Connecting Sigfox backend to Watson IoT Platform

Requirements

- A physical device must be registered and connected to the Sigfox backend.
- A Bluemix application based on the "Internet of Things Platform Starter" boilerplate.
- Access to the Sigfox backend is required to setup the integration towards the Watson IoT Platform application in Bluemix.

Steps

1. Introduction

IoT devices can be connected to the Sigfox network and take benefit from long-range broadcast capabilities. This guide describes how to create a connection between devices of a given Device Type within a Group in the SigFox Backend and an IBM Bluemix application where we'll build an IoT Gateway in order auto register the devices in the Watson Internet of Things platform as individual devices.

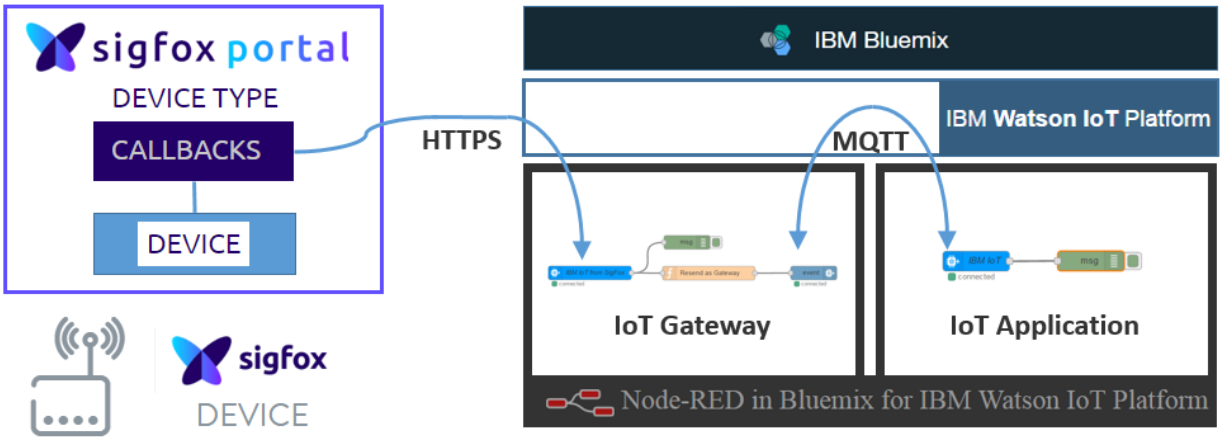
The guide takes you through the following steps:

- Create and setup a specific device type and device in the Watson IoT Platform (WIoTP) which will be needed to take care of the authentication between Sigfox backend and the Watson IoT Platform application.
- Create and setup another device type in the WIoTP which will correspond to the device type on the Sigfox backend.
- Create and setup a device type for the Gateway.
- Create the Callback definition in the Sigfox backend to connect it to your Bluemix WIoTP application using the device REST API.
- Create the Node-Red flow in Bluemix to act as an IoT Gateway, which receives data through the REST API, repackage the payload and submits that as MQTT messages
- Create the Node-Red flow to receive the MQTT messages

The starting step of this guide supposes that your Sigfox device is already registered and connected to the Sigfox network. Pushing a message from your Sigfox device is vendor-dependent and is therefore not documented here. The scope focuses on all

additional steps required to transmit your Sigfox data to the Watson IoT Platform for further processing.

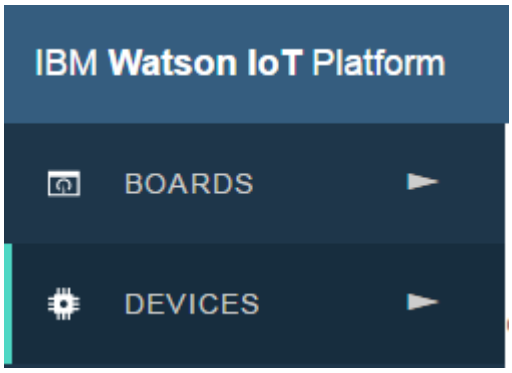
Architecture Overview:



2. *WIoTP: Create authentication device type and device*

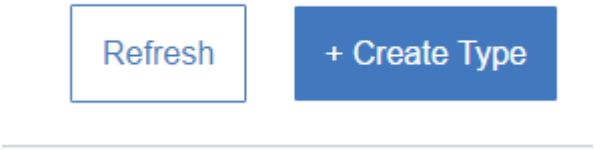
Open your application based on the "Internet of Things Platform Starter" boilerplate and navigate to the Watson IoT Platform dashboard.

2.1. Select Devices:

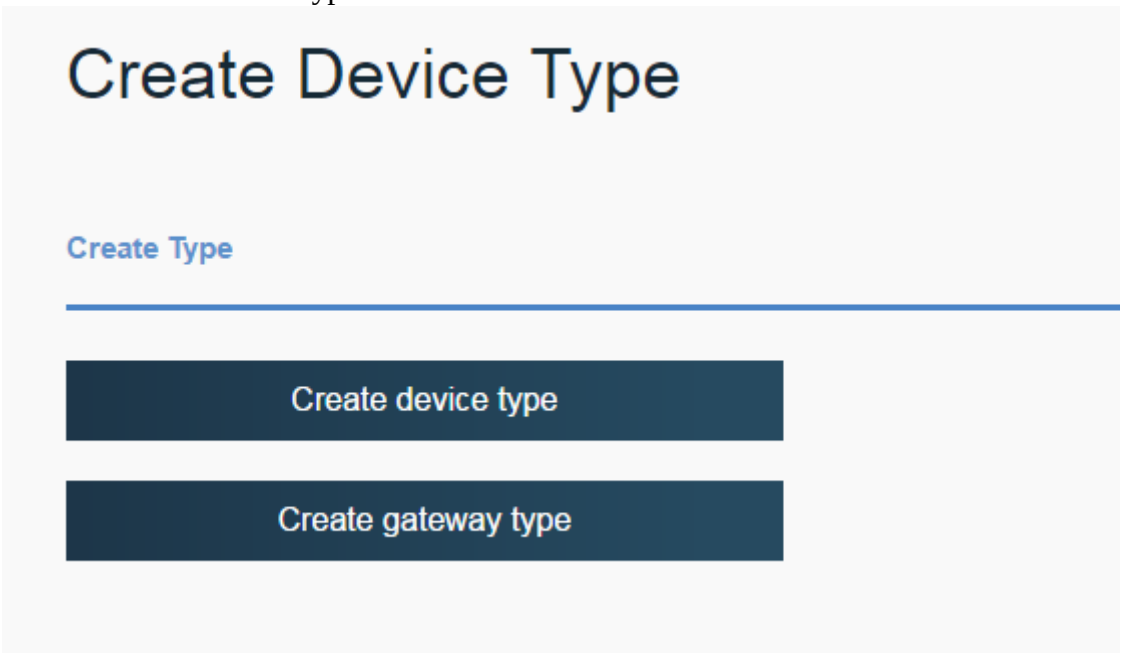


2.2. Select Device Types

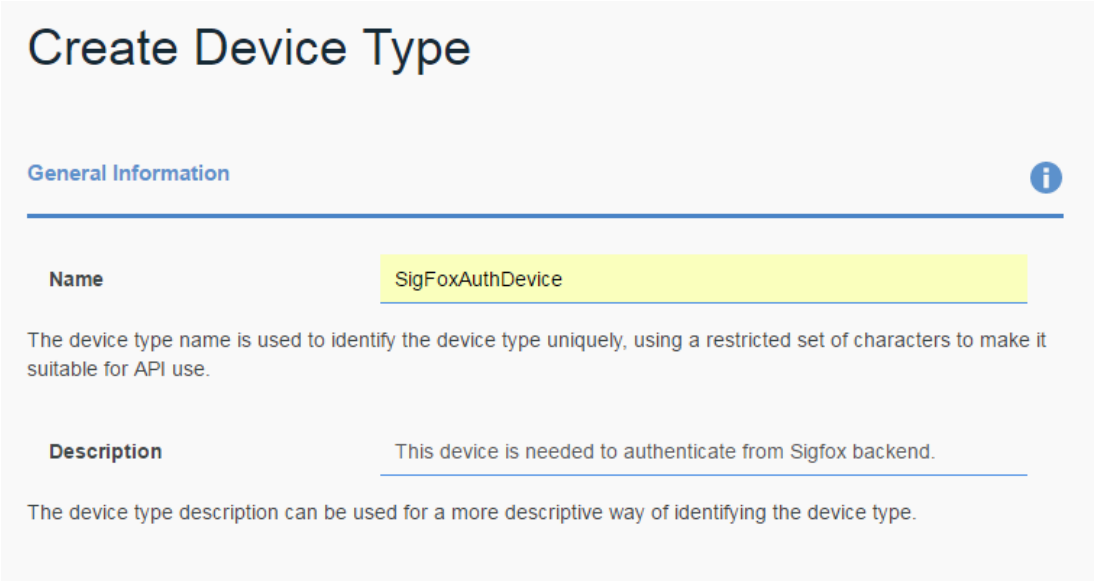
2.3. Create a new Device Type, press +Create Type in the upper right corner:



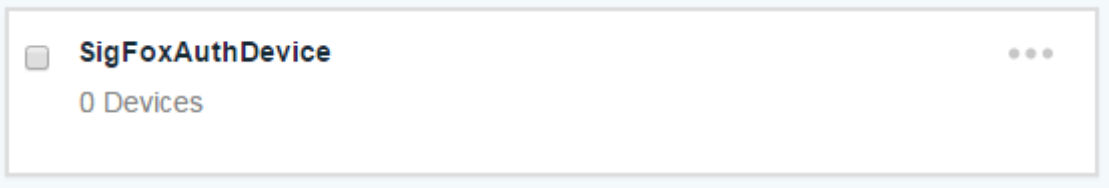
2.4. Click on Create device type



2.5. Enter a descriptive name for the device type needed for authentication, e.g. SigFoxAuthDevice – and a good description like: This device is needed to authenticate from Sigfox backend.



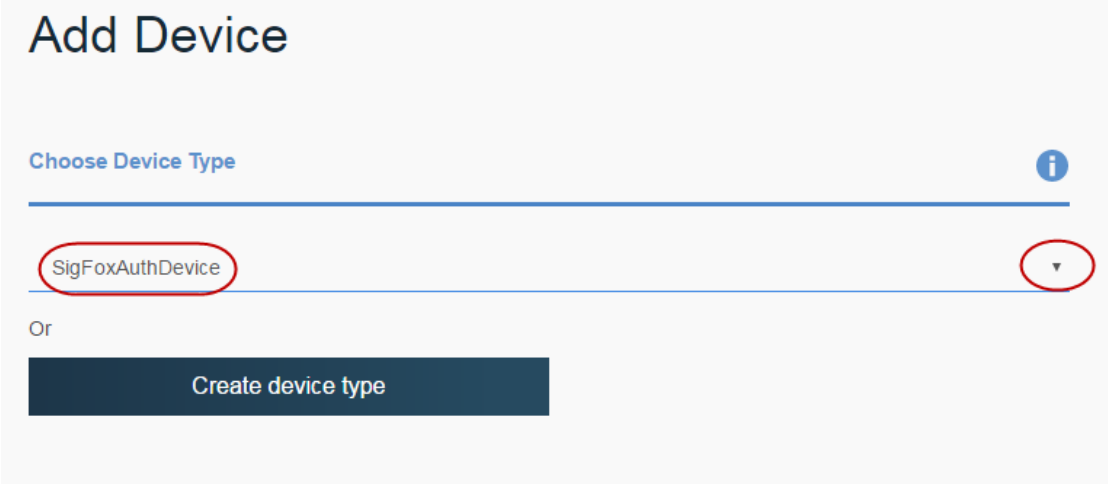
2.6. Press Next (lower right corner), Next, Next, Create and the first device type is created (for this recipe no additional information is entered):



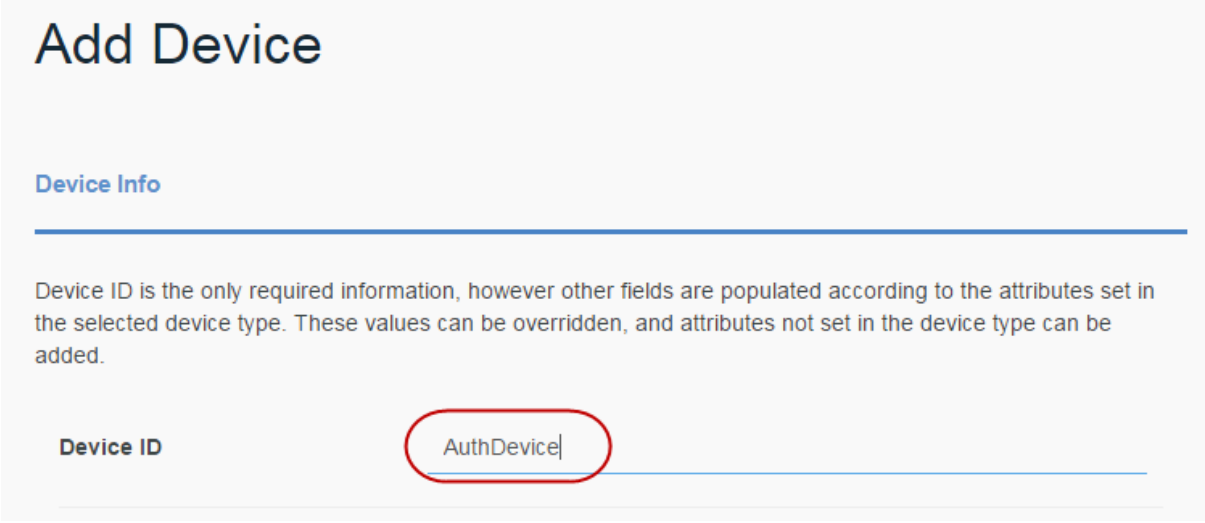
2.7. Now it's time to create the device dedicated to the authorization between Sigfox backend and your WIoTP application. Go to the Browse tab:

2.8. Then press +Add Device:

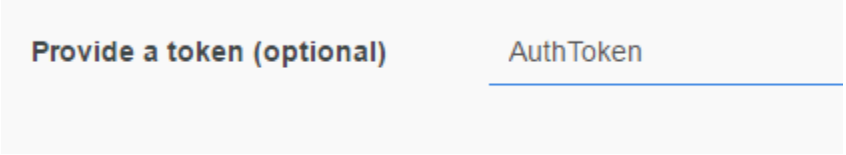
2.9. Select SigFoxAuthDevice at the small widget in the right side and click Next in the lower right corner:



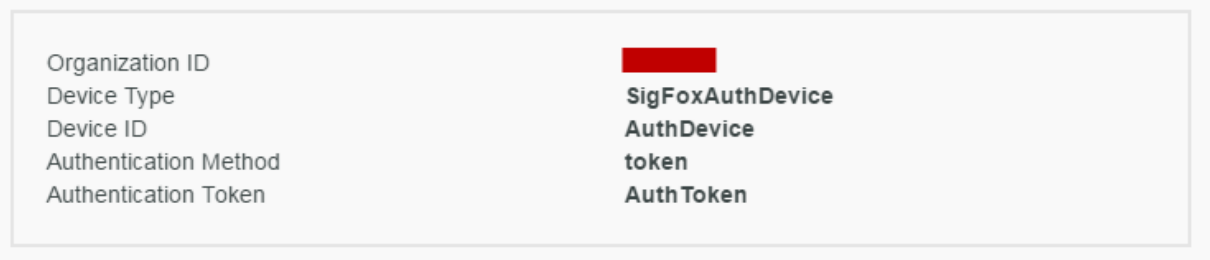
2.10. Enter a usable name, e.g. AuthDevice.



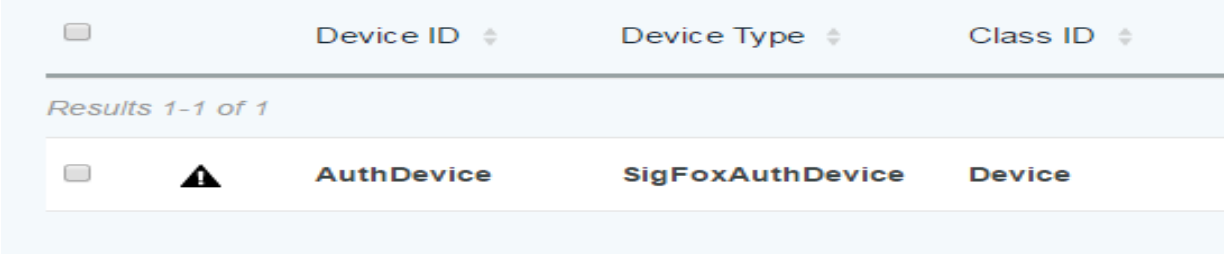
2.11. Press Next twice and you have to define an Authentication Token or let the system create in for you. For the purpose of this recipe, I'll define the Token to be AuthToken:



2.12. Press Next and Add. **Please make a note of the information, as you will need it later** (even the Organization ID, which I have hidden):



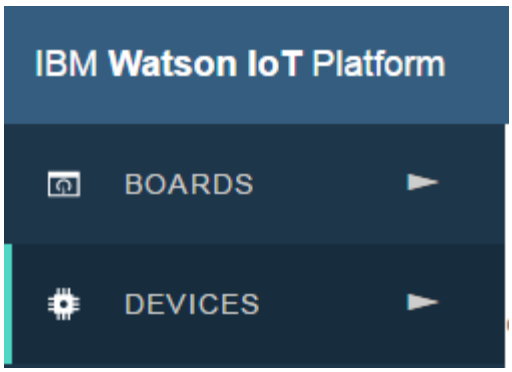
2.13. Close the dialogue, and you will now browse the devices:



3. *WIoTP: Create Sigfox device type*

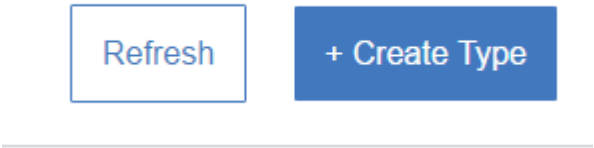
We will now create the device type for the actual Sigfox devices.

3.1. Select Devices:

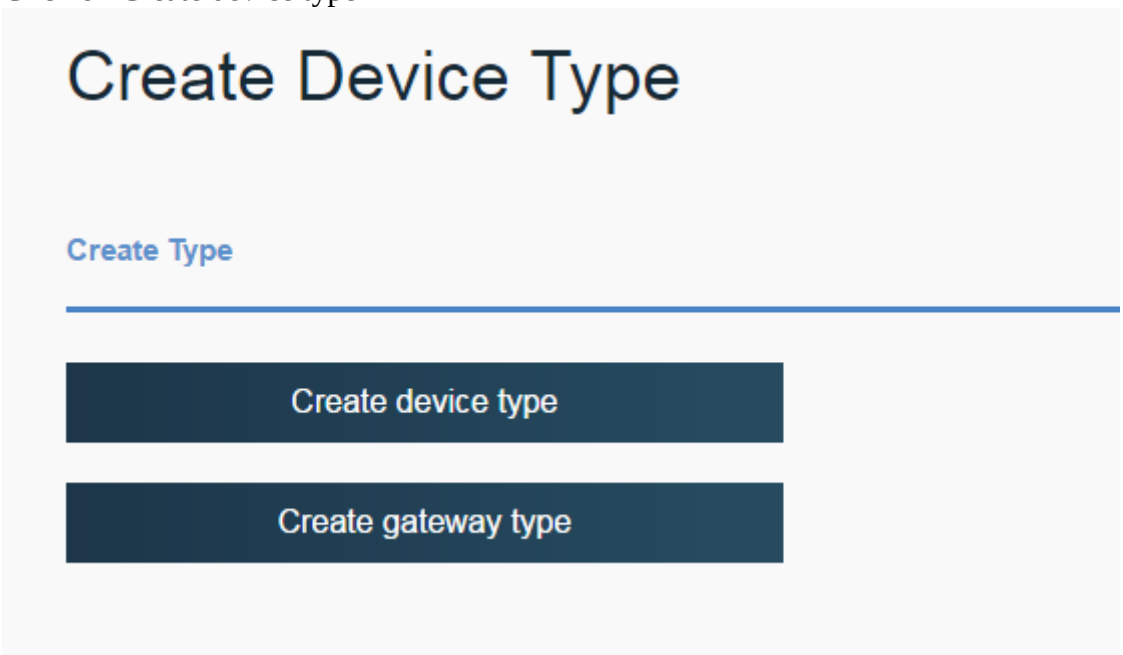


3.2. Select Device Types

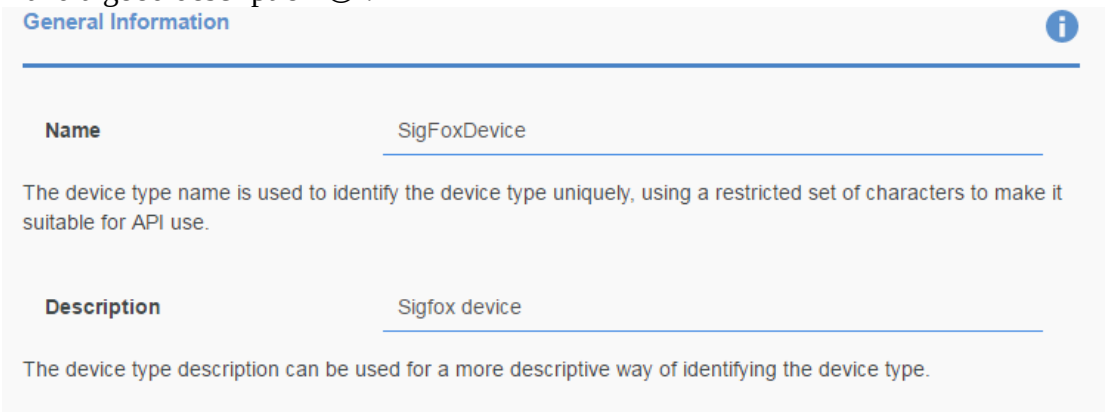
3.3. Create a new Device Type, press +Create Type in the upper right corner:



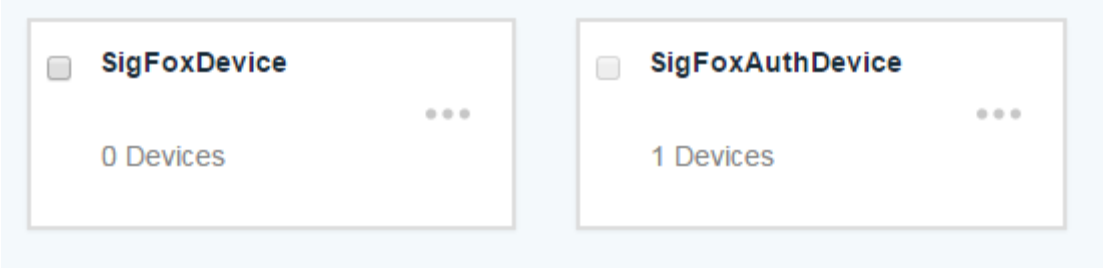
3.4. Click on Create device type



3.5. Enter a descriptive name for the device type for the Sigfox device, e.g. SigFoxDevice – and a good description ☺ :



3.6. Press Next, Next, Next and Create. You know have two Device Types defined:

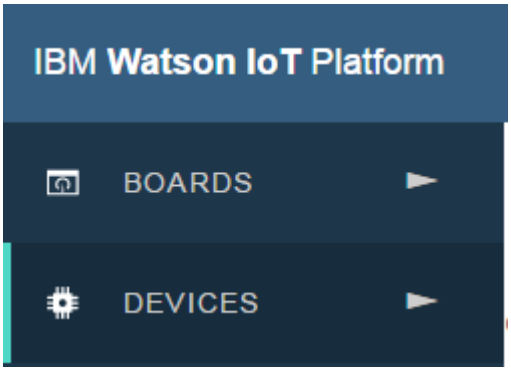


You don't need to create a device of this type, as they will later be auto registered through the Gateway we create in the Next activity.

4. *WIoTP: Create the Gateway type and a Gateway*

We will now create the gateway type for the gateway.

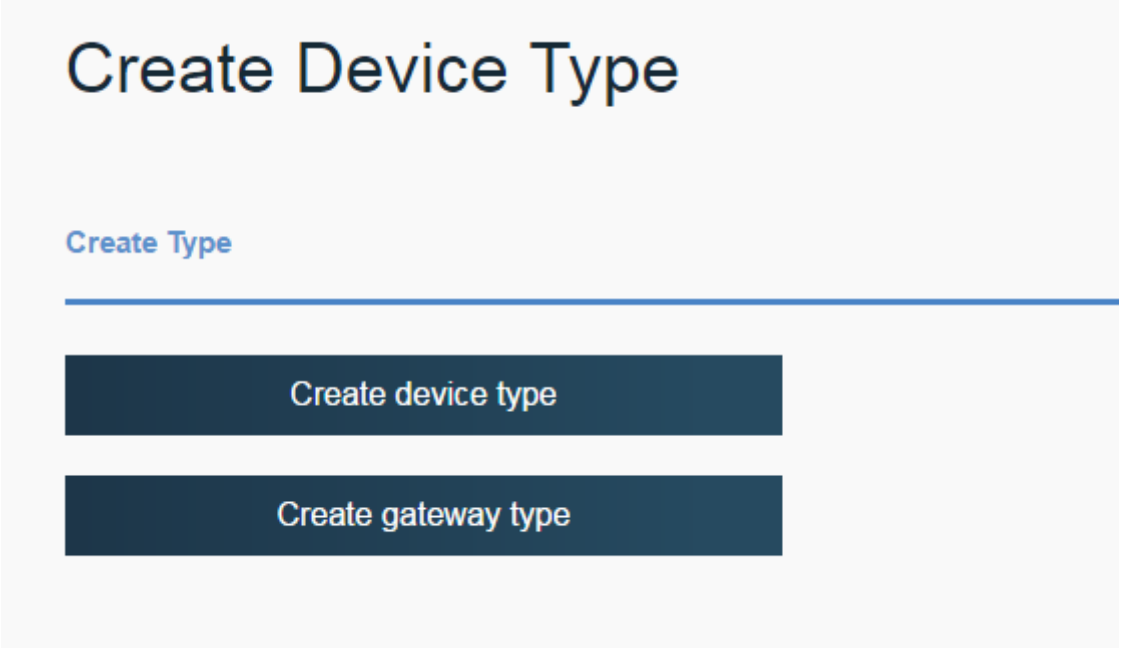
4.1. Select Devices:



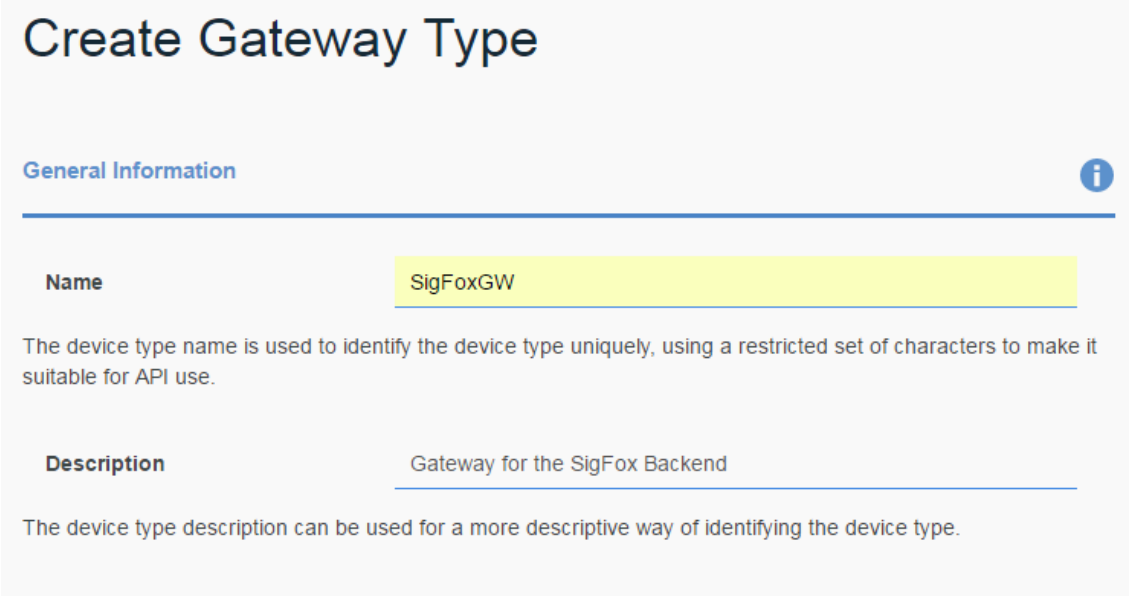
4.2. Select Device Types

4.3. Create a new Device Type, press +Create Type in the upper right corner:

4.4. This time Click on **Create gateway type:**



4.5. Enter a descriptive name for the gateway type, e.g. SigFoxGW – and a good description:

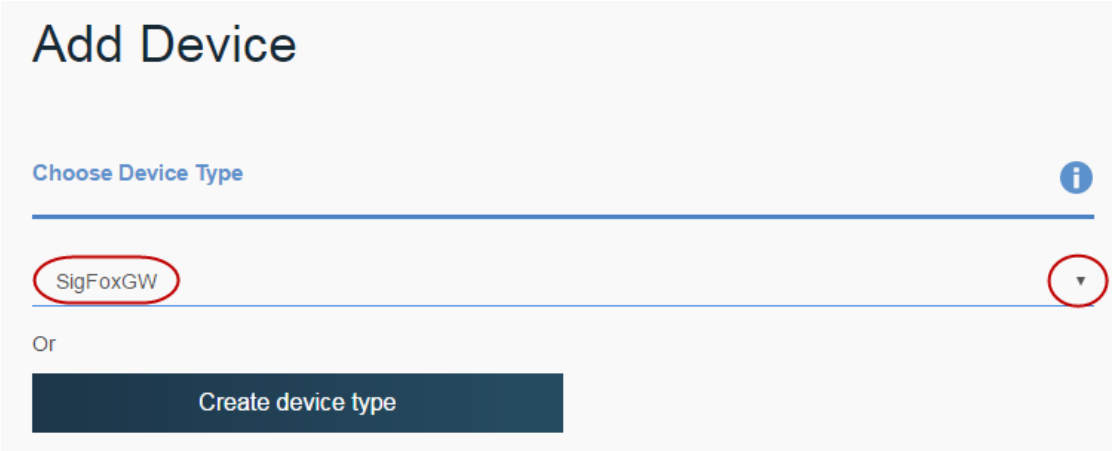


4.6. Press Next, Next, Next and Create. You know have two Device Types and a Gateway Type defined:

4.7. It is now time to create the Gateway. Go to the Browse tab:

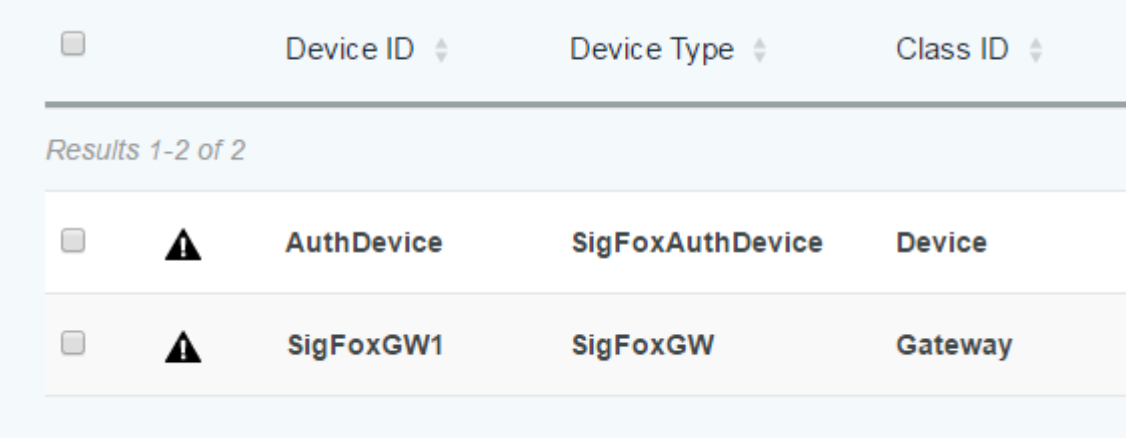
4.8. Then press +Add Device:

4.9. Select SigFoxGW at the small widget in the right side and click Next in the lower right corner:



4.10. Give it a name, e.g. SigFoxGW1 and press Next, Next, leave the Token field empty and press Next and Add. **Please write down the information**:

4.11. You have now created two devices:

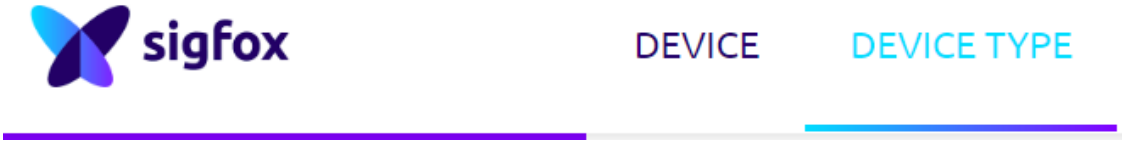


This concludes the creation of device types, gateway type and the device and gateway.

5. Sigfox: Create the backend callback functionality

Now it's time to establish the integration between the Sigfox backend and your WIoTP application in Bluemix.

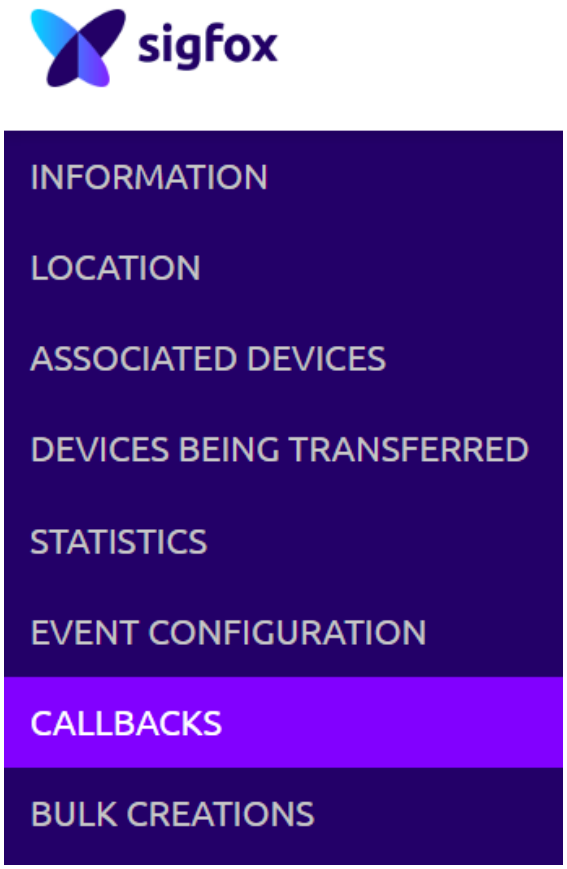
5.1. Go to the Sigfox Backend, where you login with your credentials and go to the Device Type tab:



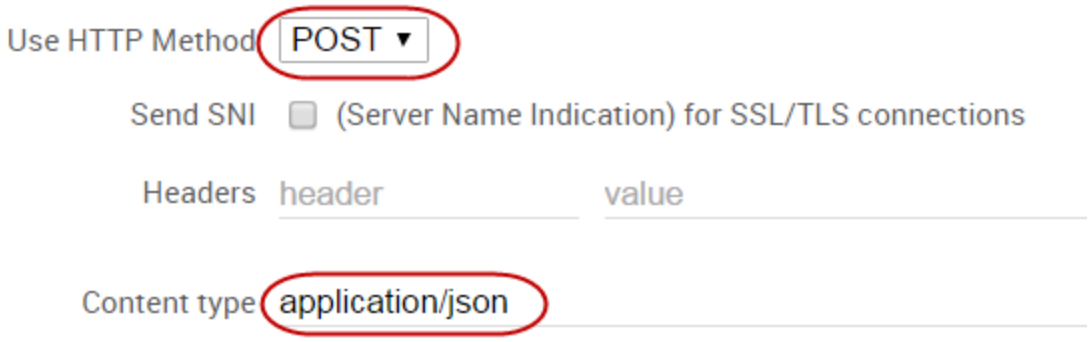
5.2. Select the Device Type you want to connect to the WIoTP. In this case I have a device type created to handle some Sens'it devices:

Description	Display type	Group	Keep alive	Name
Sensit 2.0 Demo devices	Test	IBM Denmark	N/A	Sensit 2.0

5.3. Go to the Callbacks section:



5.4. Click on “New” in upper right corner, select “Custom callback”.
Leave “Type” as “DATA UPLINK”, and “Channel” as “URL”.
Select “POST” as “Use HTTP Method” – and “application/json” as “Content type”:



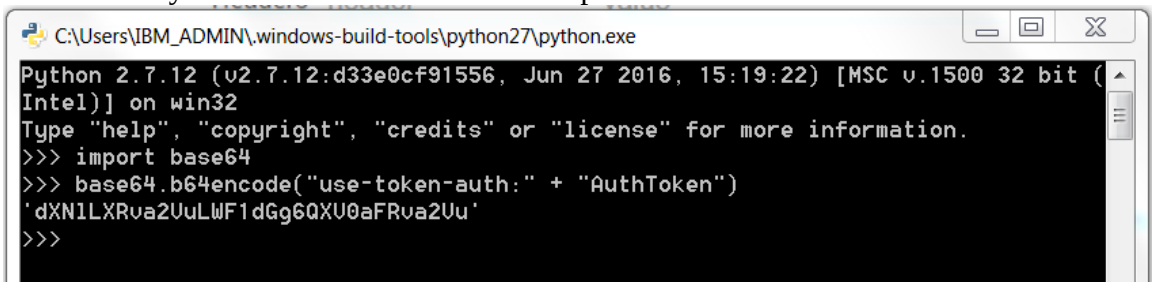
5.5. Open a terminal and execute Python with the following commands (replace all $variable with your context):

>>> import base64

>>> base64.b64encode(“use-token-auth:” + $(your device Authentication Token defined in 2.11))

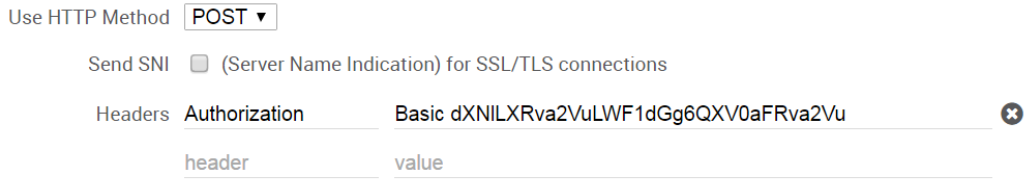
the output is your basic authentification key
– in our case: >>> base64.b64encode(“use-token-auth:” + “AuthToken”) – which returned 'dG9rZW46QXV0aFRva2Vu'

In Windows you would use CMD and the steps would look like this:



5.6. Change the Headers:

- header = “Authorization”
- value = “Basic dXNlLXRva2VuLWF1dGg6QXV0aFRva2Vu”



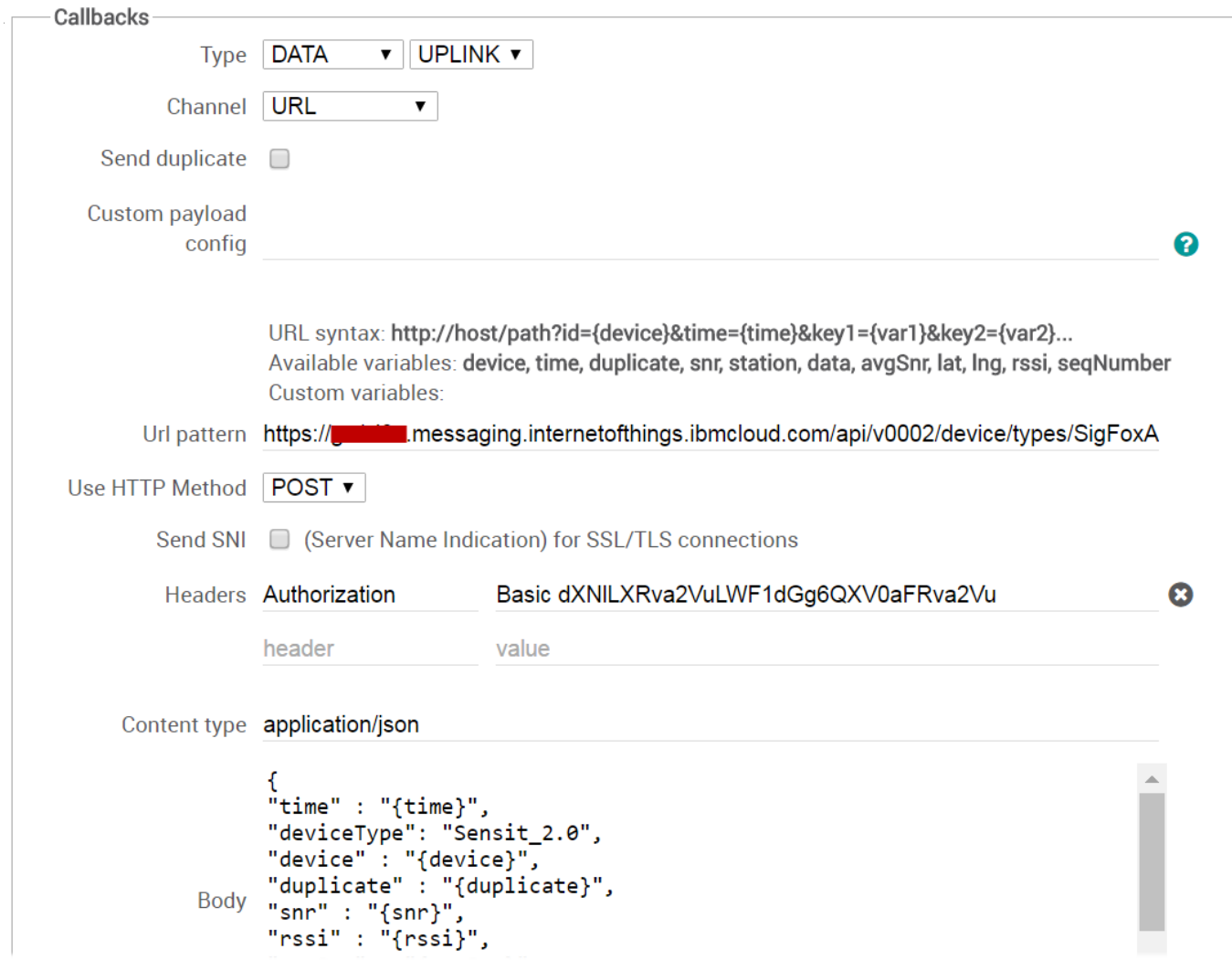
5.7. The “Url pattern” needs to be defined as:

- https://$(IBM organisation).messaging.internetofthings.ibmcloud.com/api/v0002/device/types/$(IBM device type)/devices/$(IBM device ID)/events/$(Topic)

- in this **example** the organization is “ab1cd2” – you should use your own organization: https://ab1cd2.messaging.internetofthings.ibmcloud.com/api/v0002/device/types/SigFoxAuthDevice/devices/AuthDevice/events/status

5.8. Add the content to the Body in order to send over the parameters available + the device type, so I can use that in the Gateway:

```
{
"time" : "{time}",
"deviceType": "Sensit_2.0",
"device" : "{device}",
"duplicate" : "{duplicate}",
"snr" : "{snr}",
"rssi" : "{rssi}",
"avgSnr" : "{avgSnr}",
"station" : "{station}",
"lat" : "{lat}",
"lng" : "{lng}",
"seqNumber" : "{seqNumber}",
"data" : "{data}"
}
```



5.9. Click OK

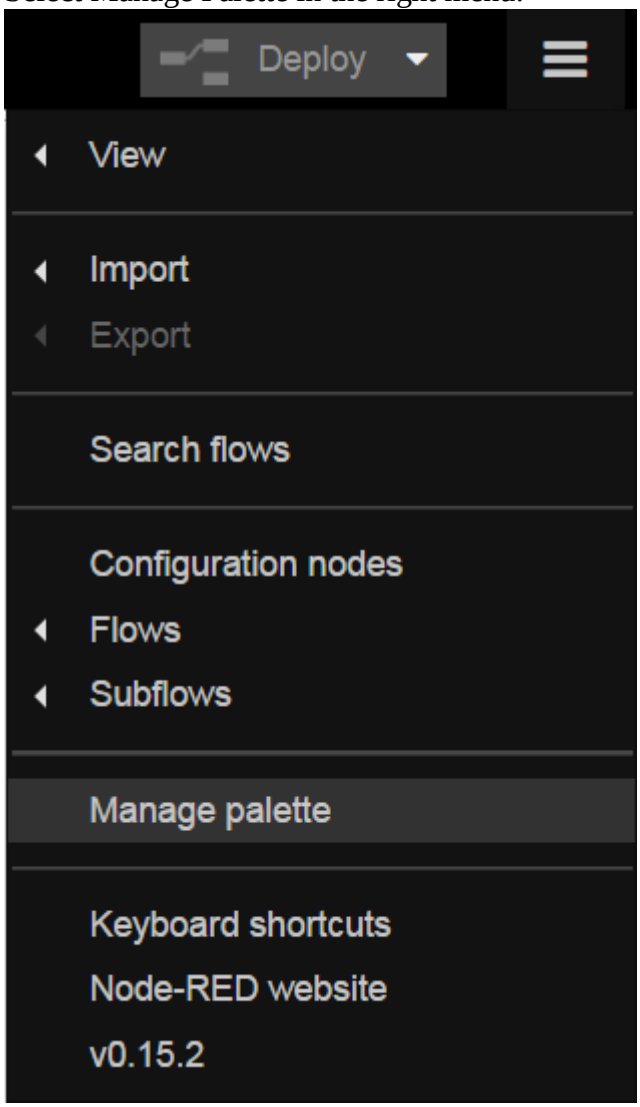
You have now created the link from the Sigfox backend towards your WIoTP application in Bluemix. The next step is to create the IoT Gateway in your WIoTP application in Bluemix.

6. *WIoTP: Create the IoT Gateway in Node-Red*

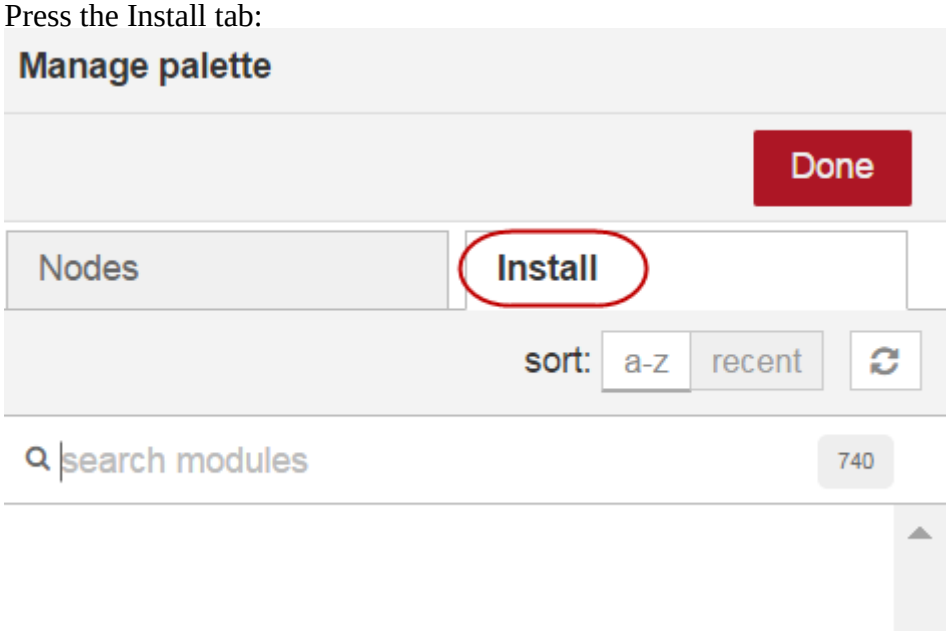
We need to add a couple of nodes to the node library of the WIoTP application before jumping to creating the node-red code for the gateway.

6.1. Go to the Node-red editor of your application, e.g. http://sigfoxgw.eu-gb.mybluemix.net/red/#.

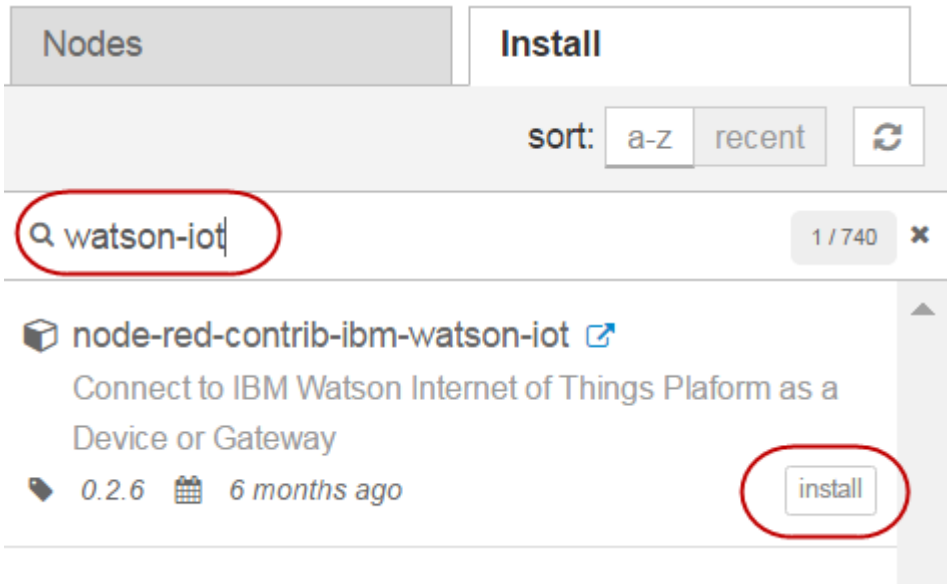
Select Manage Palette in the right menu:



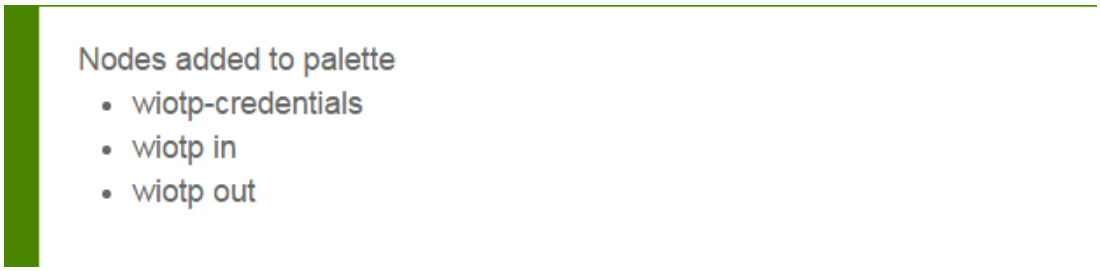
6.2. Press the Install tab:



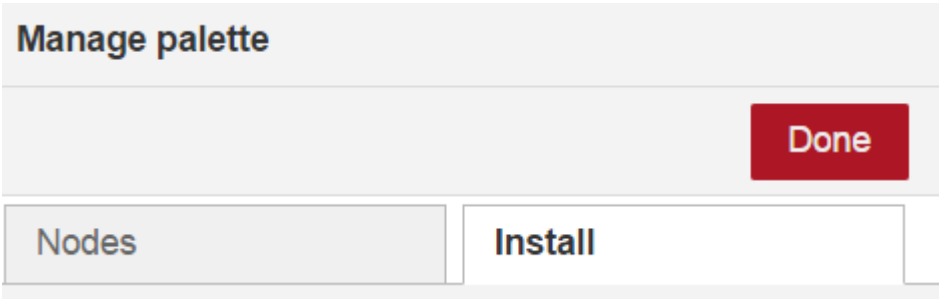
6.3. Write “watson-iot” in the search field and then press “install” once the node-red-contrib-ibm-watson-iot library is found:



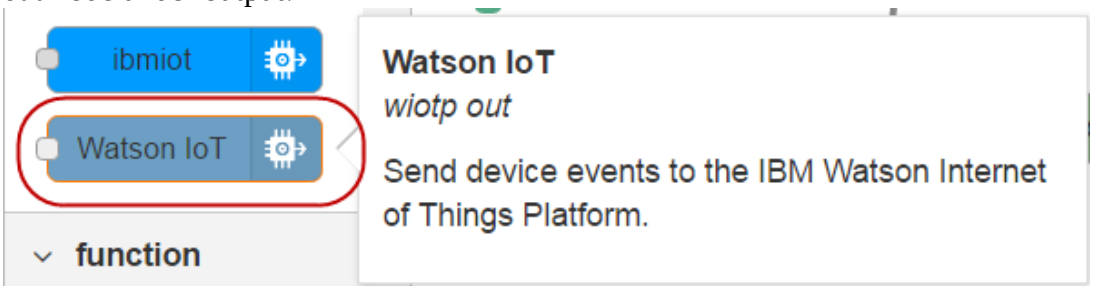
6.4. Wait until the nodes has been added:



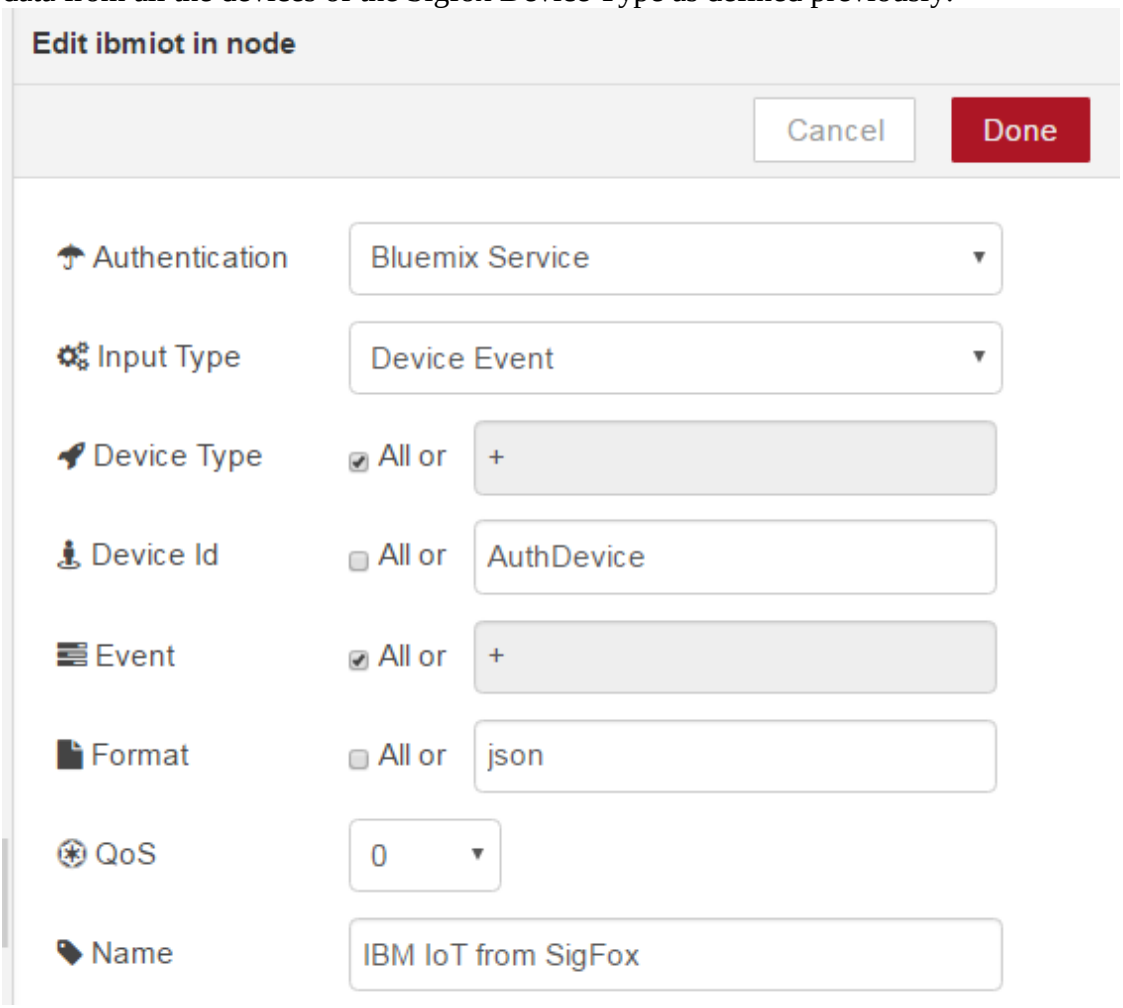
6.5. Press Done:



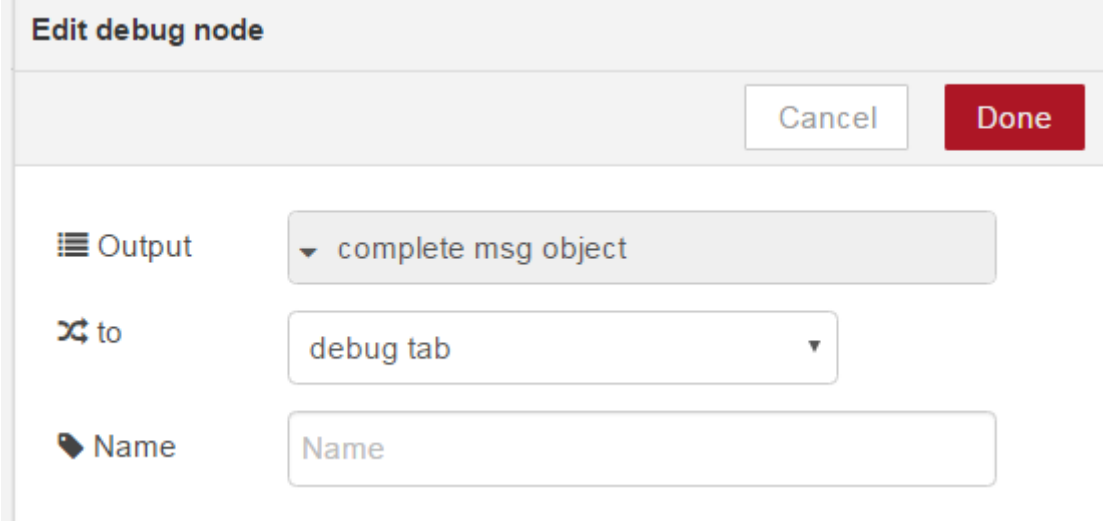
6.6. Reload the web page and you will now see the *wiotp in* node under input and the *aiotp out* node under output:



6.7. You are now ready to create some flows:
Remove the content of the existing Flow 1.
Rename Flow 1 to “Gateway”.
Drag in an ibmiot node and double-click it to open its properties.
Change Authentication to Bluemix Service, and add AuthDevice to the Device Id as this is the device that will forward the data from the Sigfox backend and thereby send data from all the devices of the Sigfox Device Type as defined previously:



6.8. Click Done and drag in a debug node and set that to Output: complete msg object:



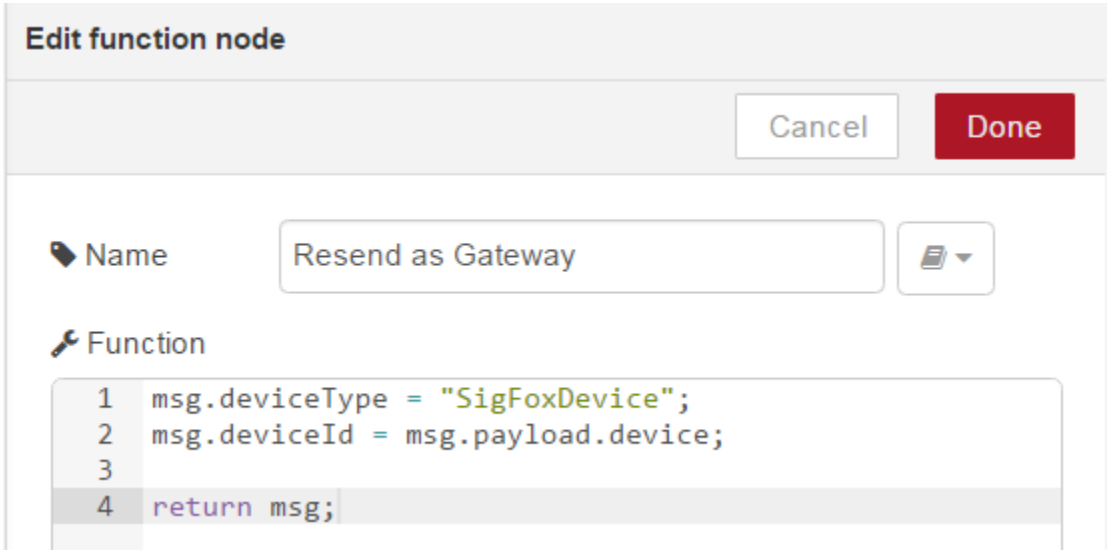
6.9. Connect the two nodes and deploy the application:

6.10. Enable the debug node, select the Debug tab, and wait until a message is received. How long time you have to wait depends on how your sigfox device is configured. It could look like this:

6.11. We will now extract the needed information from this JSON package in order to submit a modified JSON package in MQTT format as the gateway we created previously.
Drag in a function node, give it a good Name, and add the following function code:

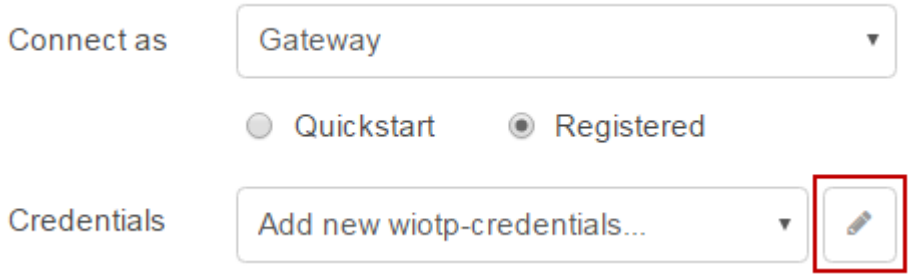
```
msg.deviceType = "SigFoxDevice";
msg.deviceId = msg.payload.device;

return msg;
```

– so it looks like this:

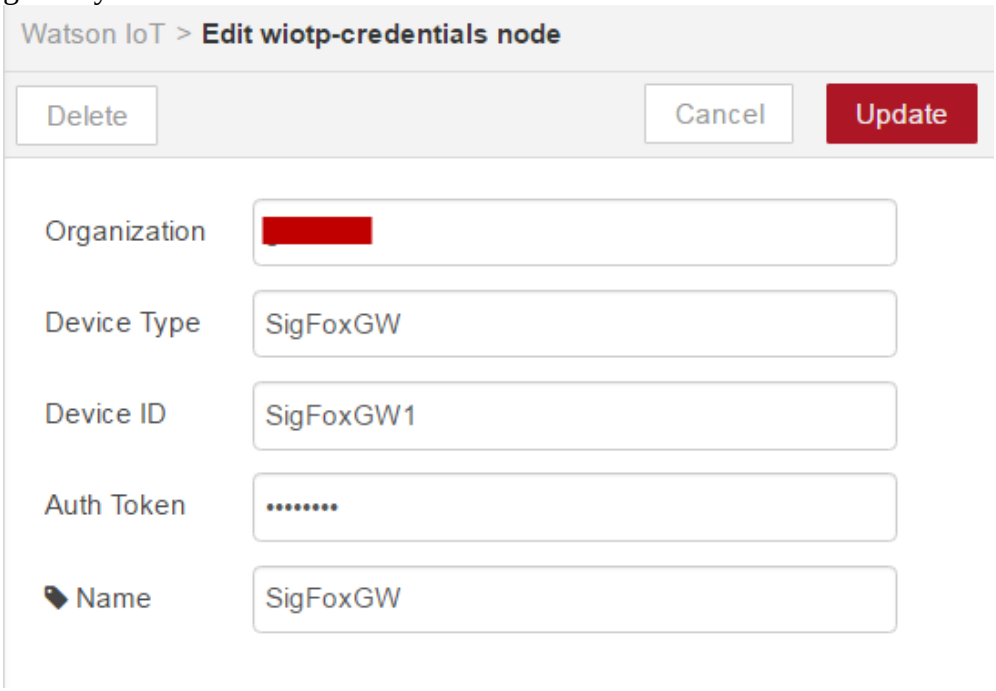


6.12. Press Done. Add a Watson IoT output node (the *wiotp out* node and not the *ibmiot* node) and configure it accordingly:

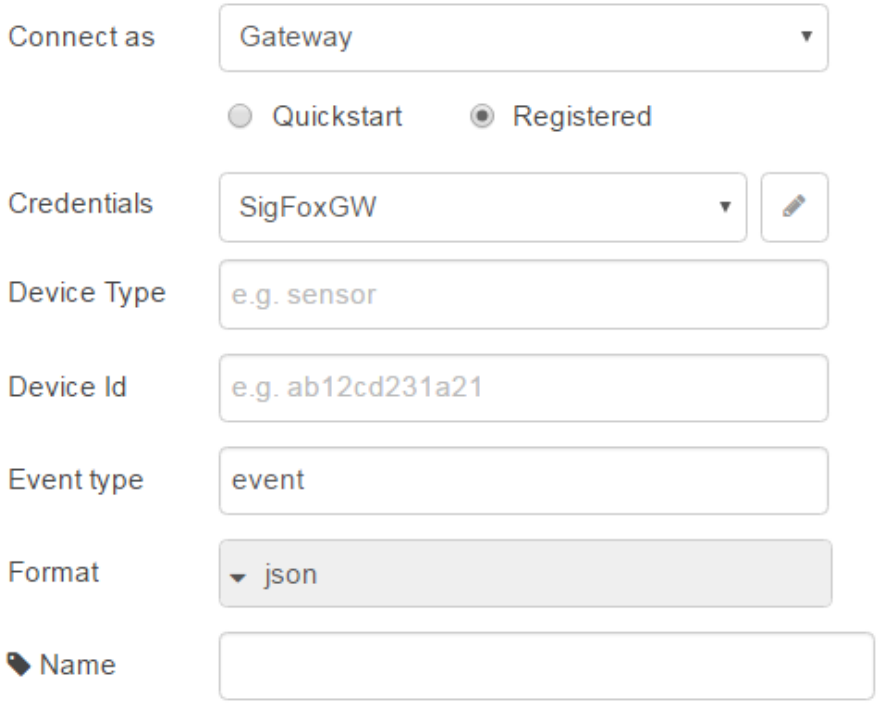
- Connect as Gateway
- Select Registered
- click at the pencil to edit the wiotp-credentials:



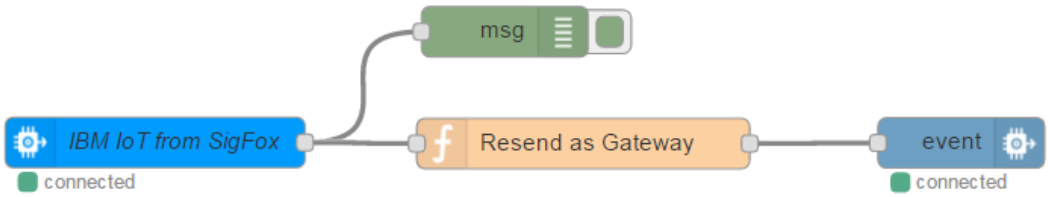
- Enter the credentials data as you produced while creating the SigFoxGW1 gateway:



- Click Update and then Done. You should now you see this:



6.13. Now connect the dots, and deploy. Then you'll see this flow:



The gateway is now created and all incoming Sigfox device messages are received and published via MQTT as separate devices. An advantage of this approach is that new Sigfox devices that use the same Sigfox Device Type in the Sigfox backend will be auto registered in the Watson IoT Platform as individual devices – we'll look at that in the end of the next section.

7. WIoTP: Create the Node-Red flow for receiving the device data

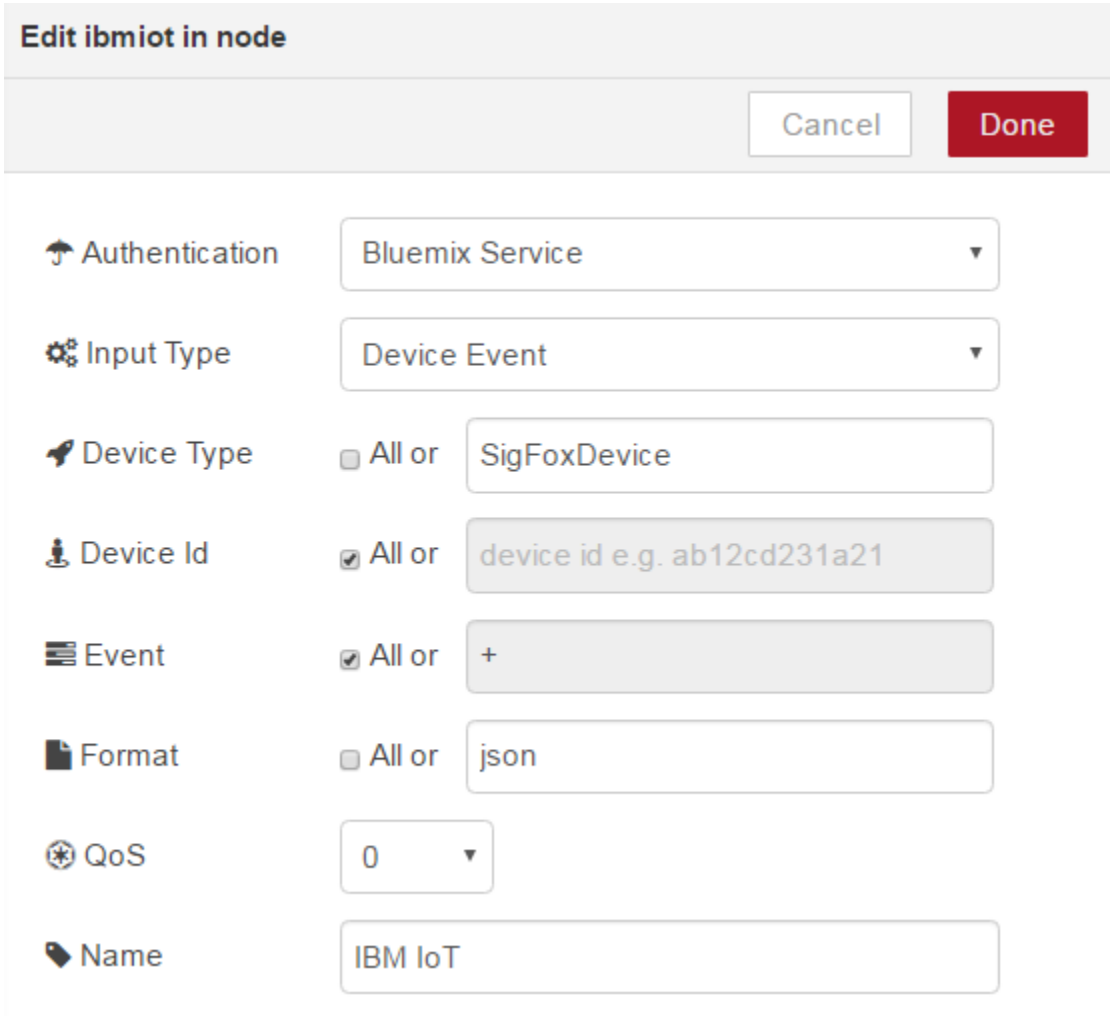
We'll now create a very simple flow to demonstrate how the data is received in a normal WIoTP application, so we can see the how the JSON package looks like.

7.1. Open your node-red editor and create another flow which you call IoT Platform:

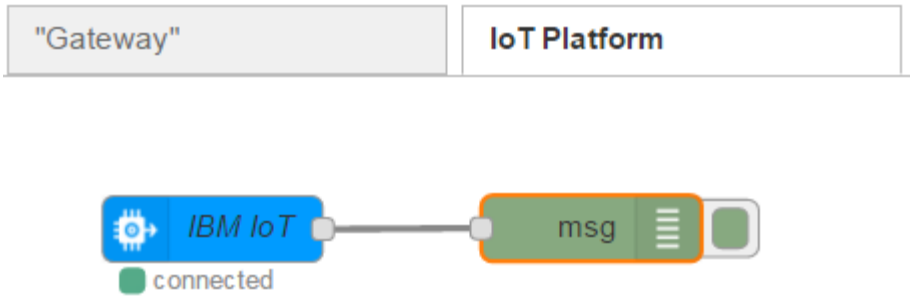
7.2. Drag an ibmiot input node into the flow.

- Change Authentication to Bluemix Service
- Deselect All after Device Type and write SigFoxDevice instead of the +

- Select All after Device Id:



7.3. Add a debug node to the flow and set its output to show complete msg object. Connect the two nodes and Deploy:



7.4. Now wait until a new message is received and then compare the two debug messages.

- The first one is the message received from the Sigfox backend, where the deviceId and deviceType is generic for all incoming messages:

iot-2/type/SigFoxAuthDevice/id/AuthDevice/evt/status/fmt/json : msg : Object

{ "topic": "iot-2/type/SigFoxAuthDevice/id/AuthDevice/evt/status/fmt/json", "payload": { "time": "1475938952", "deviceType": "Sensit_2.0", "device": "1CB1A5", "duplicate": "false", "snr": "16.75", "rssi": "-27.00", "avgSnr": "99.68", "station": "347E", "lat": "56", "lng": "12", "seqNumber": "396", "data": "8169084b" }, "deviceId": "AuthDevice", "deviceType": "SigFoxAuthDevice", "eventType": "status", "format": "json", "_msgid": "e8b5ae9b.174a5" }

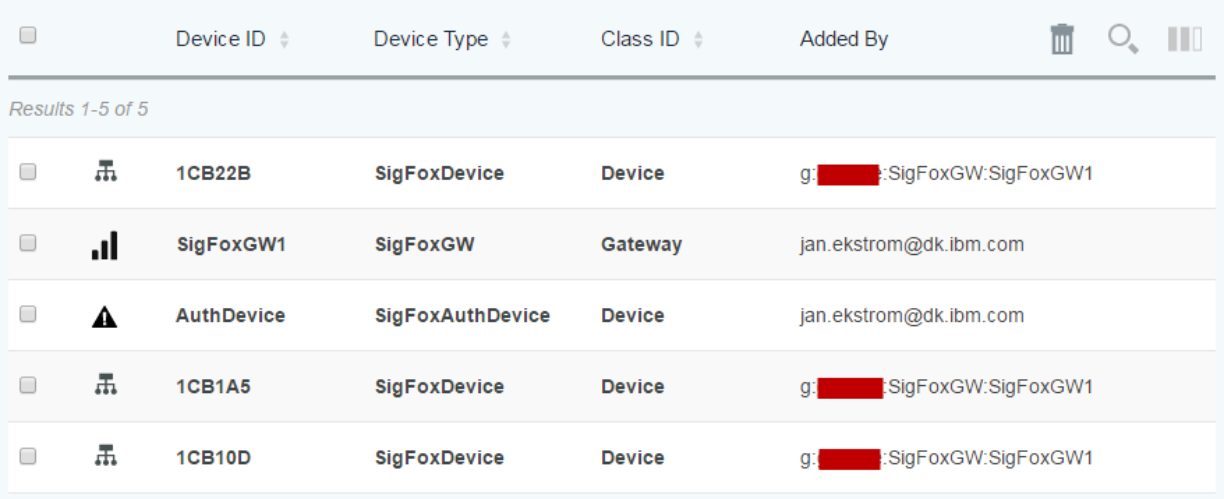
- Secondly the message received in the IoT Platform flow from the "Gateway", where the message now comply with the MQTT standard and the deviceId and deviceType is from the individual device:

iot-2/type/SigFoxDevice/id/1CB1A5/evt/event/fmt/json : msg : Object

{ "topic": "iot-2/type/SigFoxDevice/id/1CB1A5/evt/event/fmt/json", "payload": { "d": { "time": "1475938952", "deviceType": "Sensit_2.0", "device": "1CB1A5", "duplicate": "false", "snr": "16.75", "rssi": "-27.00", "avgSnr": "99.68", "station": "347E", "lat": "56", "lng": "12", "seqNumber": "396", "data": "8169084b" } }, "deviceId": "1CB1A5", "deviceType": "SigFoxDevice", "eventType": "event", "format": "json", "_msgid": "a4db292f.5b24d8" }

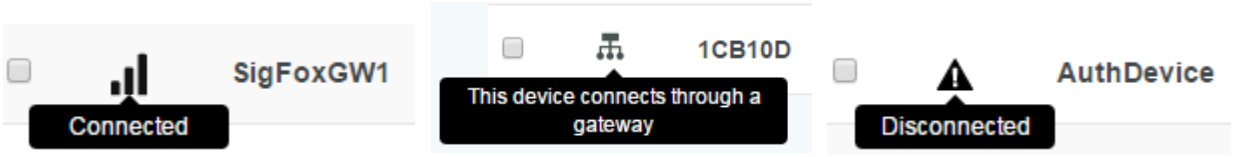
Optional:

Connect 2 additional sensors of the same type to the Sigfox backend. Data started to flow into the Watson IoT Platform and the new devices was auto registered. Go to the Watson IoT Platform and browse the devices:



Device ID	Device Type	Class ID	Added By
Results 1-5 of 5			
1CB22B	SigFoxDevice	Device	g:[redacted]:SigFoxGW:SigFoxGW1
SigFoxGW1	SigFoxGW	Gateway	jan.ekstrom@dk.ibm.com
AuthDevice	SigFoxAuthDevice	Device	jan.ekstrom@dk.ibm.com
1CB1A5	SigFoxDevice	Device	g:[redacted]:SigFoxGW:SigFoxGW1
1CB10D	SigFoxDevice	Device	g:[redacted]:SigFoxGW:SigFoxGW1

NB. the icons in front of the devices depict its state:



The AuthDevice is not connected since it doesn't utilize MQTT but is only used for authenticate the callbacks.