
General Configure/Commands to Connect to IoT server for -NB & -NS NB-IoT models

last modified by Mengting Qiu

on 2024/08/28 14:44

Table of Contents

1. The use of this guideline	5
2. Attach Network	5
2.1 General Configure to attach network	5
2.2 Speed Up Network Attach time	6
3. Configure to connect to different servers	7
3.1 General UDP Connection	7
3.1.1 Simulate UDP Connection by PC tool	7
3.1.2 Configure NB-IoT Sensor	7
3.2 General MQTT Connection	8
3.3 ThingSpeak (via MQTT)	10
3.3.1 Get MQTT Credentials	10
3.3.2 Simulate with MQTT.fx	12
3.3.3 Configure NB-IoT Sensor for connection	15
3.4 Datacake	17
3.4.1 For device Already has template	18
3.4.2 For Device already registered in DataCake before shipped	23
3.4.3 Manual Add Decoder in DataCake (don't use the template in DataCake)	26
3.4.4 For device have not configured to connect to DataCake	35
3.5 Node-Red (via MQTT)	35
3.5.1 Configure Node-Red	35
3.5.2 Simulate Connection	40
3.5.3 Configure NB-IoT Sensors	42
3.6 ThingsBoard.Cloud (via MQTT)	42
3.6.1 Configure ThingsBoard	42
3.6.2 Simulate with MQTT.fx	49
3.6.3 Configure NB-IoT Sensor	51
3.7 ThingsBoard.Cloud (via COAP)	53
3.7.1 Configure ThingsBoard	53
3.7.2 Node Configuration(Example: Connecting to the Thingsboard platform)	58
3.8 Tago.io (via MQTT)	61
3.8.1 Create device & Get Credentials	61
3.8.2 Simulate with MQTT.fx	64
3.8.3 tago data	66
3.9 TCP Connection	67
3.10 AWS Connection	68
4. MQTT/UDP/TCP downlink	68
4.1 MQTT (via MQTT.fx)	68
5. FAQ	72
5.1 What is the usage of Multi Sampling and One Uplink?	72
5.2 Why the uplink JSON format is not standard?	73
6. Trouble Shooting:	73
6.1 Checklist for debugging Network Connection issue. Signal Strenght:99 issue.	73
6.2 Issue: "NBIOT did not respond"	74
6.3 Issue: "Failed to readl MSI number"	75
6.4 Why sometime the AT Command is slow in reponse?	75
6.5 What is the Downlink Command by the NB device?	76
UDP:	76
MQTT:	76
6.6 How to obtain device logs?	77
6.7 How to find the AT Command Password if lost?	77
Why can't the password access AT command after upgrade(-NB)?	77
Version Confirmation	78
UART connection and firmware update methods	78

query the password via STM32CubeProgrammer	78
Special case	80

Table of Contents:

- [1. The use of this guideline](#)
- [2. Attach Network](#)
 - [2.1 General Configure to attach network](#)
 - [2.2 Speed Up Network Attach time](#)
- [3. Configure to connect to different servers](#)
 - [3.1 General UDP Connection](#)
 - [3.1.1 Simulate UDP Connection by PC tool](#)
 - [3.1.2 Configure NB-IoT Sensor](#)
 - [3.1.2.1 AT Commands](#)
 - [3.1.2.2 Uplink Example](#)
 - [3.2 General MQTT Connection](#)
 - [3.3 ThingSpeak \(via MQTT\)](#)
 - [3.3.1 Get MQTT Credentials](#)
 - [3.3.2 Simulate with MQTT.fx](#)
 - [3.3.2.1 Establish MQTT Connection](#)
 - [3.3.2.2 Publish Data to ThingSpeak Channel](#)
 - [3.3.3 Configure NB-IoT Sensor for connection](#)
 - [3.3.3.1 AT Commands:](#)
 - [3.3.3.2 Uplink Examples](#)
 - [3.3.3.3 Map fields to sensor value](#)
 - [3.4 Datacake](#)
 - [3.4.1 For device Already has template](#)
 - [3.4.1.1 Create Device](#)
 - [3.4.2 For Device already registered in DataCake before shipped](#)
 - [3.4.2.1 Scan QR Code to get the device info](#)
 - [3.4.2.2 Claim Device to User Account](#)
 - [3.4.3 Manual Add Decoder in DataCake \(don't use the template in DataCake\)](#)
 - [3.4.4 For device have not configured to connect to DataCake](#)
 - [3.5 Node-Red \(via MQTT\)](#)
 - [3.5.1 Configure Node-Red](#)
 - [3.5.2 Simulate Connection](#)
 - [3.5.3 Configure NB-IoT Sensors](#)
 - [3.6 ThingsBoard.Cloud \(via MQTT\)](#)
 - [3.6.1 Configure ThingsBoard](#)
 - [3.6.1.1 Create Device](#)
 - [3.6.1.2 Create Uplink & Downlink Converter](#)
 - [3.6.1.3 MQTT Integration Setup](#)
 - [3.6.2 Simulate with MQTT.fx](#)
 - [3.6.3 Configure NB-IoT Sensor](#)
 - [3.7 ThingsBoard.Cloud \(via COAP\)](#)
 - [3.7.1 Configure ThingsBoard](#)
 - [3.7.1.1 Create Uplink & Downlink Converter](#)
 - [3.7.1.2 COAP Integration Setup](#)
 - [3.7.1.3 Add COAP Integration](#)
 - [General Configure to Connect to IoT server for -NB & -NS NB-IoT models](#)
 - [3.7.2 Node Configuration\(Example: Connecting to the Thingsboard platform\)](#)
 - [3.7.2.1 Instruction Description](#)
 - [3.8 Tago.io \(via MQTT\)](#)
 - [3.8.1 Create device & Get Credentials](#)
 - [3.8.2 Simulate with MQTT.fx](#)
 - [3.8.3 tago data](#)
 - [3.9 TCP Connection](#)
 - [3.10 AWS Connection](#)
- [4. MQTT/UDP/TCP downlink](#)
 - [4.1 MQTT \(via MQTT.fx\)](#)
- [5. FAQ](#)
 - [5.1 What is the usage of Multi Sampling and One Uplink?](#)
 - [5.2 Why the uplink JSON format is not standard?](#)

- [6. Trouble Shooting:](#)
 - [6.1 Checklist for debugging Network Connection issue. Signal Strenght:99 issue.](#)
 - [6.2 Issue: "NBIOT did not respond"](#)
 - [6.3 Issue: "Failed to read MSI number"](#)
 - [6.4 Why sometime the AT Command is slow in reponse?](#)
 - [6.5 What is the Downlink Command by the NB device?](#)
 - [UDP:](#)
 - [MQTT:](#)
 - [6.6 How to obtain device logs?](#)
 - [6.7 How to find the AT Command Password if lost?](#)
 - [Why can't the password access AT command after upgrade\(-NB\)?](#)
 - [Version Confirmation](#)
 - [UART connection and firmware update methods](#)
 - [query the password via STM32CubeProgrammer](#)
 - [Special case](#)

1. The use of this guideline

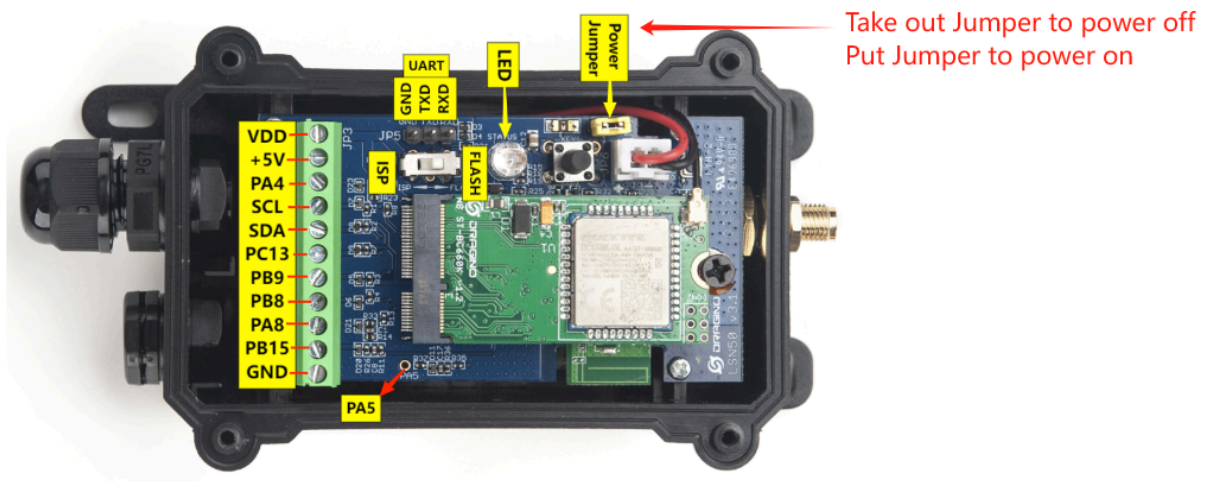
This configure instruction is for Dragino NB-IoT models with -NB or -NS suffix, for example DDS75-NB. These models use the same NB-IoT Module [BC660K-GL](#) and has the same software structure. The have the same configure instruction to different IoT servers. Use can follow the instruction here to see how to configure to connect to those servers.

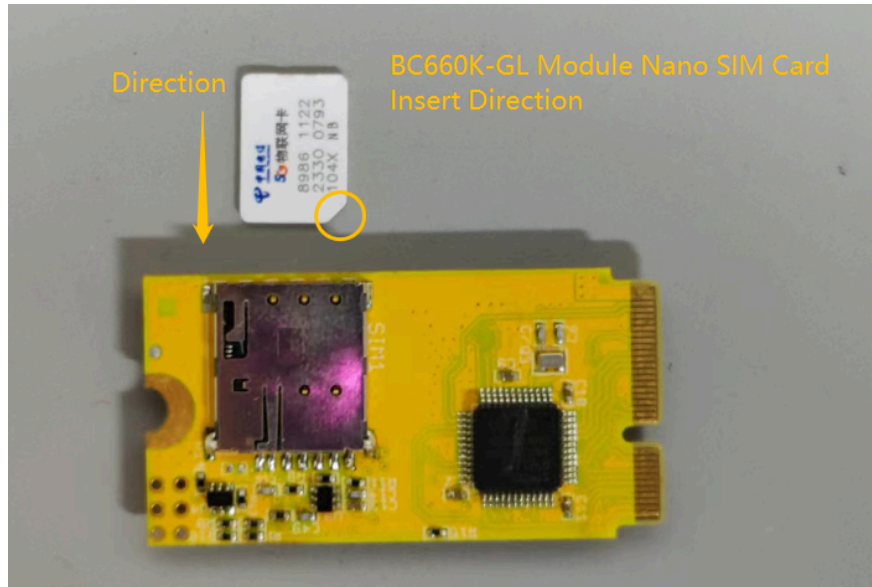
2. Attach Network

2.1 General Configure to attach network

To attache NB-IoT sensors to NB-IoT Network, You need to:

1. Get a NB-IoT SIM card from Service Provider. (Not the same as the SIM card we use in mobile phone)
2. Power Off End Node (See below for the power off/on position)
3. Insert the SIM card to Sensor. (See below for direction)
4. Power On End Node
5. [Configure APN](#) in the sensor (AT+APN=<APN>), example AT+APN=iot.1nce.net





After doing above, the NB-IoT Sensors should be able to attach to NB-IoT network .

The -NB and -NS models support **LTE Cat NB2**, with below frequency band: multiple frequency bands of **B1/B2/B3/B4/B5/B8/B12/B13/B14/B17/B18/B19/B20/B25/B28/B66/B70/B85** . Make sure you use a the NB-IoT SIM card.

SIM Provider	AT+APN=	NB-IoT Coverage
1NCE	iot.1nce.net	Coverage Reference Link Austria, Belgium, Bulgaria, Croatia, Czech Republic, Denmark, Finland, Latvia, Malta, Netherlands, Norway, Puerto Rico, Russia, Slovak , Republic of the UK, US Virgin Islands
China Mobile	No need configure	China Mainland, HongKong
China Telecom	ctnb	China Mainland

2.2 Speed Up Network Attach time

BC660K-GL supports multi bands **B1/B2/B3/B4/B5/B8/B12/B13/B14/B17/B18/B19/B20/B25/B28/B66/B70/B85**. It will search one by one and try to attach, this will take a lot of time and even cause attach fail and show **Signal Strenght:99**. User can lock the band to specify band for its operator to make this faster.

AT+QBAND? // Check what is the current used frequency band
AT+QBAND=1,4 // Set to use 1 frequency band. Band4
Europe General AT+QBAND=2,8,20 // Set to use 2 frequency bands. Band 8 and Band 20
Global General : AT+QBAND=10,8,20,28,2,4,12,13,66,85,5
Verizon AT+QBAND=1,13
AT&T AT+QBAND=3,12,4,2
Telstra AT+QBAND=1,28
Softband AT+QBAND=2,3,8

After connection is successful, user can use **AT+QENG=0** to check which band is actually in used.

By default, device will search network for 5 minutes. User can set the time to 10 minutes by **AT+CSQTIME=10** so it can search longer.

See bands used for different provider: [NB-IoT Deployment , Bands, Operator list](#)

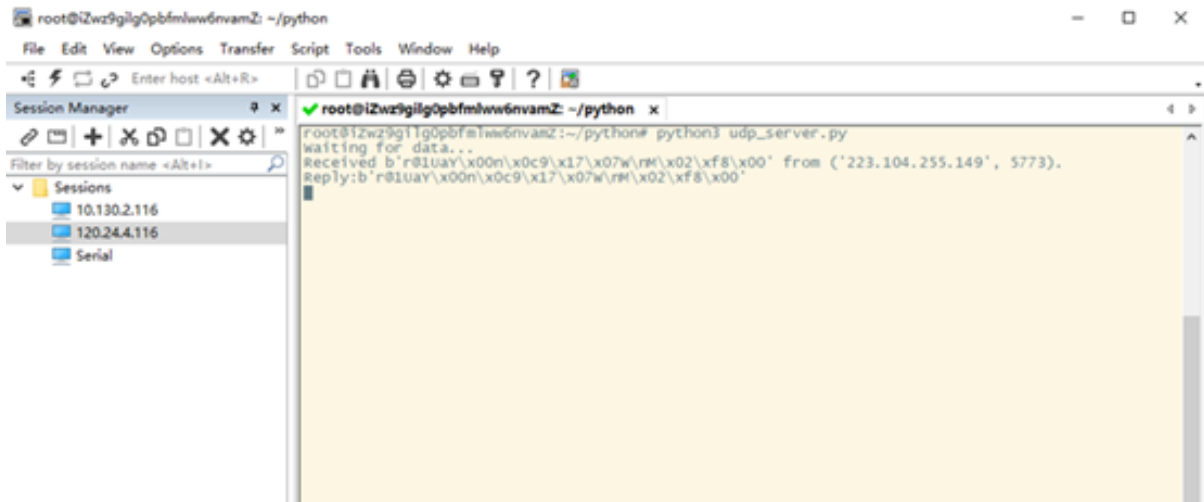
- **AT+PRO=2,0** // Set to use UDP protocol to uplink ,Payload Type select Hex payload
- **AT+SERVADDR=120.24.4.116,5601** // Set UDP server address and port

```

AT+NAME866207053462762
[816]reboot error:NRST!
DRAGINO SN50U3-NB NB-IoT Sensor Node
Image Version: v1.0.0
NB-IoT Stack : D-BC660K-001
Protocol in Used: UDP
[7669]NB-IoT has responded.
[12001]Echo mode turned off successfully.
[13347]Disable the reporting of deep sleep event URC.
[14706]Model information:BC660K-GL.
[16046]The IMEI number is:866207053462762.
[17394]The IMSI number is:460083823106206.
[19442]Set the data format for sending and receiving.
Currently set frequency band: 1,2,3,4,5,8,12,13,17,18,19,20,25,28,66,70,85
[26484]Signal Strength:24
[31518]PSM mode configured
[34151]DNS configuration is successful
[40195]No DNS resolution required
[41233]*****Upload start:0*****
[41769]remaining battery =3245 mv
[42563]DS18B20(1) temp is -0.06
[42666]adc_mV(1):53.00
[42693]No I2C device detected
[46230]Open a Socket Service successfully
[53283]Datagram is sent by RF
[54317]Send complete
[55341]*****End of upload*****
AT+PWRM2

```

3.1.2.2 Uplink Example



3.2 General MQTT Connection

The NB-IoT Sensor can send packet to server use MQTT protocol.

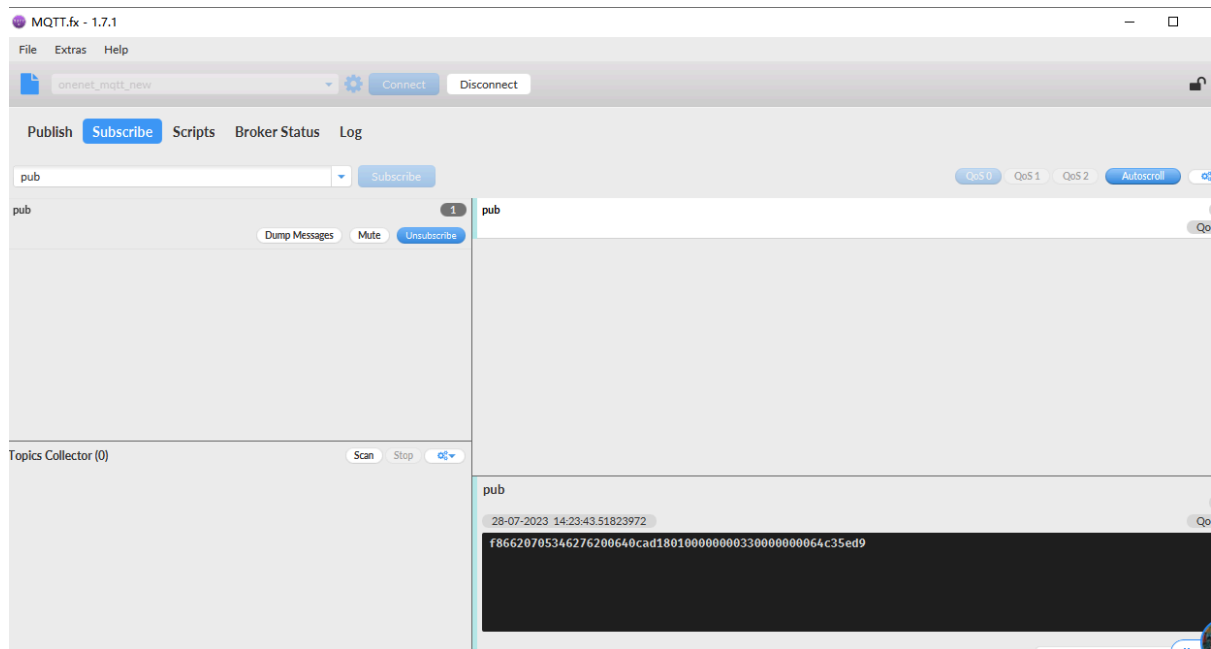
Below are the commands.

AT Commands:

- **AT+PRO=3,0** // Set to use MQTT protocol to uplink, Payload Type select Hex payload.
- **AT+SERVADDR=120.24.4.116,1883** // Set MQTT server address and port

- **AT+CLIENT=CLIENT** // Set up the CLIENT of MQTT
- **AT+UNAME=UNAME** // Set the username of MQTT
- **AT+PWD=PWD** // Set the password of MQTT
- **AT+PUBTOPIC=NSE01_PUB** // Set the sending topic of MQTT
- **AT+SUBTOPIC=NSE01_SUB** // Set the subscription topic of MQTT

```
DRAGINO SN50V3-NB NB-IoT Sensor Node
Image Version: v1.0.0
NB-IoT Stack : D-BC660K-001
Protocol in Used: MQTT
[7370]NB-IoT has responded.
[11701]Echo mode turned off successfully.
[13047]Disable the reporting of deep sleep event URC.
[14406]Model information:BC660K-GL.
[15746]The IMEI number is:866207053462762.
[17094]The IMSI number is:460083823106206.
[19142]Set the data format for sending and receiving.
Currently set frequency band: 1,2,3,4,5,8,12,13,17,18,19,20,25,28,66,70,85
[26184]Signal Strength:25
[31218]PSM mode configured
[33851]DNS configuration is successful
[39895]No DNS resolution required
[40933]*****Upload start:0*****
[41469]remaining battery =3245 mv
[42263]DS18B20(1) temp is -0.06
[42366]adc_mV(1):56.00
[43471]No I2C device detected
[48909]Opened the MQTT client network successfully
[52467]Successfully connected to the server
[57522]Subscribe to topic successfully
[61068]Close the port successfully
AT+PWRM2
[62269]Send complete
[63293]*****End of upload*****
```



Notice: MQTT protocol has a much higher power consumption compare with UDP/ CoAP protocol. Please check the power analyze document and adjust the uplink period to a suitable interval.

3.3 [ThingSpeak](#) (via MQTT)

3.3.1 Get MQTT Credentials

[ThingSpeak](#) connection uses MQTT Connection. So we need to get MQTT Credentials first. You need to point MQTT Devices to TH

thingspeak.com/devices/mqtt

ThingSpeak™ Channels ▾ Apps ▾ Devices ▾ Support ▾ Commercial Use How to Buy

MQTT Devices

[Add a new device](#)

Device Details: BC660-Test <i>No description</i>	Authorized Channels and Permissions: dragino-test (396640) ✓publish ✓subscribe	MQTT Client ID: JCoZK[REDACTED]4tBQQHMSw	Edit Delete
---	--	--	--

Point to Channel

ThingSpeak™

Channels ▾

Apps ▾

Devices ▾

Support ▾

MQTT Devices / Edit BC660-Test

Edit BC660-Test

Device Information

Name

BC660-Test

Description

Enter optional information about this device for later reference.

MQTT Credentials

Use these MQTT credentials to publish and subscribe to ThingSpeak channels. [Learn More](#)

Client ID

JCoZKTc4tBQQHMSw

Username

JCoZKTc4tBQQHMSw

Password

.....

Authorize channels to access ⓘ

-- Select a Channel -- ▾

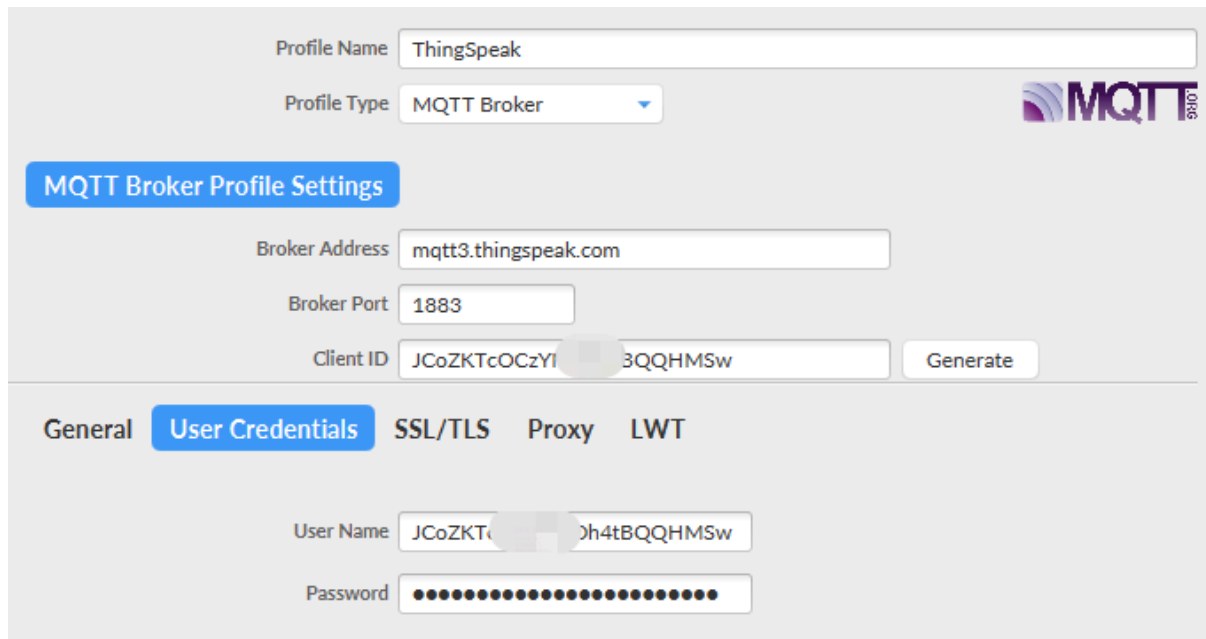
...

Add Channel

3.3.2 Simulate with MQTT.fx

3.3.2.1 Establish MQTT Connection

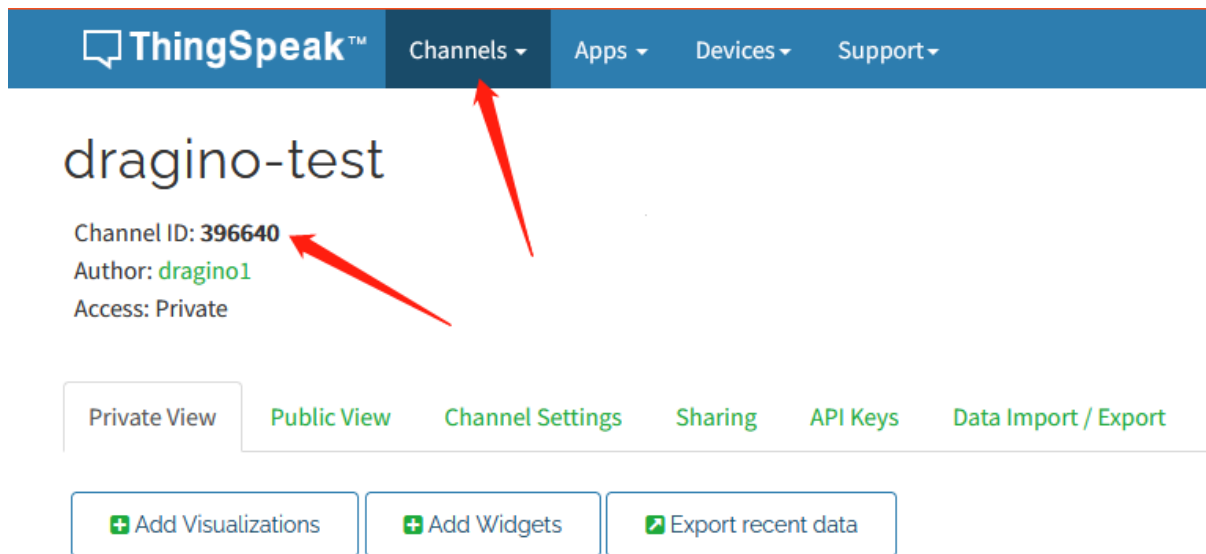
After we got MQTT Credentials, we can first simulate with PC tool MQTT.fx tool to see if the Credentials and settings are fine.



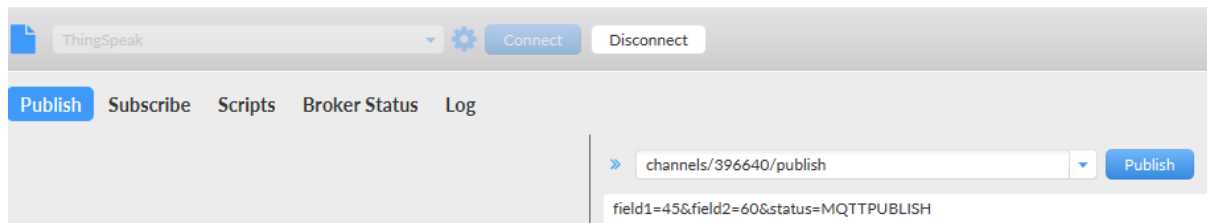
The image shows a web interface for configuring an MQTT Broker profile. At the top, there's a 'Profile Name' field with 'ThingSpeak' and a 'Profile Type' dropdown set to 'MQTT Broker'. An MQTT logo is in the top right. Below is a blue button labeled 'MQTT Broker Profile Settings'. The main configuration area includes 'Broker Address' (mqtt3.thingspeak.com), 'Broker Port' (1883), and 'Client ID' (JCoZKTcOCzYI3QQHMSw) with a 'Generate' button. At the bottom, there are tabs for 'General', 'User Credentials', 'SSL/TLS', 'Proxy', and 'LWT'. The 'User Credentials' tab is active, showing 'User Name' (JCoZKTcOCzYI3QQHMSw) and a masked 'Password' field.

- **Broker Address:** mqtt3.thingspeak.com
- **Broker Port:** 1883
- **Client ID:** <Your ThingSpeak MQTT ClientID>
- **User Name:** <Your ThingSpeak MQTT User Name>
- **Password:** <Your ThingSpeak MQTT Password>

3.3.2.2 Publish Data to ThingSpeak Channel



The image shows the ThingSpeak web interface. The top navigation bar has 'ThingSpeak™' and links for 'Channels', 'Apps', 'Devices', and 'Support'. The main content area is for a channel named 'dragino-test'. It displays 'Channel ID: 396640', 'Author: dragino1', and 'Access: Private'. Below this is a row of tabs: 'Private View' (selected), 'Public View', 'Channel Settings', 'Sharing', 'API Keys', and 'Data Import / Export'. At the bottom, there are three buttons: '+ Add Visualizations', '+ Add Widgets', and 'Export recent data'. Two red arrows point to the 'Channels' dropdown and the 'Channel ID: 396640' text.



In MQTT.fx, we can publish below info:

- **Topic:** channels/YOUR_CHANNEL_ID/publish
- **Payload:** field1=63&field2=67&status=MQTTPUBLISH

Where 63 and 67 are the value to be published to field1 & field2.

Result:

dragino-test

Channel ID: 396640

Author: dragino1

Access: Private

Private View

Public View

Channel Settings

Sharing

API Keys

Data Import / Export

+ Add Visualizations

+ Add Widgets

Export recent data

MATLAB Analysis

MATLAB Visualization

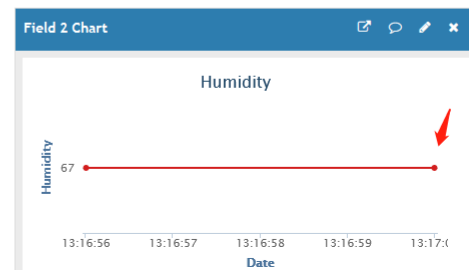
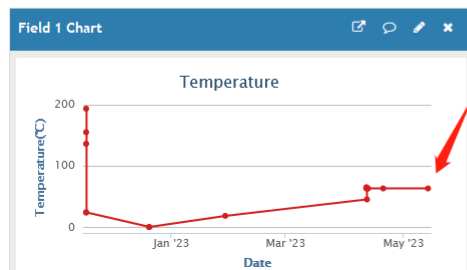
Channel 1 of 4 < >

Channel Stats

Created: 5 years ago

Last entry: less than a minute ago

Entries: 71580



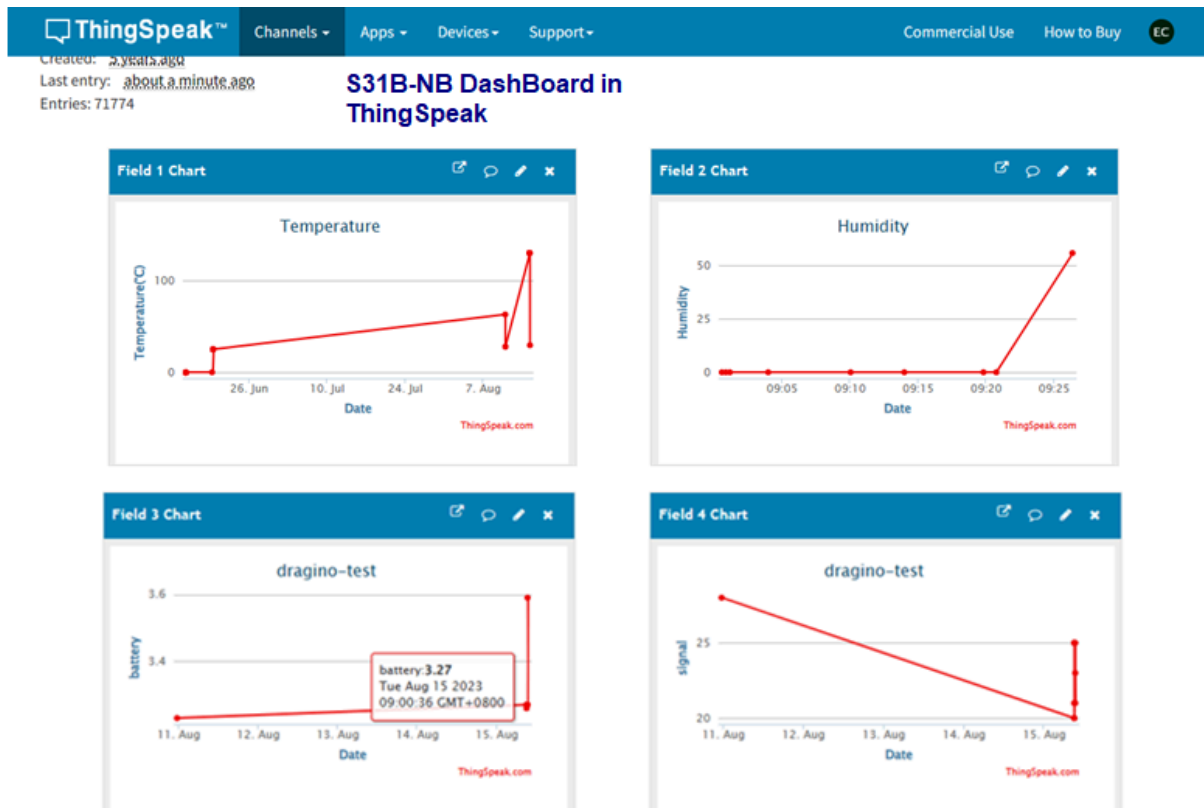
3.3.3 Configure NB-IoT Sensor for connection

3.3.3.1 AT Commands:

In the NB-IoT, we can run below commands so to publish the channels like MQTT.fx

- **AT+PRO=3,1** // Set to use ThingSpeak Server and Related Payload
- **AT+CLIENT=<Your ThingSpeak MQTT ClientID>**
- **AT+UNAME=<Your ThingSpeak MQTT User Name>**
- **AT+PWD=<Your ThingSpeak MQTT Password>**
- **AT+PUBTOPIC=<YOUR_CHANNEL_ID>**
- **AT+SUBTOPIC=<YOUR_CHANNEL_ID>**

3.3.3.2 Uplink Examples



For SE01-NB

For DDS20-NB

For DDS45-NB

For DDS75-NB

For NMDS120-NB

For SPH01-NB

For NLM01-NB

For NMDS200-NB

For CPN01-NB

For DS03A-NB

For SN50V3-NB

3.3.3.3 Map fields to sensor value

When NB-IoT sensor upload to ThingSpeak. The payload already specify which fields related to which sensor value. Use need to create fields in

dragino-test

Channel ID: 396640

Author: dragino1

Access: Private

[Private View](#)

[Public View](#)

[Channel Settings](#)

[Sharing](#)

[API Keys](#)

[Data Import / Export](#)

Channel Settings

Percentage complete 30%

Channel ID 396640

Name dragino-test

Description

Field 1

Temperature



Field 2

Humidity



Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

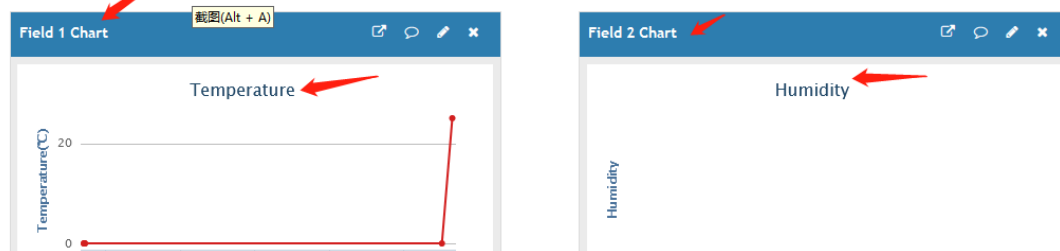
- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.

Channel Stats

Created: 5 years ago

Last entry: about a month ago

Entries: 71763



Below is the NB-IoT Product Table show the mapping.

	Field1	Field2	Field3	Field4
S31x-NB	Temperature	Humidity	Battery	RSSI
SE01-NB	Temperature	Humidity	conduct	dielectric
DDS20-NB	distance	Battery	RSSI	
DDS45-NB	distance	Battery	RSSI	
DDS75-NB	distance	Battery	RSSI	
NMDS120-NB	distance	Battery	RSSI	
SPH01-NB	ph	Temperature	Battery	RSSI
NLM01-NB	Humidity	Temperature	Battery	RSSI
NMDS200-NB	distance1	distance2	Battery	RSSI
CPN01-NB	alarm	count	door open duration	calculation
DS03A-NB	level	alarm	pb14door open num	pb14door open num
SN50V3-NB mod1	mod	Battery	RSSI	DS18B20
SN50V3-NB mod2	mod	Battery	RSSI	DS18B20
SN50V3-NB mod3	mod	Battery	RSSI	adc0
SN50V3-NB mod4	mod	Battery	RSSI	DS18B20
SN50V3-NB mod5	mod	Battery	RSSI	DS18B20
SN50V3-NB mod6	mod	Battery	RSSI	count

3.4 Datalog

Dragino NB-IoT sensors has its template in [Datalog](#) Platform. There are two version for NB Sensor,

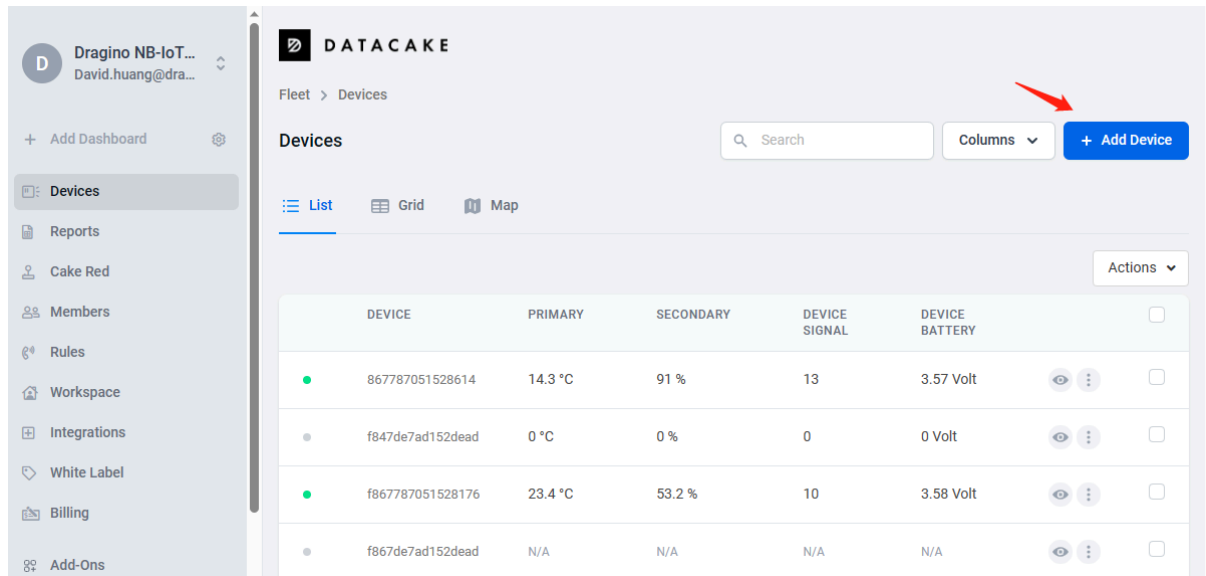
As example for S31B-NB. there are two versions: **S31B-NB-1D** and **S31B-NB-GE**.

- **S31B-NB-1D**: This version have pre-configure DataCake connection. User just need to Power on this device, it will auto connect send data to DataCake Server.
- **S31B-NB-GE**: This version doesn't have pre-configure Datacake connection. User need to enter the AT Commands to connect to Datacake. See below for instruction.

3.4.1 For device Already has template

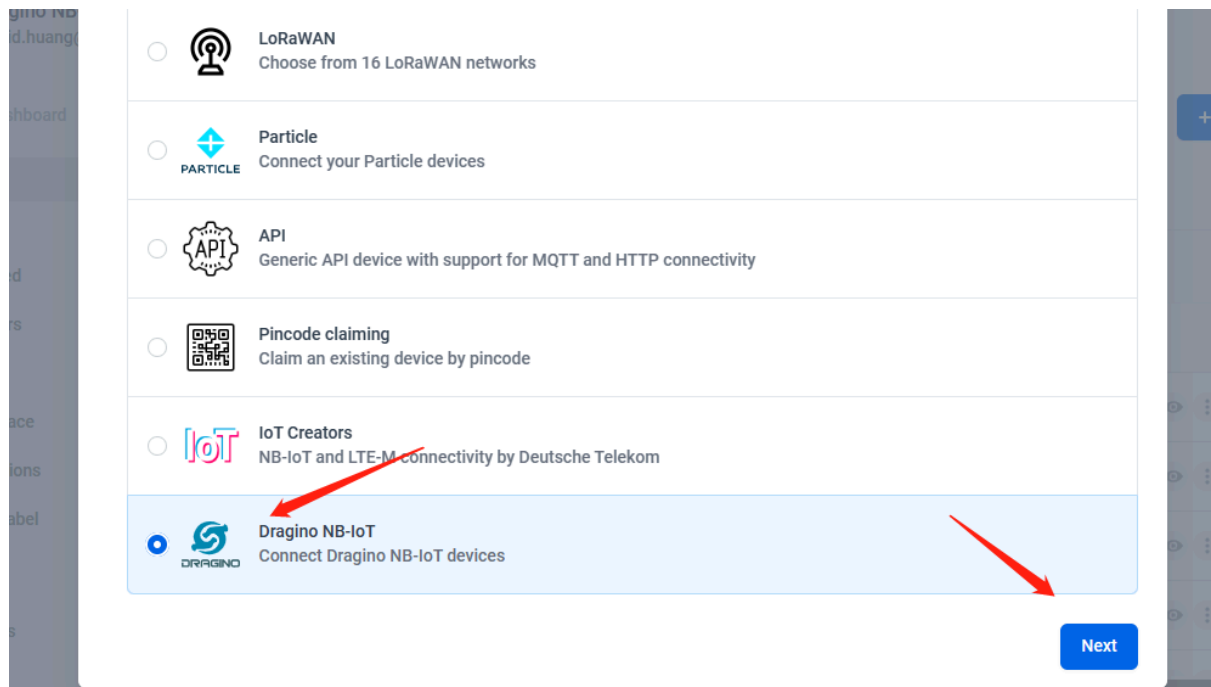
3.4.1.1 Create Device

Add Device in DataCake.



The screenshot shows the DataCake web interface. On the left is a sidebar with navigation options: Devices (selected), Reports, Cake Red, Members, Rules, Workspace, Integrations, White Label, Billing, and Add-Ons. The main content area is titled 'Fleet > Devices'. It features a search bar, a 'Columns' dropdown, and a '+ Add Device' button highlighted with a red arrow. Below these are tabs for 'List' (selected), 'Grid', and 'Map'. An 'Actions' dropdown is in the top right of the table. The table lists four devices with columns for status, device ID, primary/secondary values, device signal, device battery, and actions.

	DEVICE	PRIMARY	SECONDARY	DEVICE SIGNAL	DEVICE BATTERY	
●	867787051528614	14.3 °C	91 %	13	3.57 Volt	👁️ ⋮ □
●	f847de7ad152dead	0 °C	0 %	0	0 Volt	👁️ ⋮ □
●	f867787051528176	23.4 °C	53.2 %	10	3.58 Volt	👁️ ⋮ □
●	f867de7ad152dead	N/A	N/A	N/A	N/A	👁️ ⋮ □



The screenshot shows a configuration screen with the following options:

- ☐ **LoRaWAN**
Choose from 16 LoRaWAN networks
- ☐ **Particle**
Connect your Particle devices
- ☐ **API**
Generic API device with support for MQTT and HTTP connectivity
- ☐ **Pincode claiming**
Claim an existing device by pincode
- ☐ **IoT Creators**
NB-IoT and LTE-M connectivity by Deutsche Telekom
- ☒ **Dragino NB-IoT**
Connect Dragino NB-IoT devices

A blue **Next** button is located at the bottom right of the configuration area.

Choose the correct model from template.

Add Dragino NB-IoT Device

STEP 1
Product

STEP 2
Devices

STEP 3
Plan

Datacake product

You can add devices to an existing product on Datacake, or create a new empty product. Products allow you to share the same configuration (fields, dashboard and more) between devices.

Existing product
Add devices to an existing product

New product
Create a new empty product

New Product from template
Create new product from a template

☒ **Dragino N95S31B**
Dragino NB-IoT Temperature and Humidity Sensor UDP

Back

Next

Fill Device ID. The device ID needs to be filled in with IMEI, and a prefix of 'f' needs to be added.

Add Dragino NB-IoT Device

STEP 1
Product

STEP 2
Devices

STEP 3
Plan

Add Devices

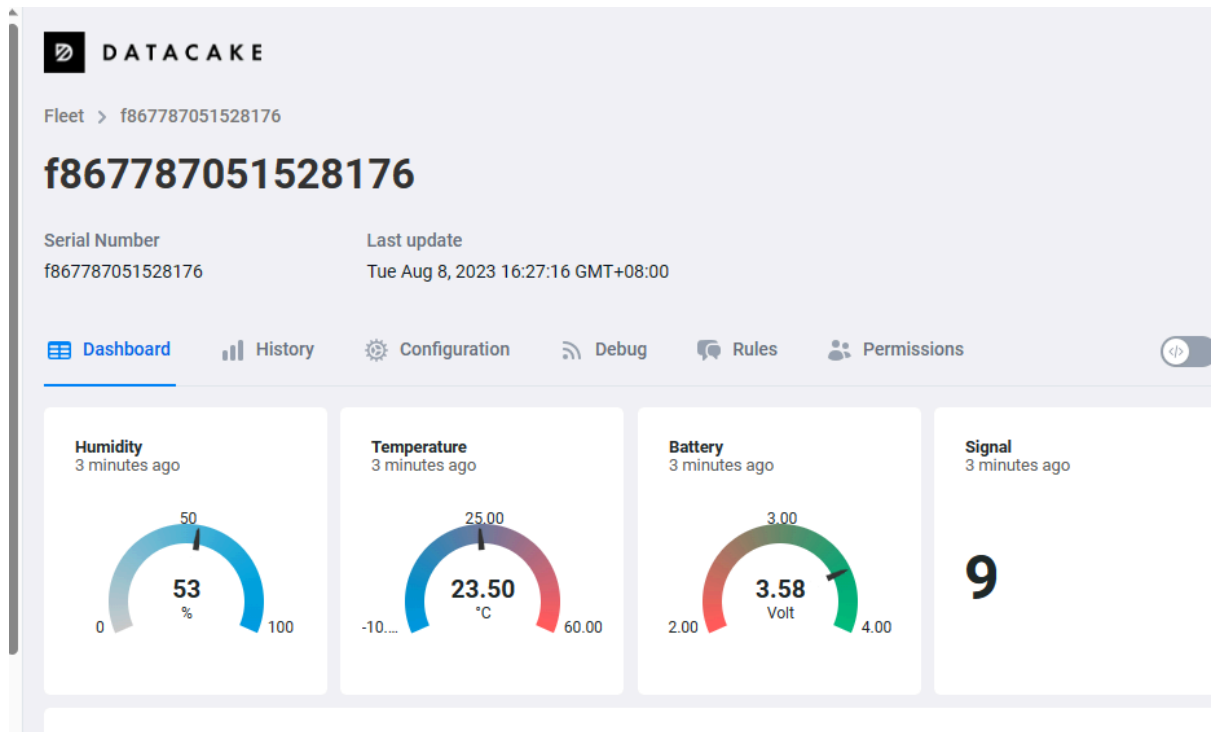
You can add one or more Dragino NB-IoT devices at a time. Devices are identified by their device IDs.

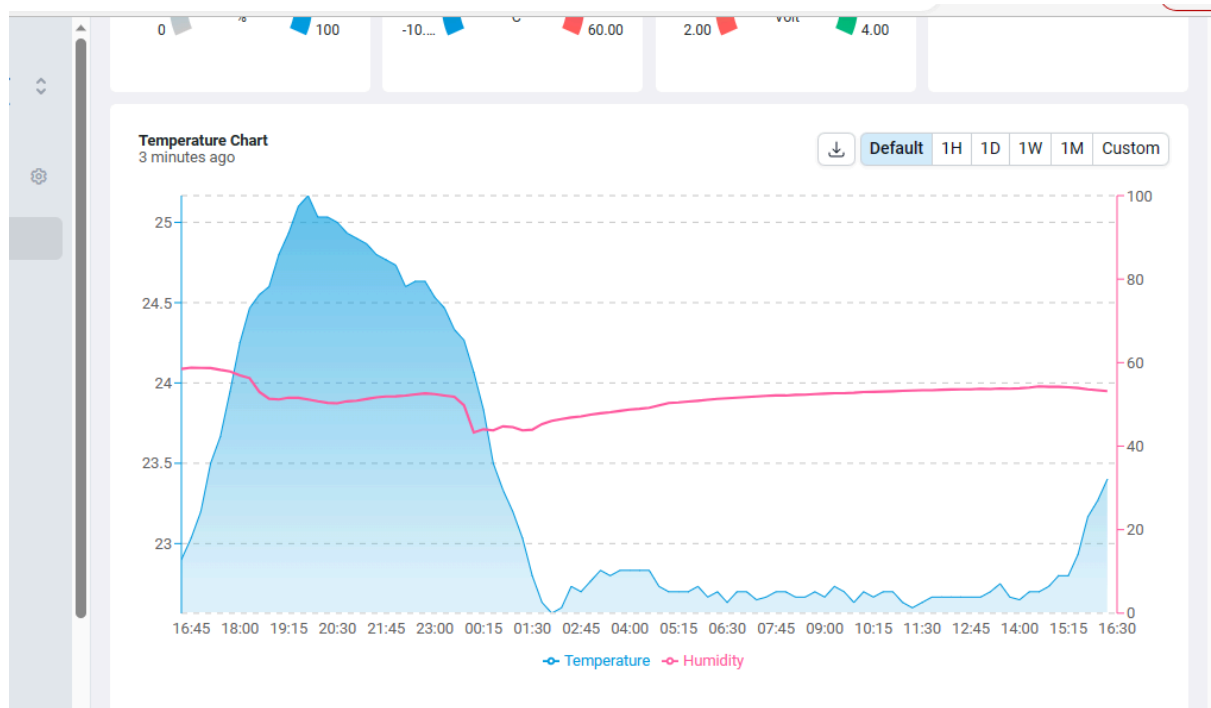
DEVICE ID	NAME	LOCATION	TAGS
f867787051528176	imei Name Please enter a device name	Location	Add tag

+ Add another device

Back

Next



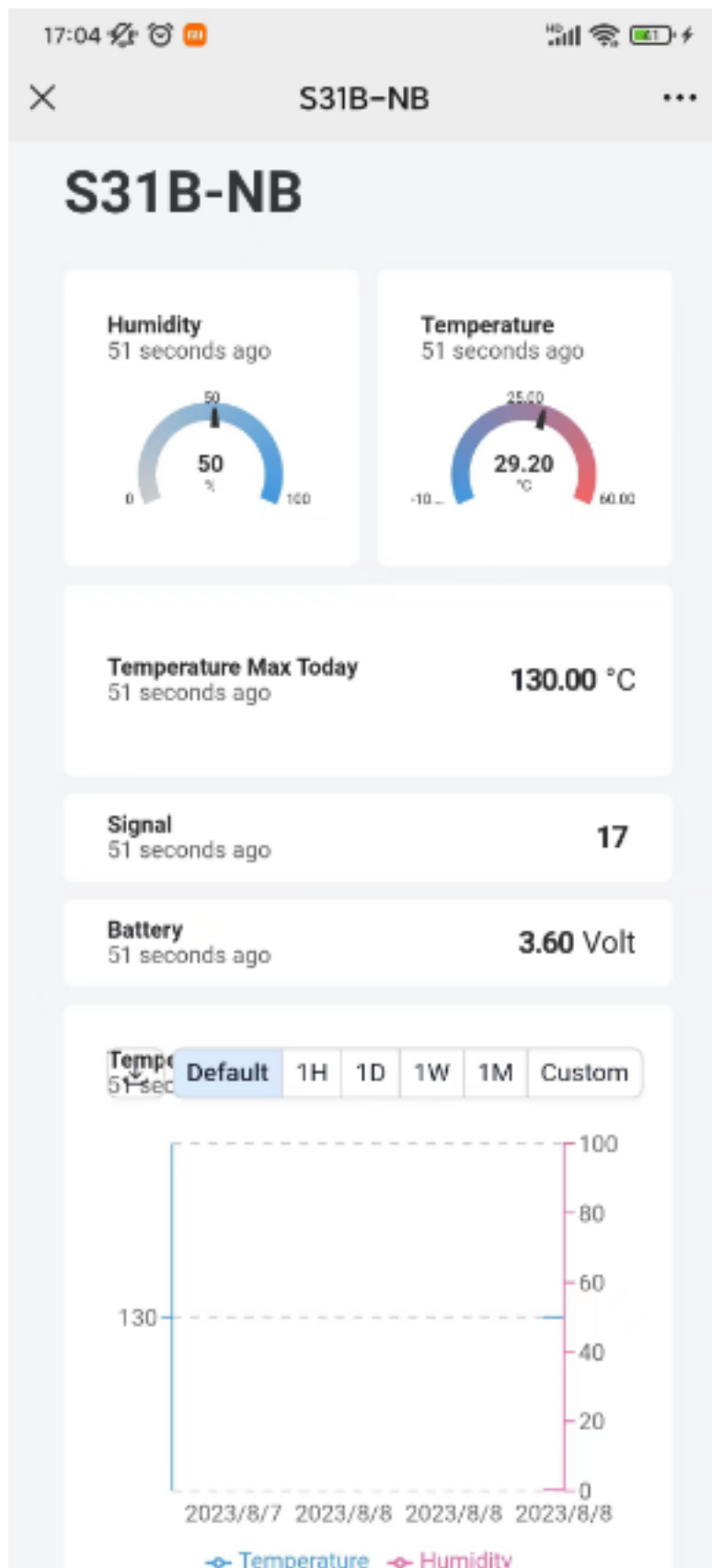


3.4.2 For Device already registered in DataCake before shipped

3.4.2.1 Scan QR Code to get the device info

Users can use their phones or computers to scan QR codes to obtain device data information.



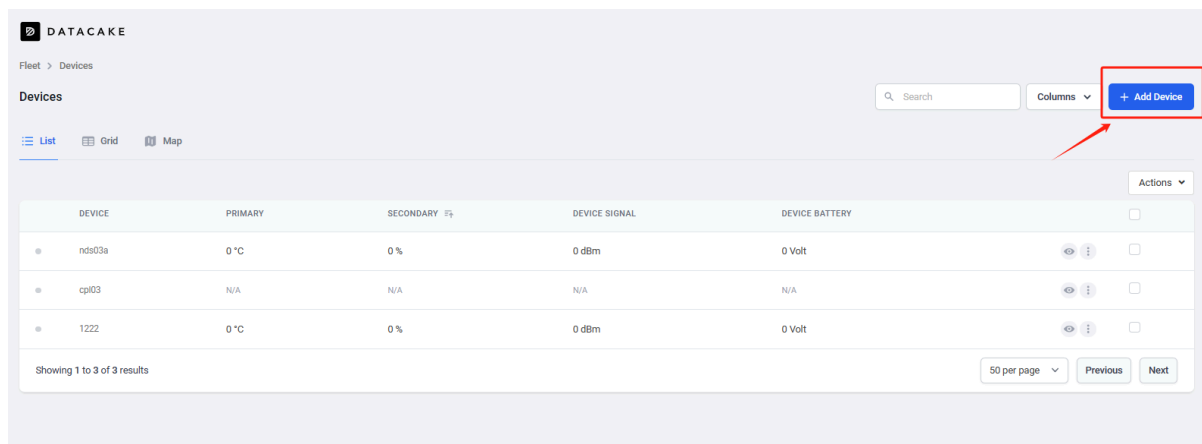


3.4.2.2 Claim Device to User Account

By Default, the device is registered in Dragino's DataCake Account. User can Claim it to his account.

3.4.3 Manual Add Decoder in DataCake (don't use the template in DataCake)

Step1: Add a device



Step2: Choose your device type,please select dragino NB-IOT device

Add Device

First, choose the connectivity type of your device.

☐

**LoRaWAN**
Choose from 16 LoRaWAN networks

☐

**Particle**
Connect your Particle devices

☐

**API**
Generic API device with support for MQTT and HTTP connectivity

☐

**Pincode Claiming**
Claim an existing device by pincode

☐

**IoT Creators**
NB-IoT and LTE-M connectivity by Deutsche Telekom

☒

**Dragino NB-IoT**
Connect Dragino NB-IoT devices

☐

**1NCE**
Connect 1NCE devices

Next

Step3: Choose to create a new device

Add Dragino NB-IoT Device

STEP 1
Product

STEP 2
Devices

STEP 3
Plan

Datacake product
You can add devices to an existing product on Datacake, or create a new empty product. Products allow you to share the same configuration (fields, dashboard and more) between devices.

Existing product
Add devices to an existing product

New product
Create a new empty product

New Product from template
Create new product from a template

New Product
If your device is not available as a template, you can start with an empty device. You will have to create the device definition (fields, dashboard) and provide the payload decoder in the device's configuration.

Product name
xxxx

BackNext

Step4: Fill in the device ID of your NB device

Add Dragino NB-IoT Device

STEP 1
Product

STEP 2
Devices

STEP 3
Plan

Add Devices

You can add one or more Dragino NB-IoT devices at a time. Devices are identified by their device IDs.

DEVICE ID	NAME	LOCATION	TAGS
 Device ID is a required field	 12	Location	Add tag Add

+ Add another device

Back

Next

Step5: Please select your device plan according to your needs and complete the creation of the device

Add Dragino NB-IoT Device

STEP 1
Product

STEP 2
Devices

STEP 3
Plan

Free
€0.00 / month
7 days data retention
500 datapoints / day
max 5 per workspace
Cancel any time

Light
€1.00 / month
1 month data retention
1,000 datapoints / day
Cancel any time

Standard
€3.00 / month
3 months data retention
2,500 datapoints / day
Cancel any time

Plus
€5.00 / month
12 months data retention
7,500 datapoints / day
Cancel any time

Looking for a way to consolidate your billing or a more affordable package pricing?
[Explore our package and enterprise pricing options](#)

Have a code?

Apply

Back

Add 1 device

Step6: Please add the decoder at the payload decoder of the device configuration.

Decoder location: [dragino-end-node-decoder/Datacake-Dragino_NB at main · dragino/dragino-end-node-decoder \(github.com\)](#)

Due to version update, please use the following decoder for the new version firmware:
[dragino-end-node-decoder/Datacake-Dragino_NB_New_Version at main · dragino/dragino-end-node-decoder \(github.com\)](#)

The screenshot displays the Datacake IoT management interface. On the left is a sidebar with a user profile 'cn' and a list of dashboards: 'LHT52Test', '1', and 'Igt92'. Below these is a 'Devices' section with a sub-menu containing 'Reports', 'Members', 'Rules', 'Workspace', 'Integrations', 'White Label', 'Billing', and 'Add-Ons'. The 'Devices' menu item is highlighted with a red box. The main content area is titled 'DATA CAKE' and shows a breadcrumb 'Fleet > 1222' with '1222' highlighted in a red box. Below the breadcrumb, the device ID '1222' is prominently displayed, followed by its 'Serial Number' (121123123) and 'Last update' status (Never). A horizontal navigation bar contains 'Dashboard', 'History', 'Configuration' (highlighted with a red box), 'Debug', 'Rules', and 'Permissions'. The 'Configuration' section is titled 'General Configuration' and includes fields for 'Device name' (containing '1222'), 'Icon' (a dropdown menu showing 'No icon selected'), and 'Location description'. Below these is a 'Tags' section with an 'Add tag' input field and an 'Add' button. At the bottom, there is a 'Metadata' section. The interface is clean and modern, using a light gray color scheme.

DATA CAKE

Fleet > 1222

1222

Serial Number 121123123 Last update Never

Dashboard History **Configuration** Debug Rules Permissions

General Configuration

Device name

1222

Icon

No icon selected

You can override the default product icon for this device

Location description

Tags

Add tag Add

You can use tags to group and filter your devices on dashboards and in rules

Metadata

Payload Decoder

Product-wide setting

When your device sends a payload, it will be passed to the payload decoder, where it is transformed into Datacake measurements.

```

1  function Decoder(payload, payloadSize) {
2      // payload is the decoded payload
3      // payloadSize is bytes-encoded
4
5      function bytesToHex(bytes) {
6          var hex = '';
7          for (var i = 0; i < bytes.length; i++) {
8              var current = bytes[i] < 0 ? bytes[i] + 256 : bytes[i];
9              var hexByte = current.toString(16);
10             hex += hexByte.length === 1 ? '0' + hexByte : hexByte;
11         }
12         return hex;
13     }
14
15     function parsePayload(hex) {
16         var offset = 0;
17
18         function toInt(hexstring, lengthInbits) {
19             var value = parseInt(hexstring, 16);
20
21             if (lengthInbits && (value & (1 << (lengthInbits - 1)))) { // Check the MSB
22                 value = (~value << lengthInbits); // two's complement conversion
23             }
24
25             return value;
26         }
27
28         var deviceID = hex.substr(offset, 16); offset += 16;
29         var versionStr = hex.substr(offset, 8); offset += 8;
30         var version = toInt(versionStr) / 100;
31         var battery = toInt(hex.substr(offset, 4)) / 1000; offset += 4;
32         var signal = toInt(hex.substr(offset, 2)); offset += 2;
33         var mod = toInt(hex.substr(offset, 2)); offset += 2;
34         var interrupt = toInt(hex.substr(offset, 2)); offset += 2;
35         var Soil Moisture = toInt(hex.substr(offset, 4)) / 100; offset += 4;

```

Paste your decoder here

Payload

Paste a payload here

Try decoder

Output

console.log Output

Recognized measurements

Save

Step7: Add the output of the decoder as a field

Fields
Fields describe the data the device will store.

+ Add Field

Live data

NAME	IDENTIFIER	TYPE	ROLE	CURRENT VALUE	LAST UPDATE	
Battery	BATTERY	Float	Device Battery	0 Volt	23 days ago	
Signal	SIGNAL	Integer	Device Signal	0 dBm	23 days ago	
Temperature	SHTTEMP	Float	Primary	0 °C	23 days ago	
Humidity	SHTHUM	Float	Secondary	0 %	23 days ago	
Mod	MOD	Integer	N/A	0	23 days ago	
Adc	ADC	Integer	N/A	0	23 days ago	
Interrupt	INTERRUPT	Integer	N/A	0	23 days ago	
Tempsds18b20	TEMPSDS18B20	Integer	N/A	0	23 days ago	

Configuration Fields
Configuration fields hold a static value and can have a product-wide default value, that can be overwritten on a device level. They can be accessed in decoders.

+ Add Configuration Field

!

No fields have been created, yet
Create configuration fields to define configuration variables.

Step8: Customize the dashboard and use fields as parameters of the dashboard

Fleet > 1222

1222

Serial Number
121123123

Last update
Never

Dashboard

History

Configuration

Debug

Rules

Permissions

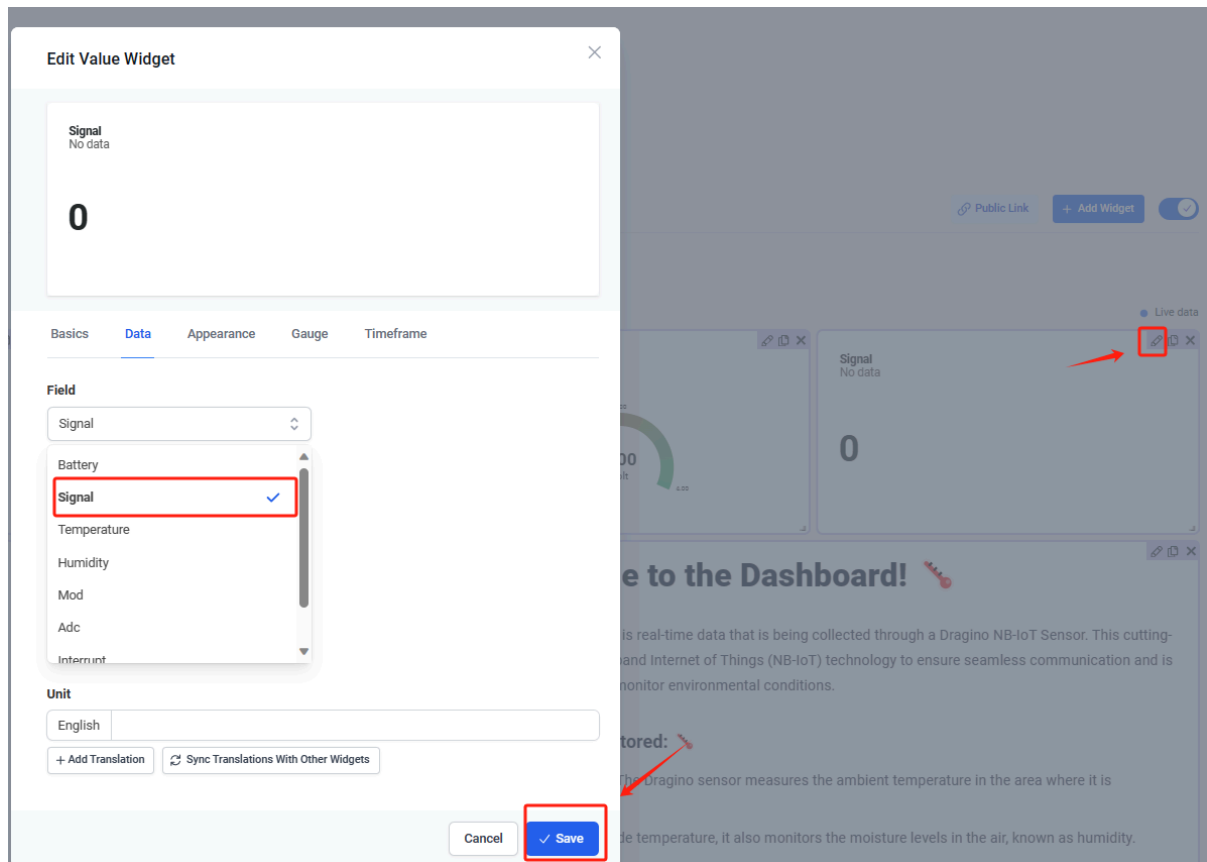
Home - General Configure/Commands to Connect to IoT server for -NB & -NS NB-IoT models

The screenshot displays the Datacake dashboard configuration interface. On the left, a sidebar shows the 'Dashboard' tab selected, with a '1222' value and 'Serial Number 121123123' and 'Last update Never' displayed. The main area features a central widget selection menu with 14 options:

- Boolean**: Displays a boolean state
- Chart**: Displays a chart
- Headline**: Displays a headline
- Histogram**: Displays a histogram
- Map**: Displays a map
- Text**: Displays a text widget
- Value**: Displays a measurement
- Switch**: Displays a switch widget
- Slider**: Displays a slider
- Downlink**: Button that sends a message to a device
- Image**: Displays a static image
- Image Map**: Display floor plans or interactive diagrams
- Iframe**: Displays an iframe
- Device Fields**: Displays a table with device fields

On the right, a preview of the dashboard is shown. It includes a 'Public Link' button, an 'Add Widget' button (highlighted with a red box and an arrow), and a 'Live Data' toggle. Below these, a 'Signal' widget is displayed with the value '0' and the text 'add a widget' in red. The dashboard preview also shows a 'the Dashboard!' heading and a list of data points:

- Temperature (°C/°F): The Dragino sensor measures the ambient temperature in the area where it is deployed.
- Humidity (%): Alongside temperature, it also monitors the moisture levels in the air, known as humidity.



3.4.4 For device have not configured to connect to DataCake

Use AT command for connecting to DataCake

AT+PRO=2,0

AT+SERVADDR=67.207.76.90,4445

3.5 Node-Red (via MQTT)

3.5.1 Configure [Node-Red](#)

Take S31-NB UDP protocol as an example.

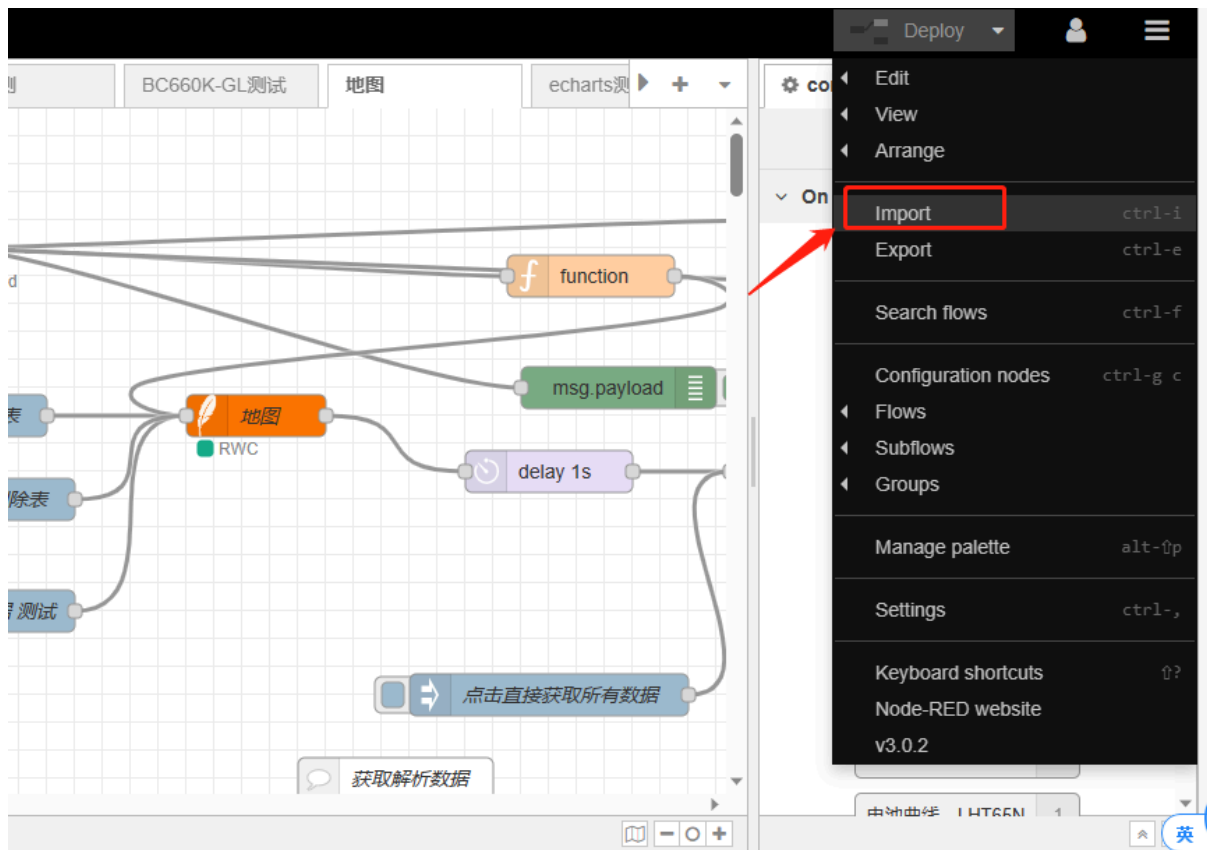
Dragino provides input flow examples for the sensors.

User can download the required JSON file through Dragino Node-RED input flow template.

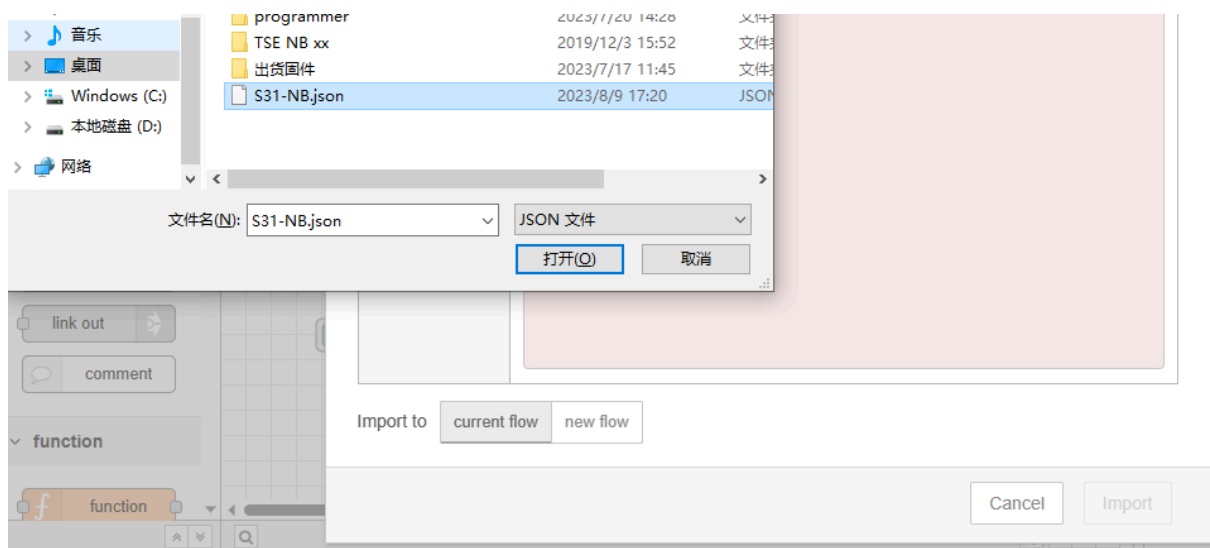
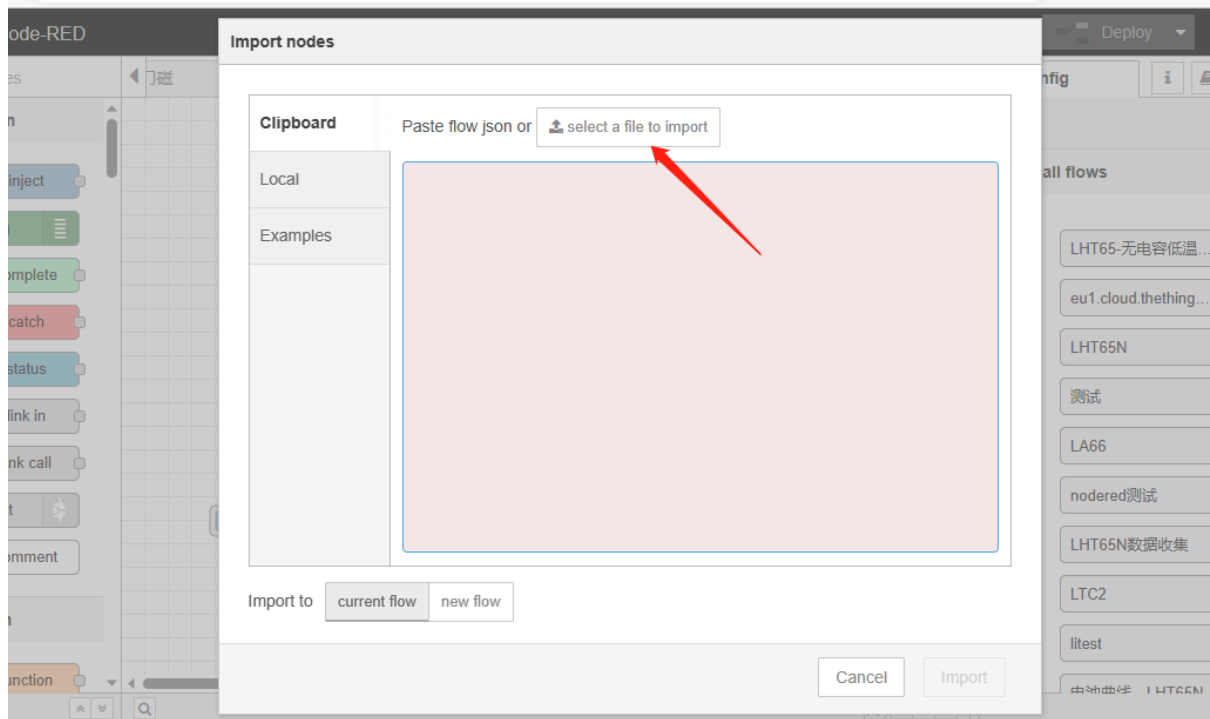
Download sample JSON file link: https://www.dropbox.com/sh/mduw85jcuwsua22/AAAvwPhg9z6dLjJhmZjqBf_ma?dl=0

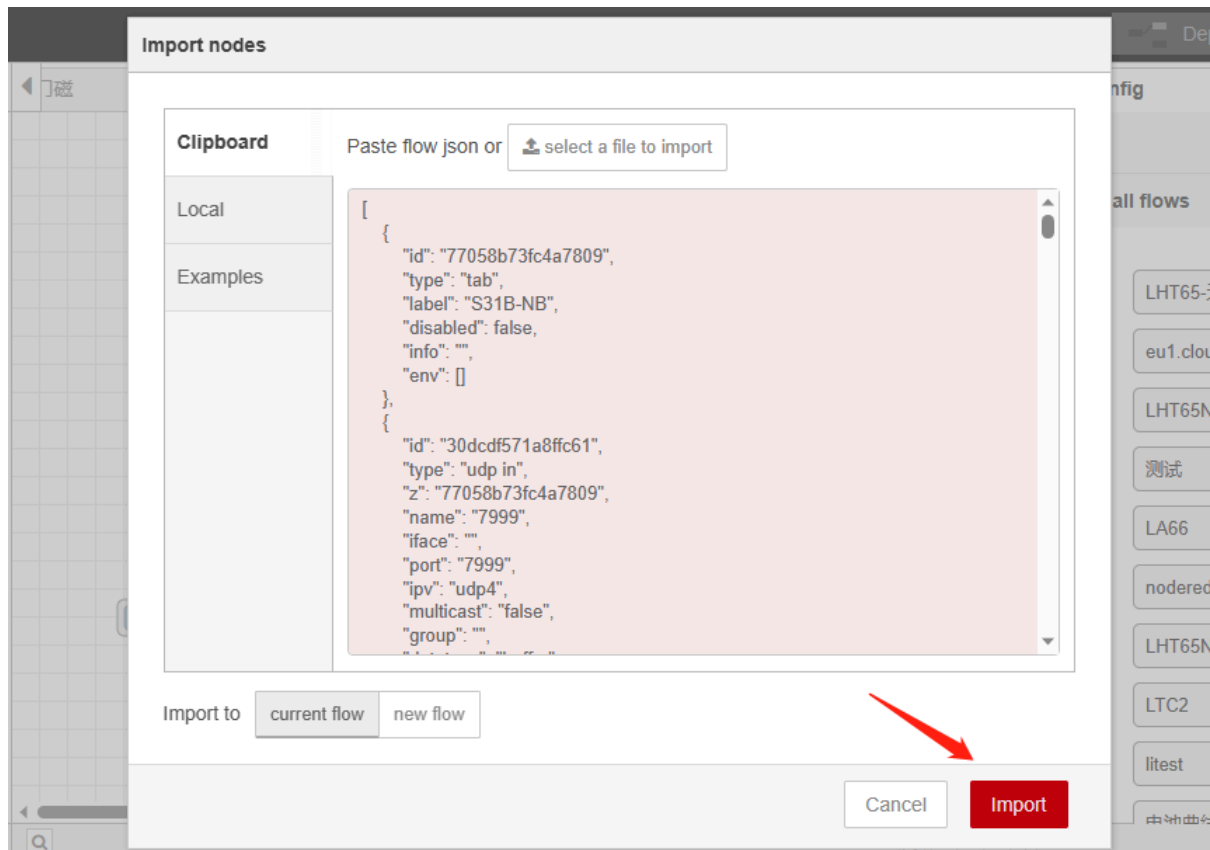
We can directly import the template.

The templates for S31-NB and NB95S31B are the same.

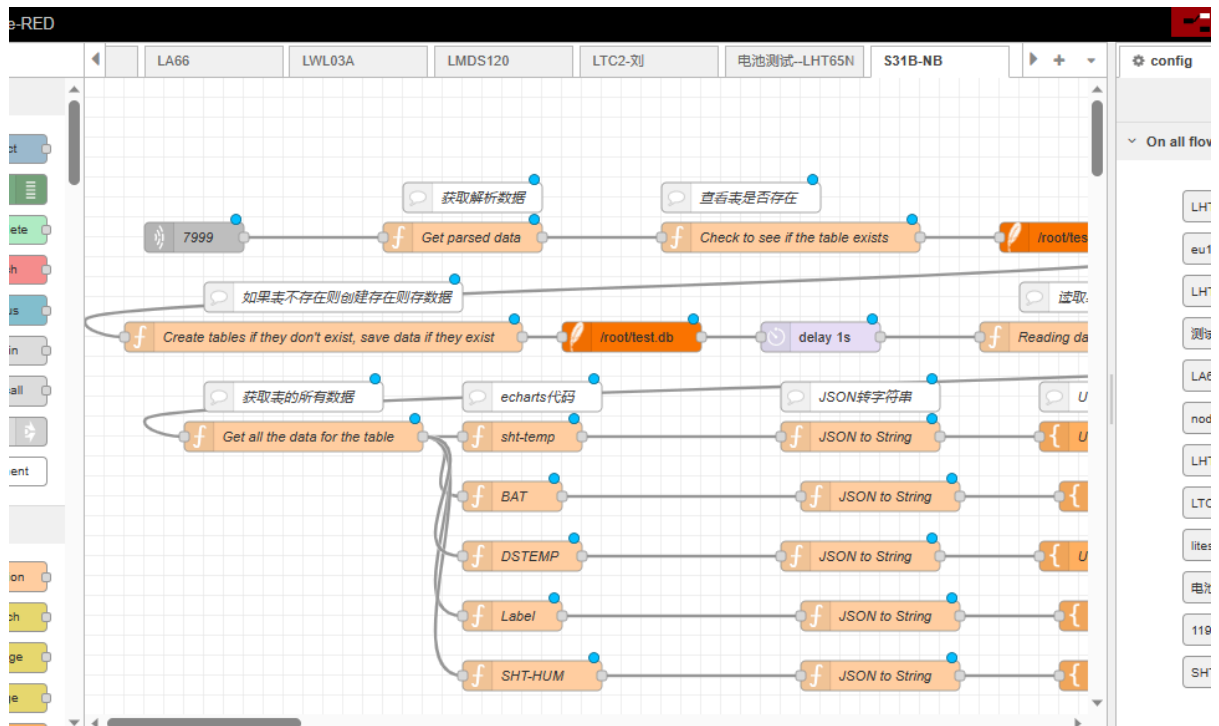


Please select the NB95S31B template.

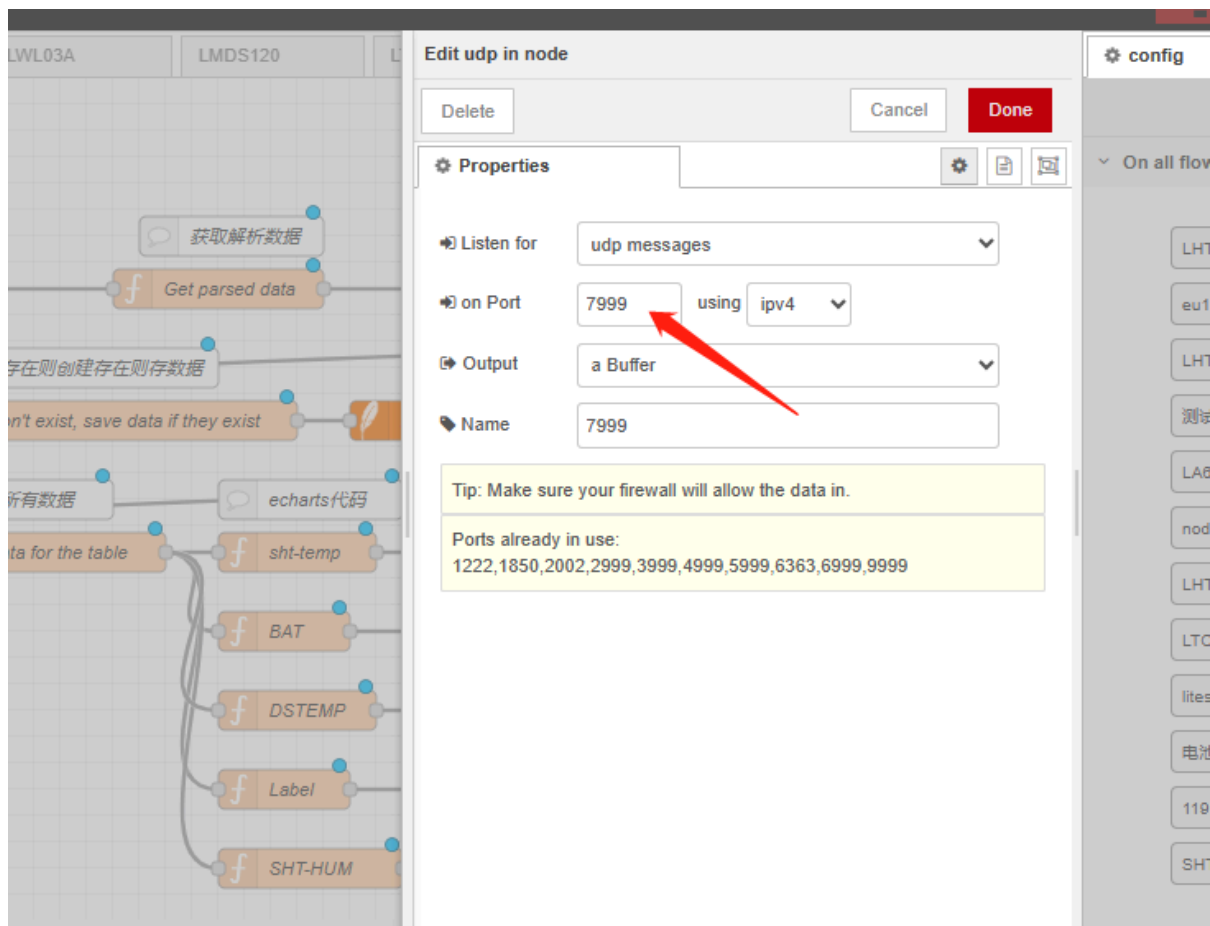




Successfully imported template.

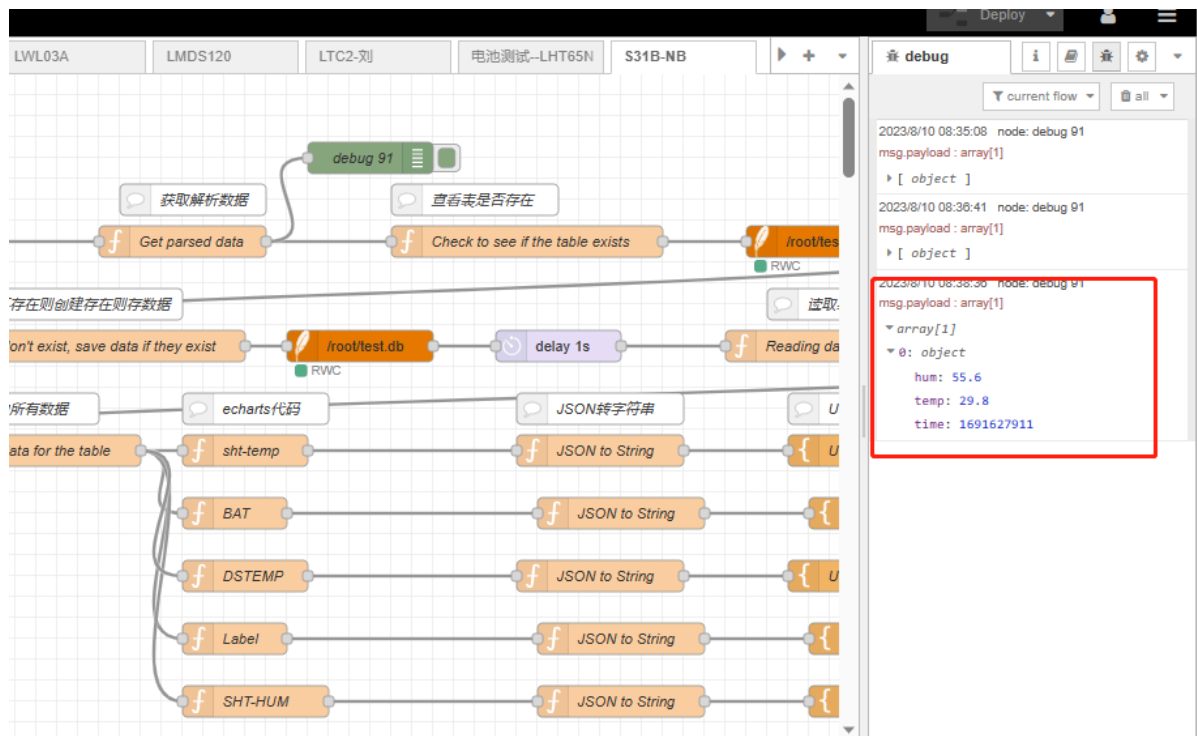


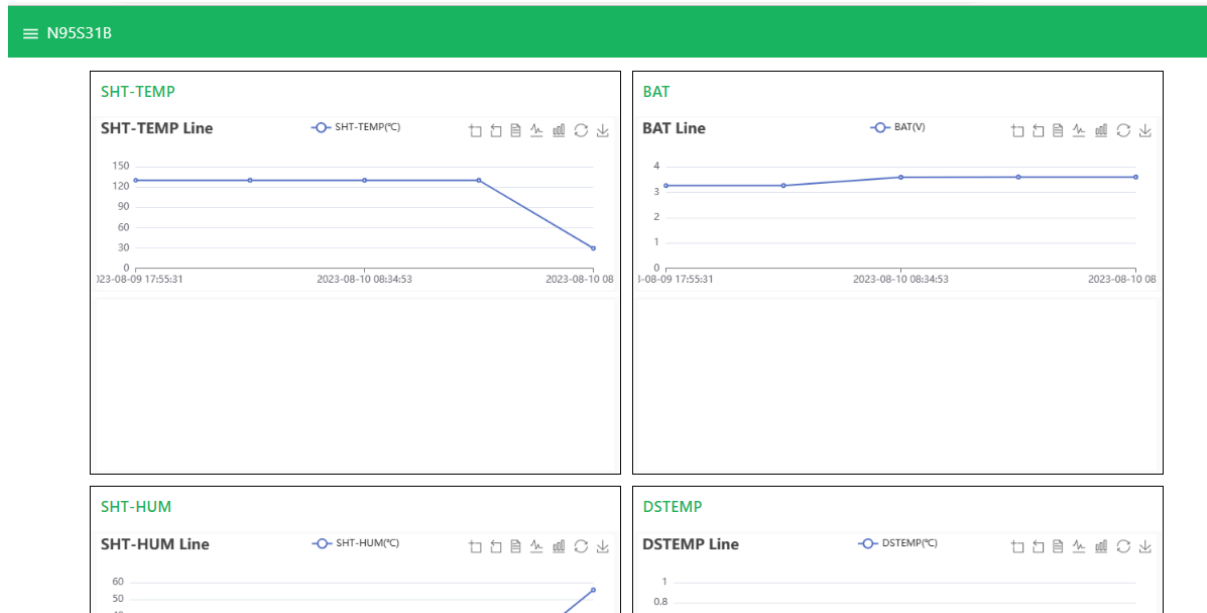
Users can set UDP port.



3.5.2 Simulate Connection

We have completed the configuration of UDP. We can try sending packets to node red.





3.5.3 Configure NB-IoT Sensors

- **AT+PRO=3,0 or 3,5** // hex format or json format
- **AT+SUBTOPIC=<device name>or User Defined**
- **AT+PUBTOPIC=<device name>or User Defined**
- **AT+CLIENT=<device name> or User Defined**
- **AT+UNAME=<device name> or User Defined**
- **AT+PWD="Your device token"**

3.6 ThingsBoard.Cloud (via MQTT)

3.6.1 Configure ThingsBoard

3.6.1.1 Create Device

Create a New Device in [ThingsBoard](#). Record Device Name which is used for MQTT connection.

Add new device

1 Device details 2 Credentials Optional 3 Owner and groups Optional

Name*

Label

☒ Select existing device profile ☐ Create new device profile

Device profile* default

Next: Credentials

Cancel Add

3.6.1.2 Create Uplink & Downlink Converter

Uplink Converter

The purpose of the decoder function is to parse the incoming data and metadata to a format that ThingsBoard can consume. device value objects. Nested objects are not supported.

To create an uplink converter go to the **Integrations center** -
> **Data converters** page and click **plus** button. Name it **"MQTT Uplink Converter"** and select type **"Uplink"**. Use debug mode for now.

The screenshot displays the ThingsBoard Professional interface. The left sidebar contains navigation links: Home, Alarms, Dashboards, Solution, Entities, Profiles, Customers, Users, Integrations center, Integrations, Data converters, Rule chain, Edge rules, Advanced features, Resources, Notification center, Api Usage, White Labeling, Settings, and Security. The main area shows the 'Data converters' page with a list of converters. A modal window titled 'Add Data Converter' is open, showing the following fields and options:

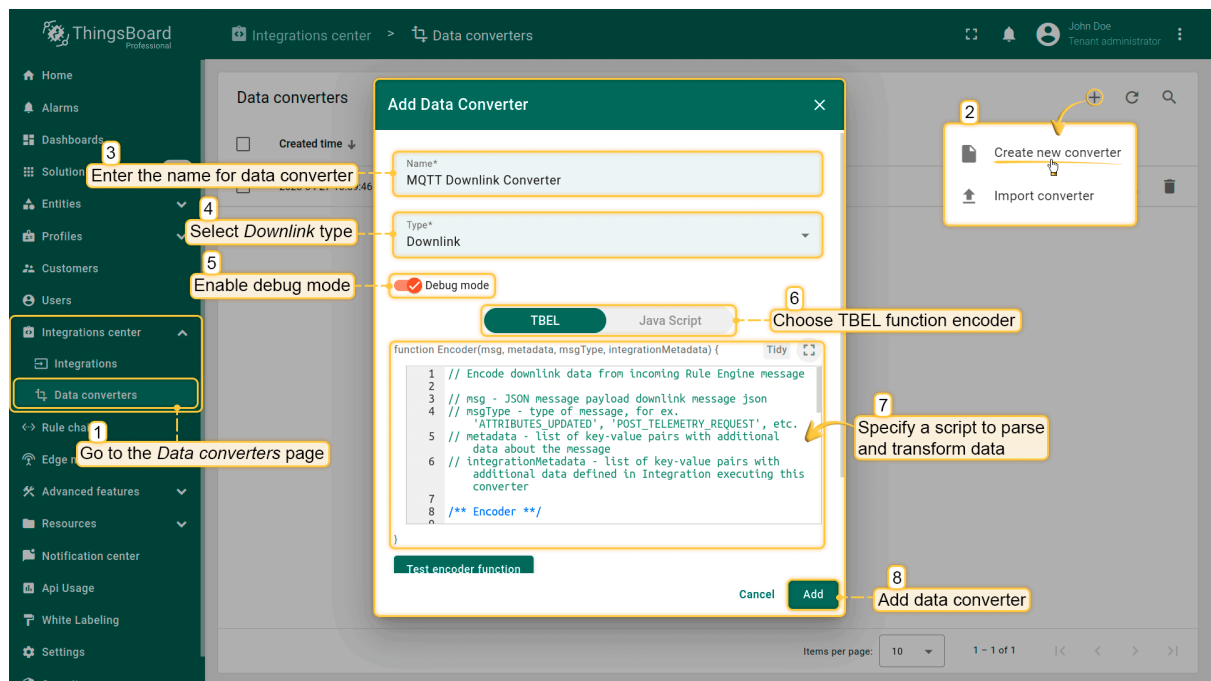
- Name***: MQTT Uplink Converter (Annotation 3: Enter the name for data converter)
- Type***: Uplink (Annotation 4: Select Uplink type)
- Debug mode**: Enabled (Annotation 5: Enable debug mode)
- Decoder**: TBEL (Annotation 6: Choose TBEL function decoder)
- Script**: A JavaScript function for decoding MQTT payloads (Annotation 7: Specify a script to parse and transform data)
- Buttons**: Test decoder function, Cancel, Add (Annotation 8: Add data converter)

The script in the 'Script' field is as follows:

```
function Decoder(payload, metadata) {  
  1  
  2  
  3  /** Decoder **/  
  4  // decode payload to string  
  5  var payloadStr = decodeURIComponent(payload);  
  6  var data = JSON.parse(payloadStr);  
  7  
  8  var deviceName = metadata.topic.split("/")[3];  
  9  // decode payload to JSON  
 10  var deviceType = 'sensor';  
 11  
 12  // Result object with device attributes/telemetry data  
 13  var result = {  
    deviceName: deviceName,  
    deviceType: deviceType,  
    data: data  
  };  
}
```

Downlink Converter

The Downlink converter transforming outgoing RPC message and then the Integration sends it to external MQTT broker

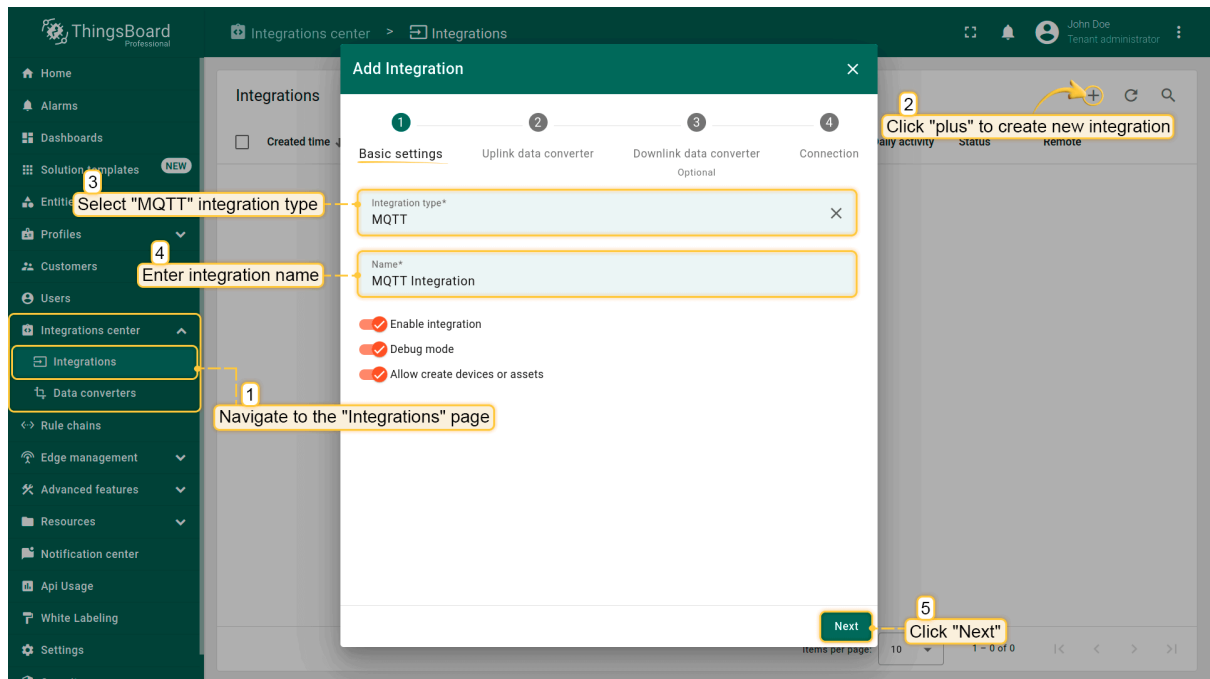


Note:

Our device payload is already human readable data. Therefore, users do not need to write decoders. Simply create by default

3.6.1.3 MQTT Integration Setup

Go to the **Integrations center** -> **Integrations** page and click “plus” icon to add a new integration. Name it “**MQTT Integration**”, select type **MQTT**;



- The next steps is to add the recently created uplink and downlink converters;

ThingsBoard Professional

Integrations center > Integrations

Home Alarms Dashboards Solution templates NEW Entities Profiles Customers Users Ingest Data converters Rule chains Edge management Advanced features Resources Notification center Api Usage White Labeling Settings Security

Integrations

Created time

Basic settings MQTT Uplink data converter Downlink data converter Optional Connection

Select existing uplink data converter

Uplink data converter* MQTT Uplink Converter

Create new uplink data converter

Back Next

Items per page: 10 1 - 0 of 0

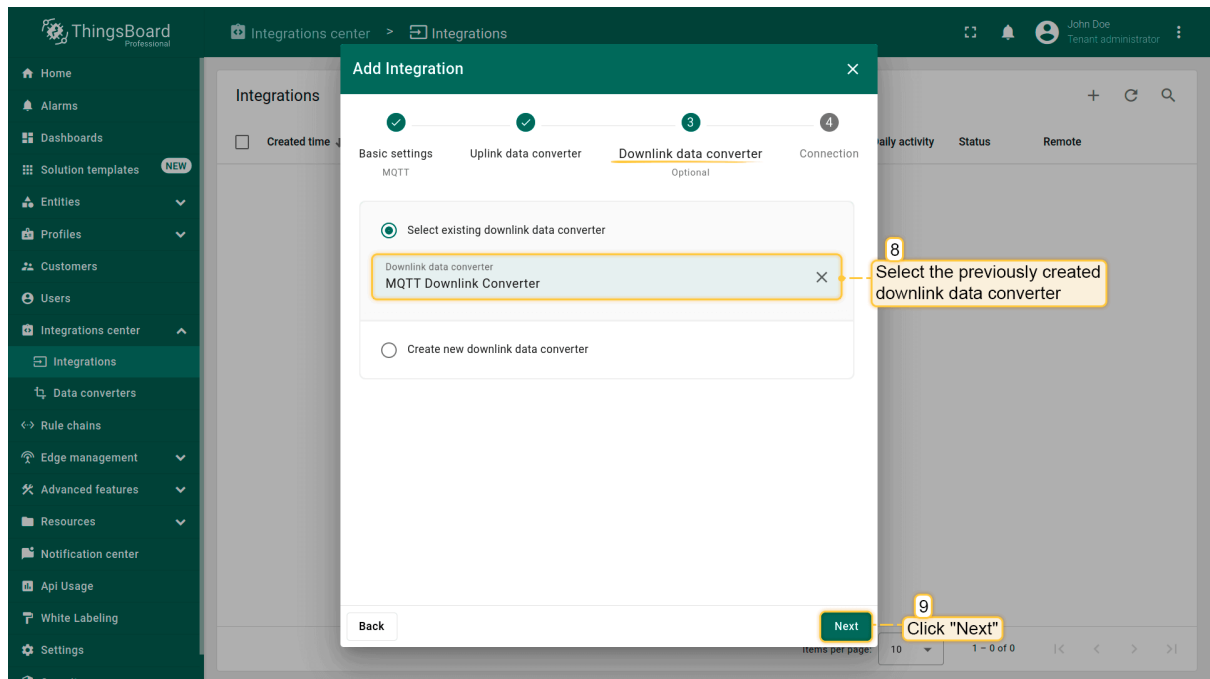
John Doe Tenant administrator

daily activity Status Remote

At this step, you can either choose a previously created uplink data converter or create a new one directly in this window

6 Select the previously created uplink data converter

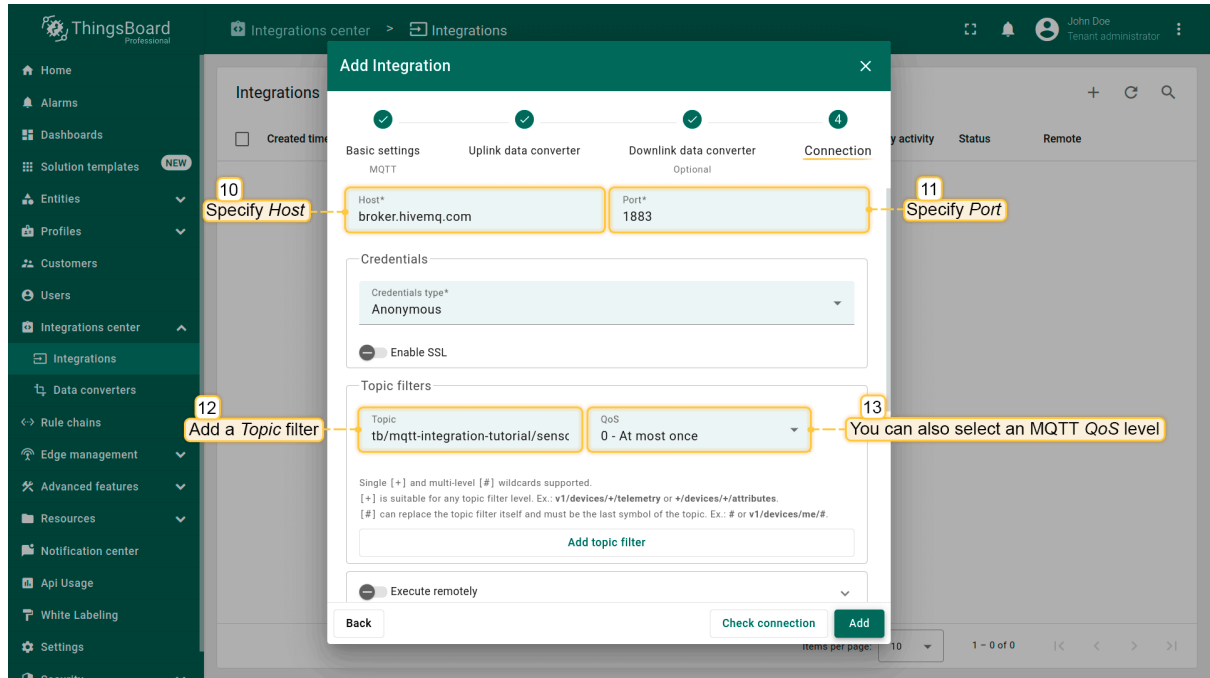
7 Click "Next"



Add a topic filter:

Consistent with the theme of the node setting.

You can also select an MQTT QoS level. We use MQTT QoS level 0 (At most once) by default;



3.6.2 Simulate with MQTT.fx

Profile Name

Profile Type MQTT Broker



MQTT Broker Profile Settings

Broker Address

Broker Port

Client ID Generate

General

User Credentials

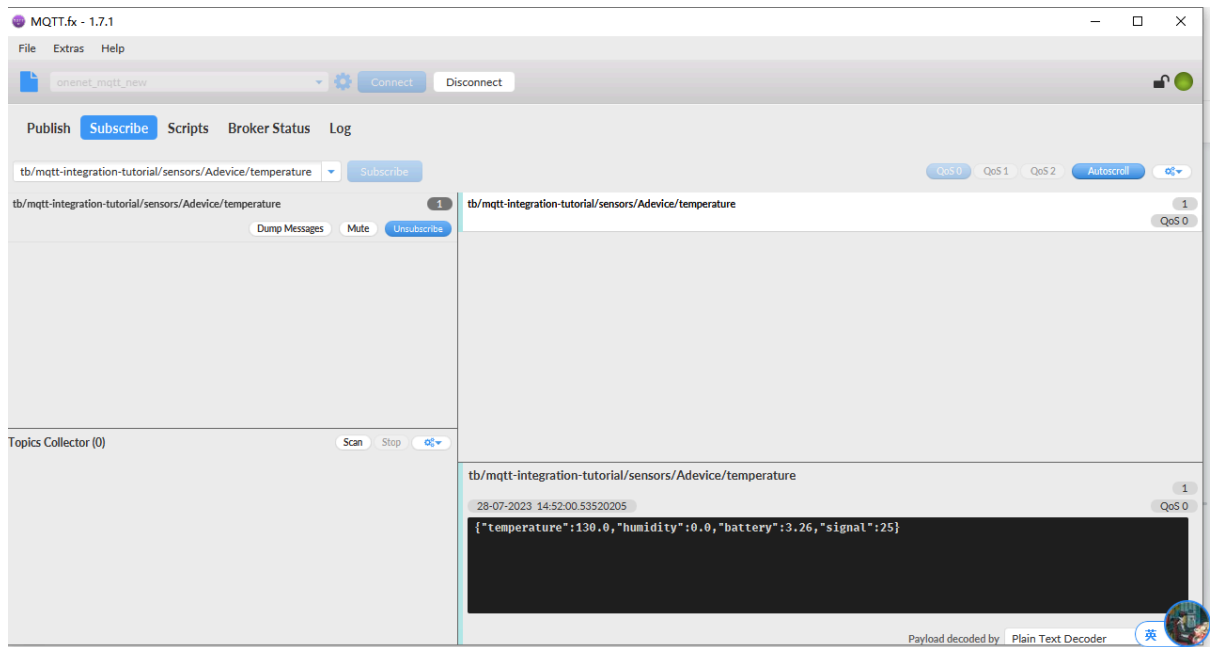
SSL/TLS

Proxy

LWT

User Name

Password



3.6.3 Configure NB-IoT Sensor

AT Commands

- **AT+PRO=3,3** // Use MQTT to connect to ThingsBoard. Payload Type set to 3.
- **AT+SUBTOPIC=<device name>**
- **AT+PUBTOPIC=<device name>**
- **AT+CLIENT=<device name> or User Defined**
- **AT+UNAME=<device name> or User Defined**
- **AT+PWD=<device name> or User Defined**

Test Uplink by click the button for 1 second

Device A

Device details

?

✕

<

Details

Attributes

Latest telemetry

Alarms

Events

Relations

>

Latest telemetry

🔍

☐	Last update time	Key ↑	Value
☐	2023-07-20 17:21:10	humidity	80
☐	2023-07-20 17:21:10	rawData	{\"temperature\":130.0,\"humidity\":0.0,\"battery...}
☐	2023-07-20 17:21:10	temperature	42

Items per page: 10 1 - 3 of 3 << < > >>

Integrations

INtest

Integration details

Message

✕

```
{  \"topic\": \"tb/mqtt-integration-tutorial/sensors/Adevice/temper\"  \"payload\": {    \"temperature\": 130.0,    \"humidity\": 0.0,    \"battery\": 3.25,    \"signal\": 23  }
```

Close

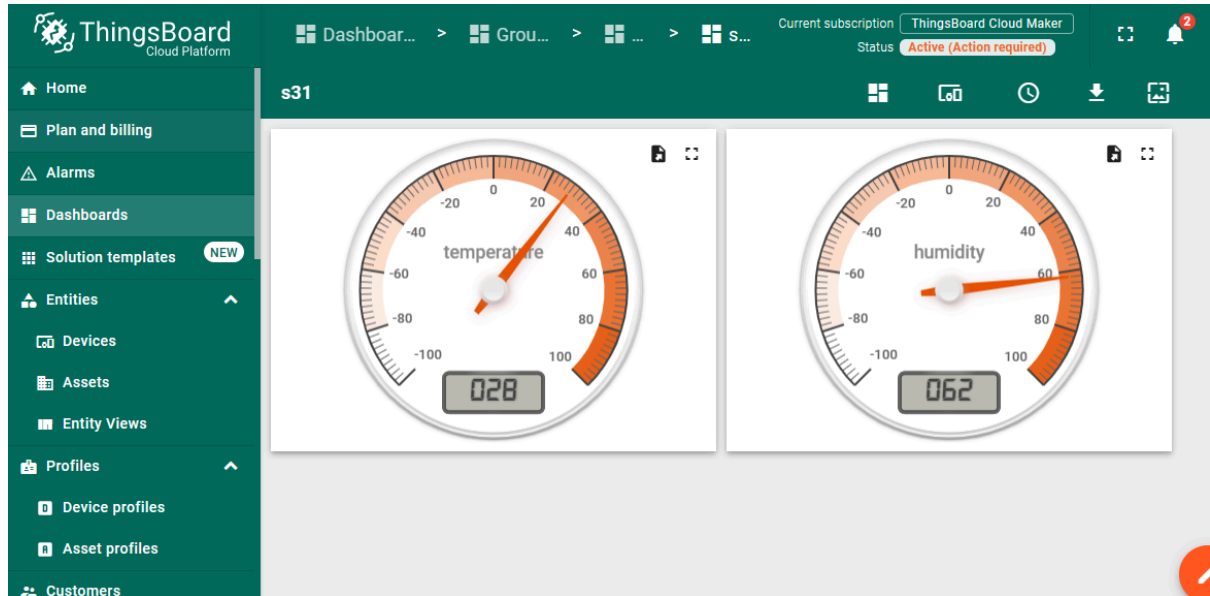
2023-07-20 16:49:06

tb-ie-main-1

Uplink

...

OK



3.7 ThingsBoard.Cloud (via COAP)

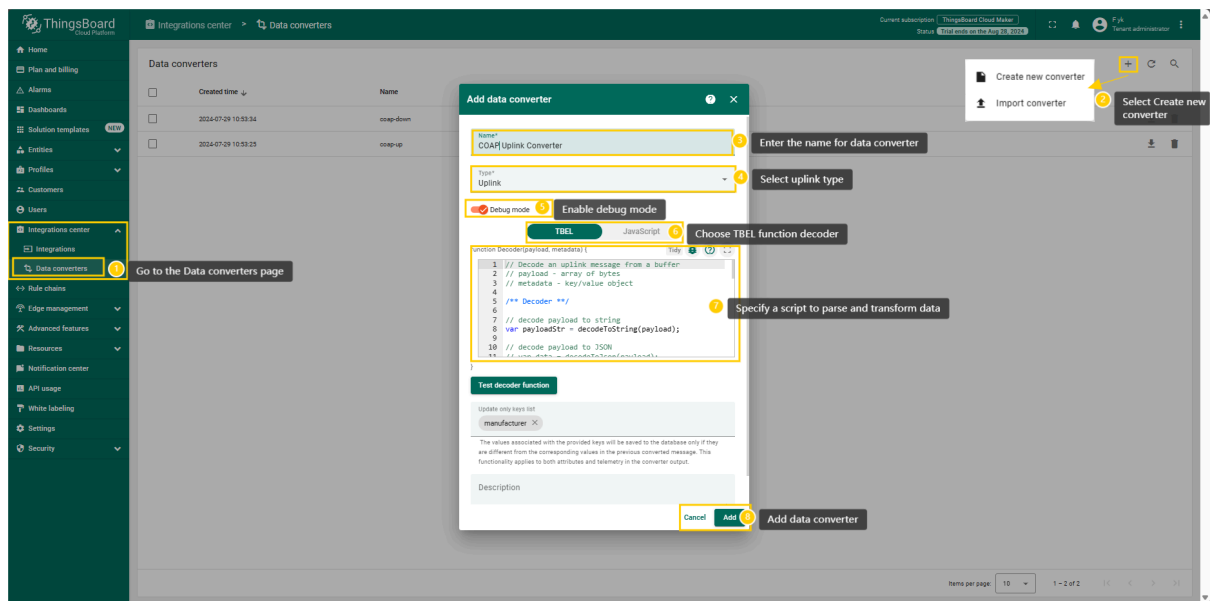
3.7.1 Configure ThingsBoard

3.7.1.1 Create Uplink & Downlink Converter

Uplink Converter

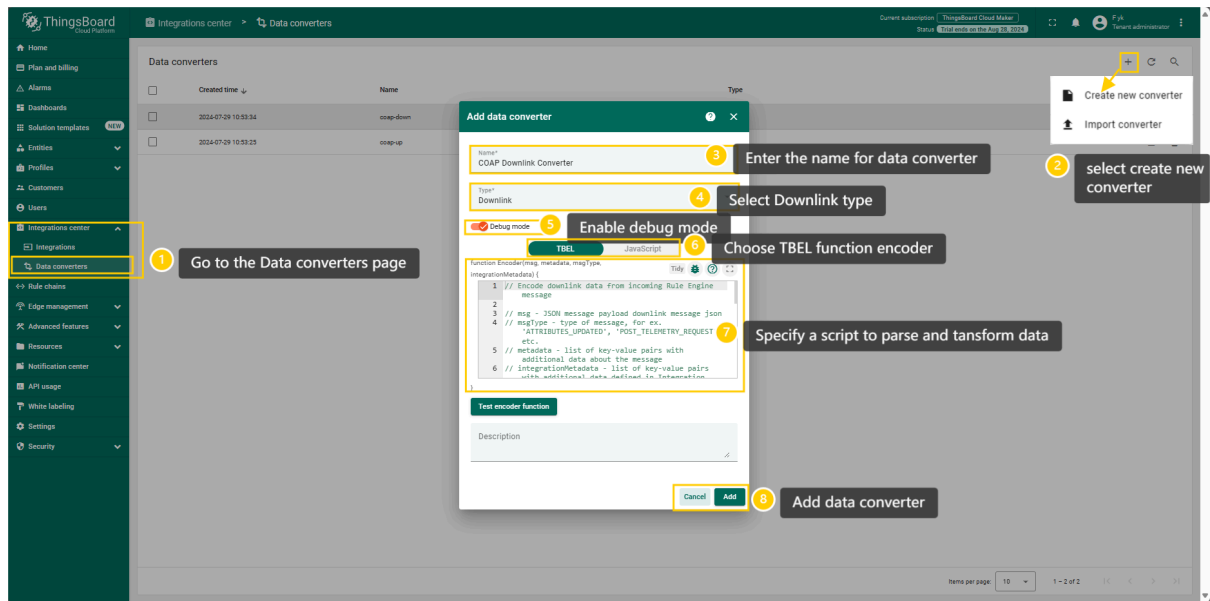
The purpose of the decoder function is to parse the incoming data and metadata to a format that ThingsBoard can consume. device value objects. Nested objects are not supported.

To create an uplink converter go to the [Integrations center](#) -
> [Data converters](#) page and click “plus” button. Name it “[COAP Uplink Converter](#)” and select type “[Uplink](#)”. Use debug mode for now.



Downlink Converter

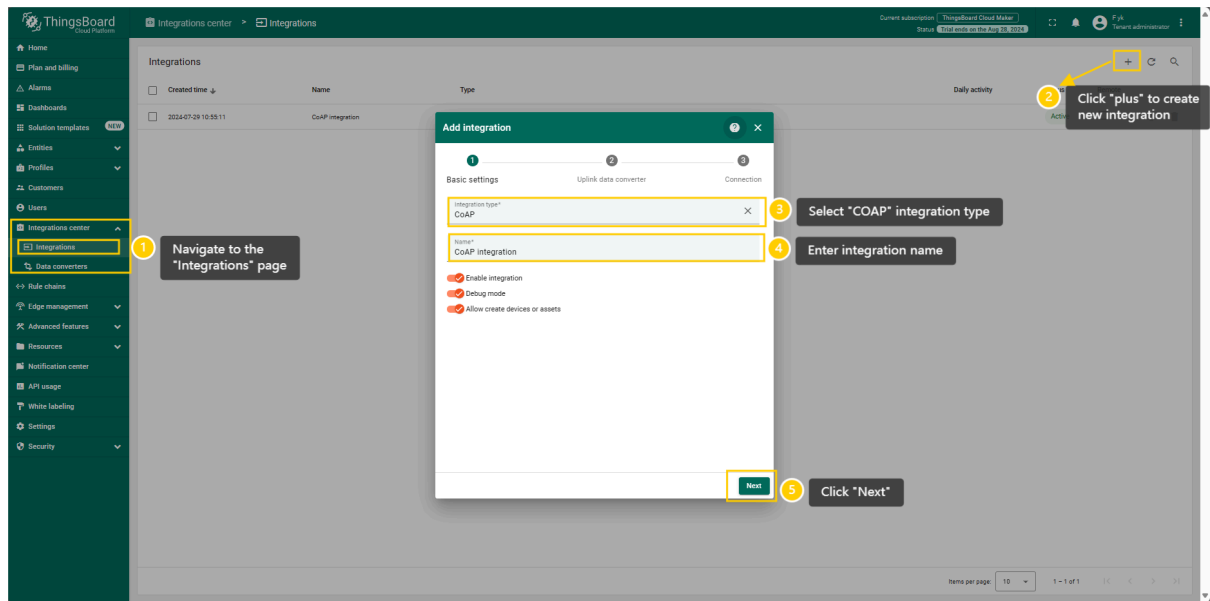
The Downlink converter transforming outgoing RPC message and then the Integration sends it to external COAP broker.



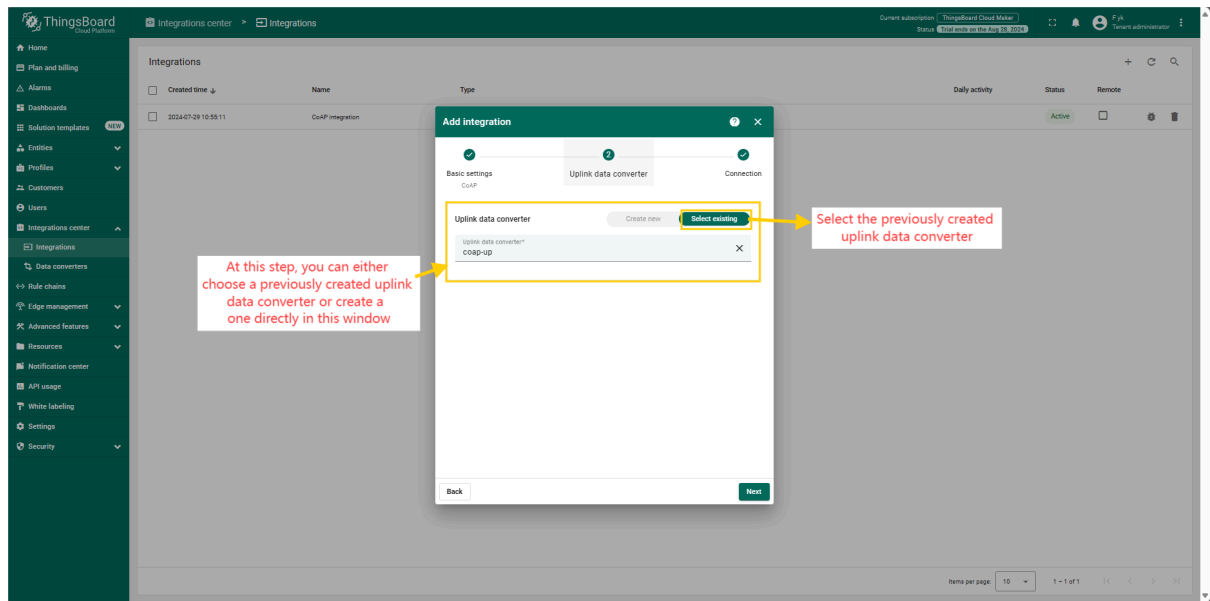
3.7.1.2 COAP Integration Setup

Go to the [Integrations center](#) -> [Integrations page](#) and click “plus” icon to add a new integration. Name it “**CoAP Integration**”, select type **COAP** ;

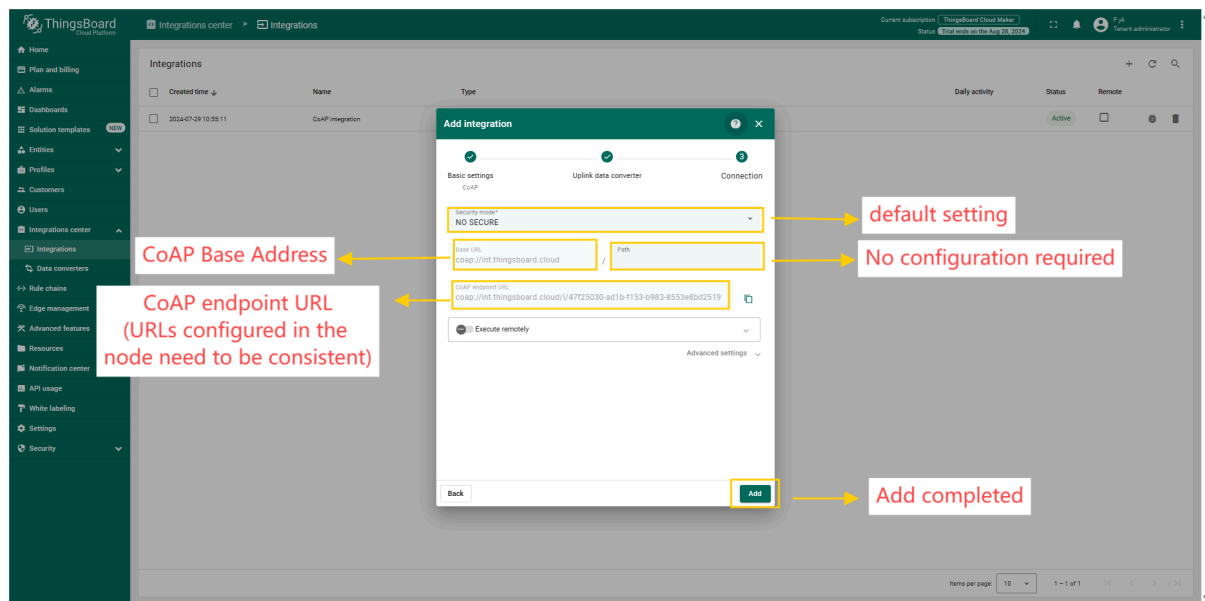
Home - General Configure/Commands to Connect to IoT server for -NB & -NS NB-IoT models



The next steps is to add the recently created uplink converters;



3.7.1.3 Add COAP Integration

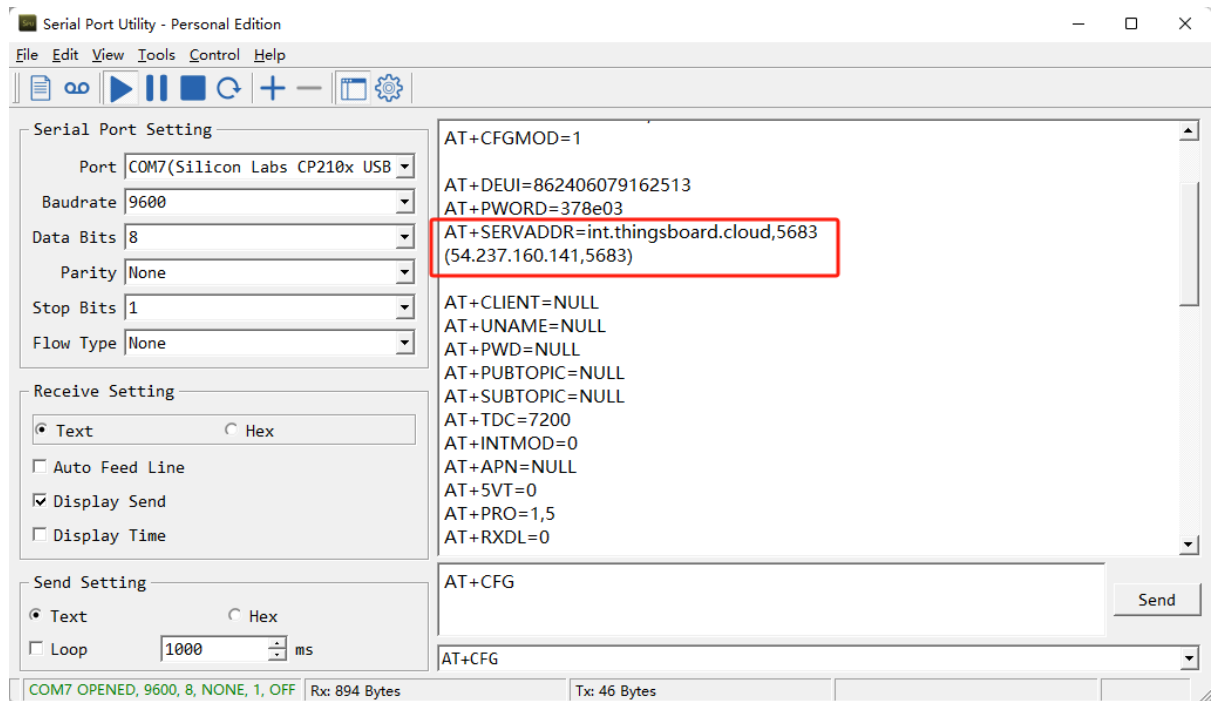


3.7.2 Node Configuration(Example: Connecting to the Thingsboard platform)

3.7.2.1 Instruction Description

- AT+PRO=1,0(HEX format uplink) &AT+PRO=1,5(JSON format uplink)
- AT+SERVADDR=COAP Server Address,5683

Example: AT+SERVADDR=int.thingsboard.cloud,5683(The address is automatically generated when the COAP integration is created)



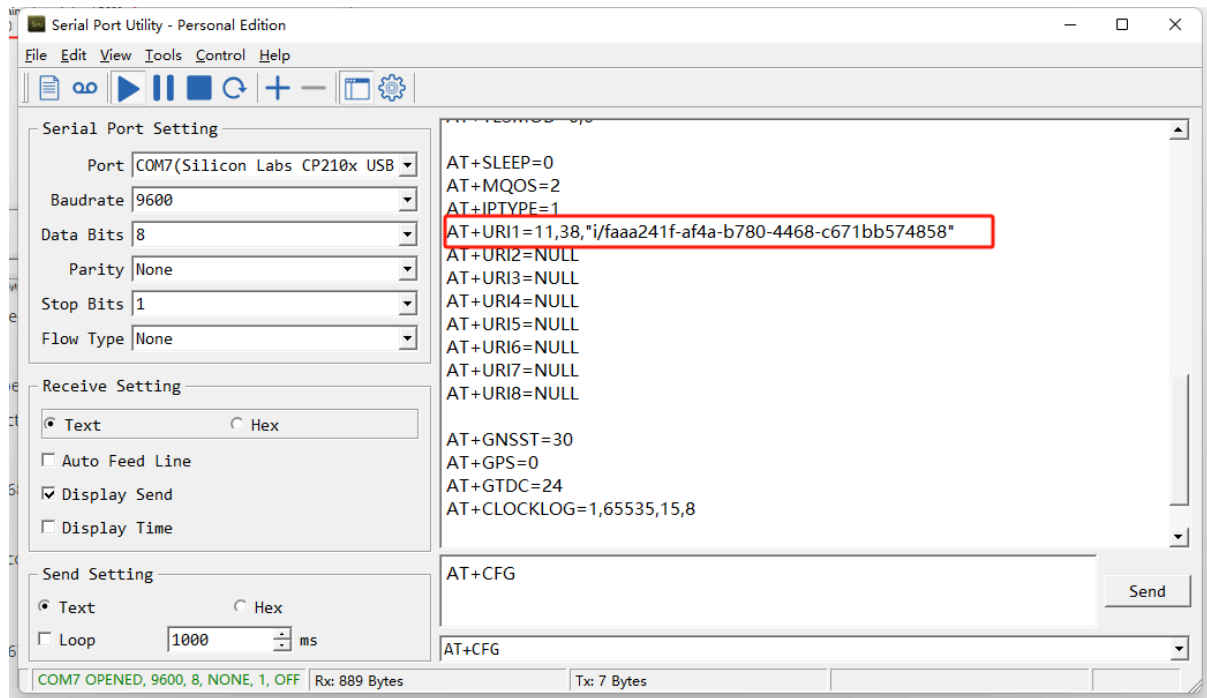
Note: The port for the COAP protocol has been fixed to 5683

- AT+URL1=11, **character length**, "Needs to be consistent with the CoAP endpoint URL in the platform"

If the module used is **BC660K**, **only one** URL directive needs to be configured,

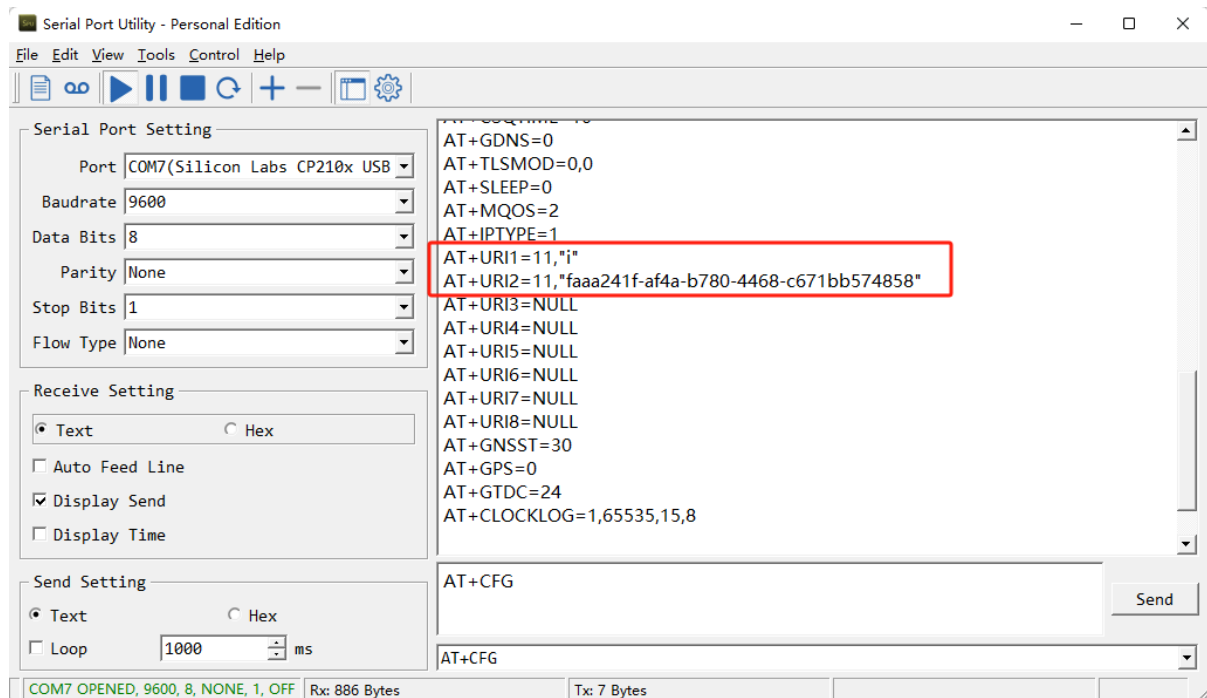
e.g.

- AT+URL1=11,38, "i/faaaa241f-af4a-b780-4468-c671bb574858"



If you are using a **BG95-M2** module, you need to configure **TWO** URL commands,
e.g.

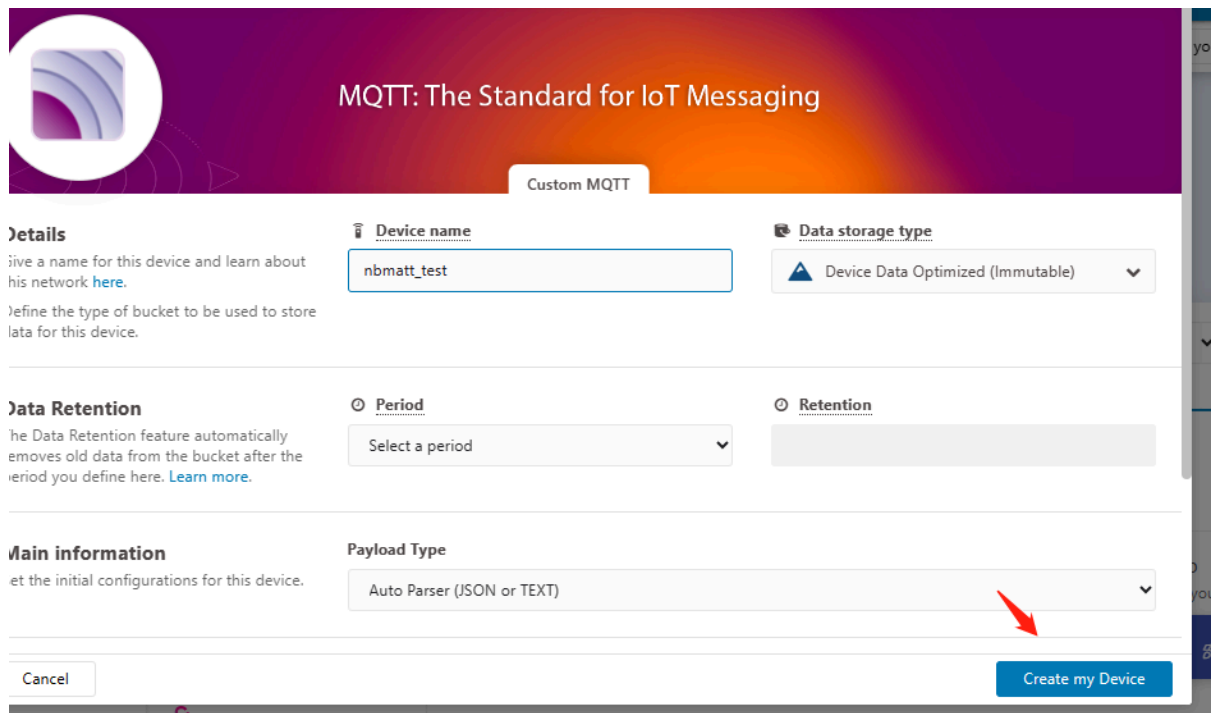
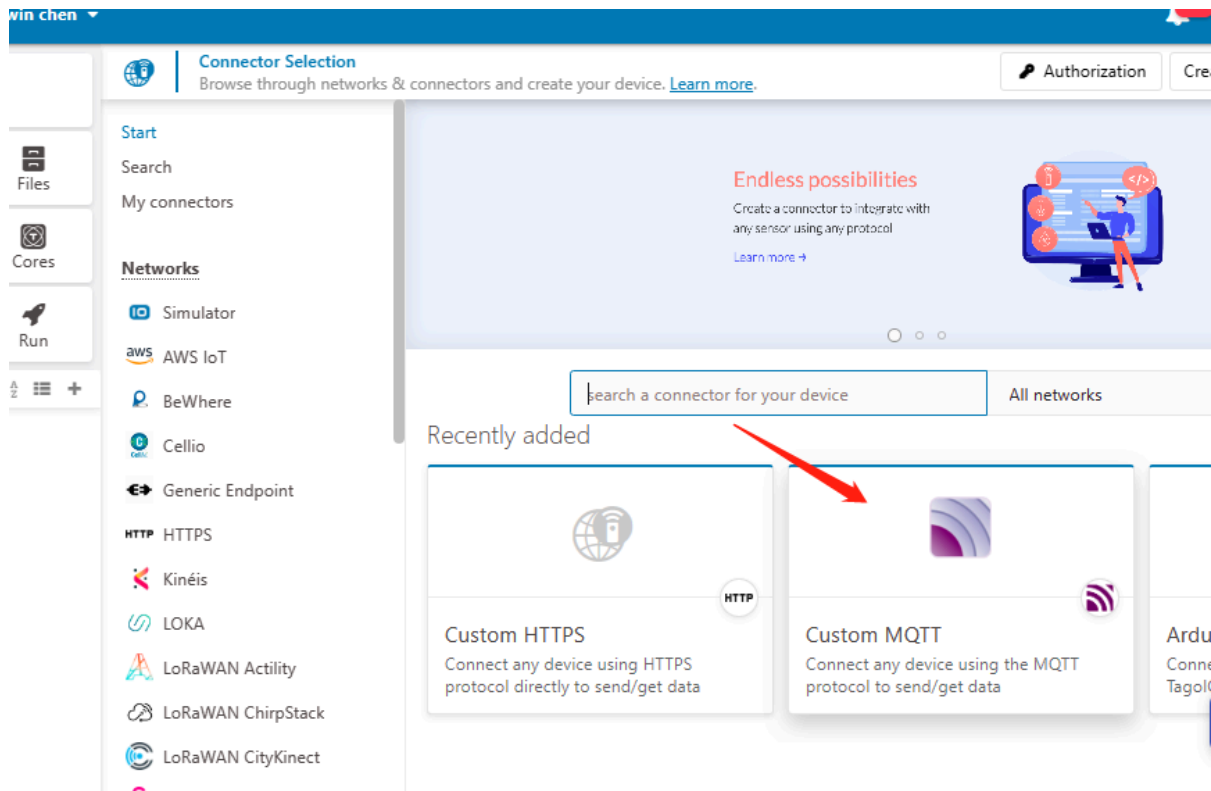
- AT+URL1=11, "i";
- AT+URL2=11, "/faaaa241f-af4a-b780-4468-c671bb574858"



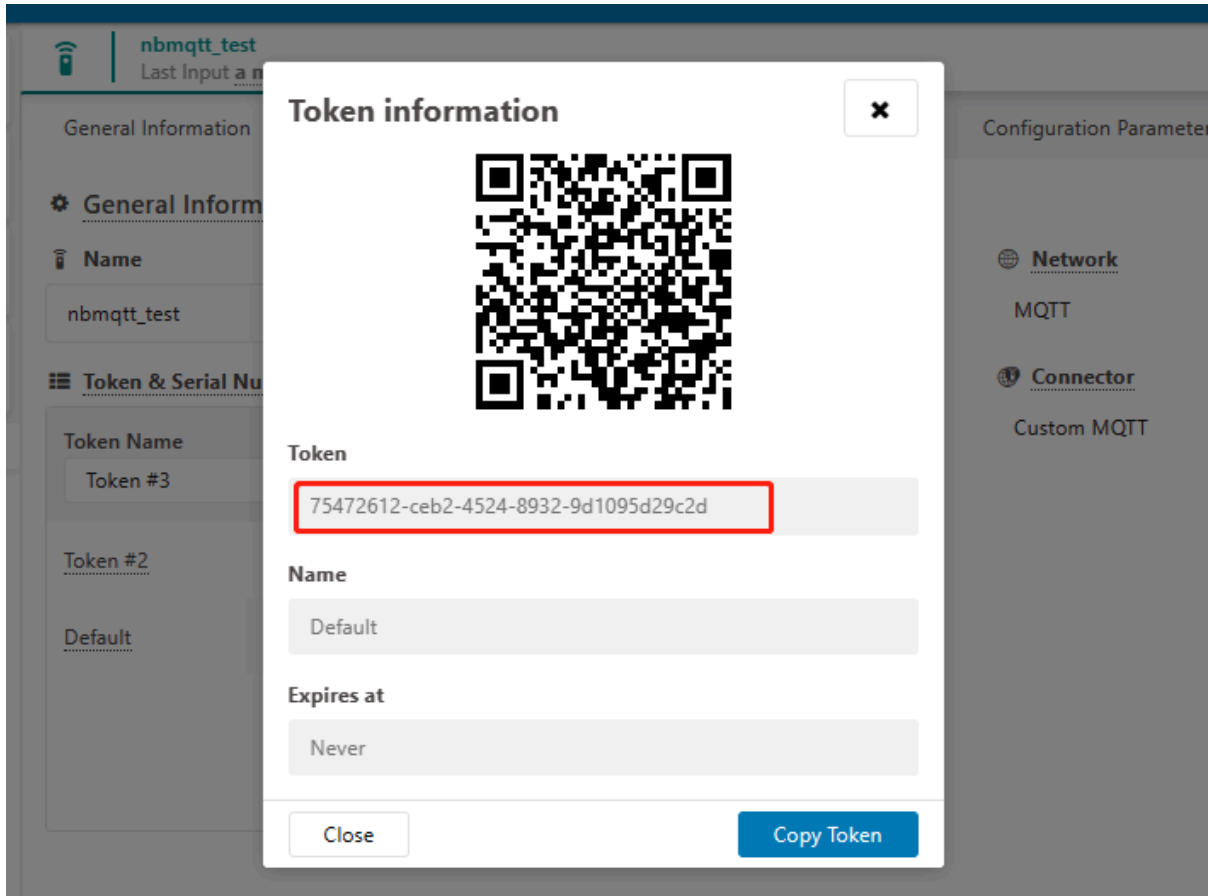
3.8 [Tago.io](#) (via MQTT)

3.8.1 Create device & Get Credentials

We use MQTT Connection to send data to [Tago.io](#). We need to Create Device and Get MQTT Credentials first.



Go to the Device section and create a device. Then, go to the section tokens and copy your device-token.



The device needs to enable the TLS mode and set the **AT+TLSMOD=1,0** command.

On the Connection Profile window, set the following information:

- **Profile Name:** “Any name”
- **Broker Address:** mqtt.tago.io
- **Broker Port:** 8883
- **Client ID:** “Any value”

On the section User credentials, set the following information:

- **User Name:** “Any value” // Tago validates your user by the token only
- **Password:** “Your device token”
- **PUBTOPIC:** “Any value”
- **SUBTOPIC:** “Any value”

AT command:

- **AT+PRO=3,0 or 3,5** // hex format or json format
- **AT+SUBTOPIC=<device name>or User Defined**
- **AT+PUBTOPIC=<device name>or User Defined**
- **AT+CLIENT=<device name> or User Defined**
- **AT+UNAME=<device name> or User Defined**
- **AT+PWD=“Your device token”**

3.8.2 Simulate with MQTT.fx

Edit Connection Profiles

Tago

Connection Profile

Profile Name:

Broker Address:

Broker Port:

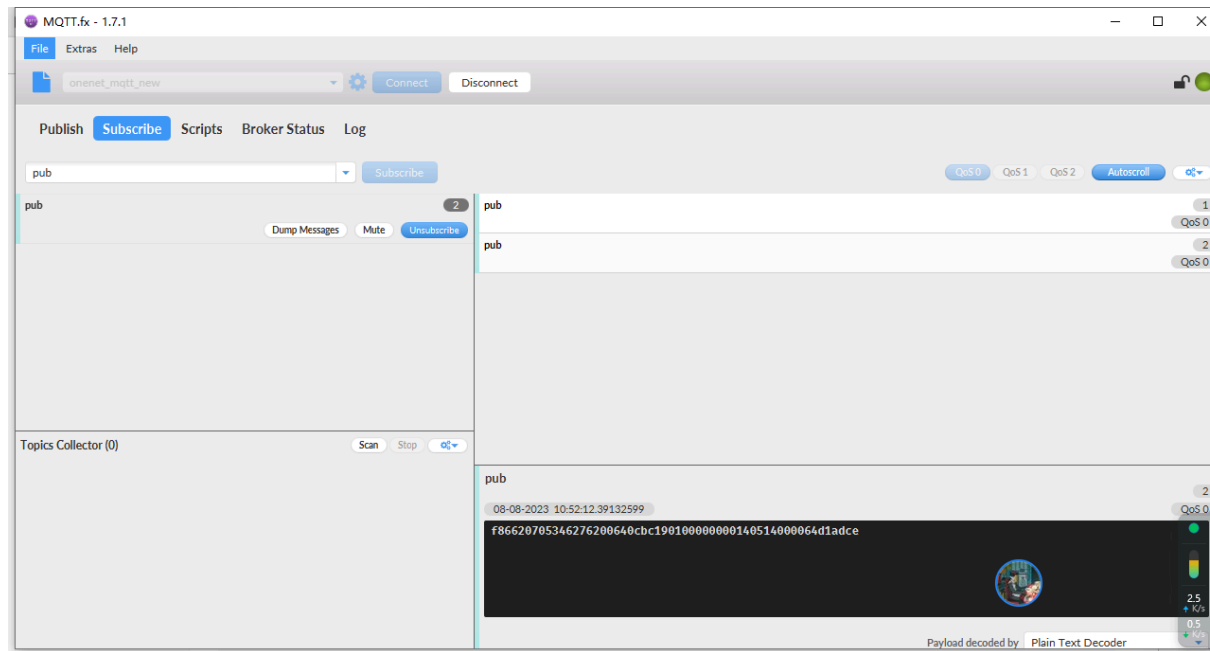
Client ID:

General | User Credentials | **SSL/TLS** | Proxy | Last Will and Testament

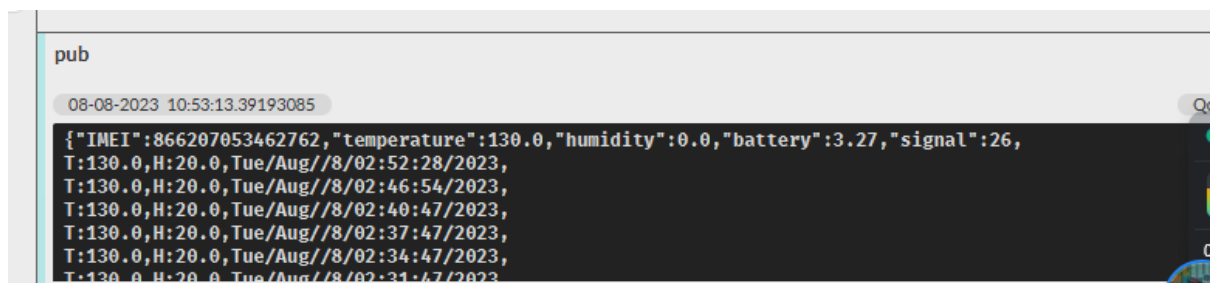
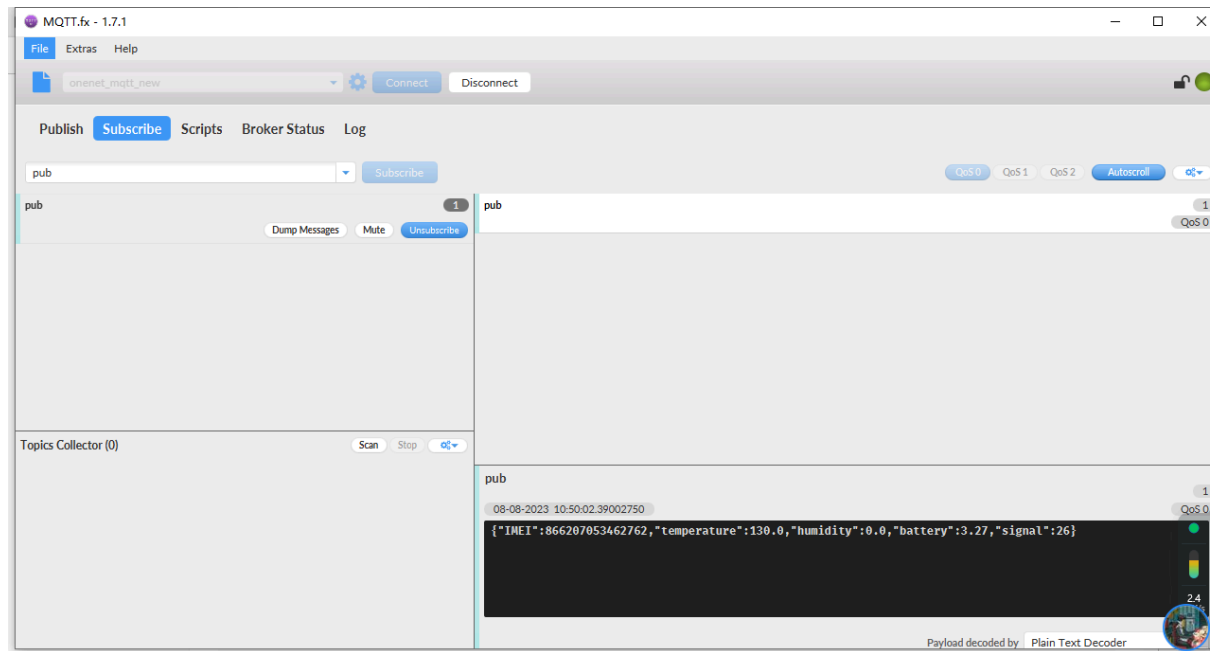
Enable SSL/TLS ☒

Protocol:

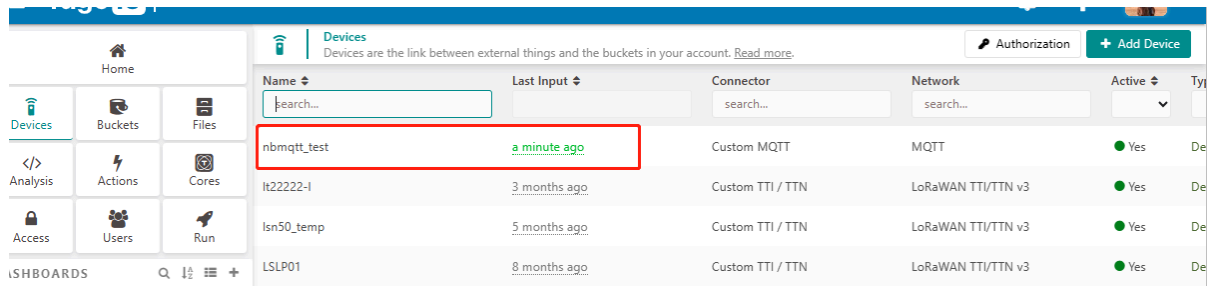
- ☐ CA signed server certificate
- ☐ CA certificate file
- ☐ CA certificate keystore
- ☐ Self signed certificates
- ☐ Self signed certificates in keystores



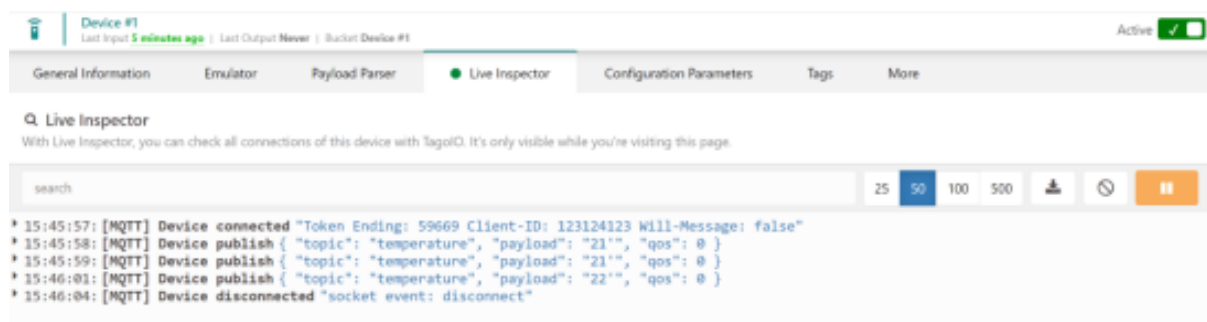
Users can run the **AT+PRO=3,5** command, and the payload will be converted to **JSON format**.



3.8.3 tago data



Name	Last Input	Connector	Network	Active	Type
nbmqtt_test	a minute ago	Custom MQTT	MQTT	Yes	De
lt22222-l	3 months ago	Custom TTI / TTN	LoRaWAN TTI/TTN v3	Yes	De
lsm50_temp	5 months ago	Custom TTI / TTN	LoRaWAN TTI/TTN v3	Yes	De
LSP01	8 months ago	Custom TTI / TTN	LoRaWAN TTI/TTN v3	Yes	De



Device #1
Last Input 5 minutes ago | Last Output Never | Bucket Device #1 | Active

General Information | Emulator | Payload Parser | **Live Inspector** | Configuration Parameters | Tags | More

Live Inspector

With Live Inspector, you can check all connections of this device with TagUI. It's only visible while you're visiting this page.

search [25] 50 100 500 [Download] [Refresh] [Pause]

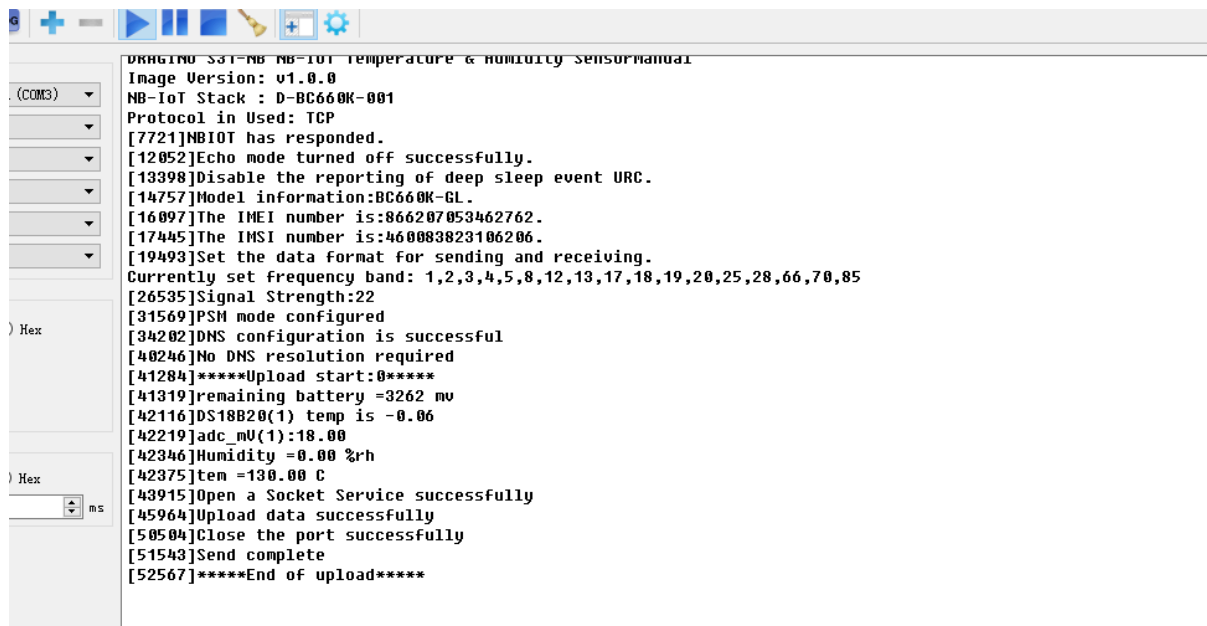
```
* 15:45:57: [MQTT] Device connected "Token Ending: 59669 Client-ID: 123124123 Will-Message: false"
* 15:45:58: [MQTT] Device publish { "topic": "temperature", "payload": "21", "qos": 0 }
* 15:45:59: [MQTT] Device publish { "topic": "temperature", "payload": "21", "qos": 0 }
* 15:46:01: [MQTT] Device publish { "topic": "temperature", "payload": "22", "qos": 0 }
* 15:46:04: [MQTT] Device disconnected "socket event: disconnect"
```

3.9 TCP Connection

AT command:

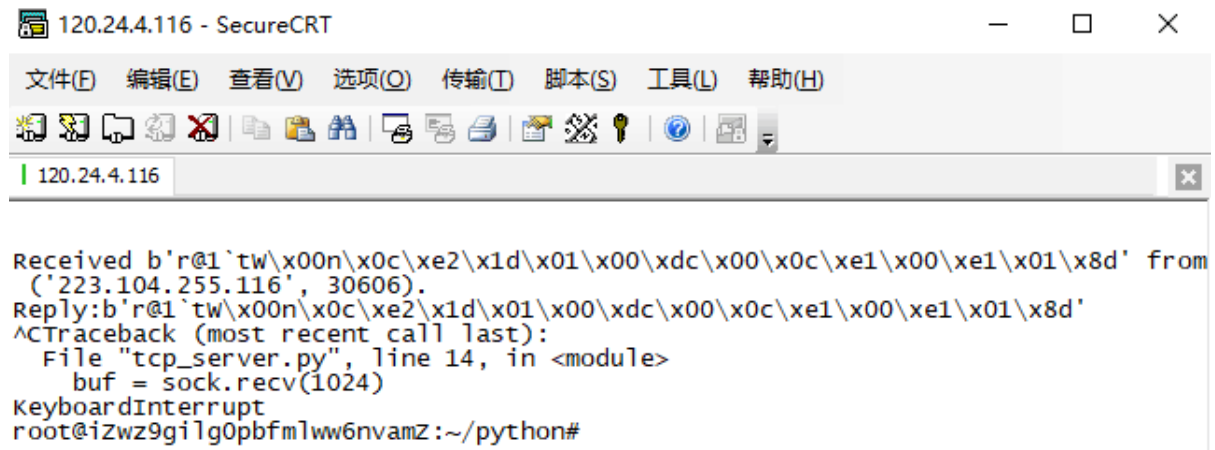
- **AT+PRO=4,0** // Set to use TCP protocol to uplink(HEX format)
- **AT+PRO=4,1** // Set to use TCP protocol to uplink(JSON format)
- **AT+SERVADDR=120.24.4.116,5600** // to set TCP server address and port

Sensor Console Output when Uplink:



```
DRAGINO S31-NB NB-IoT Temperature & Humidity Sensor module
Image Version: v1.0.0
NB-IoT Stack : D-BC660K-001
Protocol in Used: TCP
[7721]NB-IoT has responded.
[12052]Echo mode turned off successfully.
[13398]Disable the reporting of deep sleep event URC.
[14757]Model information:BC660K-GL.
[16097]The IMEI number is:866207053462762.
[17445]The IMSI number is:460083823106206.
[19493]Set the data format for sending and receiving.
Currently set Frequency band: 1,2,3,4,5,8,12,13,17,18,19,20,25,28,66,70,85
[26535]Signal Strength:22
[31569]PSM mode configured
[34202]DNS configuration is successful
[40246]No DNS resolution required
[41284]*****Upload start:0*****
[41319]remaining battery =3262 mv
[42116]DS18B20(1) temp is -0.06
[42219]adc_mv(1):18.00
[42346]Humidity =0.00 %rh
[42375]ten =130.00 C
[43915]Open a Socket Service successfully
[45964]Upload data successfully
[50504]Close the port successfully
[51543]Send complete
[52567]*****End of upload*****
```

See result in TCP Server:



```
120.24.4.116 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
120.24.4.116
Received b'r@1'tw\x00n\x0c\xe2\x1d\x01\x00\xdc\x00\x0c\xe1\x00\xe1\x01\x8d' from
('223.104.255.116', 30606).
Reply:b'r@1'tw\x00n\x0c\xe2\x1d\x01\x00\xdc\x00\x0c\xe1\x00\xe1\x01\x8d'
^CTraceback (most recent call last):
  File "tcp_server.py", line 14, in <module>
    buf = sock.recv(1024)
KeyboardInterrupt
root@izwz9gilg0pbfmlww6nvamZ:~/python#
```

3.10 AWS Connection

Users can refer to [Dragino NB device connection to AWS platform instructions](#)

4. MQTT/UDP/TCP downlink

4.1 MQTT (via MQTT.fx)

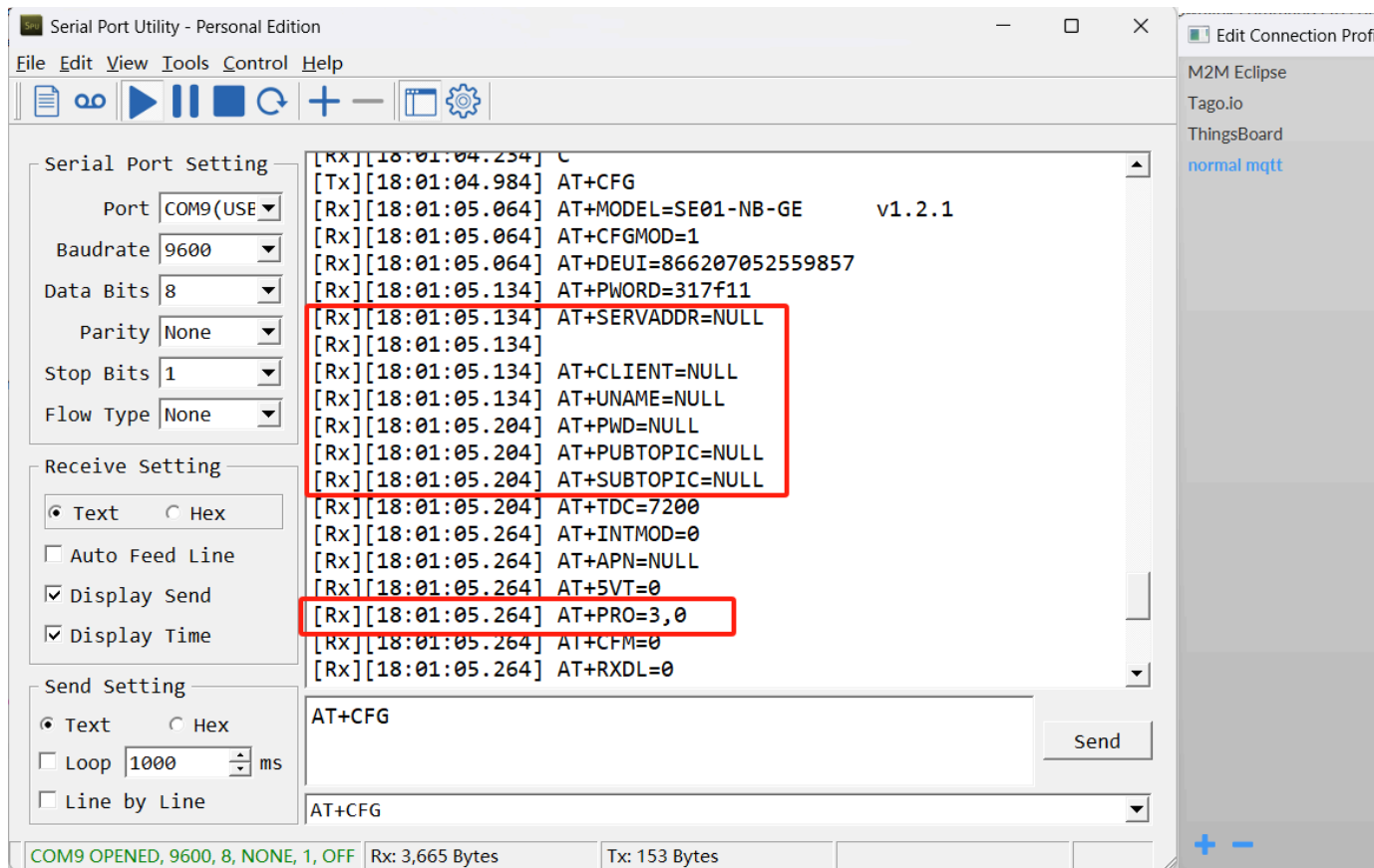
Configure MQTT connections properly and send downlink commands to configure nodes through the Publish function of MQTT.fx.

1. Configure node MQTT connection (via MQTT.fx):

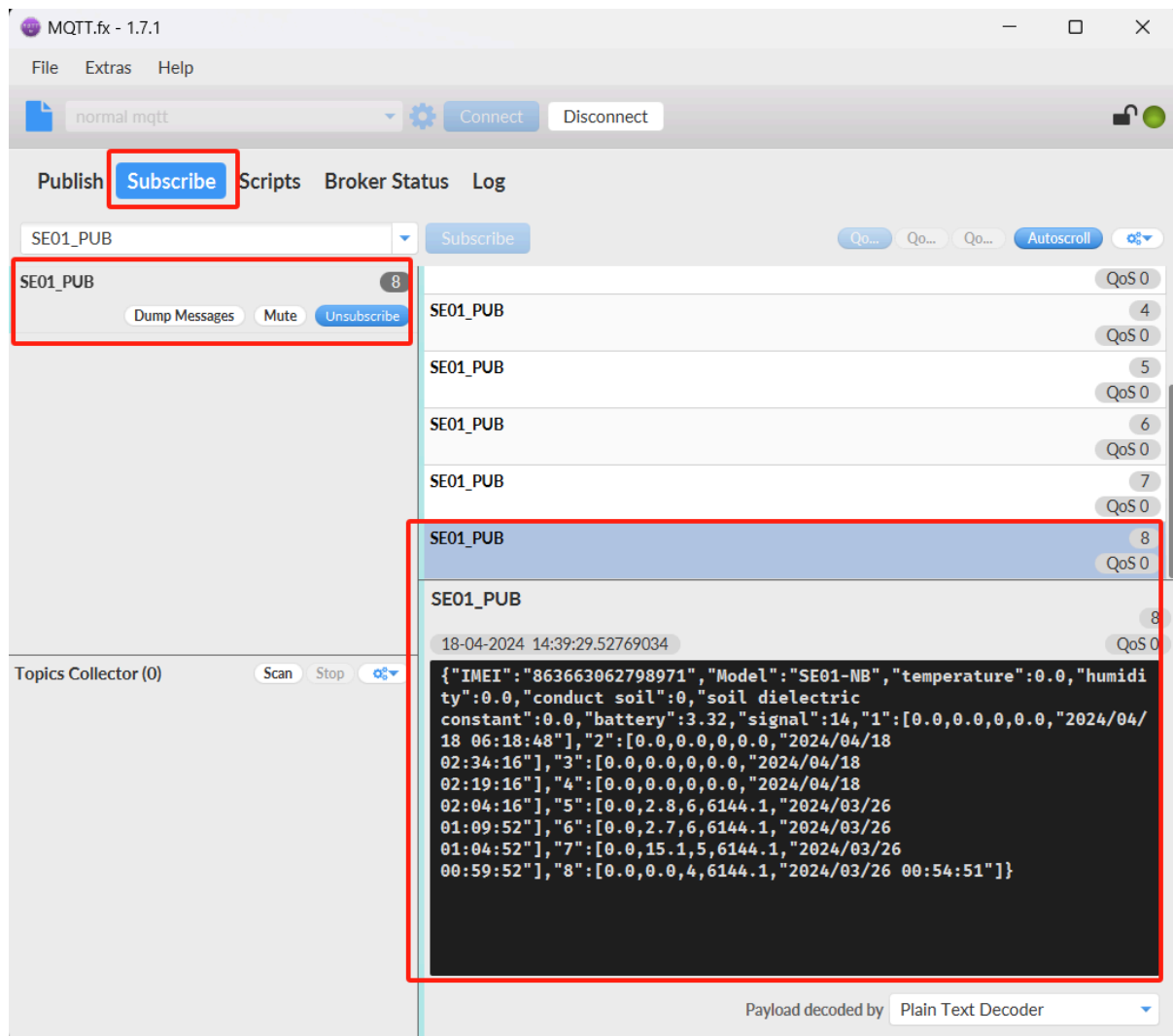
AT command:

- **AT+PRO=3,0 or 3,5** // hex format or json format
- **AT+SUBTOPIC=User Defined**
- **AT+PUBTOPIC=User Defined**
- **AT+UNAME=<device name> or User Defined**
- **AT+PWD=<device name> or User Defined**
- **AT+SERVADDR=8.217.91.207,1883** // to set MQTT server address and port

Note: To uplink and downlink via MQTT.fx, we need set the publish topic and subscribe topic different, for example: AT+SUBTOPIC=SE01_SUB & AT+PUBTOPIC=SE01_PUB.



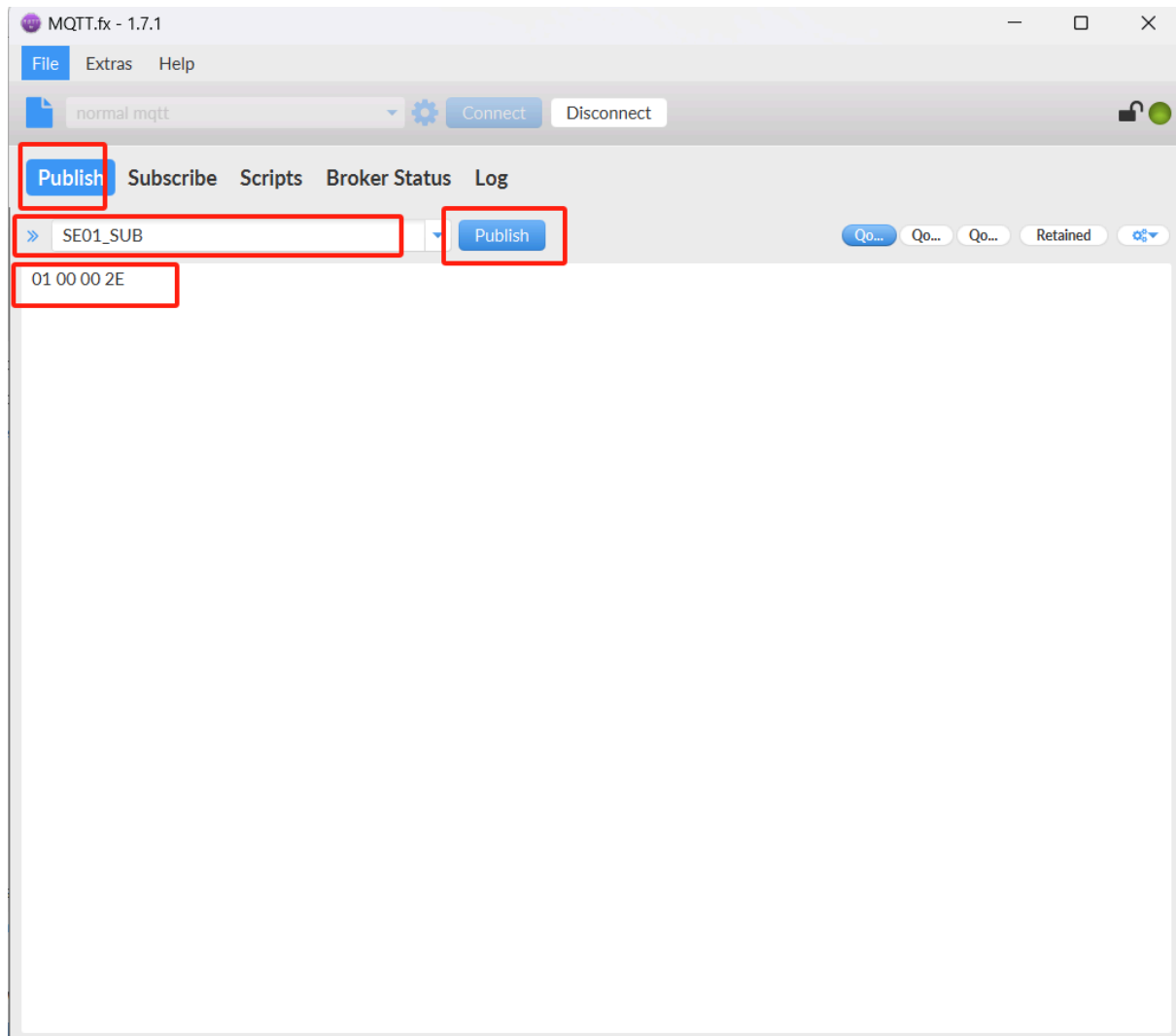
2. When the node uplink packets, we can observe the data in MQTT.fx.

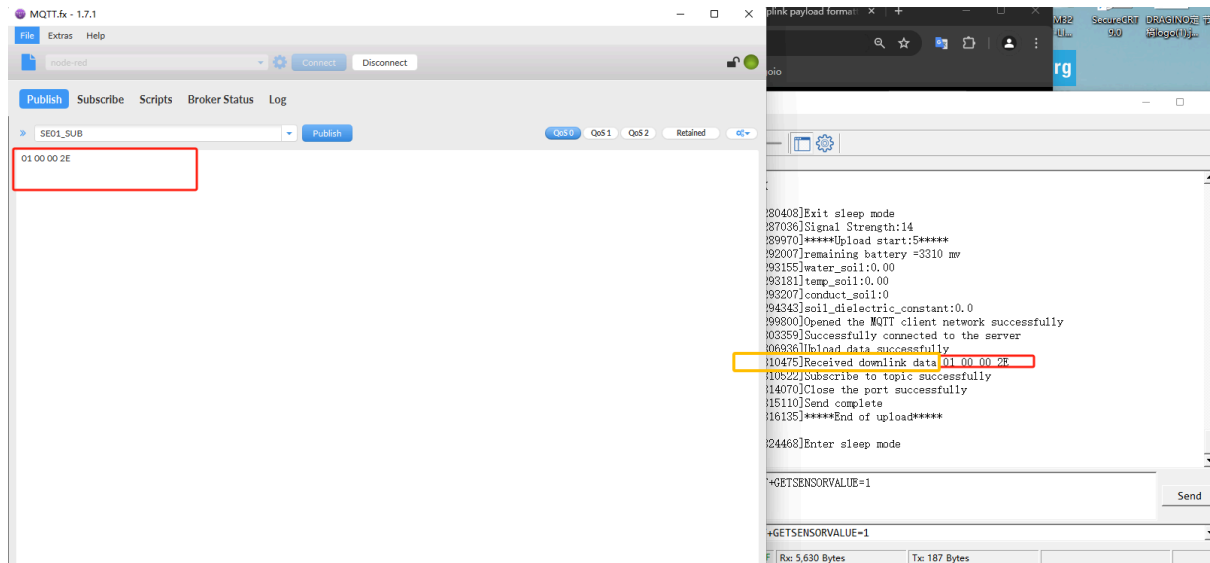


3. The downlink command can be successfully sent only when the downlink port is open.

The downlink port is opened for about 3 seconds after uplink packets are sent.

Therefore, when we see the node uplink packets in the **Subscribe** window, we need to immediately switch to the **publish** window to publish the **hex format** command.





Note: Users can edit the hex command in advance. When the node uplink, directly click the publish button several times to increase the success rate of command configuration.

5. FAQ

5.1 What is the usage of Multi Sampling and One Uplink?

The NB series has the feature for Multi Sampling and one uplink. See one of them

http://wiki.dragino.com/xwiki/bin/view/Main/User%20Manual%20for%20LoRaWAN%20End%20Nodes/SN50v3-NB_BN-IoT_Sensor_Node_User_Manual/#H2.5Multi-SamplingsandOneuplink

User can use this feature for below purpose:

1. **Reduce power consumption.** The NB-IoT transmit power is much more higher than the sensor sampling power. To save battery life, we can sampling often and send in one uplink.
2. Give more sampling data points.

3. Increase reliable in transmission. For example. If user set
- **AT+TR=1800** // The unit is seconds, and the default is to record data once every 1800 seconds (30 minutes, the minimum can be set to 180 seconds)
 - **AT+NOUD=24** // The device uploads 24 sets of recorded data by default. Up to 32 sets of record data can be uploaded.
 - **AT+TDC=7200** // Uplink every 2 hours.
 - this will mean each uplink will actually include the 6 uplink data (24 set data which cover 12 hours). So if device doesn't lost 6 continue data. There will not data lost.

5.2 Why the uplink JSON format is not standard?

The json format in uplink packet is not standard Json format. Below is the example. This is to make the payload as short as possible, due to NB-IoT transmit limitation, a standard Json is not able to include 32 sets of sensors data with timestamp.

The firmware version released after 2024, Mar will use change back to use Json format. Detail please check changelog.

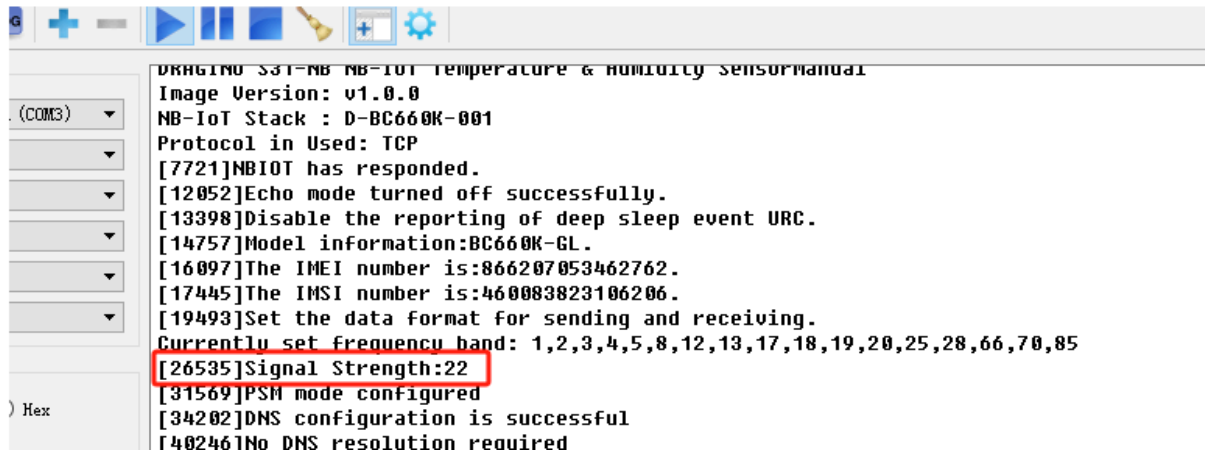
```
1  {
2    "IMEI": "80000000000000000000000000000000",
3    "Model": "D23-NB",
4    "temperature1": -18.6,
5    "temperature2": -409.5,
6    "temperature3": -409.5,
7    "battery": 3.33,
8    "signal": 28,
9    "1": {
10      -18.1,
11      -409.5,
12      -409.5,
13      2024/02/28 09: 01: 38
14    },
15    "2": {
16      -17.8,
17      -409.5,
18      -409.5,
19      2024/02/28 07: 51: 18
20    },
21  }
```

6. Trouble Shooting:

6.1 Checklist for debugging Network Connection issue. Signal Strenght:99 issue.

There are many different providers provide NB-IoT service in the world. They might use different band, different APN & different operator configuration. Which makes connection to NB-IoT network is complicate.

If end device successfully attached NB-IoT Network, User can normally see the signal strength as below (between 0~31)



```
DRAGINO S31-NB NB-IoT Temperature & Humidity Sensor module
Image Version: v1.0.0
NB-IoT Stack : D-BC660K-001
Protocol in Used: TCP
[7721]NB-IoT has responded.
[12052]Echo mode turned off successfully.
[13398]Disable the reporting of deep sleep event URC.
[14757]Model information:BC660K-GL.
[16097]The IMEI number is:866207053462762.
[17445]The IMSI number is:460083823106206.
[19493]Set the data format for sending and receiving.
Currently set frequency band: 1,2,3,4,5,8,12,13,17,18,19,20,25,28,66,70,85
[26535]Signal Strength:22
[31569]PSM mode configured
[34202]DNS configuration is successful
[40246]No DNS resolution required
```

If fail to attach network, it will shows signal 99. as below:

```
Protocol in Used: UDP
[6588]NB-IoT has responded.
[10919]Echo mode turned off successfully.
[12865]Model information:BC95-GV.
[14203]The IMEI number is:8677.....
[15551]The IMSI number is:9012.....
Currently set frequency band:5,8,3,28,20,1
[23247]The full frequency band is set successfully,and the module is being restarted...
Currently set frequency band:1,3,5,8,20,28
[29991]Set automatic network access successfully.
[31546]Signal Strength:99
[33076]Signal Strength:99
[34606]Signal Strength:99
[36136]Signal Strength:99
[37666]Signal Strength:99
```

When see this issue, below are the checklist:

- Does your SIM card support NB-IoT network? If SIM card doesn't not specify support NB-IoT clearly, normally it doesn't support. You need to confirm with your operator.
- Do you configure the correct APN? [Check here for APN settings](#).
- Do you lock the frequency band? This is the most case we see. [Explain and Instruction](#).
- Check if the device is attached to Carrier network but reject. (need to check with operator).
- Check if the antenna is connected firmly.

If you have check all above and still fail. please send console log files (as many as possible) to support@dragino.com so we can check.

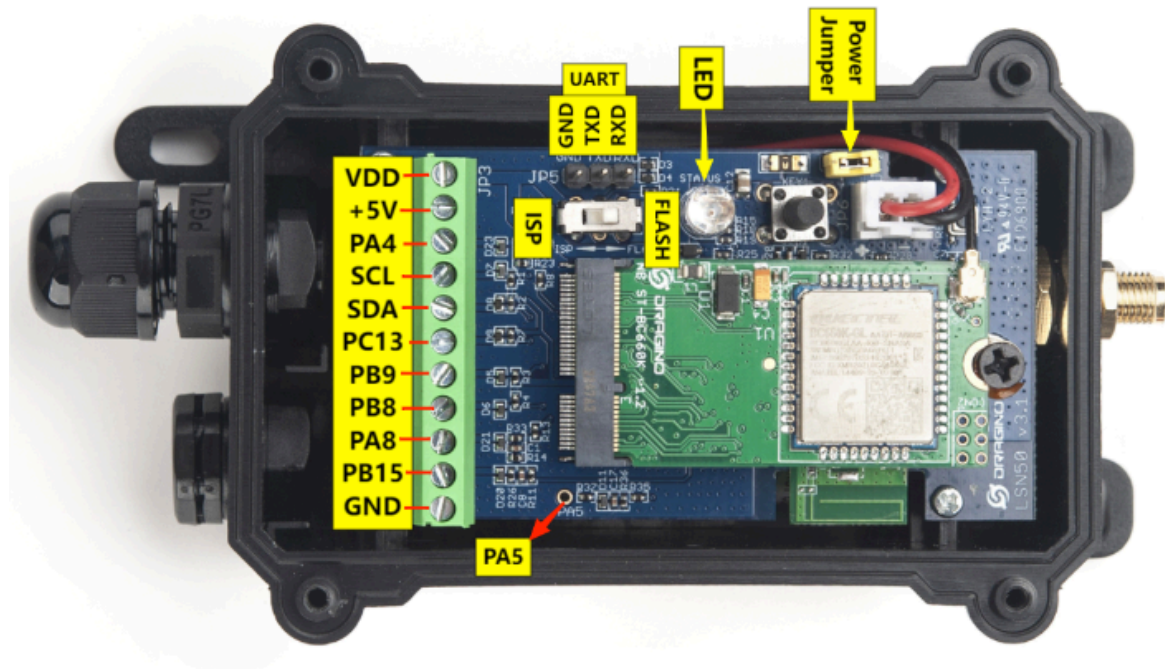
6.2 Issue: "NBIOT did not respond"

```
11:24:22.397 [44596]NBIOT did not respond.
11:24:24.315 [46530]NBIOT did not respond.
11:24:26.256 [48464]NBIOT did not respond.
11:24:28.196 [50398]NBIOT did not respond.
11:24:30.115 [52332]NBIOT did not respond.
11:24:32.127 [54266]NBIOT did not respond.
```

11:24:32.127 [54299]Restart the module...
11:24:39.181 [61332]No response when shutting down

This issue might due to initiate issue for NB-IoT module. In this case, please try:

- 1) Open Enclosure
- 2) Power off device by pull out the power on Jumper
- 3) Power on device by connect back the power jumper.
- 4) push reset button.



6.3 Issue: "Failed to readl MSI number"

[18170]Failed to read IMSI:1umber.
[20109]Failed to read IMSI numoer.
[22048]Failed to read IMSI number.
[29842]Restart the module...

Make sure that the SIM card is insert in correct direction and device is power off/on during insert. Here is reference link: [Insert SIM Card](#).

6.4 Why sometime the AT Command is slow in reponse?

When the MCU is communicating with the NB-IoT module, the MCU response of AT Command will become slower, it might takes several seconds to response.

```

[Rx][11:11:00.120] OK
[Rx][11:11:00.120]
[Rx][11:11:03.210] [60883]Signal Strength:26
[Tx][11:11:06.053] AT+TDC=60 Command Lost
[Rx][11:11:06.140] [63816]*****Upload start:1*****
[Rx][11:11:06.180] [63851]remaining battery =2978 mv
[Tx][11:11:10.339] AT+TDC=60 3 seconds to reponse
[Rx][11:11:13.229] [70891]distance:0
[Rx][11:11:13.830]
[Rx][11:11:13.830] OK
[Rx][11:11:13.830]
[Tx][11:11:13.839] AT+TDC=60 3 seconds to reponse
[Rx][11:11:16.399]
[Rx][11:11:16.399] OK
[Rx][11:11:16.399]
[Rx][11:11:18.409] [76057]Datagram is sent by RF
[Rx][11:11:19.459] [77091]Send complete
[Rx][11:11:20.469] [78115]*****End of upload*****
[Rx][11:11:20.500]

```

6.5 What is the Downlink Command by the NB device?

UDP:

Its downlink command is the same as the AT command, but brackets are required.

Example:

```
{AT+TDC=300}
```

MQTT:

Json:

The Json format in MQTT mode needs to be configured with all commands.

If you have configurations that need to be changed, please change them in the template below.

Template:

```

{
  "AT+SERVADDR":"119.91.62.30,1882",
  "AT+CLIENT":"JwcXKjQBNhQ2JykDDAA5Ahs",
  "AT+UNAME":"username dragino",
  "AT+PWD":"password dragino",
  "AT+PUBTOPIC":"123",
  "AT+SUBTOPIC":"321",
  "AT+TDC":"7200",
  "AT+INTMOD":"0",
  "AT+APN":"NULL",
  "AT+5VT":"0",
  "AT+PRO":"3,5",
  "AT+TR":"900",
  "AT+NOUD":"0",
  "AT+CSQTIME":"5",
  "AT+DNSTIMER":"0",

```

```
"AT+TLSMOD": "0,0",  
"AT+MQOS": "0",  
"AT+TEMPALARM1": "0",  
"AT+TEMPALARM2": "10",  
"AT+TEMPALARM3": "0"  
}
```

Hex:

MQTT's hex format. Since many commands need to support strings, only a few commands are supported.

The supported commands are consistent with LoRaWAN's hex commands.

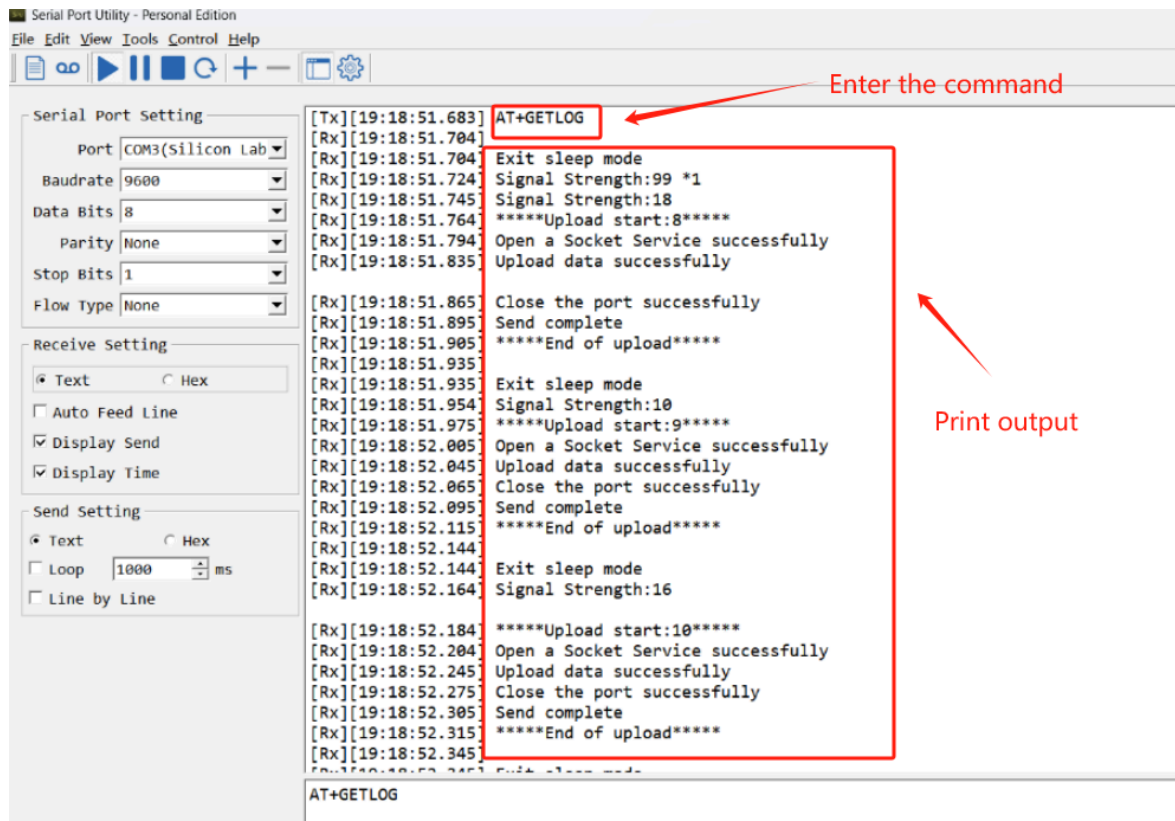
Please refer to the following link to obtain the hex format:

<http://wiki.dragino.com/xwiki/bin/view/Main/End%20Device%20AT%20Commands%20and%20Downlink%20Command/>

6.6 How to obtain device logs?

- **AT Command: AT +GETLOG**

This command can be used to query upstream logs of data packets.



6.7 How to find the AT Command Password if lost?

Why can't the password access AT command after upgrade(-NB)?

Because the new version of -NB firmware has updated the factory reset function, users can choose to restore all parameters to factory Settings, or keep the password to restore the rest of the parameters to factory Settings.

This update changes the password address of the firmware, so the password will be invalid after the customer upgrades from the old version of firmware (without FDR1 function) to the new version of firmware (with FDR1 function).

Two different restore factory Settings configurations.

AT command:

- **AT+FDR** // Reset Parameters to Factory Default.
- **AT+FDR1** // Reset parameters to factory default values except for passwords.(new)

Version Confirmation

We are now dividing the **old firmware**(without FDR1 function) with the **new firmware**(with FDR1 function) by whether it contains FDR1 functionality. Please refer to the table:

General Model	Firmware version (without FDR1 function)	Firmware version (with FDR1 function)
CPL03-NB, S31-NB, SN50V3-NB, TS01-NB, D20-NB, DS03A-NB, DDS04-NB, DDS45-NB, DDS20-NB, DDS75-NB, LDS12-NB, LDS40-NB, LMS01-NB, MDS120-NB, MDS200-NB, SE01-NB, SPH01-NB;	Before V1.2.1	After V1.2.1 (including V1.2.1)
WL03A-NB, SDI-12-NB;	Before V1.0.2	After V1.0.8 (including V1.0.2)
SW3L-NB, PS-NB;	Before V1.0.5	After V1.0.5 (including V1.0.5)
RS485-NB	Before V1.0.8	After V1.0.8 (including V1.0.8)

UART connection and firmware update methods

Users can query passwords only using the UART interface via the STM32CubeProgrammer.

See [UART Connection](#).

update firmware through UART TTL interface : [Instruction](#).

query the password via STM32CubeProgrammer

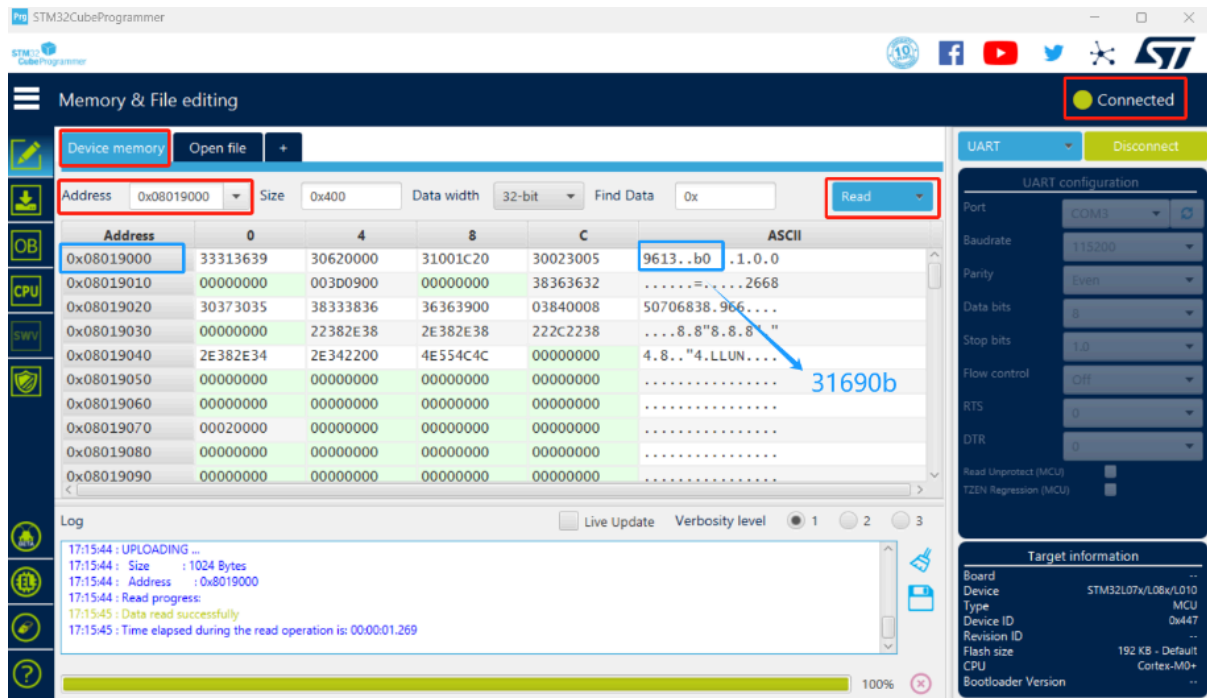
Users can use the password address to query the password through STM32CubeProgrammer.

- The password address for old firmware(without FDR1 function) : **0x08019000**
- The password address for new firmware(with FDR1 function) : **0x08025D00**

Notice: The password can only be queried after the firmware is run once.

Procedure for querying the password(old firmware):

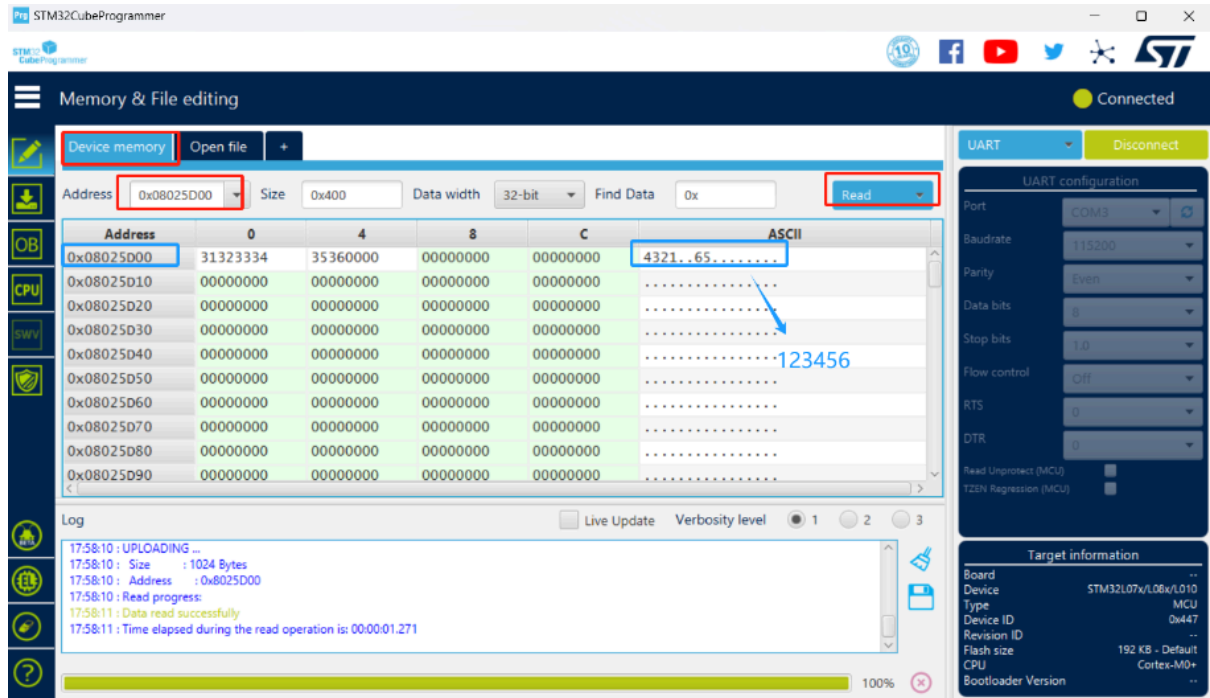
- After the firmware upgrade is complete, switch back to the **FLASH** and reset the node to **run the firmware once**.
- Then place the switch at the **ISP** and connect to the STM32CubeProgrammer (same as when burning the firmware).
- Click "Device memory", enter **0x08019000** in "Address", and click "Read"
- Find the 0x08019000 address field and then read the current password as shown in the screenshot below.



Procedure for querying the password(new firmware):

Refer to [the old and new firmware division](#) above, and run the firmware first after updating the firmware.

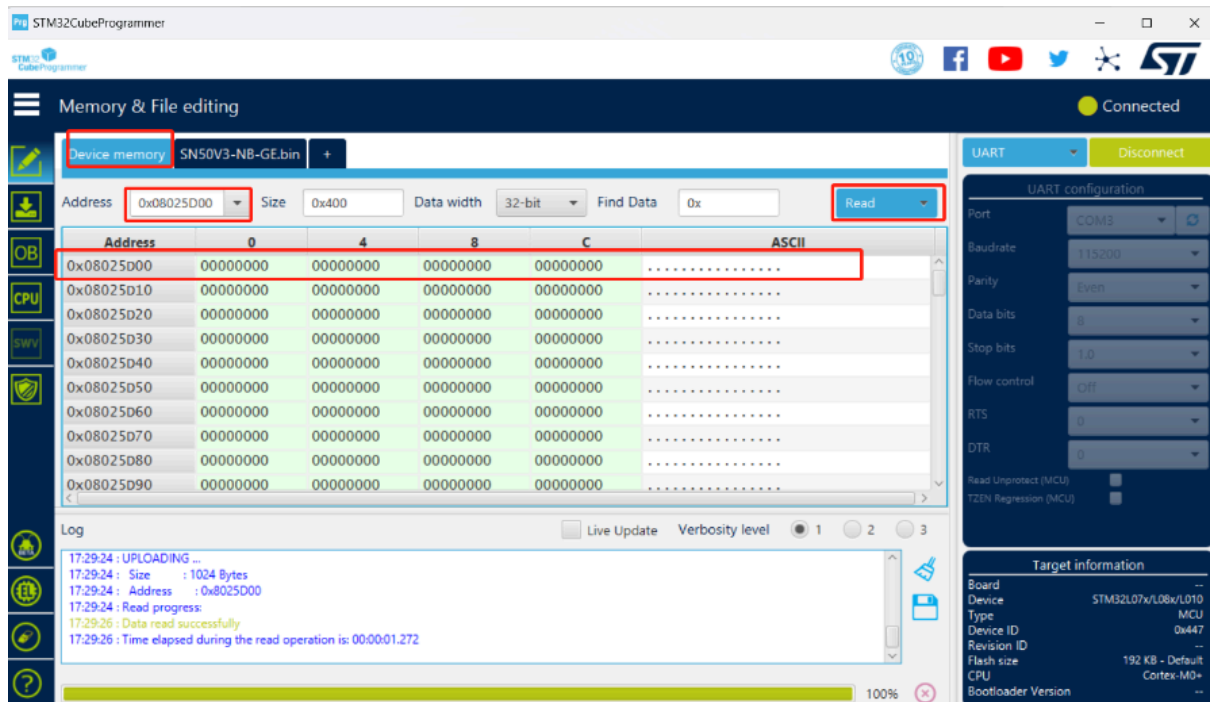
- After the firmware upgrade is complete, switch back to the **FLASH** and reset the node to **run the new firmware once**.
- Then place the switch at the **ISP** and connect to the STM32CubeProgrammer (same as when burning the firmware).
- Click "Device memory", enter **0x08025D00** in "Address", and click "Read"
- Find the 0x08025D00 address field and then read the current password as shown in the screenshot below.



Special case

If the user has never changed the password manually, the user cannot find the valid password through the above two password addresses. In this case, the valid password is still the original password on the node box label (**AT +PIN**).

Invalid query screenshot example:



The screenshot displays the STM32CubeProgrammer application window. The main area is titled "Memory & File editing" and shows a memory map for the "Device memory" (SNS0V3-NB-GE.bin). The address range is 0x08019000 to 0x08019090, with a size of 0x400. The data width is set to 32-bit. The "Read" button is highlighted in red. Below the memory map, a table shows the progress of the read operation, with the first row highlighted in red:

Address	0	4	8	C	ASCII
0x08019000	08018B5C	08018B60	00000008	000007D0	\>..`>.....D...
0x08019010	08015478	0801547C	08015471	08018B5C	]T...T...qT...\\...
0x08019020	08018B60	00000009	000001F4	08015EB5	`>.....ô...µA...
0x08019030	08015EE5	08015EA9	08018B5C	08018B60	âA..ôA..\\>..`>..
0x08019040	0000000A	000001F4	08015F35	08015F65	...ô...5...e...
0x08019050	08015F29	08018B5C	08018B60	0000000B	)...\\>..`>.....
0x08019060	0000012C	080159CD	08015A1D	08015969	,...IY...Z...iY..
0x08019070	08018B5C	08018B60	0000000C	0000012C	\\>..`>.....
0x08019080	0801569D	08015965	08015671	08018B5C	.V...eY...qV...\\>..
0x08019090	08018B60	0000000D	0000012C	08015419	`>.....T...

The "Log" window at the bottom shows the following messages:

```

17:30:02 : UPLOADING ...
17:30:02 : Size : 1024 Bytes
17:30:02 : Address : 0x8019000
17:30:02 : Read progress:
17:30:03 : Data read successfully
17:30:03 : Time elapsed during the read operation is: 00:00:01.272
    
```

The "UART configuration" panel on the right shows the following settings:

- Port: COM3
- Baudrate: 115200
- Parity: Even
- Data bits: 8
- Stop bits: 1.0
- Flow control: Off
- RTS: 0
- DTR: 0
- Read Unprotect (MCU): ☐
- TZEN Regression (MCU): ☐

The "Target information" panel on the right shows the following details:

- Board: STM32L07x/L06x/L010
- Device: MCU
- Type: Cortex-M0+
- Device ID: 0x447
- Revision ID: ..
- Flash size: 192 KB - Default
- CPU: Cortex-M0+
- Bootloader Version: ..