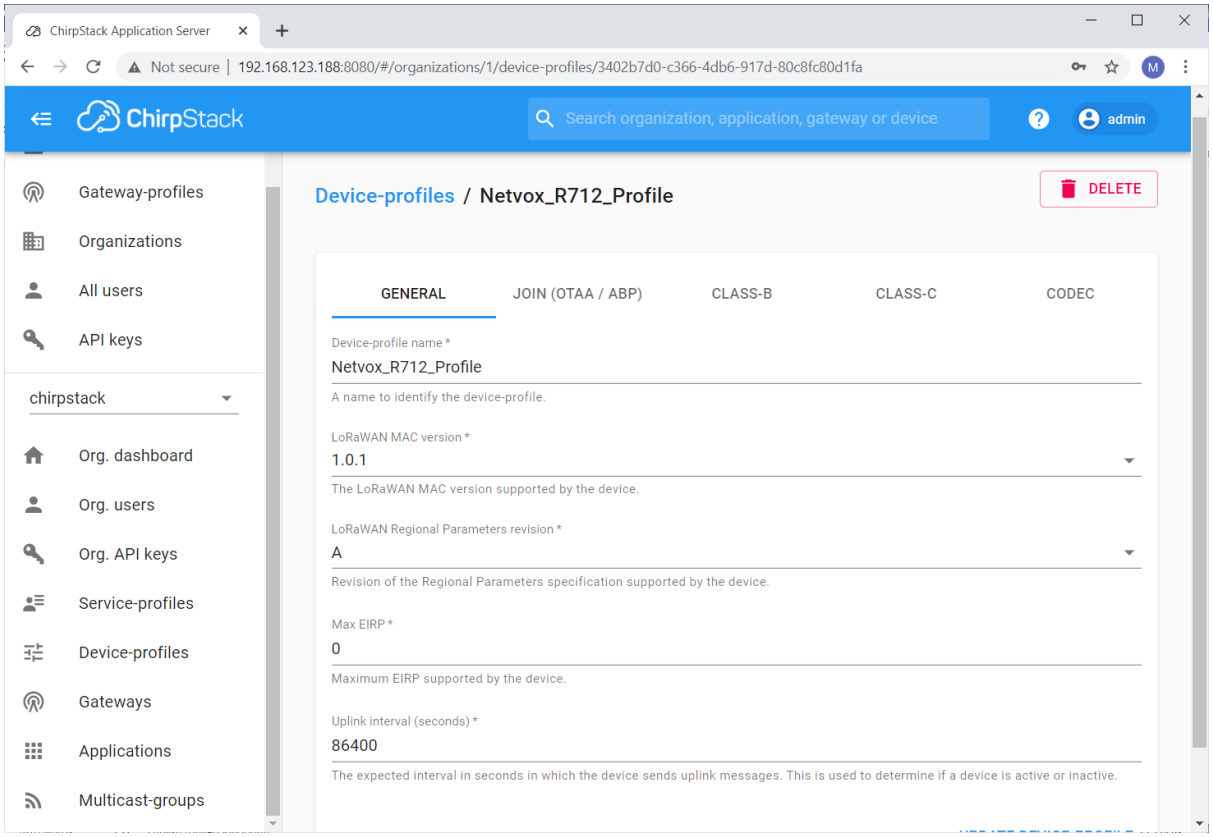


From the CODEC tab, specify a Payload coded of Custom JavaScript codec functions with the content from the example in the cloned github repository, as shown below:

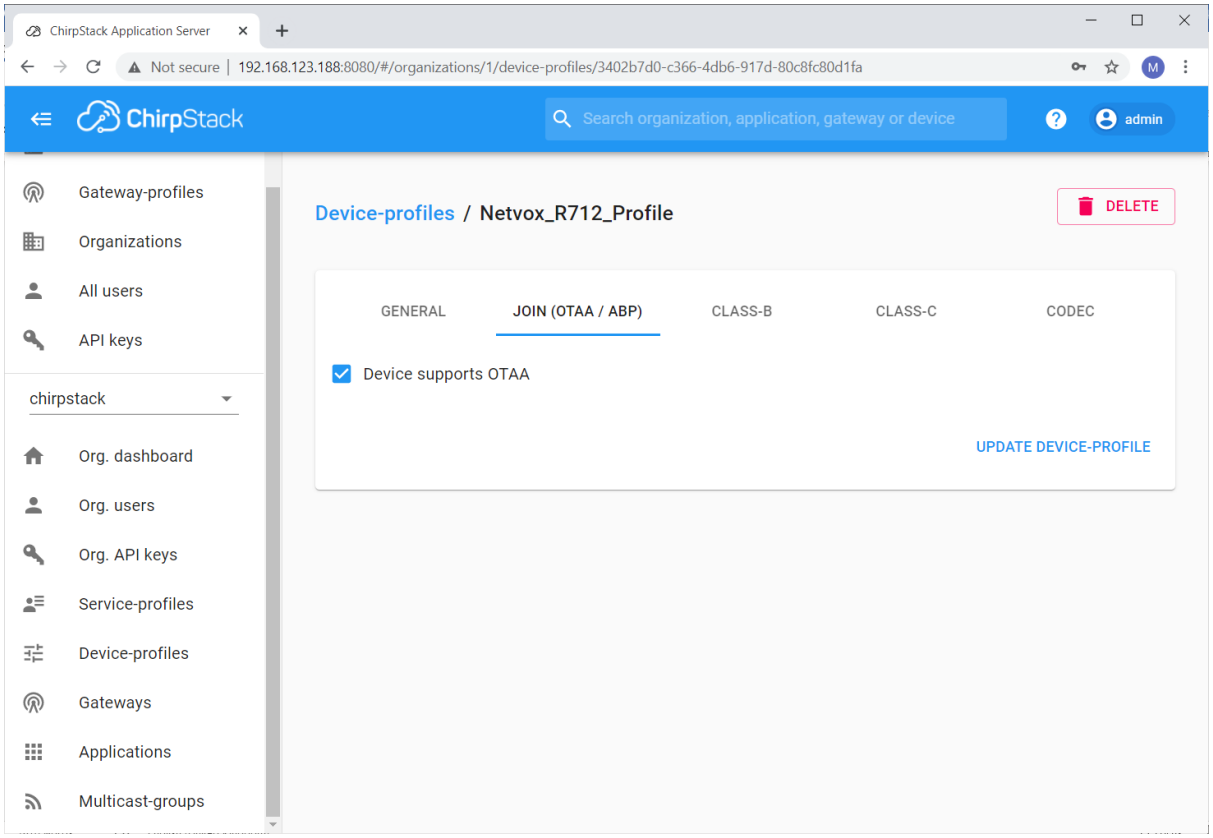




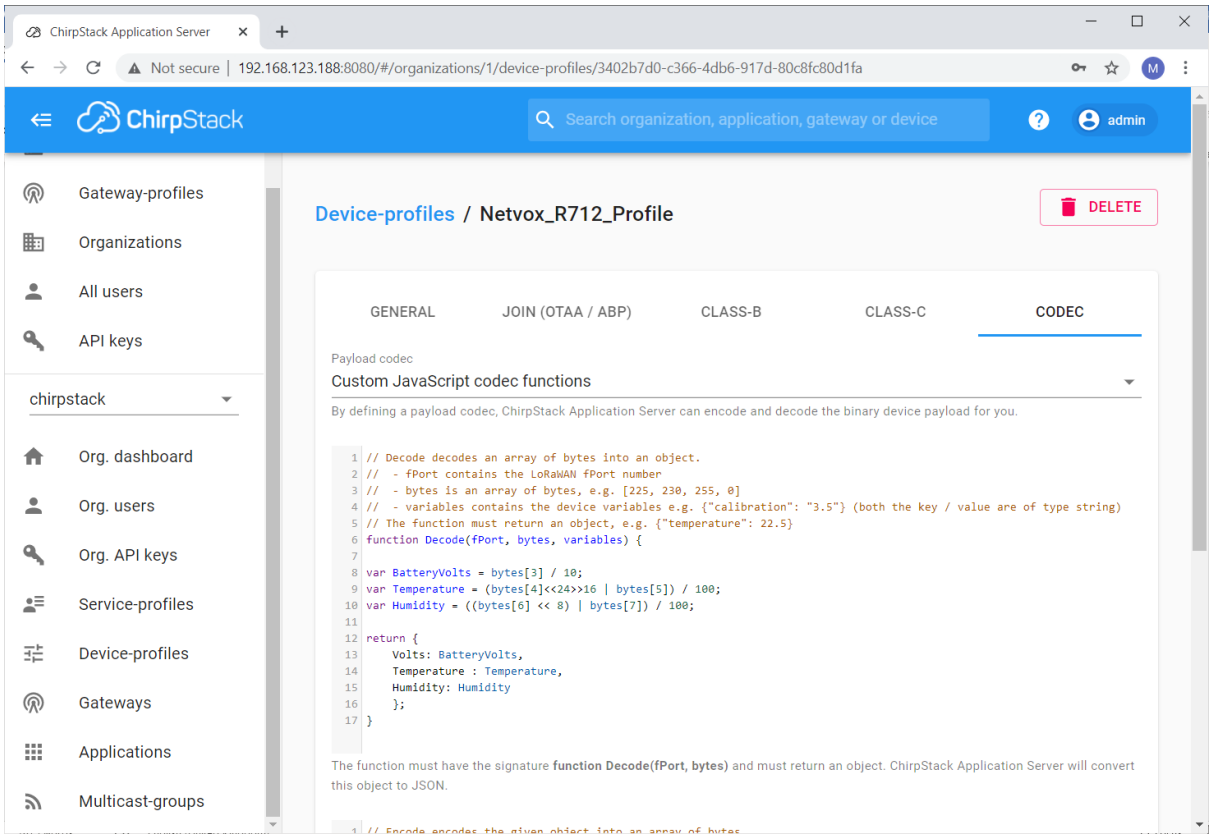
Set up a Device Profile for the Netvox R712 as shown below, by adding data in the general tab as shown below:



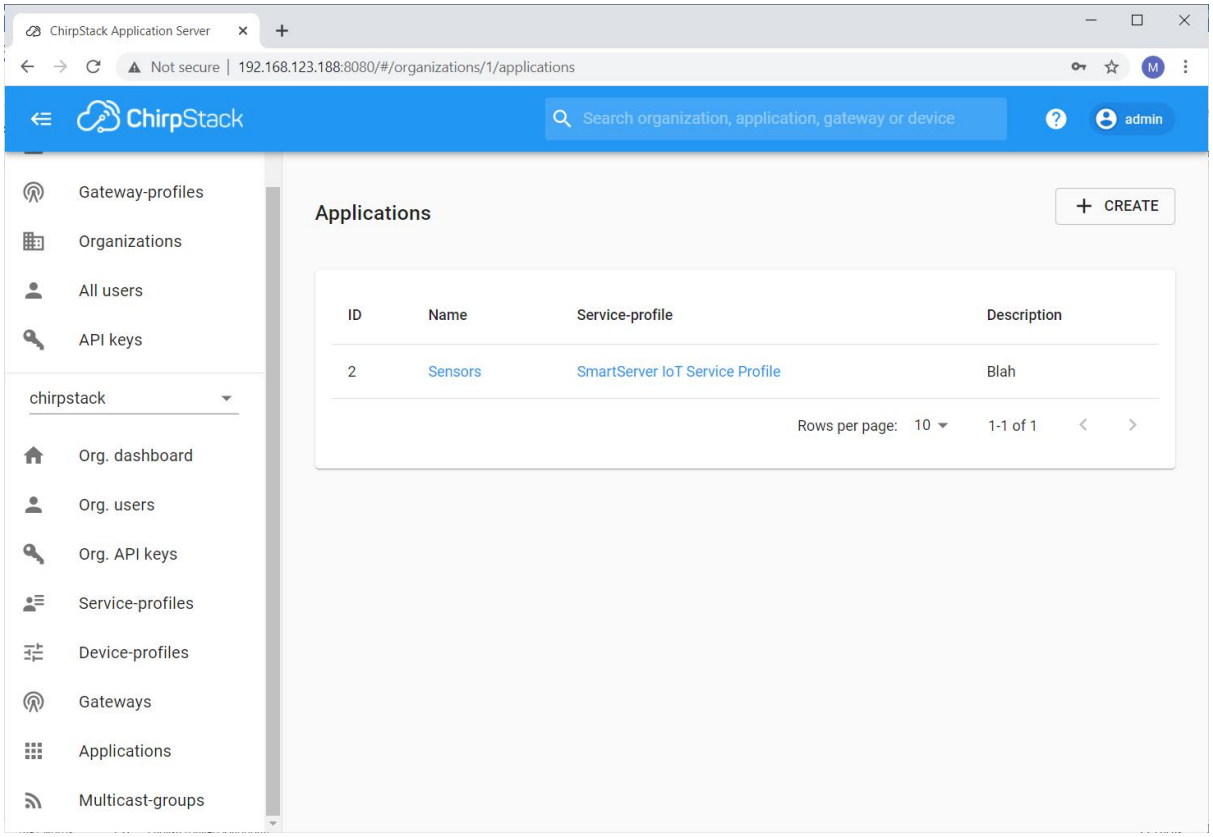
Enable Over The Air Activation in the JOIN (OTTA/ABP) tab, as shown below:

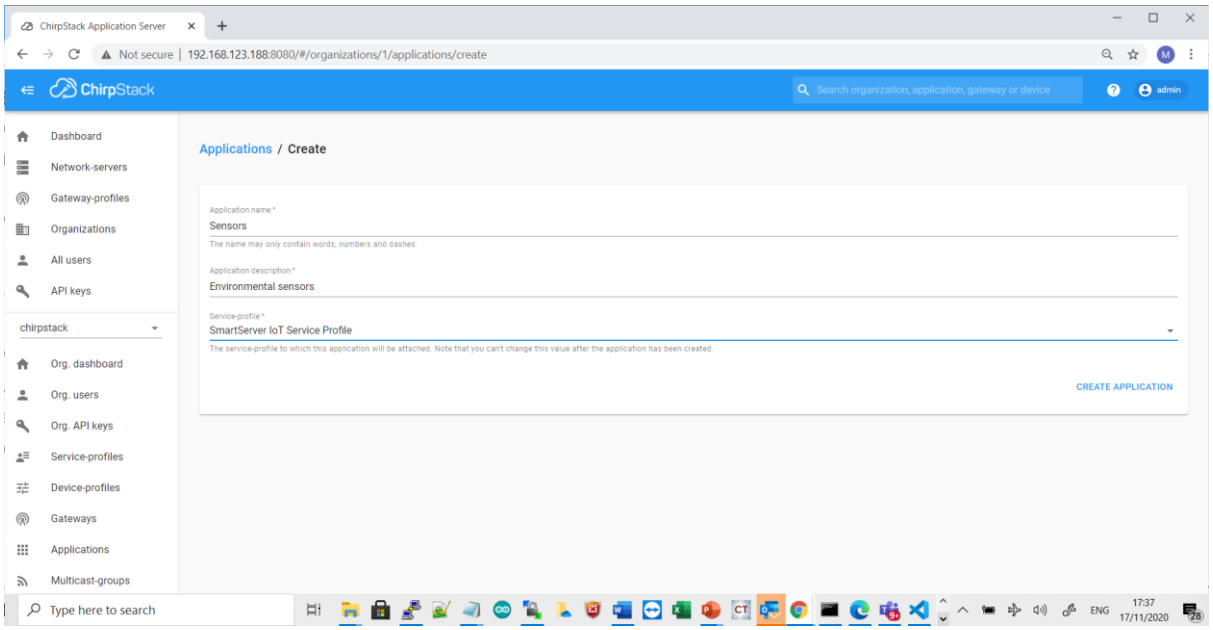


From the CODEC tab, specify a Payload codec of Custom JavaScript codec functions with the content from the example in the cloned GitHub repository, as shown below:

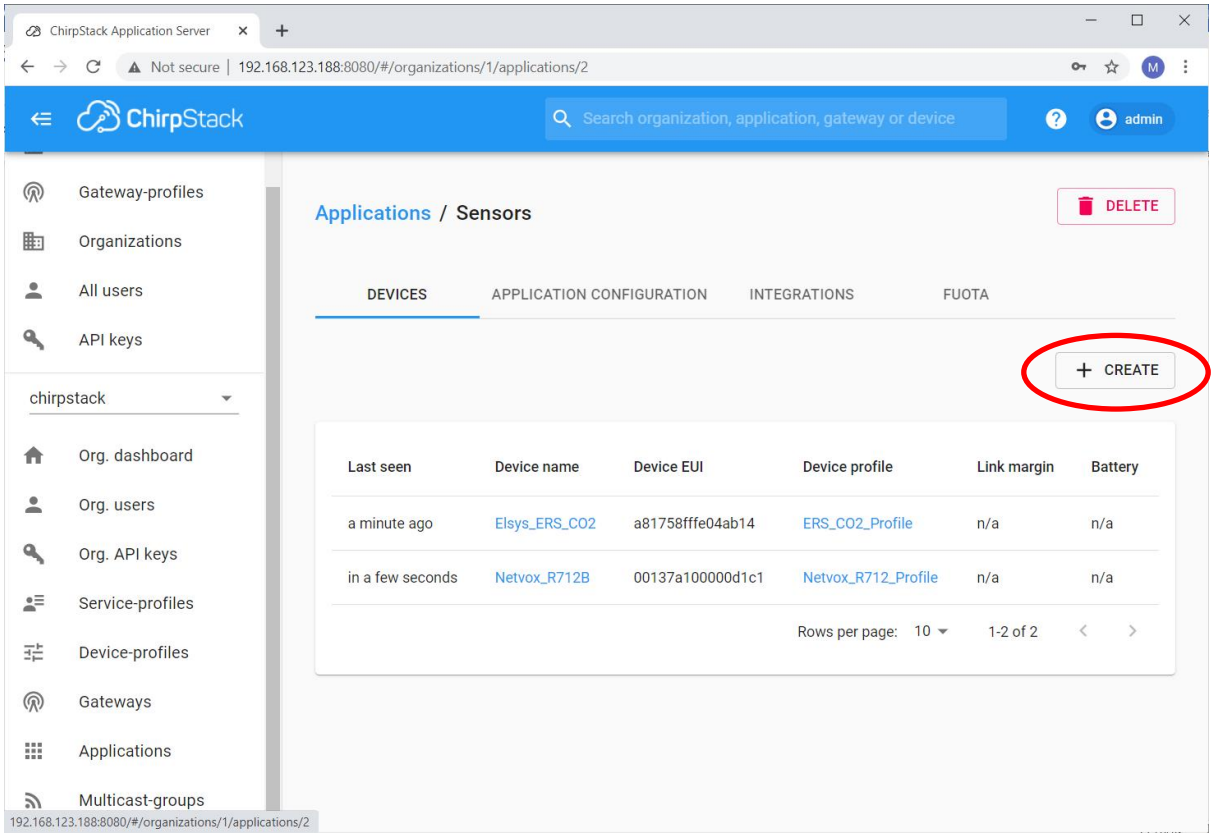


Setup up an Application, as shown below:

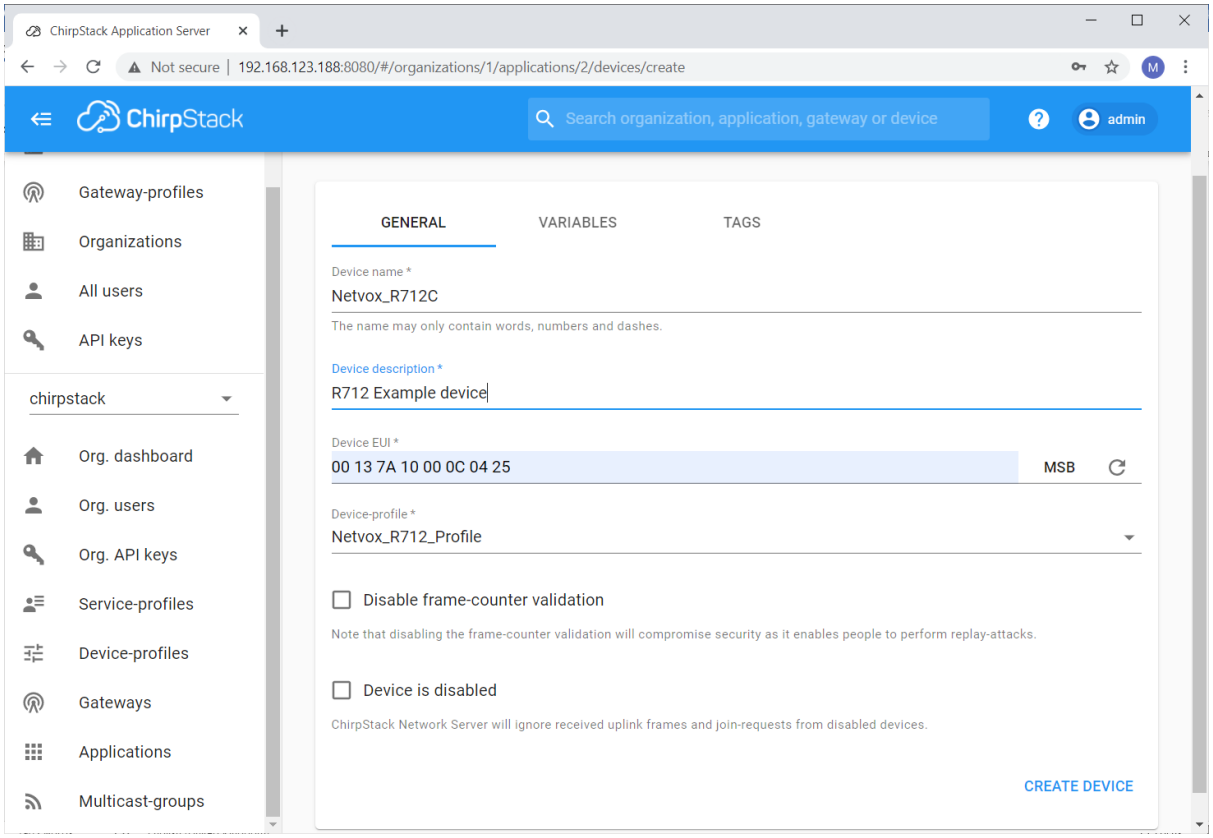




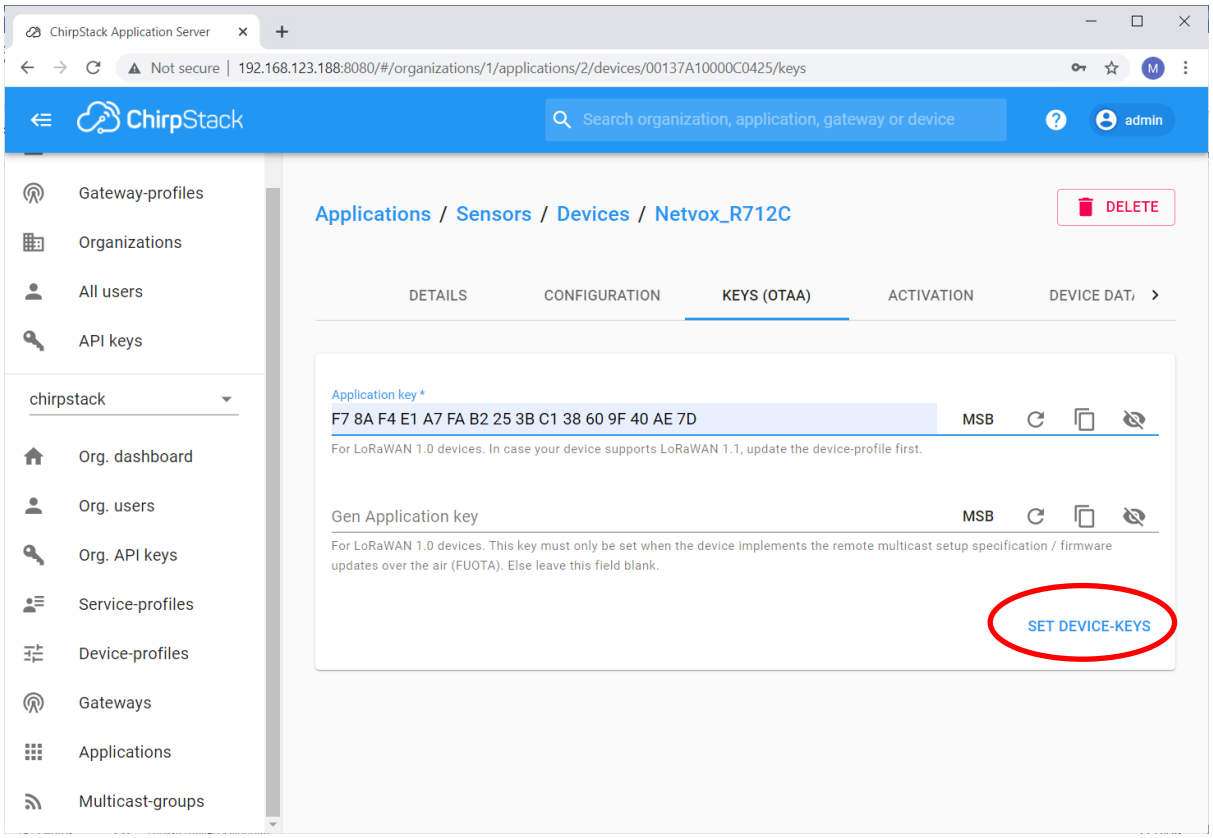
Create Devices in the Application, as shown below:



Create the Netvox R712 device using the DEVEUI from the rear of the device and the supplied application key as shown below:



In the KEYS (OTAA) tab add the application key supplied with the sensor as shown below and click SET DEVICE-KEYS.



Create the Elysys ERS CO2 device in a similar fashion.

Once both devices are created, if needed Base64 encoded configuration data can be sent down to the device using Enqueue Downlink Payload using the relevant port. Check confirmed downlink.

For example to set reporting to 60s:

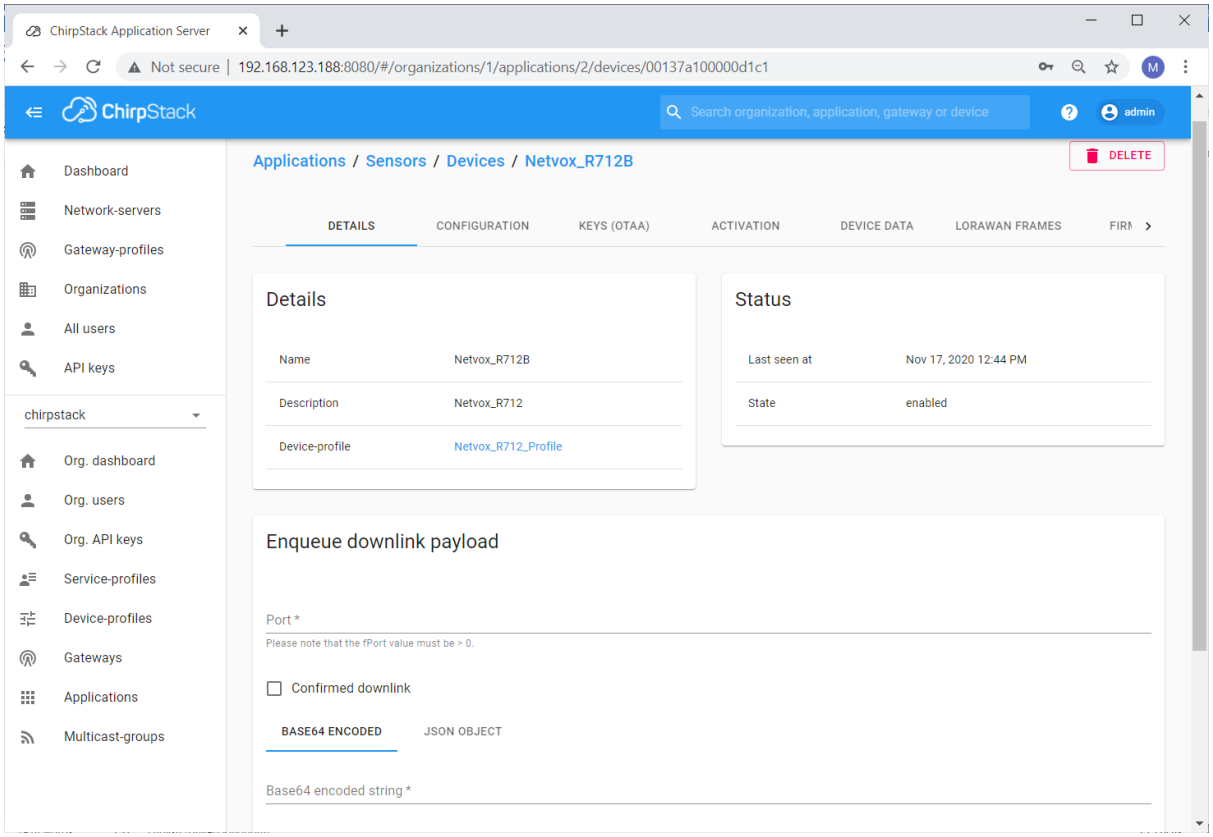
For the NetVox R712 you need to use port 7 and send down the following base64 data:

Hex	Base64
0x0101003C003C0100640064	AQEAPAA8AQBkAGQ

For the Elsys ERS C02 you need to use port 6 and send down the following base64 data:

Hex	Base64
0x3E06140000003CFE	PgYUAAAAPP4=

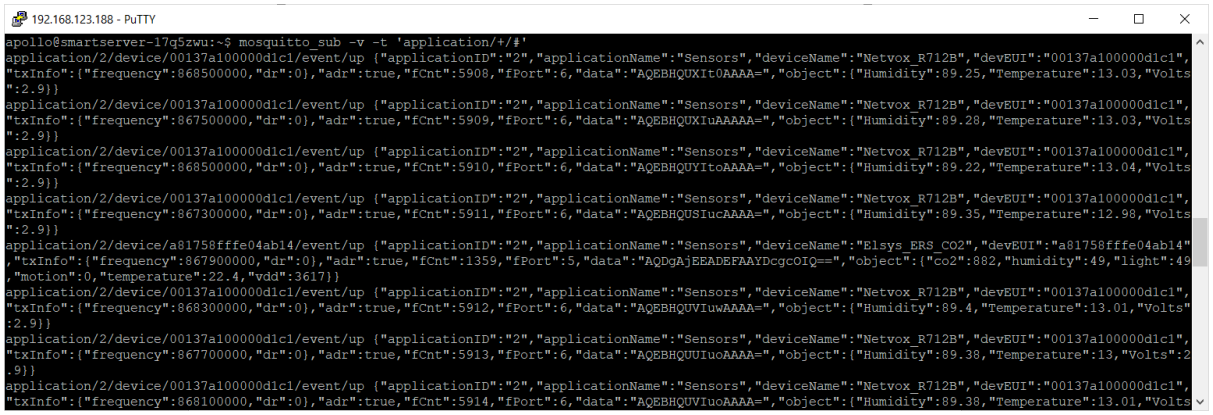
You can find more information on configuring the ERS CO2 at: https://www.elsys.se/en/downlink-generator/



Check that the LoRa Application Server is publishing data to the message broker from the console connection using the following command:

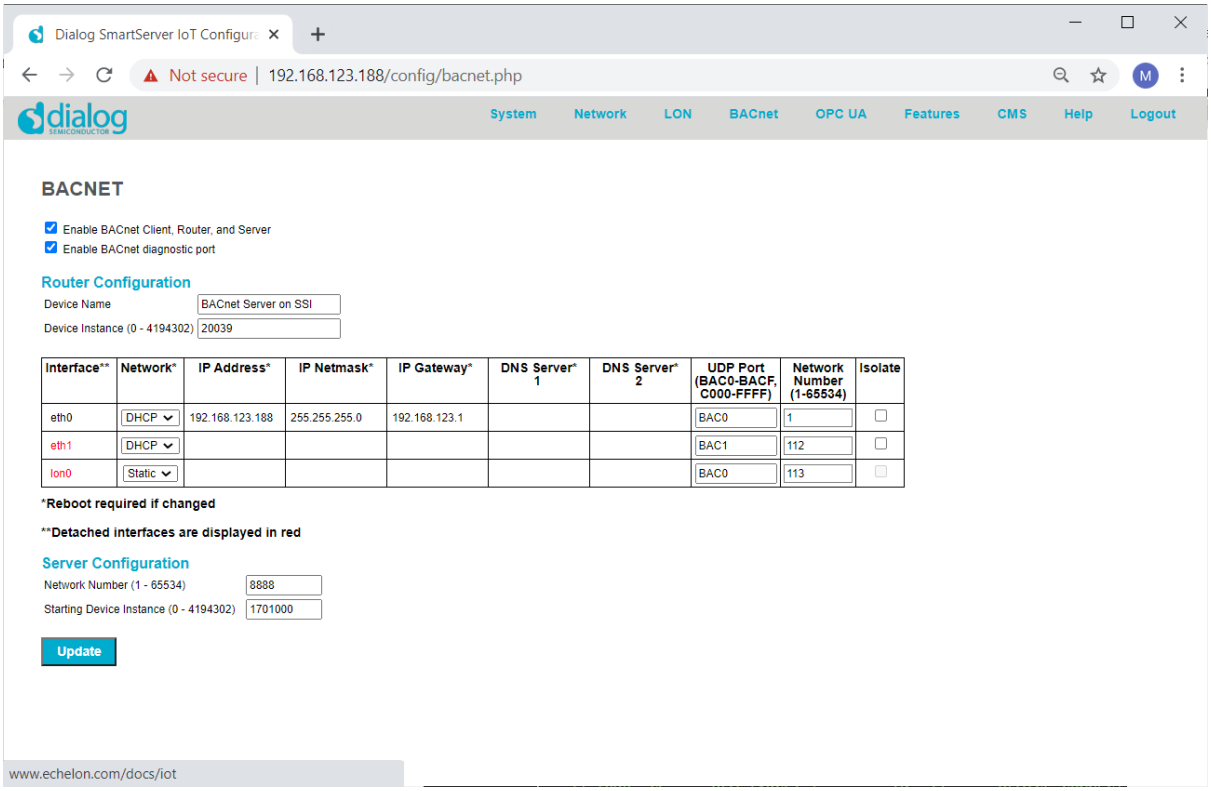
$ mosquitto_sub -v -t 'application/+/#'

You should see similar data to that shown below:



Node.js LON Internal App Installation

1. Enable BACnet from the SmartServer IoT Configuration UI, as shown below:



2. From the GitHub repository download, import the lora.dtp file using the SmartServer IoT's Device widget import device types feature.

   The .dtp includes the following:

   - lora_gw.xif
   - lora_gw.dtd
   - Type files for ufptLoRaGateway
   - lora_gw.btm

3. You can run the application under Visual Studio Code from your PC or on the SmartServer IoT but before you do either, change the DEV_TARGET IP address of the SmartServer IoT in lora_gw\.vscode\launch.json file, as shown below using a suitable editor.

```
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=83
0387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Launch Program",
            "env":{"DEV_TARGET":"192.168.123.188",
```

```
            },  // TODO: Change DEV_TARGET for your IP address, DEVICE_CONNECT
ION_STRING for your credentials
            "program": "${workspaceFolder}\\lora_gw.js"
        }
    ]
}
```

5. Change the lora_deveui_1 and lora_deveui_2 variables to match those of your sensors.

```
// LoRa Topics
// Add or change topic references here for other LoRa configurations
let lora_application_topic = 'application/+/#';   // Catch all application topic
let lora_app_id = 2; // Application #2
let lora_deveui_1 = "00137a100000d1c1"; // Device DEVEUI 00137a100000c425 R712 Tem
p & RH
let lora_deveui_2 = "a81758fffe04ab14"; // Device DEVEUI a81758fffe04ab14 ERS CO2
& Illuminance
let lora_up_topic_1 = "application/" + lora_app_id + "/device/" + lora_deveui_1 +
"/event/up"; // Specific uplink message from sensor #1
let lora_up_topic_2 = "application/" + lora_app_id + "/device/" + lora_deveui_2 +
"/event/up"; // Specific uplink message from sensor #2

function initializeInputs (interfaceObj) {
    // This function must not be called before the MQTT connection to the SmartSer
ver has be established,
    // and the Internal device has been created and provision by IAP/MQ
    clearTimeout (myDevCreateTmo); // Cancel the auto internal device create

    const pointDriverProps = {
        rate: 0,
        "lon.cfg" : {
            propagationHeartbeat: 180,
            propagationThrottle: 0,
            maxRcvTime: 0,
            propagationThreshold: 5
        }
    };
```

6. To run on your PC, please review the section
   http://docs.adestotech.com/display/PortSSIoT/Programming+Tutorial

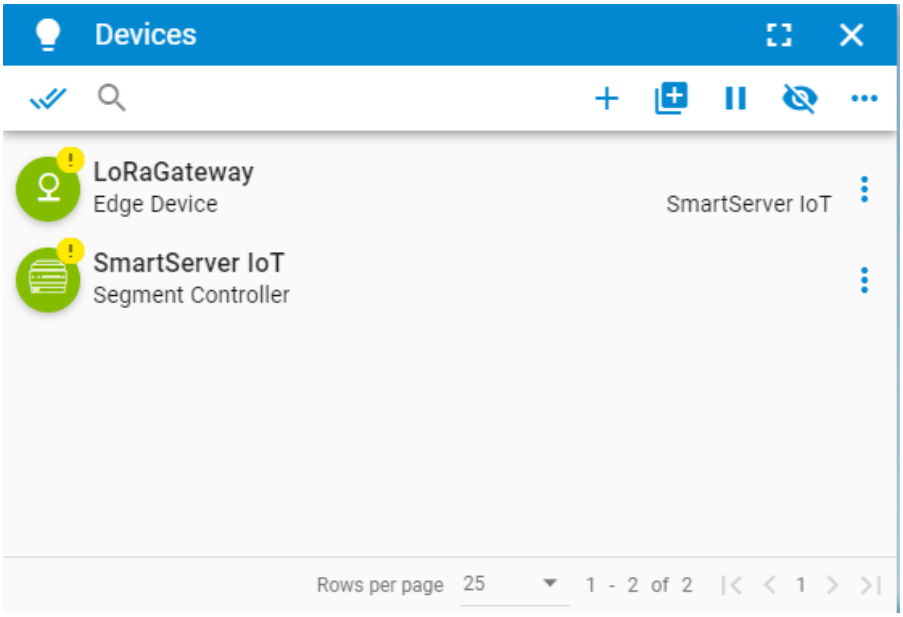
7. To run on the SmartServer IoT, once edited as above, from the cloned GitHub respository copy the lora_gw folder to the /var/apollo/data directory on the SmartServer IoT and copy the lora_gw.conf file to the /etc/supervisor/conf.d directory.

Reload supervisord using the following command:

$ sudo supervisorctl reload

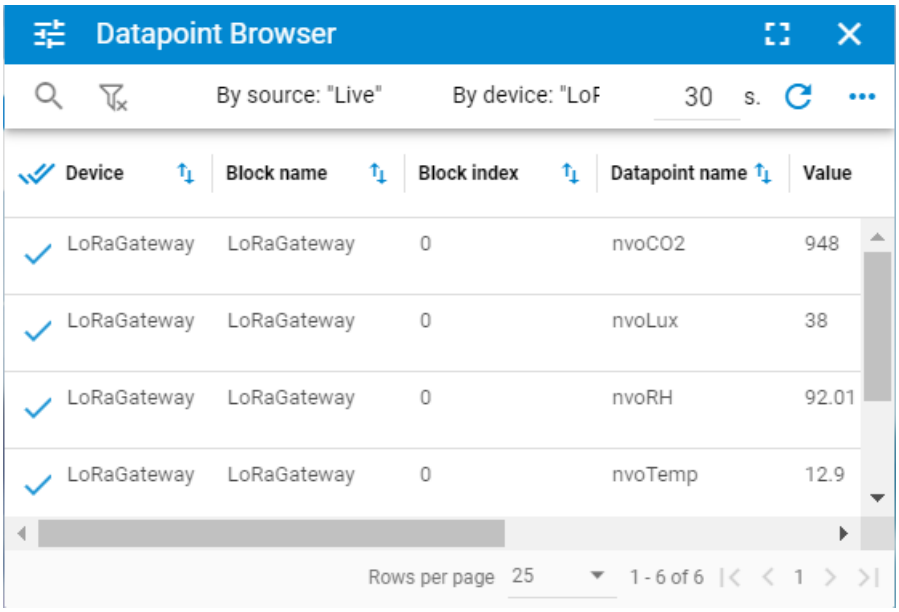
Supervisord will ensure that stdout is written to /var/log/supervisor/stdout-lora_gw.log and stderr is written to /var/log/supervisor/stderr-lora_gw.log

7. Provision the Internal Application from the Device widget

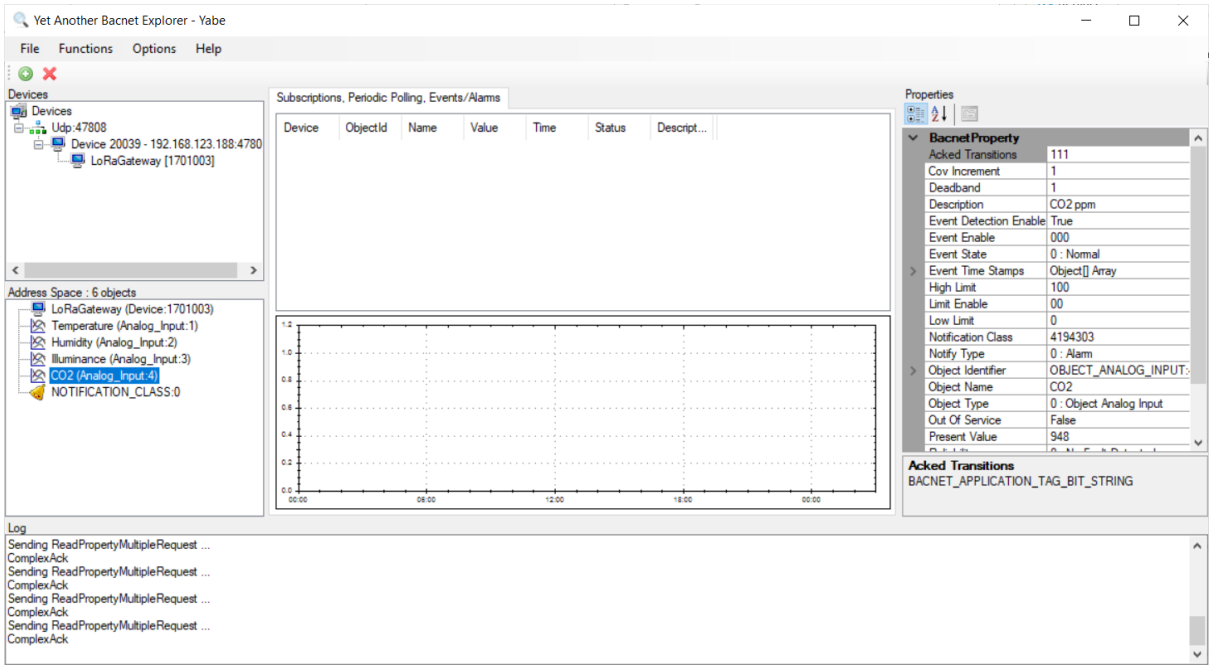


Test the Installation

1. Make sure that sensible non-zero values are present in the Datapoint Browser widget as shown below:



2. Check that data is being published to BACnet using a suitable browser such as Yabe as shown below:



Updating the LON Application's Gateway Interface

Should you wish to change or add datapoint to the gateway, in the first instance, you will need to modify or create your own UFPT using the Resource Editor. Please read the following guide:

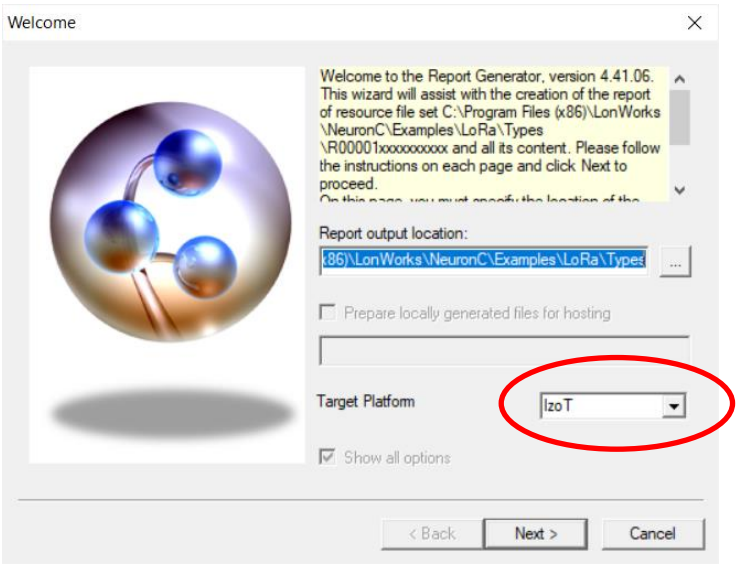
http://docs.adestotech.com/display/PortSSIoT/Creating+an+XIF+for+an+Internal+LON+App

The Resource Editor can be downloaded from here:
http://docs.adestotech.com/display/PortTL/IzoT+Resource+Editor

III can be downloaded from here:
http://docs.adestotech.com/display/PortTL/IzoT+Interface+Interpreter

Then follow the following steps:

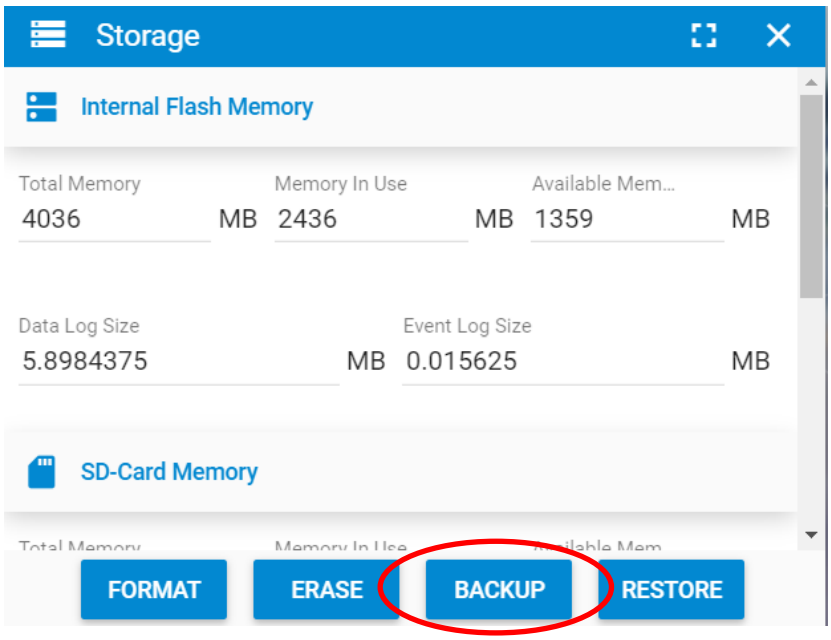
- Stop the lora_gw from within supervisord using
  - `sudo supervisorctl stop lora_gw`
- Run a report within the Resource Editor on the new UFPT type to generate xml versions of the resource files by specifying the Target Platform as xml (the default)
- Run a report within the Resource Editor on the new UFPT type to generate IzoT python versions of the resource files by specifying the Target Platform as IzoT as shown below:



- Create the .xif by running iii.exe with reference to the examples in the Interface Dev folder
- Modify lora_gw.js to support the new interface in the places highlighted in the comments
- Update the .dtd and .btm
- Recreate the .dtp with the new .xif, xml type files, .dtd, .dla and .btm
- Remove the existing gateway device in the CMS Devices widget
- Remove the existing gateway type in the CMS Device Type Widget
- Import the .dtp
- Debug lora_gw from within Visual Studio Code
- Copy the new lora_gw.js to the SmartServer
- Test accordingly

Backing up the Installation

Once everything is up and running, backup your installation using the Storage widget's Backup System feature as shown below. You may have to enable the widget by editing the dashboard's settings.



If this configuration needs to be rolled out to more than one SmartServer IoT, you might want to use cloning as described here:
http://docs.adestotech.com/display/PortSSIoT/Cloning+and+Deploying+a+SmartServer+to+a+Multiple+Sites