Quick Start Guide

Prerequisite

What do you need?

Before going through each and every step in the installation guide of the RAK7431 WisNode Bridge Serial, make sure to prepare the necessary items listed below:

Hardware Tools

1. RAK7431 WisNode Bridge Serial
2. Micro USB Cable
3. Gateway in Range, for Testing
4. A Windows/Mac OS/Linux Computer

Software Tools

- RAK Serial Port Tool
- MQTTfx Tool

Product Configuration

Typical Network Application

RAK7431 converts data from the RS485 protocol into LPWAN wireless messages and delivers it to a cloud server through an LPWAN gateway. Cloud servers can also proactively send data to RAK7431 for two-way data transmission. Using the RAK7431, you can convert data from a conventional RS485 wired network to a wireless network.

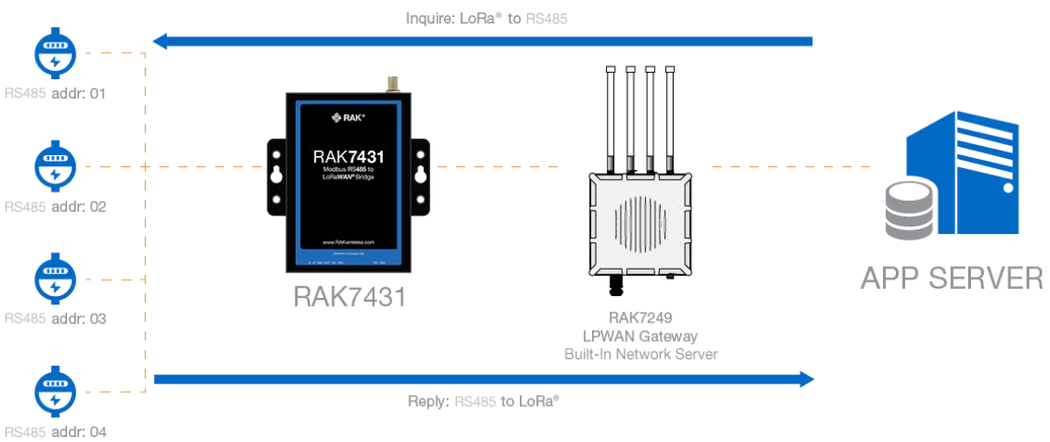

Figure 1: Example communication with RS485 enabled devices

Connect the RAK7431 to the Sensor

Power Interface Configuration

The RAK7431 device can be powered either by:

- DC (VIN/GND) terminals
- Micro USB.

The DC screw terminals are supporting 8 to 48 VDC. The Micro USB port can be used to power the RAK7431, up to 5 V / 500 mA DC. At the same time, the USB port is used as the configuration port for the device. Using the USB cable to connect the RAK7431 to a computer's USB port, you can import your configuration settings.

> 📝 **NOTE**
>
> The Micro USB port can be used only for powering the device. It cannot provide power to VOUT and power other devices in the RS485 network.

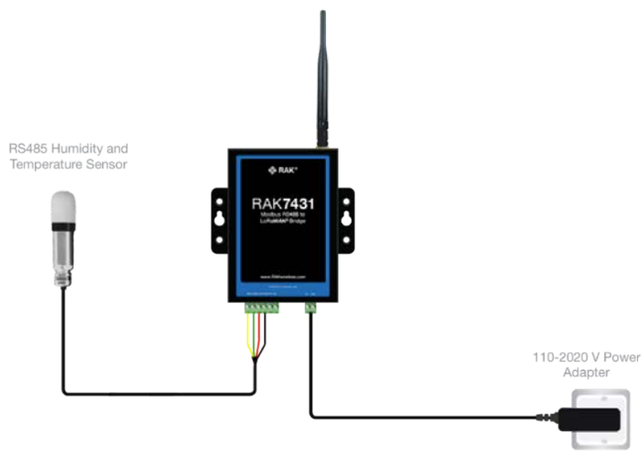

Figure 2: RAK7431 bridge with connected sensor and power supply

Data Interface Configuration

The RAK7431 - RS485 serial interface can support up to **16 RS485 devices**. VOUT on the data interface can supply external power to the RS485 connected devices (only when the device is powered from the DC input). The VOUT output voltage is the same as the DC input voltage VIN.

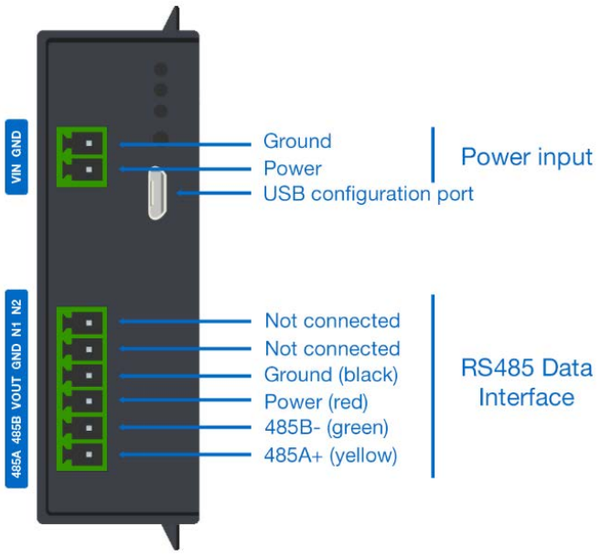

Figure 3: RAK7431 Interface pin definition

Gateway Connection Settings

In this section, the **RAK7431 WisNode Bridge Serial** shall be connected into the RAKwireless Gateway. For this demonstration, a **RAK7249 WisGate Edge Max** shall be used. Listed below are the requisites for this section.

- RAK Serial Port Tool - used to configure the RAK7431 WisNode Bridge Serial
- Web Management Platform Documentation - guide on how to configure the RAK7249 WisGate Edge Max

Gateway Configuration

Set-up the Built-in Network Server

1. Sign in to the gateway by following the Accessing the Web Management section of the WEB Management Platform documentation.

2. Setup the RAK7249 WisGate Edge Max using its Built-in Network Server by following this guide.

Adding Application

1. To enter the application configuration interface click: **LoRaNetwork > Application**. Enter a name for the application and click the **Add** button.

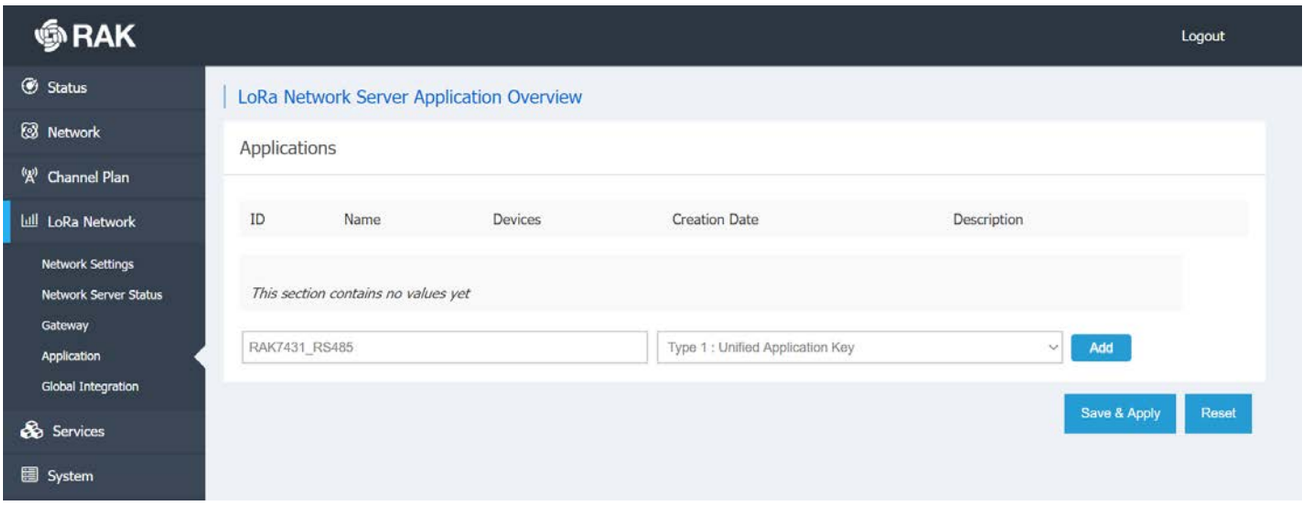

Figure 4: Create Application in the Buil-In Network Server

2. Turn on the **Auto Add LoRa Device** slider.

3. Generate **Application EUI** and **Application Key** by pressing the generate icon marked in the image below.

> 📝 **NOTE**
>
> The description is optional.

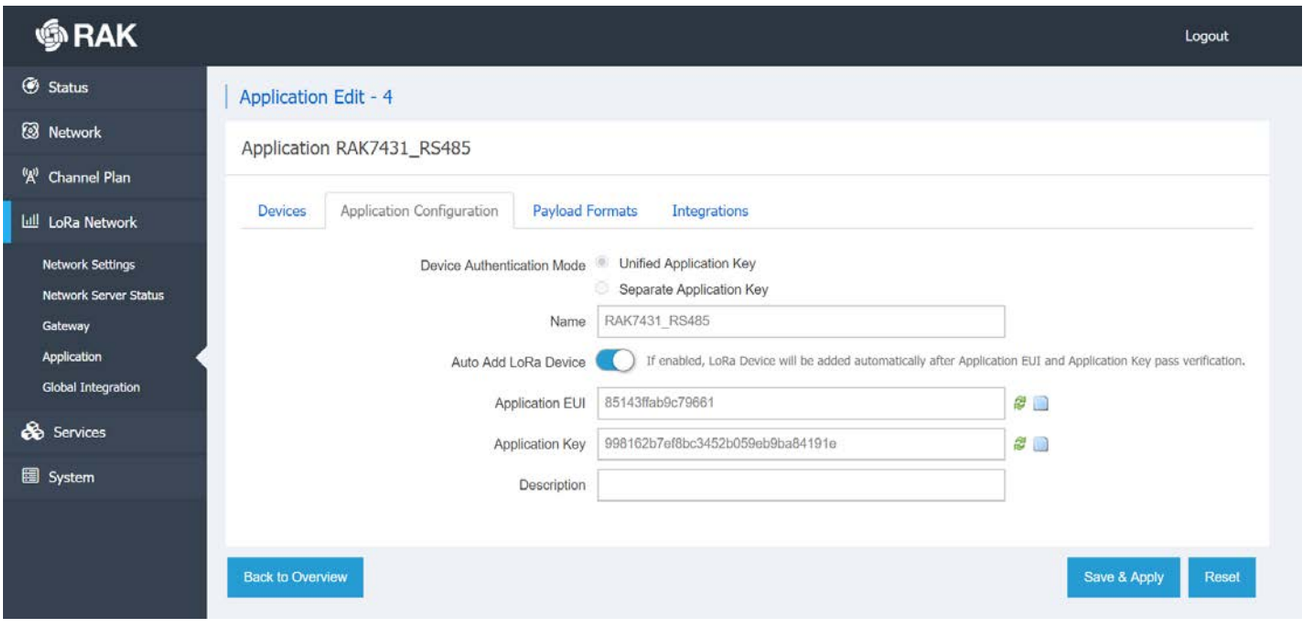

Figure 5: Registering an application

4. After which, press **Save & Apply**.

5. You will be returned to the Application page. Select **Edit** on the created application.

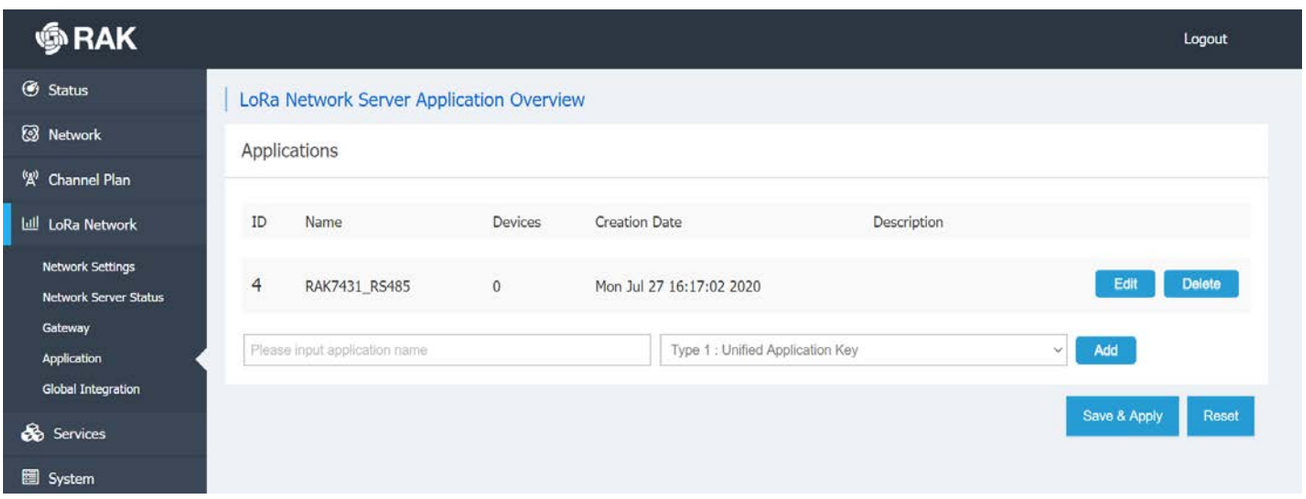

Figure 6: Application list

6. Enter the **Device EUI** and press **Add**.

> 📝 **NOTE**
>
> The RAK7431 Device EUI can be seen at the label on the back

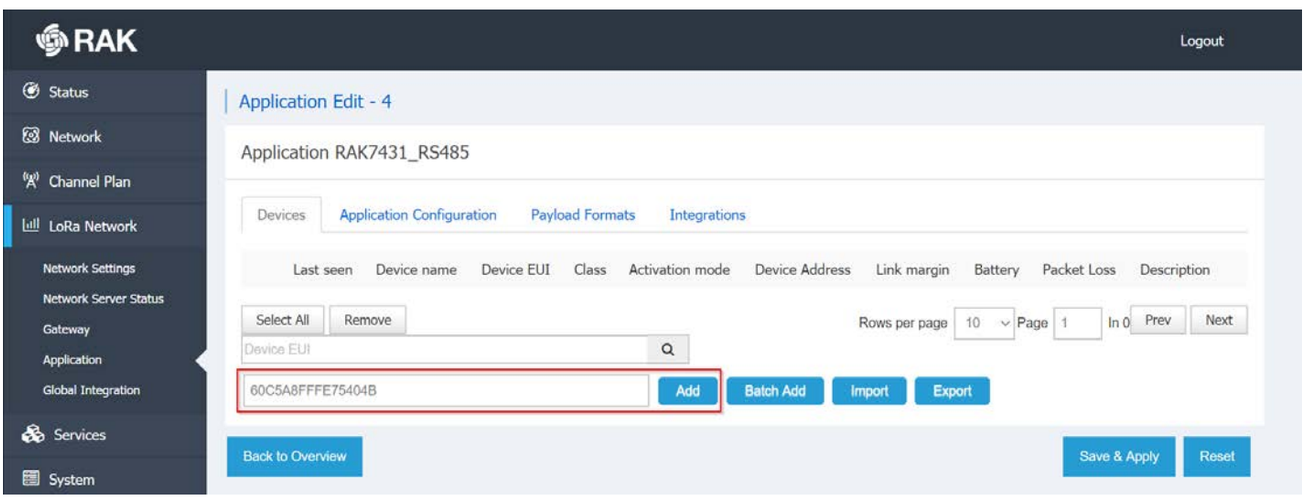

Figure 7: Adding the RAK7431

7. On the next page, select the settings provided below:

- **LoRaWAN Class**: C
- **Join Mode**: OTAA
- **Description**: Optional

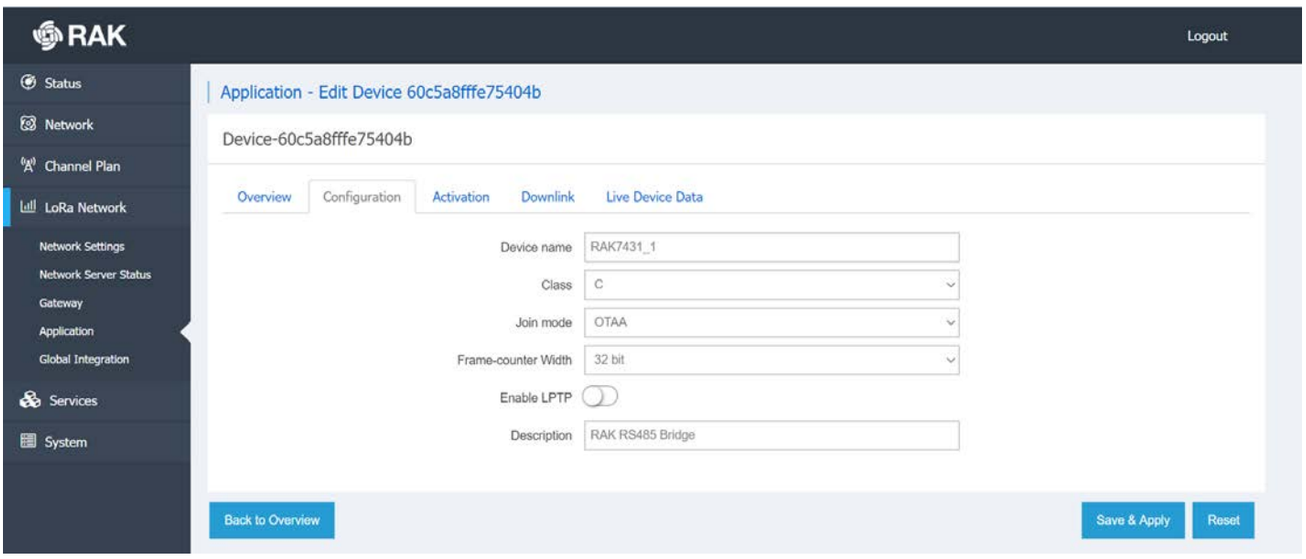

Figure 8: Adding the RAK7431 to the Built-In Server

RAK7431 Configuration

Connect the RAK7431 to your Network

1. Connect the RAK7431 to a computer using the Micro USB cable.

2. Open the RAK Serial Tool and select the correct COM port. The default baud rate is **115200**.

3. After selecting, press **Open**.

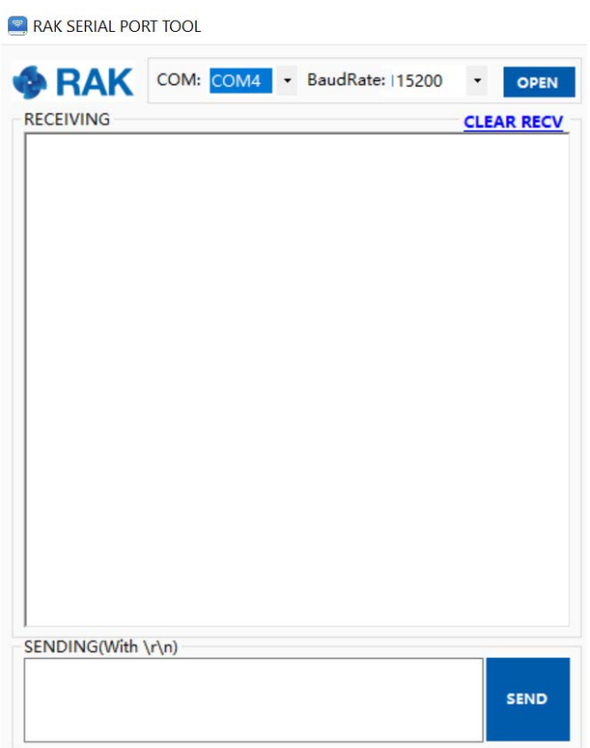

Figure 9: RAK Serial Tool

- To set up the Device EUI, run the command:

```sh
AT+DEVEUI=<Device EUI>
```

- To check the Device EUI run:

```sh
AT+DEVEUI
```

- To set up the Application EUI run the command:

```sh
AT+APPEUI=<application EUI>
```

- To set up the Application Key run the command:

```sh
AT+APPKEY=<application Key>
```

- To check the previously configured Application EUI and Key, run the commands:

```sh
AT+APPEUI
```

```sh
AT+APPKEY
```

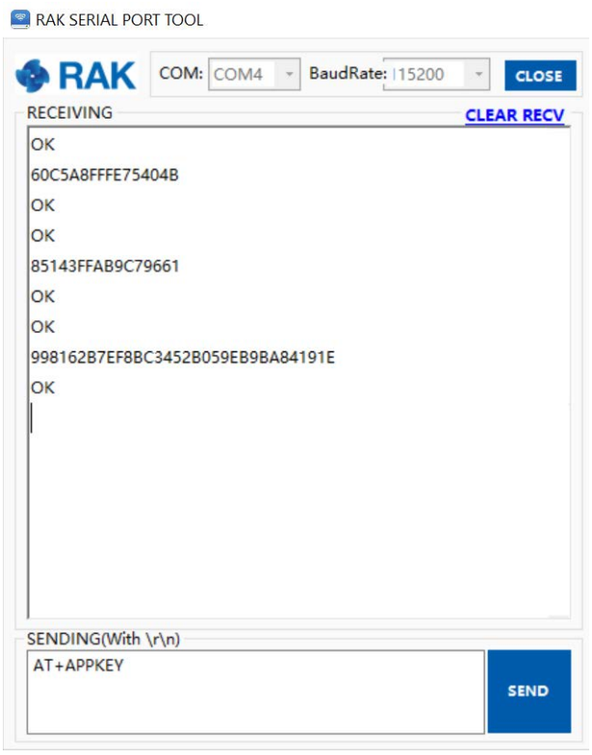

Figure 10: Configuring the RAK7431

Set the Frequency Region

The node supports the following Regional Frequencies:

- EU433
- CN470
- CN470ALI
- RU864
- IN865
- EU868
- US915
- AU915
- KR920
- AS923

For this demonstration, EU868 shall be used. To set the desired regional frequency band use the command:

```sh
AT+REGION=EU868
```

> 📝 **NOTE**
>
> The regional frequency settings need to be consistent with the RAK commercial gateway supported band.

Data Serial Port Rate Setting

> 📝 **NOTE**
>
> The baud rate setting needs to be consistent with the baud rate of the sensor, which is **9600**.

The AT command for execution is:

```sh
AT+BAUDRATE=9600
```

Operating and Activation Mode Settings

1. Supported operating modes are two: **Class A** and **Class C**. To set the operating mode (Class C in this case), you need to execute the AT command:

```sh
AT+CLASS=C
```

> 📝 **NOTE**
>
> The changes take effect as soon as they are made.

2. Activation mode supports the following two modes: **ABP** and **OTAA**. To set the activation mode (OTAA in this case), you need to execute the AT command:

```sh
AT+JOINMODE=OTAA
```

3. **Restart** is needed for the modification to take effect. To restart the RAK7431, execute the command:

```sh
AT+RESTART
```

4. If everything is configured right, after the execution of the restart command this output pops up in the RAK Serial Tool:

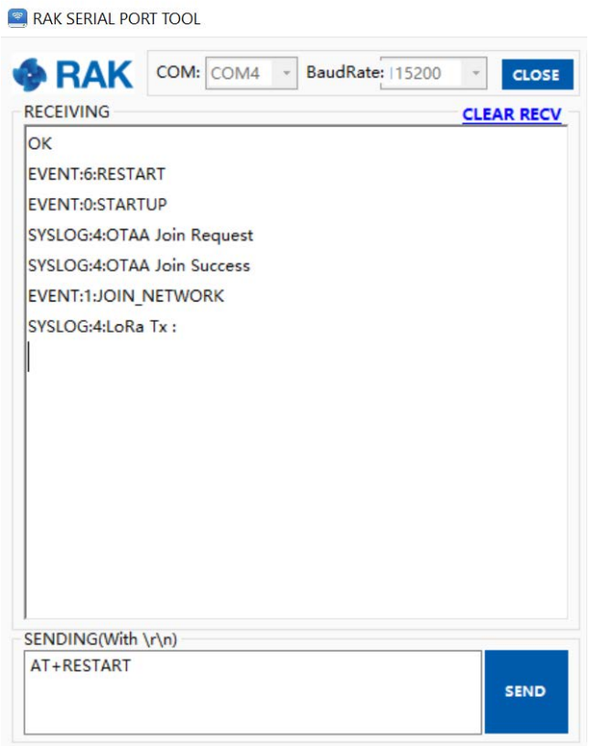

Figure 11: RAK7431 Successful Join

Configure RAK7431 Working Modes

Data Transparent Mode

When the RS485 data interface works in Modbus mode, the data encapsulation format can be divided into two types: **transparent mode** and **non-transparent mode.**

- In **transparent mode**, the Modbus execution instruction response data (data, received by the node) will be directly forwarded through the LoRaWAN network.
- In the **non-transparent mode**, the Modbus execution instruction response data (data, received by the node) will be encapsulated in the message header according to the Modbus protocol, and then transmitted to the server through LoRaWAN.

> 📝 **NOTE**
>
> The non-transparent mode is the default one.

Enter the following AT command in the RAK Serial Tool to change the mode:

```sh
AT+TRANSPARENT=n
```

n	Condition
0	transparent mode is turned off
1	it is turned on

> 📝 **NOTE**
>
> The change takes effect immediately after modification.

Scheduled Polling Function

When the device works in MODBUS mode, it supports the scheduled polling function.

This means that the device will perform a polling operation every given period (polling cycle). During polling, the device will send the pre-added MODBUS instructions in turn and forward the corresponding response data through the LoRaWAN network.

The device turns on the scheduled polling by default. The AT command for this is:

```sh
AT+ENABLEPOLL=n
```

n	Condition
0	turns scheduled polling off
1	turns it on

> 📝 **NOTE**
>
> The modification takes effect after restart.

Figure 12: Scheduled polling example

Scheduled Polling Cycle

This command sets/reads the scheduled polling cycle. This command only works if scheduled polling is enabled. The modification takes effect after the next polling cycle or a restart.

Example: To set the polling cycle to 60 seconds, use this command:

```sh
AT+POLLPERIOD=60
```

RAK7431 supports polling mode, which stores up to 32 query instructions at a maximum length of 128 bytes per instruction. Polling intervals and wait times can be adjusted as needed. RAK7431 converts the data returned by the RS485 node into a LoRaWAN message, which can be sent to the LoRaWAN gateway as is or encapsulated. In transparent mode, the data for the RS485 is sent in the payload of the LoRa message as is, and in non-transparent mode, the data of RS485 is encapsulated in the LoRa message with a header and validation.

Add Polling Instructions

To add polling instruction, execute the AT command:

```sh
AT+ADDPOLL=<n>:<xxxx>
```

Parameter	Description	Value Range
n	polling instruction ID	1 to 127
xxxx	polling instruction content; hexadecimal string	128 bytes max

According to the temperature and humidity register address of the temperature and humidity sensor in the example and the RS485 address, the polling instruction should be:

```sh
AT+ADDPOLL=1:010300000002C40B
```

Example: If you have added multiple RS485 temperature and humidity sensors, continue to increase the polling instructions based on the RS485 address and register address, for example:

- RS485 Temperature and humidity sensor addr: 01, Polling 1: 010300000002C40B
- RS485 Temperature and humidity sensor addr: 04, Polling 2: 040300000002C45E
- RS485 Temperature and humidity sensor addr: 08, Polling 3: 080300000002C492
- RS485 Temperature and humidity sensor addr: 0F, Polling 4: 0F0300000002C525

You will need to increase the polling instruction by the following AT commands:

```sh
AT+ADDPOLL=1:010300000002C40B
```

```sh
AT+ADDPOLL=2:040300000002C45E
```

```sh
AT+ADDPOLL=3:080300000002C492
```

```sh
AT+ADDPOLL=4:0F0300000002C525
```

The RAK7431 sends an instruction to the sensor every 1 minute to obtain temperature and humidity data, and the following is the result of 3 consecutive scheduled polls:

- **DTU Tx**: The polling instruction sent to the Sensors over RS485 Data Interface
- **DTU Rx**: The sensor data received.
- **LoRa Tx**： Send the received data through a LoRaWAN network.

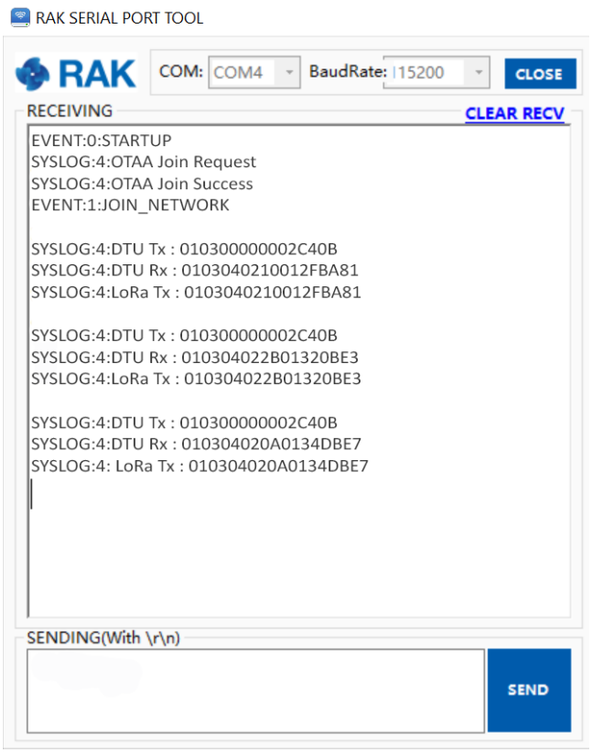

Figure 13: Data in transparent mode

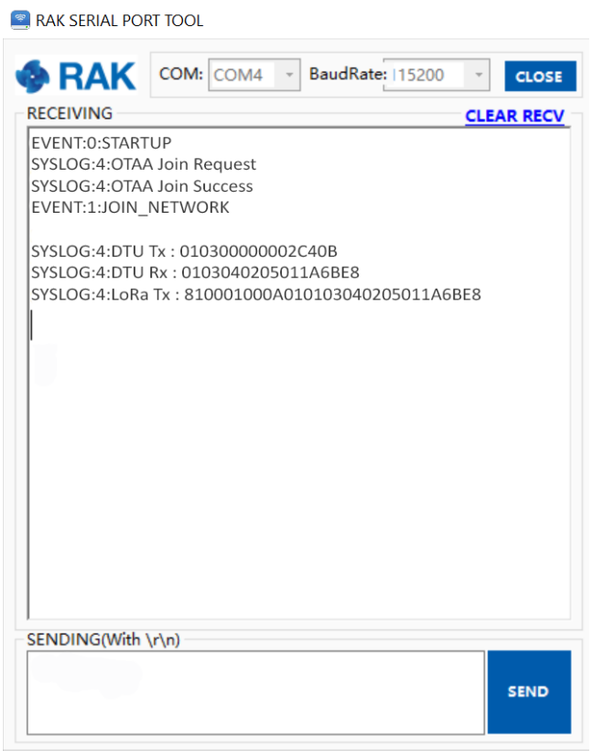

Figure 14: Data in non-transparent mode

- **Humidity calculation**: hex is 0210, the decimal is 528, converted humidity is 52.8% RH.
- **Temperature calculation**: hex is 012F, the decimal is 303, converted temperature is 30.3 ℃.

MQTT Subscribe to Data Server

To better demonstrate the functionality we will use the Application Server Integration feature to subscribe to the Built-In Network Server Topics, using the MQTT client, to obtain data and send instructions to the RAK7431.

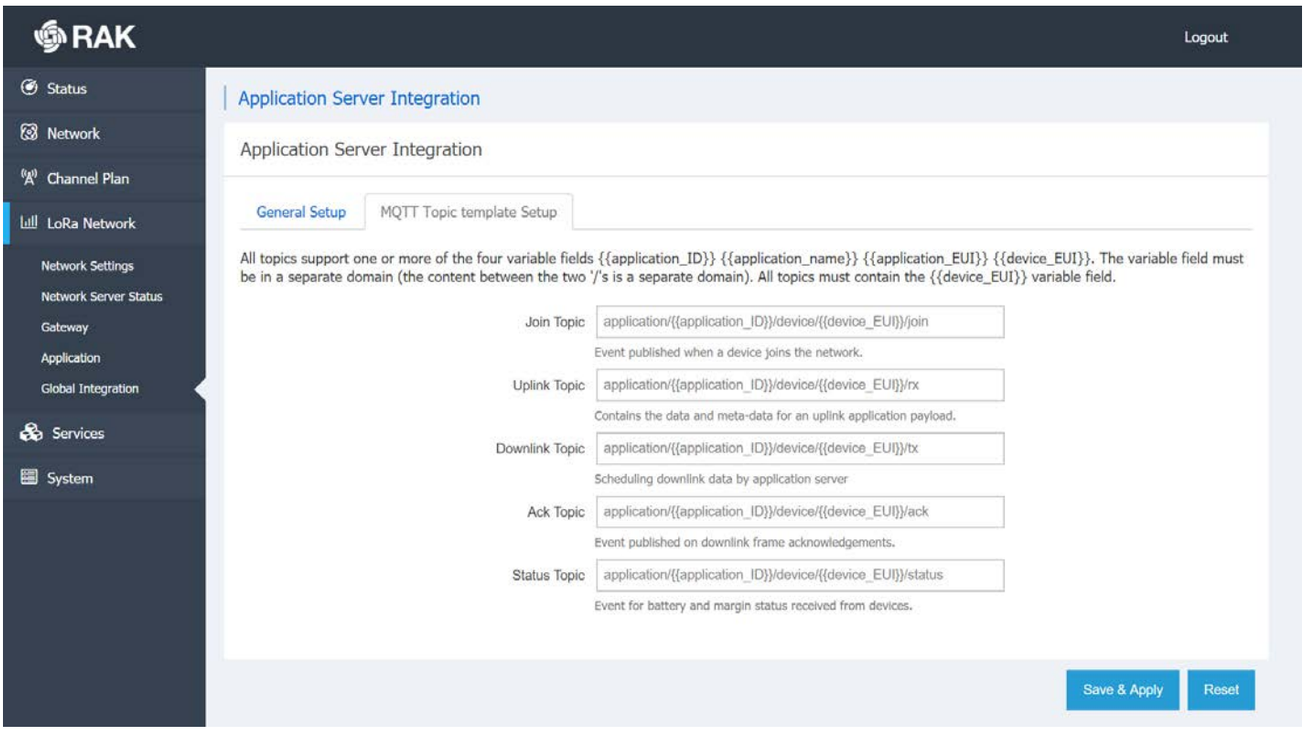

Figure 15: Gateway MQTT Topic Templates

To communicate with the MQTT bridge in the gateway we need to use MQTT Topic Templates.

MQTT Topic Configuration:

```sh
Application/{{application_ID}}/device/{{device_EUI}}/join
Application/{{application_ID}}/device/{{device_EUI}}/rx
Application/{{application_ID}}/device/{{device_EUI}}/tx
Application/{{application_ID}}/device/{{device_EUI}}/ack
Application/{{application_ID}}/device/{{device_EUI}}/status
```

1. Download and install MQTTfx tool to read the topics and send data to the gateway and node.

2. After installation, the MQTT Client must be configured. Select **local mosquitto** from the drop-down list and click the **edit connection profiles** icon marked in the image below to open the settings page.

Figure 16: MQTT.fx Client

3. On the next window, input the **Broker Adress** and **Broker Port**. If the Client ID is empty press **Generate**. Then click **OK**.

- **Broken Address**: Address of MQTT server – the gateway IP.
- **Broker Port**: Consistent with MQTT Broker Port set by the gateway - by default 1883.

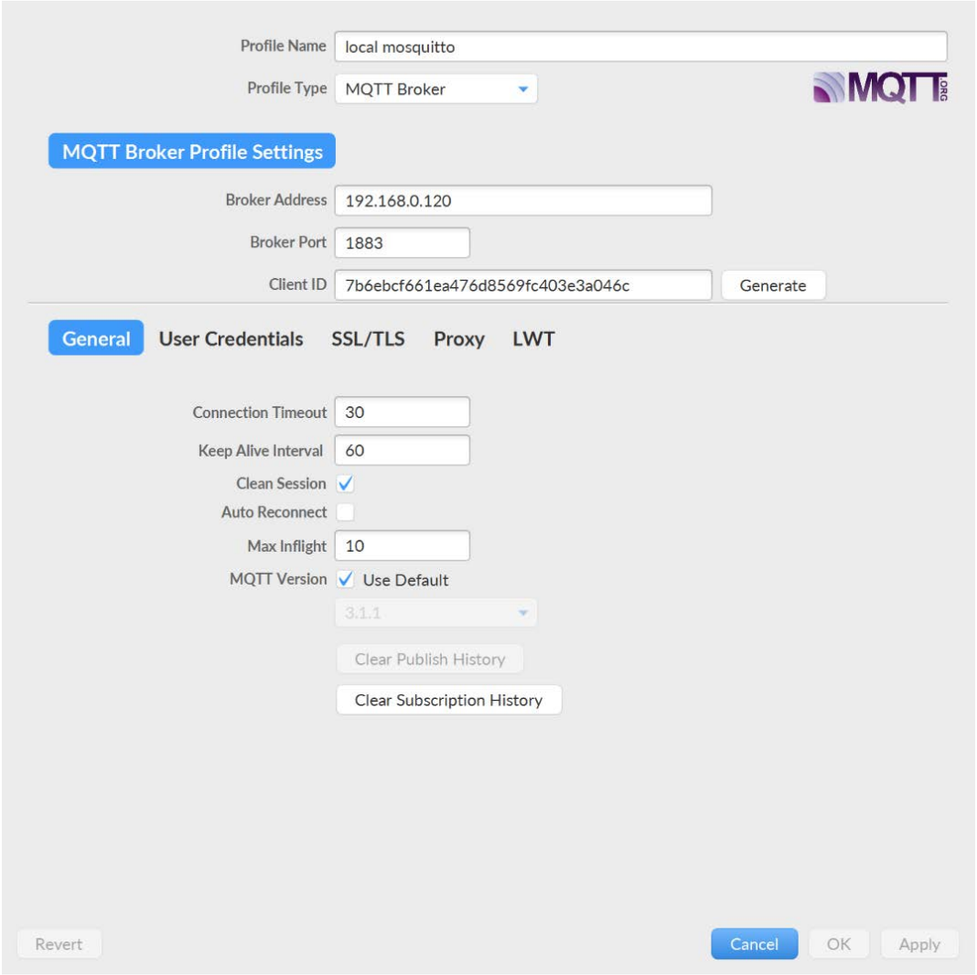

Figure 17: MQTT.fx settings

4. Click on the **Connect** button. The green dot indicates that the connection is successfully subscribed to the MQTT Broker.

Figure 18: MQTT.fx connected successfully

- If we want to receive all data from the MQTT Bridge, we can use the wildcard character #.

5. Choose the **Subscribe tab**, enter the wildcard and press **Subscribe**.

Figure 19: Subscribing to MQTT Broker with wildcard

- If the node sends data, the MQTT client will display it as it is subscribed to the topic.

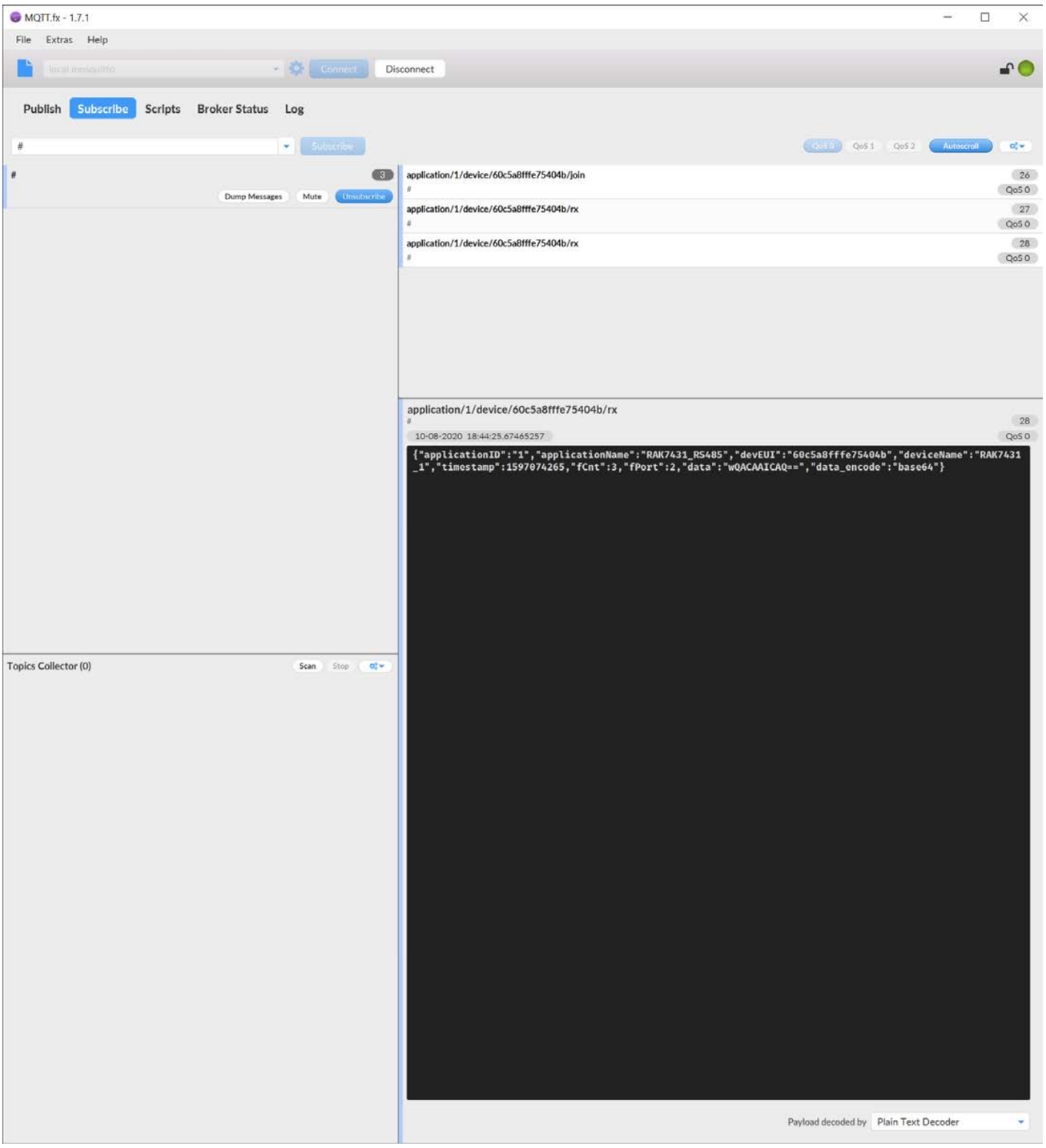

Figure 20: Subscribed topic data

- Notice that the data field is in **base64** format, which has to be converted to hex string to be useful. We can change the data format from the built-in server settings.

6. This is done by going to **Gateway>Application>Integrations>Data Encode/Decode Type** and chose **HEX String** form the drop-down menu. Press **Save & Apply**.

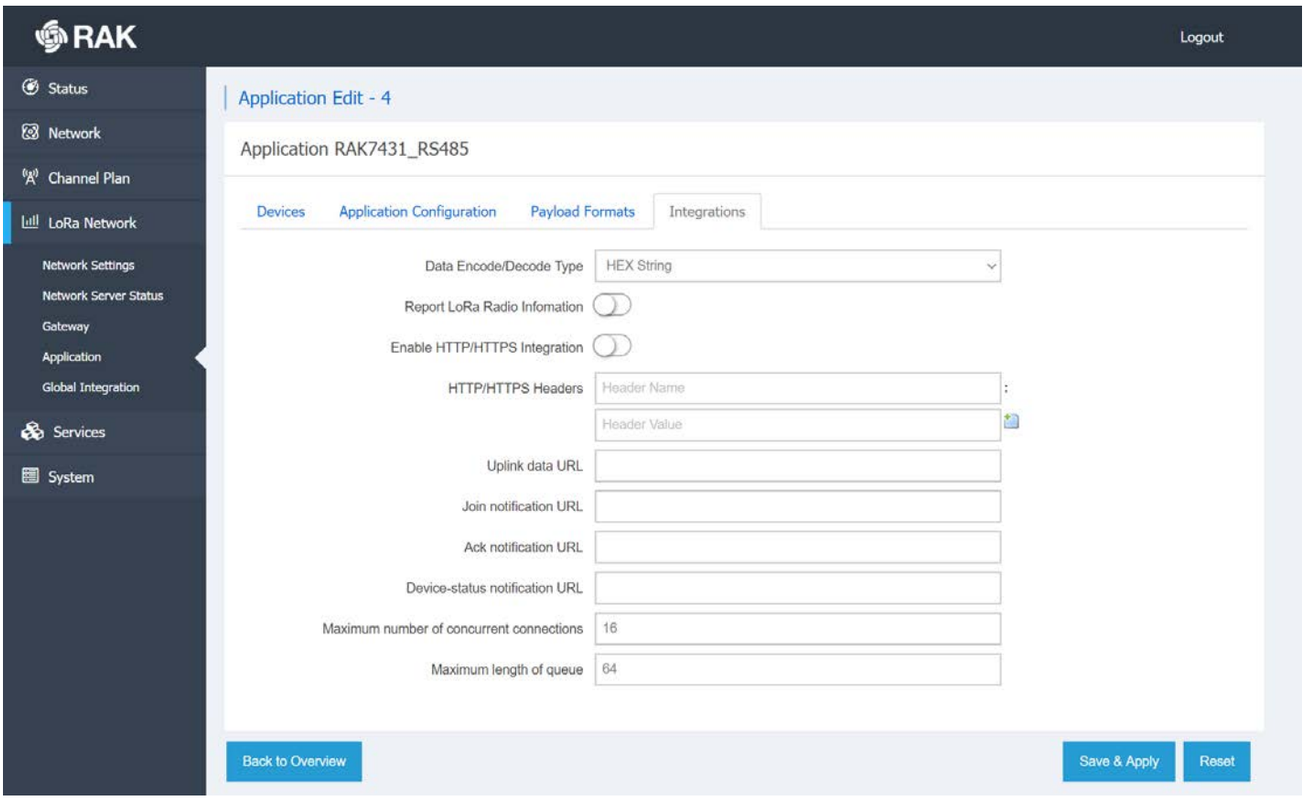

Figure 21: Change the Data Encode/Decode Type

- Now, all received data will be in HEX String.

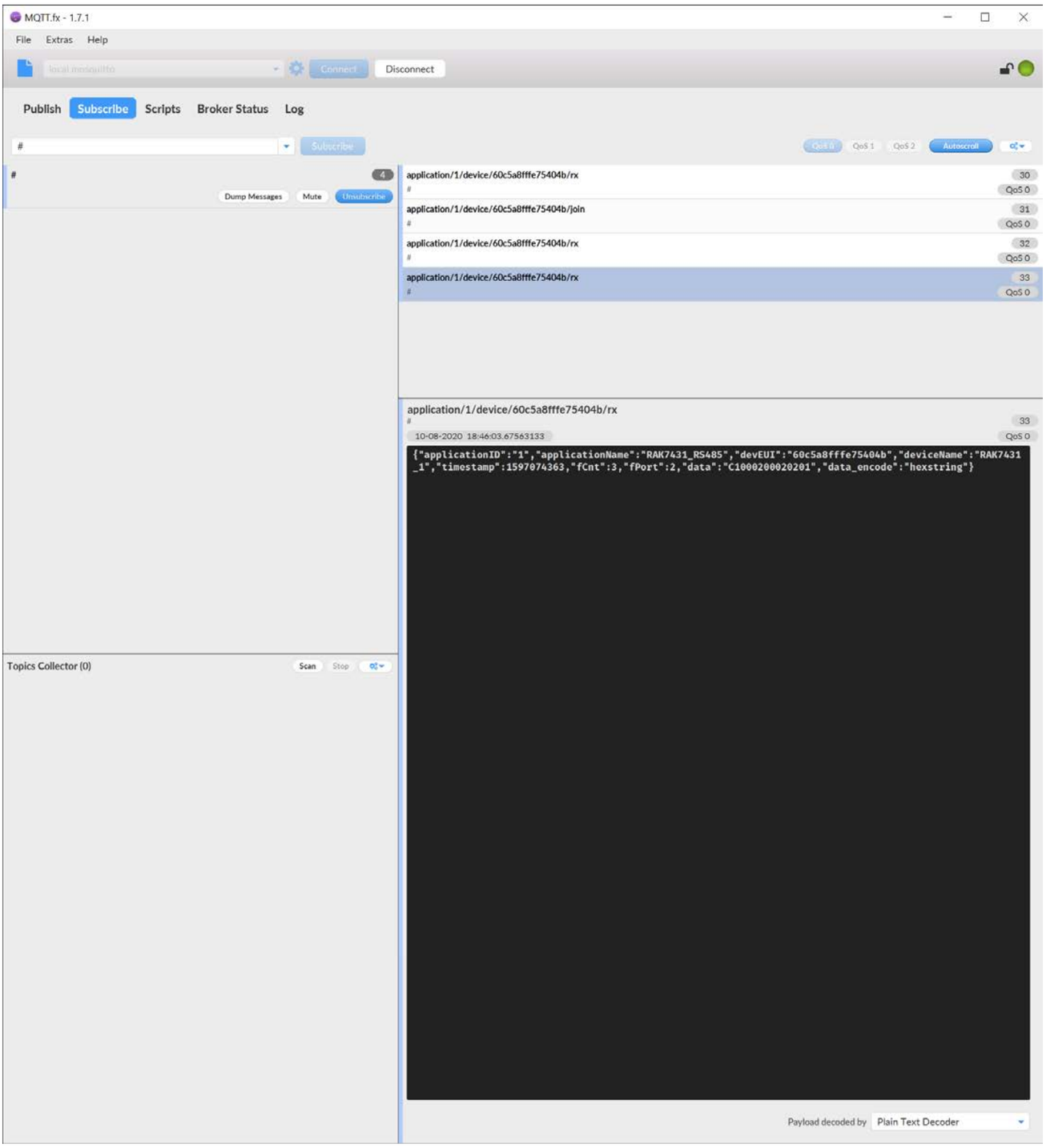

Figure 22: Received data field in HEX format

RAK7431 Remote Control and Configuration via MQTT.fx

To remotely control the RAK7431 you need to publish messages to the **Gateway's Network Server MQTT "TX" topic**.

Add a Scheduled Polling Task List

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA	
0x03	2Byte	2Byte	TASK_ID	DATA
			1Byte	nByte

> 📝 **NOTE**
>
> The message length does not contain the header

Example: We will add a polling instruction.

Publish topic:

```sh
application/1/device/60c5a8fffe75404b/tx
```

> 📝 **NOTE**
>
> Application ID and Device EUI should be consistent with the settings within the gateway.

- To successfully complete this, the JSON data format must be followed.

Content of the uplink:

```sh
{
"confirmed":true,
"fPort":129,
"data":"03000100090101030000002C40B"
}
```

Parameter	Description
"confirmed":true	This indicates that the downlink to the RAK7431 will be confirmed for successful receiving.
"fPort":129	Defines the port that we want to send the command. (For more information on the fPort see the AT Command Manual for RAK7431)
"data":"03000100090101030000002C40B"	The data of the task in hexadecimal format.

The content of the data that we will send is:

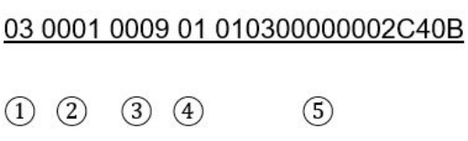

Figure 23: Data arrangement

1. DTU command word

2. The message number
3. Message length (excluding header)
4. The task ID
5. The content of the task

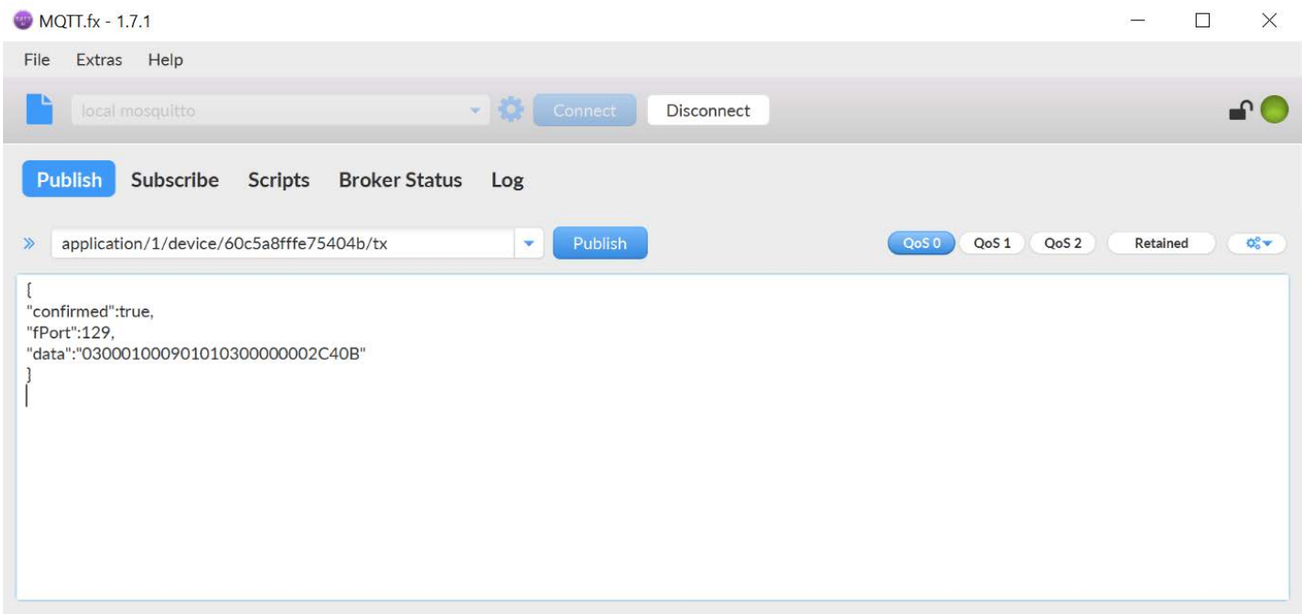

Figure 24: Publishing data to RX topic

- After publishing the data, we can see the downlink instruction and uplink answer from the RAK Serial Tool:

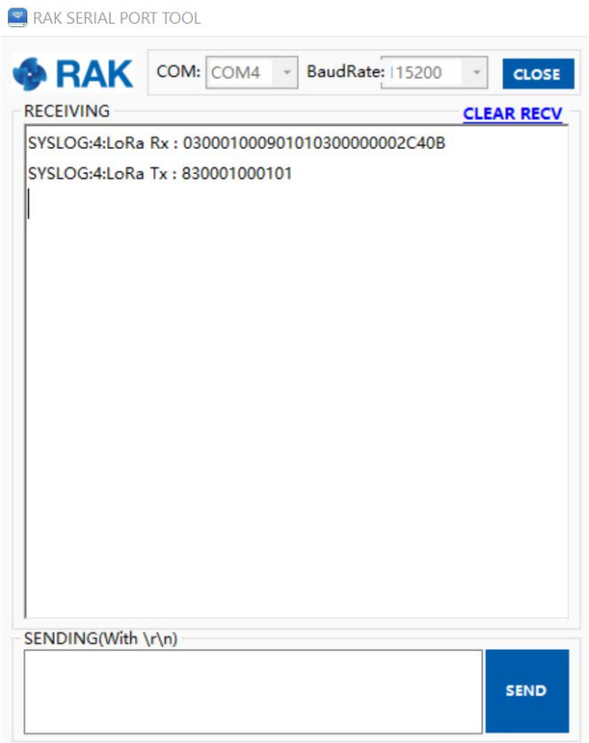

Figure 25: Received data and sent an answer

Message format when execution is successful:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x83	2Byte	2Byte	TASK_ID
			1Byte

- The MQTT subscription bar can see the upstream message "**83000100010101**" for successful execution.

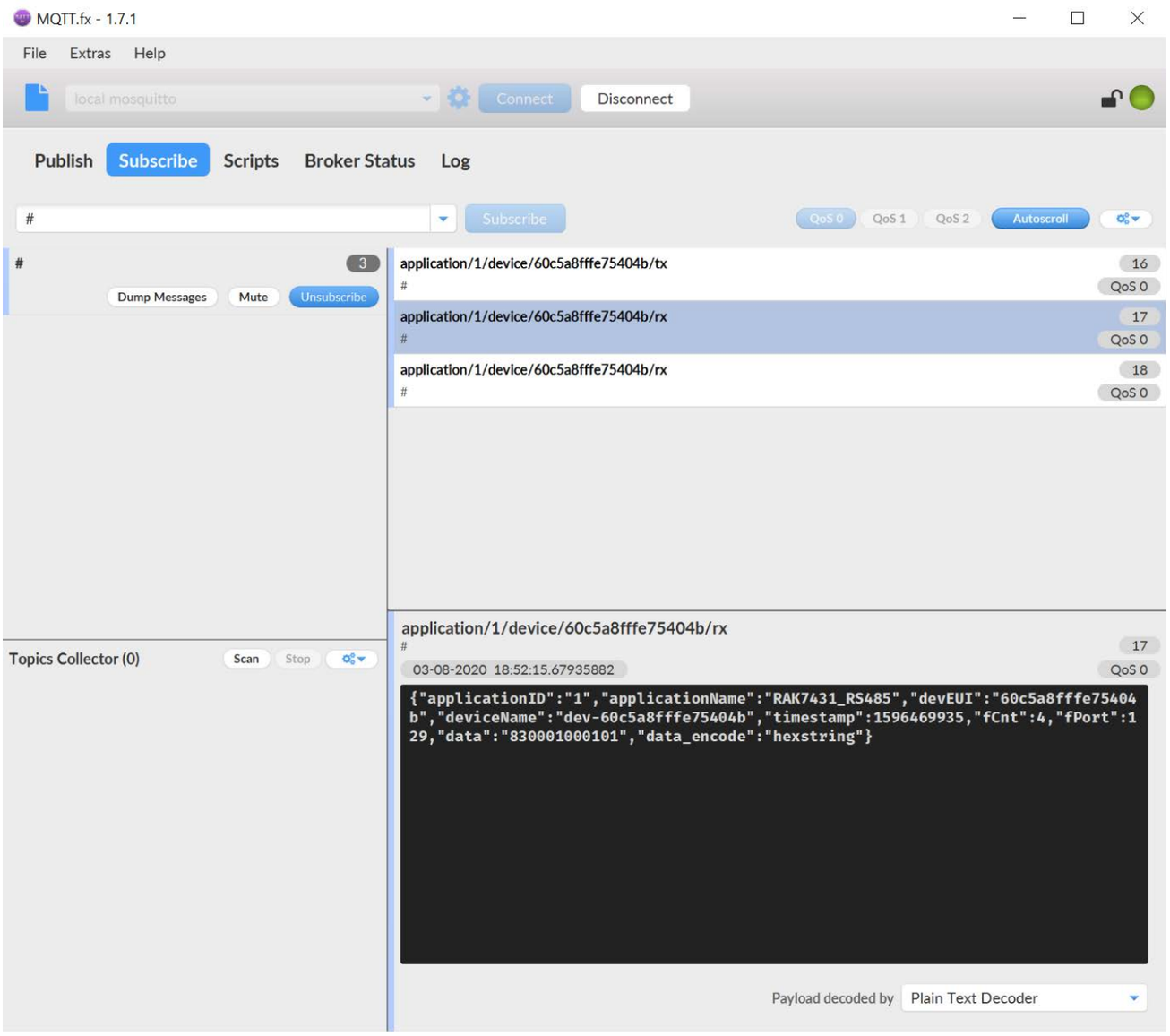

Figure 26: Received confirmation of the task

Remove the Scheduled Polling Task List

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x04	2Byte	2Byte	TASK_ID 1Byte

Example: Removal of timed polling temperature and humidity sensor task order on a node:

Publish the topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"confirmed":true,
 "fPort":129,
"data":"040001000101"
}
```

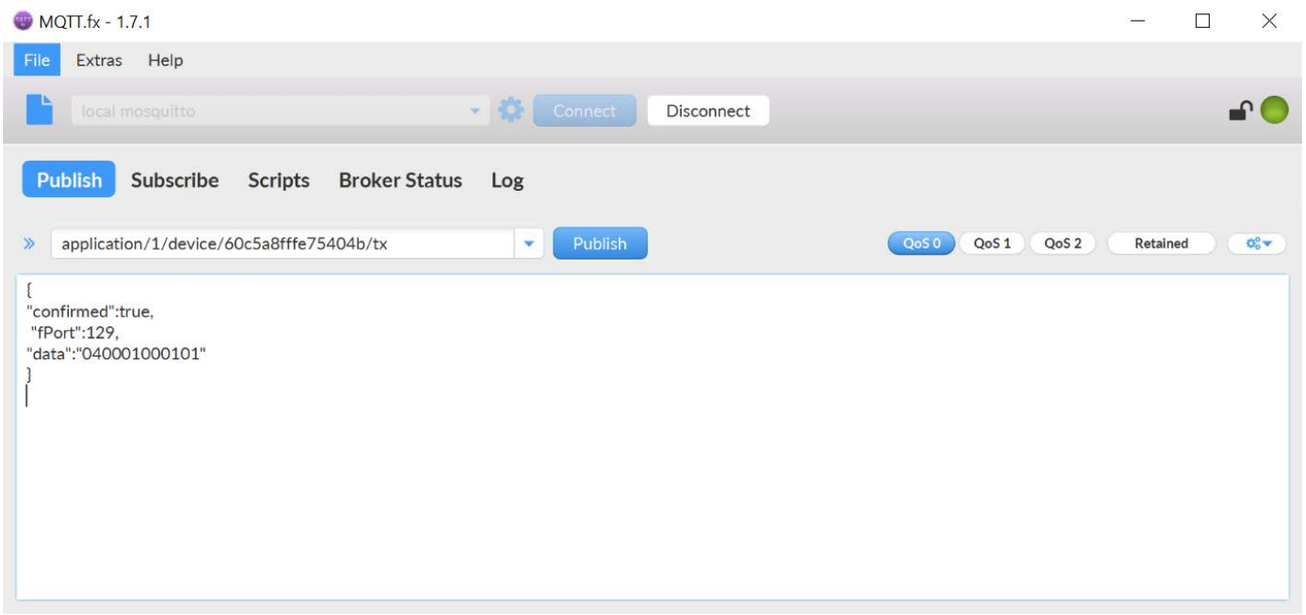

Figure 27: Remove poll downlink message

Message format when execution is successful:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x84	2Byte	2Byte	TASK_ID
			1Byte

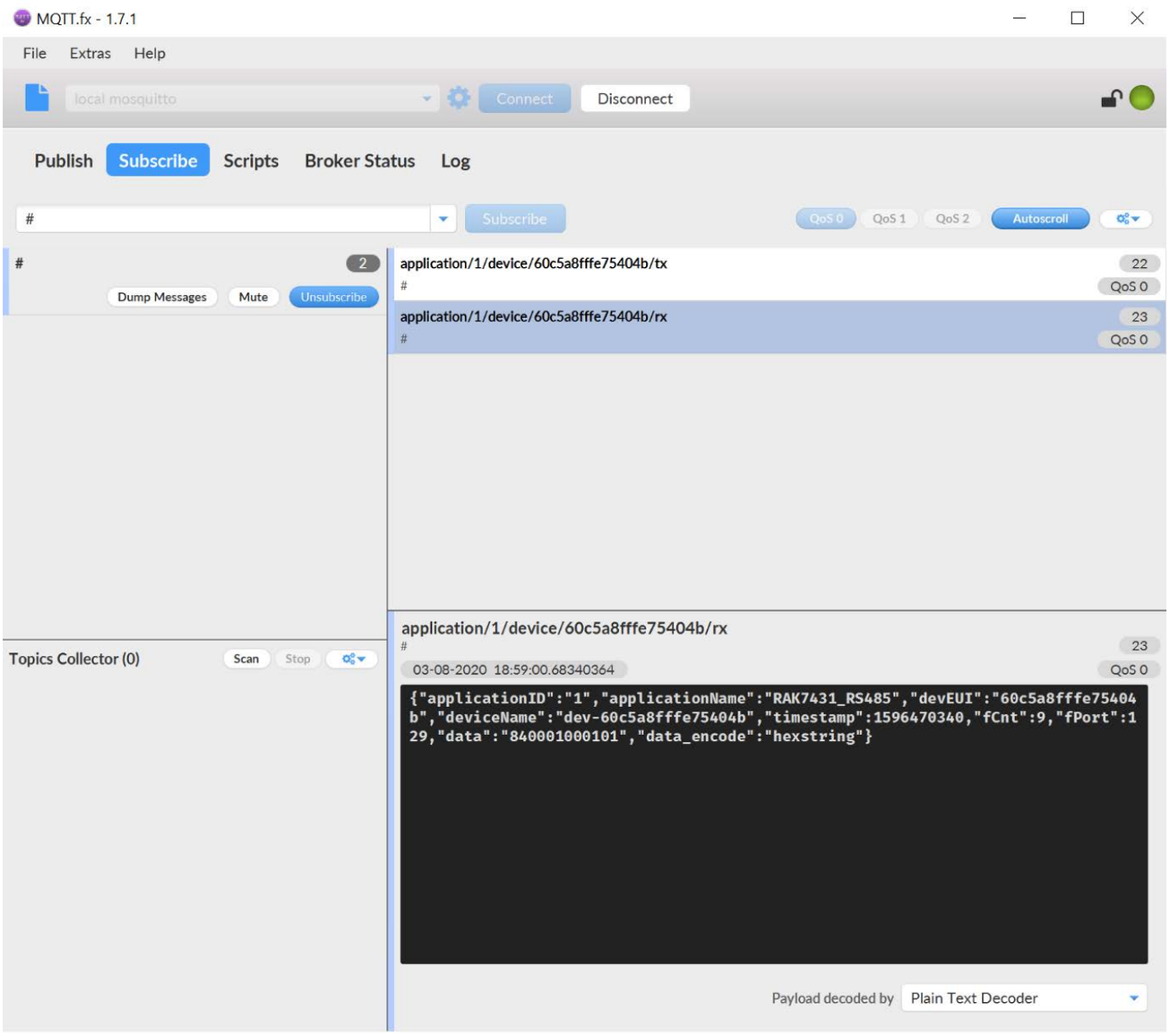

Figure 28: Poll removed successfully message

- The MQTT subscription bar sees the upstream message "**84000100010101**", which means the task was successfully removed.

Read the Scheduled Polling Task List

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x05	2Byte	2Byte	TASK_ID 1Byte

Publish topic:

```sh
application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"confirmed":true,
"fPort":129,
"data":"050001000101"
}
```

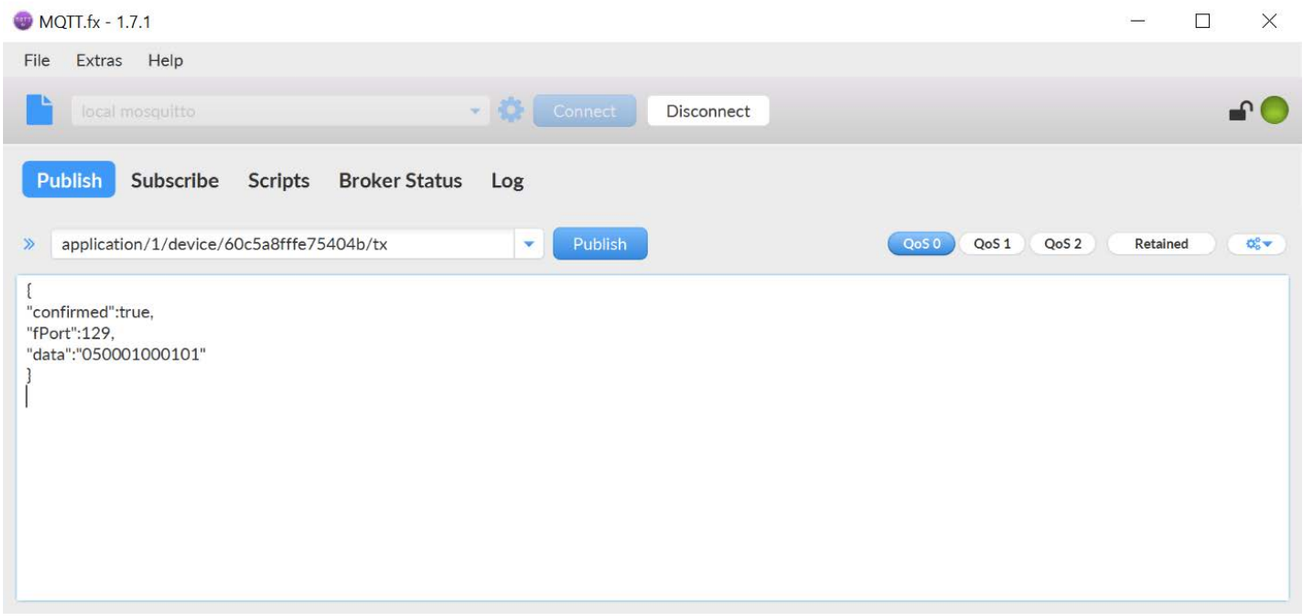

Figure 29: Publishing the read poll task message

Perform successful upstream message format:

DTU_CMD	MSER	MDATA_LEN	MDATA	
			TASK_ID	DATA
0x85	2Byte	2Byte	1Byte	nByte

- Open the MQTT subscription column that is to see to the performance of the above line: "**85000100090101030000000002C40B**" is the query to the task, the order ID is 1, the task order content is 010300000002C40B (example registers).

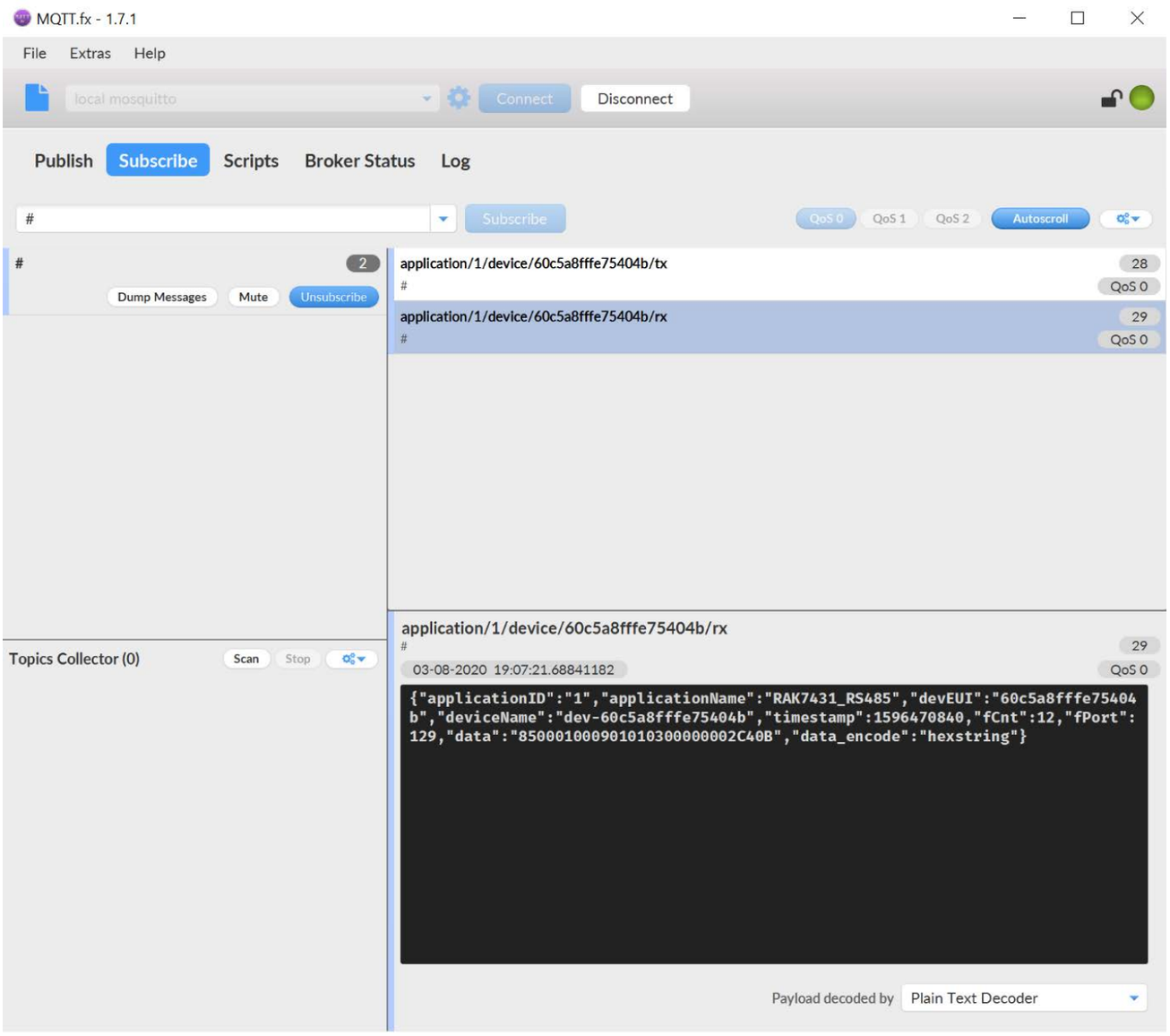

Figure 30: Received message from the node

Read the LoRa Configuration

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x06	2Byte	2Byte	0Byte

Publish topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"confirmed":true,
"fPort":129,
 "data":"0600010000"
}
```

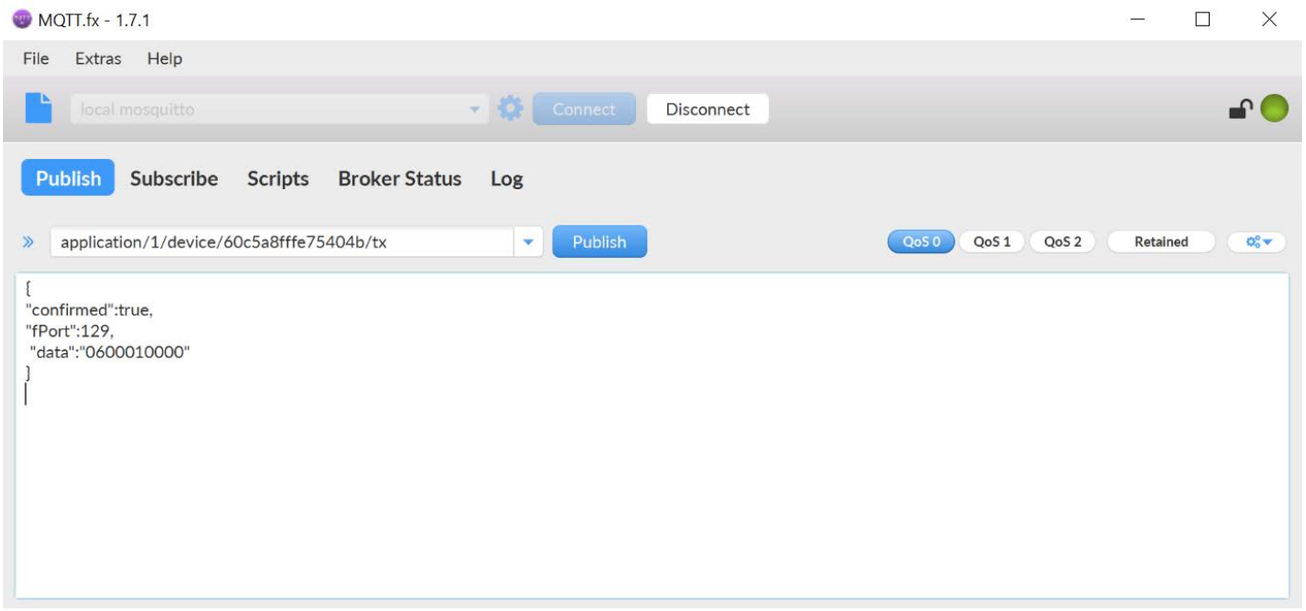

Figure 31: Publish LoRa configuration read message

Perform successful upstream message format:

DTU_CMD	MSER	MDATA_LEN	MDATA					
0x86	2Byte	2Byte	DATA RATE	TXPWR	CONFIRM	RETRY	ADR	C
			1Byte	1Byte	1Byte	1Byte	1Byte	

- **DATARATE**: Speed rate (0 – 5)
- **TXPOWER**: The transmit power level (0 – 20)
- **CONFIRM**: Whether to turn on ACK 0 – off, 1 – on
- **RETRY**: Maximum re-transmission times when ACK is on (0 ~ 15)
- **ADR**: Whether to turn on the dynamic rate adjustment 0 – off, 1 - on
- **DUTY CYCLE**: Whether to turn on duty cycle limit 0 – off, 1 – on

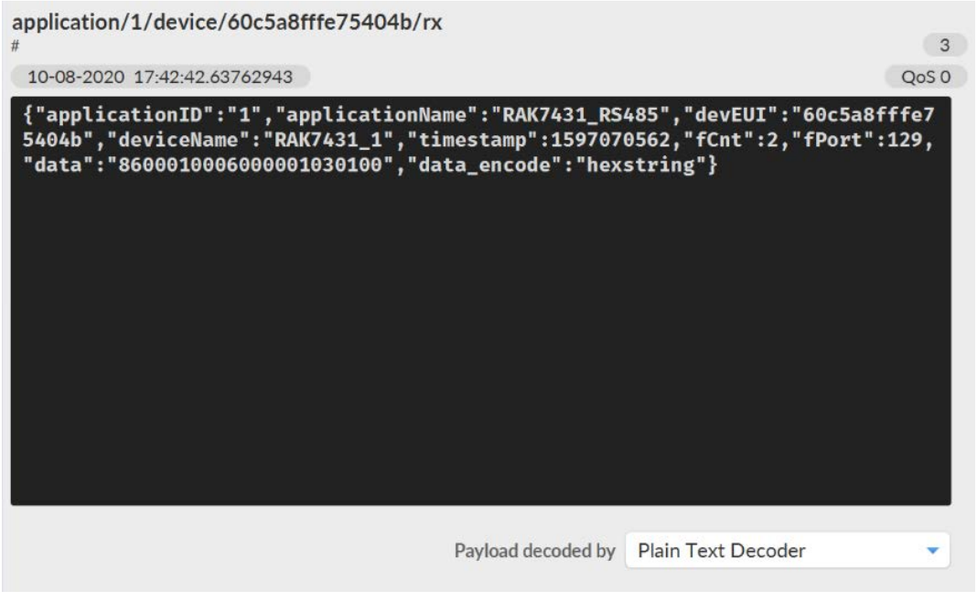

Figure 32: Received message with LoRa configuration

- Open the MQTT subscription bar to see the upstream message "**860001000006000010301000000**" to read the LoRa configuration based on the upstream message format for the successful execution above.

Change the LoRa Configuration

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA					
0x07	2Byte	2Byte	DATA RATE	TXPWR	CONFIRM	RETRY	ADR	C
			1Byte	1Byte	1Byte	1Byte	1Byte	

Publish topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"comfirmed":true,
"fPort":129,
"data":"070001000601050103010"
}
```

- The above command changes the **data rate to "1"** and the **transmit power to "5"**.

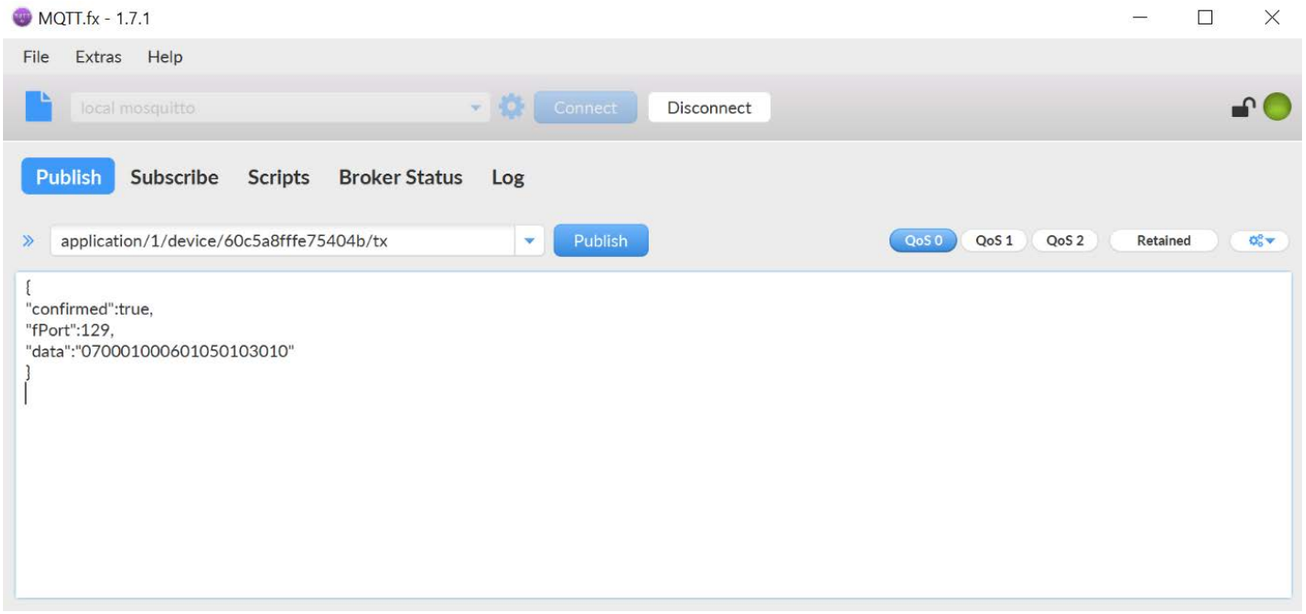

Figure 33: Publish change LoRa configuration data

Perform successful upstream message format:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x87	2Byte	2Byte	0Byte

- Open the MQTT subscription bar to see the upstream message for successful execution: "**8700010000**".

Figure 34: Received confirmation message

Reset the default LoRa Configuration

Publish topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"comfirmed":true,
"fPort":129,
"data":"1D00010000"
}
```

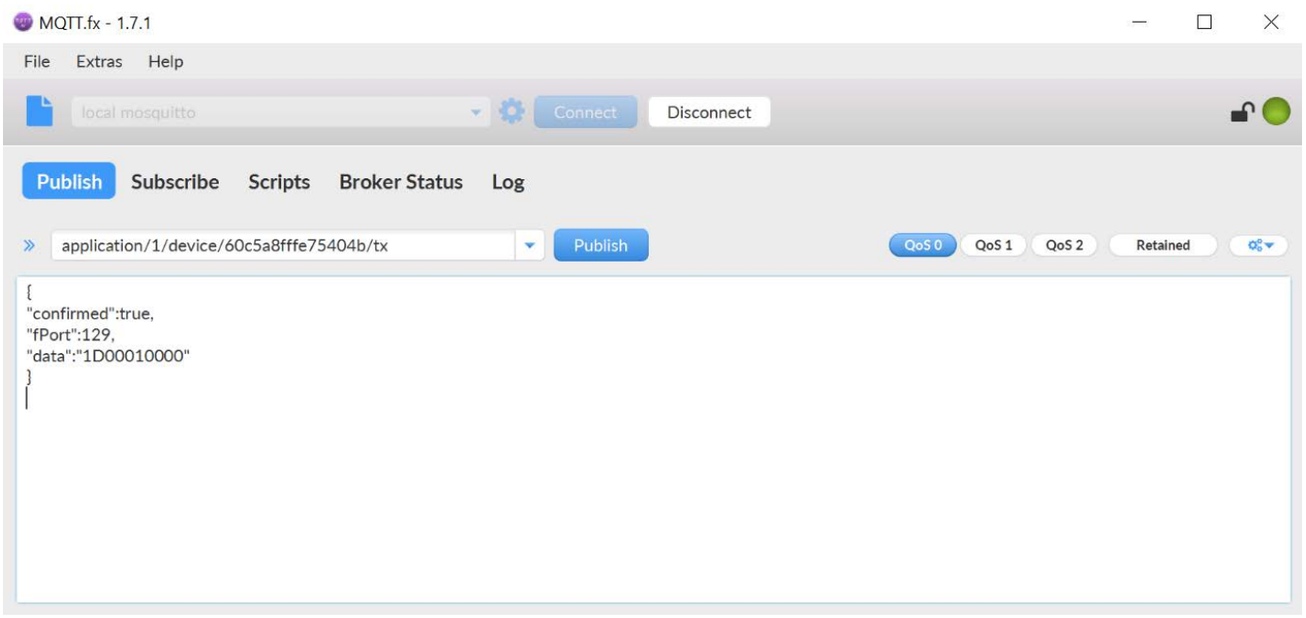

Figure 35: Publish reset the default LoRa configuration

- Open the MQTT subscription bar to see the upstream message for successful execution: "**9D00010000**".

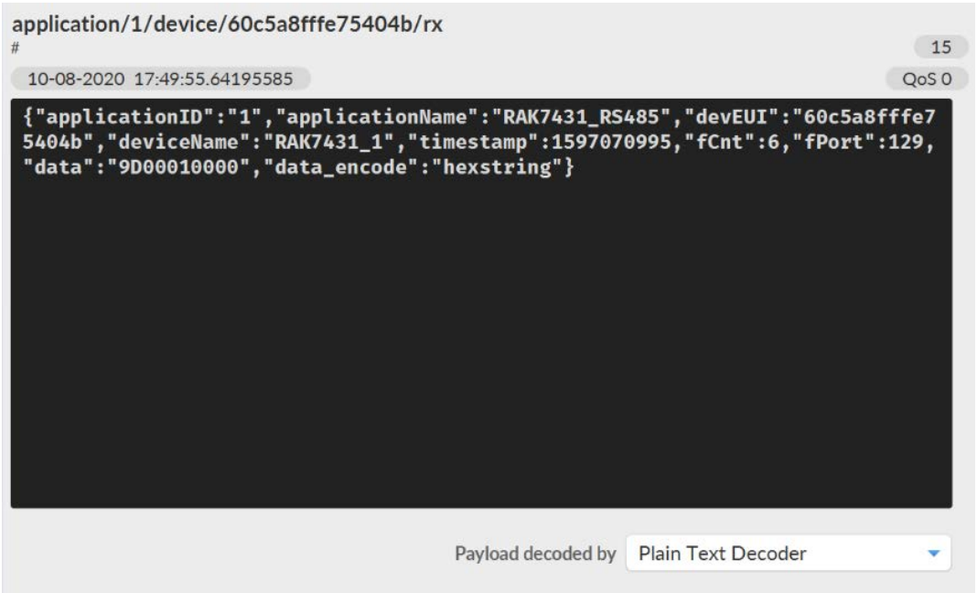

Figure 36: Received Data

LORA configuration default values:

DATARATE	TXPOWER	CONFIRM	RETRY	ADR_ENABIE	DUTYCYCLE_ENABLE
0 – DR_0	19 -19dBm	1 – open	3 times	1 – open	0 – close

Read the DTU Configuration

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x08	2Byte	2Byte	0Byte

Publish topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"comfirmed":true,
"fPort":129,
"data":"0800010000"
}
```

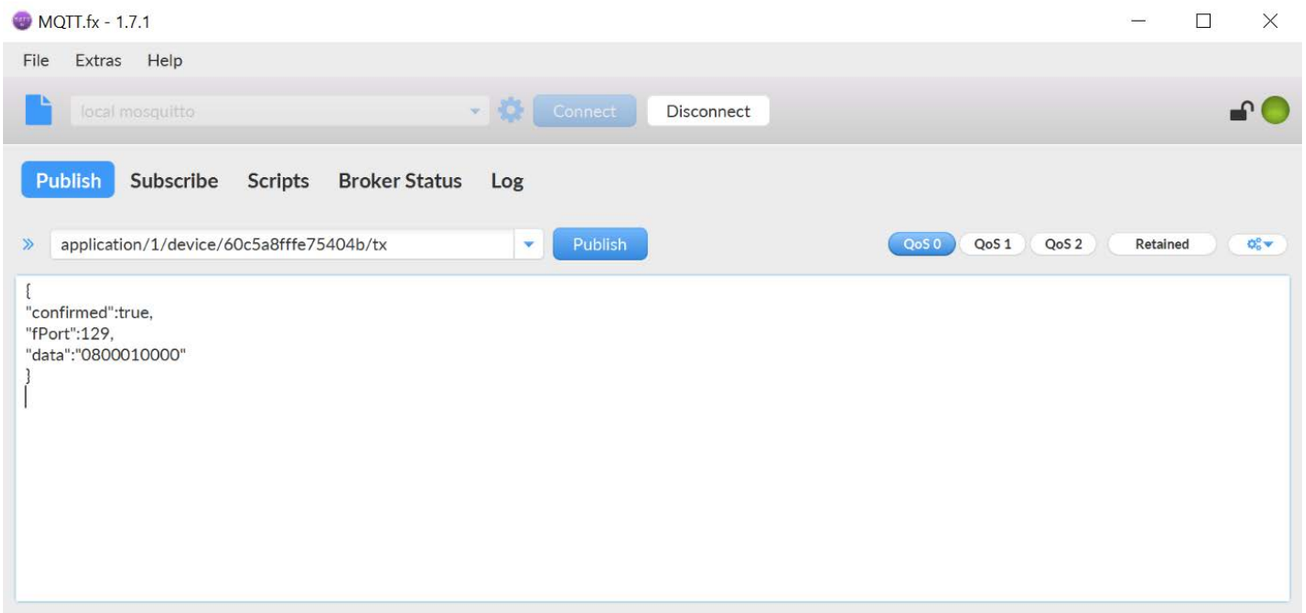

Figure 37: Publish message for reading the DTU configuration

Uplink data message format when execution successful:

DTU_CMD	MSER	MDATA_LEN	MDATA				
0x88	2Byte	2Byte	POLL ENABLE	POLL PERIOD	BUS TIMEOUT	RETRY	RS485
			1Byte	4Byte	1Byte	1Byte	1Byte

- **POLL ENABLE**: Enables scheduled polling, 0 - off, 1 - on
- **POLL PERIOD**: Polling period, in seconds
- **BUS TIMEOUT**: Bus timeout. The unit is seconds
- **RETRY**: Number of retries after bus timeout. 0 - turn off retry function
- **RS485**: 485 bus parameters

Open the MQTT subscription bar to see the upstream message "**880001000080000000 3C010050**" to read the DTU configuration according to the successful upstream message format above.

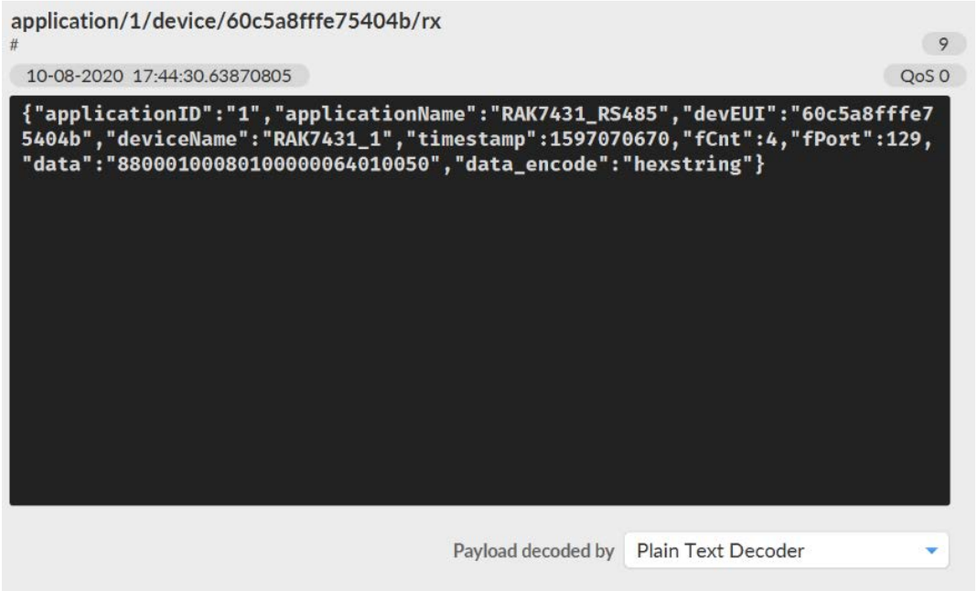

Figure 38: Received message with current DTU configuration

Change the DTU POLL configuration

Downlink instruction message format:

DTU_CMD	MSER	MDATA_LEN	MDATA				
0x09	2Byte	2Byte	POLL ENABLE	POLL PERIOD	BUS TIMEOUT	RETRY	RS485
			1Byte	4Byte	1Byte	1Byte	1Byte

Publish topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"comfirmed":true,
"fPort":129,
"data":"090001000801000000E10010050"
}
```

- The above command changes the polling period to only 1 hour.

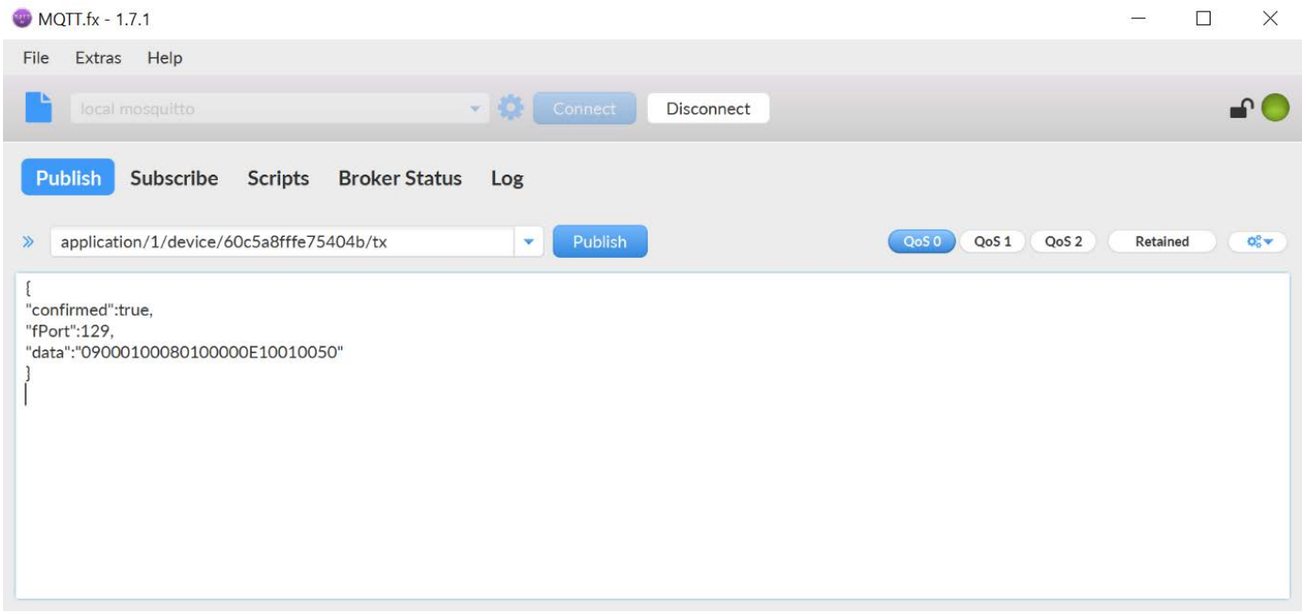

Figure 39: Publish message for change the DTU configuration

Uplink data message format when execution successful:

DTU_CMD	MSER	MDATA_LEN	MDATA
0x89	2Byte	2Byte	0Byte

- Open the MQTT subscription bar to see the upstream message for successful execution: "**8900010000**".

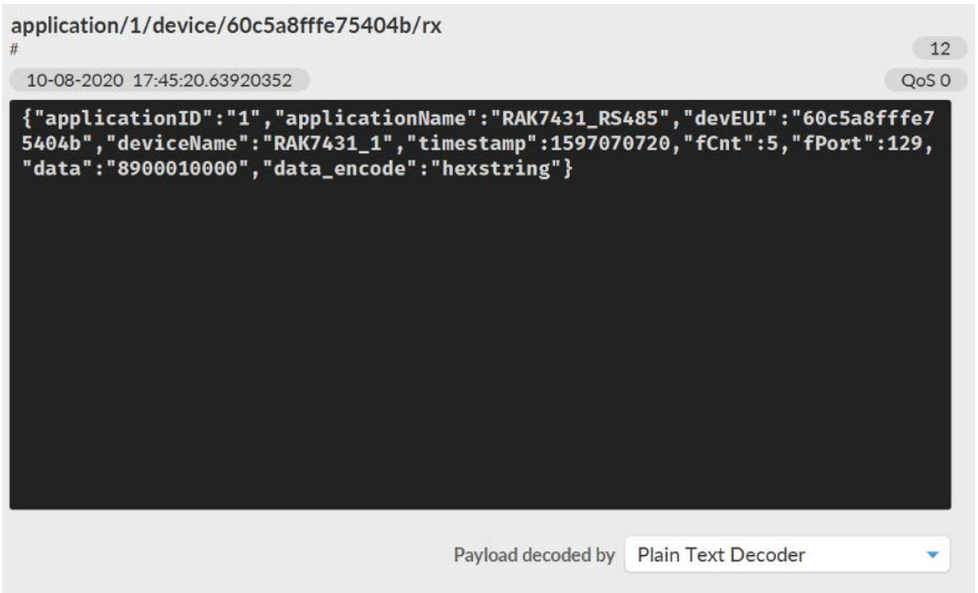

Figure 40: Received confirmation message

Reset the default DTU Configuration

Publish topic:

```sh
Application/1/device/60c5a8fffe75404b/tx
```

Content:

```sh
{
"comfirmed":true,
"fPort":129,
"data":"1E00010000"
}
```

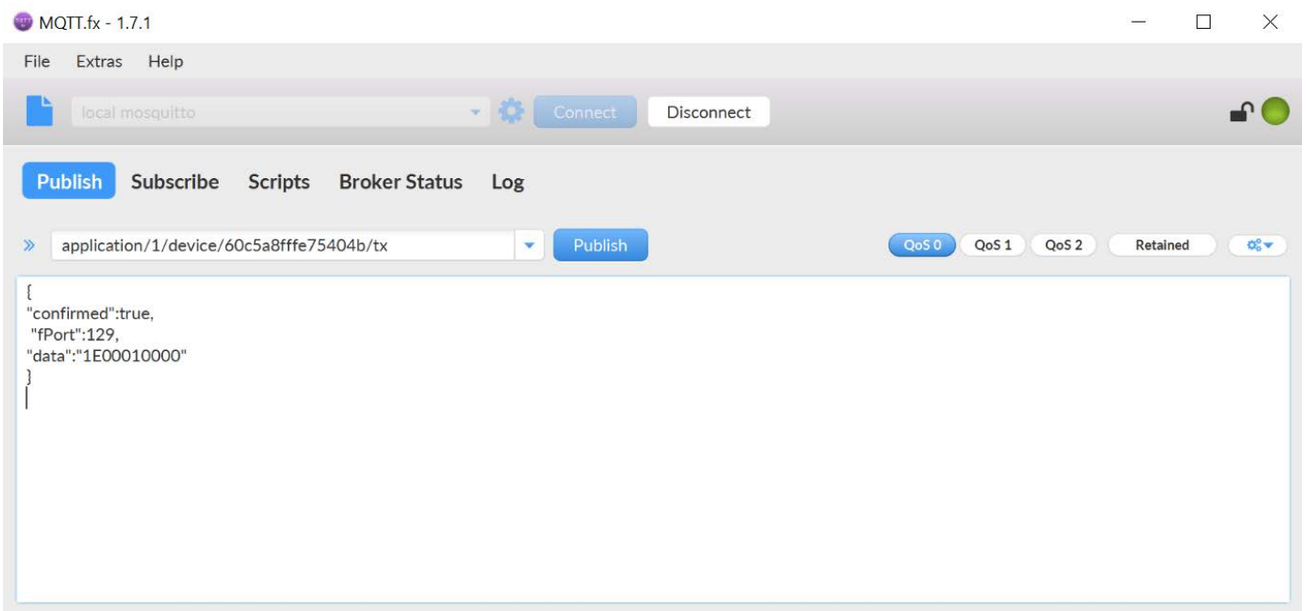

Figure 41: Publish reset the default DTU configuration

- Open the MQTT subscription bar to see the upstream message for successful execution: "**9E00010000**".

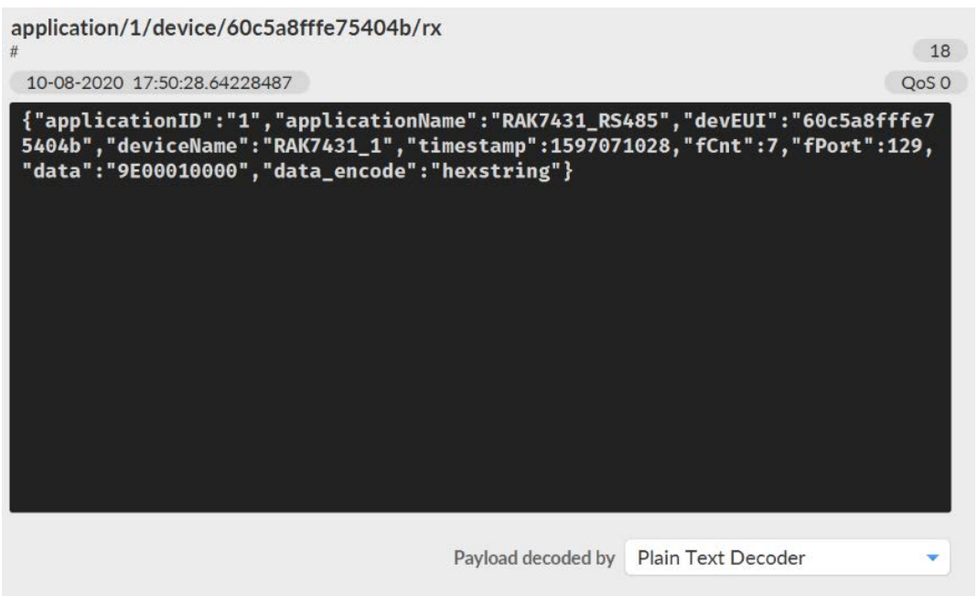

Figure 42: Received Data

DTU Configure the initial value:

POLL_ENABLE	POLL_PERIOD	BUS_TIMEOUT	RS485
1 - on	3600 seconds	1 second	0xE0