

# 内置 LoRa Network



# Server 第三方服务器集成 指南—MQTT 篇

版本 1.0 | 2020 年 7 月

## 目录

1. 目的.....	3
2. LoRaWAN 及 MQTT 简介.....	3
2.1. LoRaWAN 简述.....	4
2.2. MQTT 简述.....	5
2.3. MQTT 在 LoRaWAN 网络中的作用.....	6
3. 订阅节点数据.....	7
3.1. 商业网关使用内置 MQTT 服务器.....	7
3.1.1. 商业网关配置内置 MQTT 服务器.....	7
3.1.2. 通过 mqtt.fx 订阅节点信息.....	12
3.1.3. 通过 mqtt.fx 向节点发送信息.....	17
3.2. 商业网关使用私有 MQTT 服务器.....	20
3.2.1. 商业网关配置私有的 MQTT 服务器.....	21
3.2.2. 通过 mqtt.fx 连接私有的 MQTT 服务器.....	22
4. 通过 MQTT 获取的节点数据格式定义.....	24
4.1. Uplink.....	24
4.2. Downlink.....	25
4.3. Join.....	26
4.4. Ack.....	26

---

4.5. Device Status.....	26
5. 程序示例.....	27
6. 修订历史.....	31

## 1. 目的

本文旨在针对购买 RAK 商业网关产品，并且使用内置 LoRa Server 功能的用户，通过本文可以帮助用户了解如何通过 MQTT 订阅网关的内置 LoRa Server 得到的数据，使用户了解商业网关的工作原理，方便用户在自己的应用服务器获取节点数据，达到便捷使用应用服务器完成数据展示和数据分析的目的。

本文将以 RAK 公司的商业网关产品（室内网关 RAK7258 或者室外网关 RAK7249）为例，介绍如何使用网关内置 MQTT 服务器和私有（外置）MQTT 服务器；介绍如何订阅节点数据以及节点数据格式的解析。

## 2. LoRaWAN 及 MQTT 简介

在本章节内容中，我们将介绍 LoRaWAN 网络和 MQTT 网络，使大家对 LoRaWAN 和 MQTT 的工作原理有一个简单直观的了解。如果您已经熟悉了 LoRaWAN 以及 MQTT 的工作原理，可跳过此章节内容。

## 2.1. LoRaWAN 简述

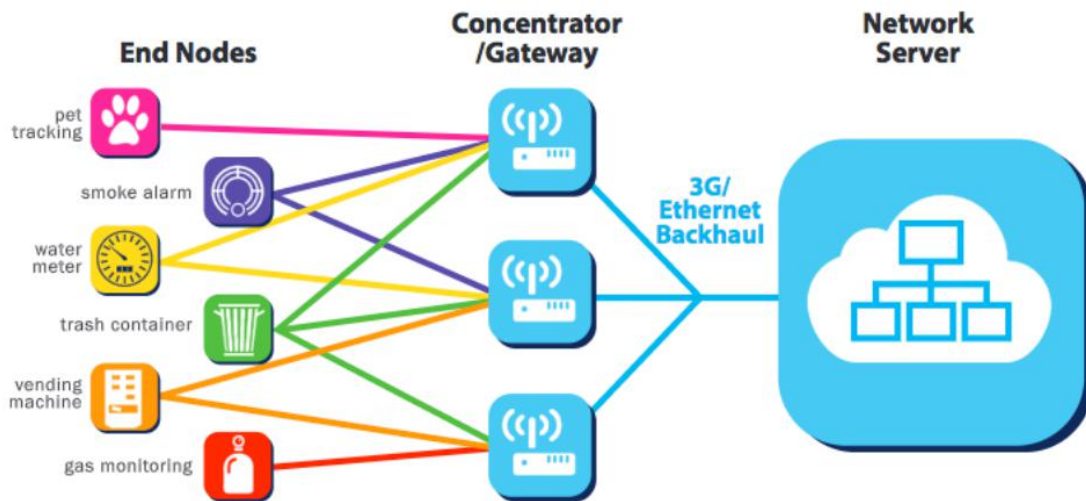


图 1

如图 1 所示，LoRaWAN 网络角色分为：

- 终端设备（End Nodes）
- 网关（Concentrator/Gateway）
- NS 服务器（Network Server）

LoRaWAN 网络中的角色作用：

**终端设备**即节点设备，负责数据采集，并将数据加密后以无线信号的形式传递给网关；

**网关**将终端发送的数据透传给 NS 服务器；

**NS 服务器**根据数据身份信息、密钥对网关转发的数据进行解密、处理。

RAK 简化 LoraWAN 实际部署条件，商业网关集合了 NS 服务器。配合 RAK 的节点可以更轻松搭建起 LoraWAN 网络。

## 2.2. MQTT 简述

MQTT 代表 MQ 遥测传输。是一种发布/订阅，极其简单和轻量级的消息传递协议，旨在用于受限设备和低带宽、高延迟或不可靠的网络。设计原则是使网络带宽和设备资源要求最小化，同时还要确保可靠性。这些原则也使该协议成为物联网世界的理想选择。

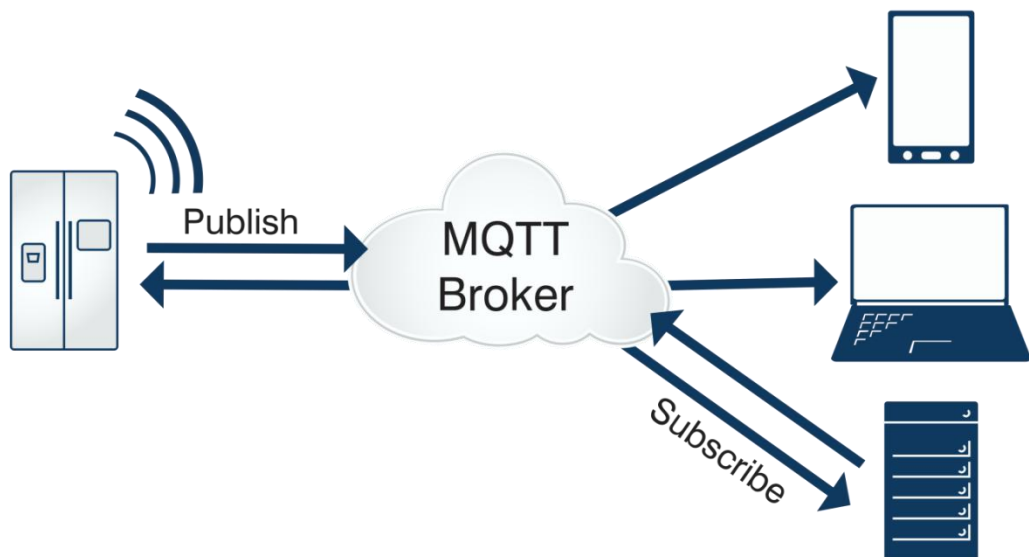


图 2

如图 2 所示，MQTT 网络角色分为：

- 发布者 (Publisher)
- 订阅者 (Subscriber)
- MQTT Broker

**发布者 (Publisher)** 发布信息；

**订阅者 (Subscriber)** 收集发布者发布的信息；

**MQTT Broker** 接收发布信息并将信息向订阅者进行展示。

MQTT Broker 同比是新闻发布网站，发布者是新闻发布成员，订阅者是浏览、查看新闻的用户。

### 2.3. MQTT 在 LoRaWAN 网络中的作用

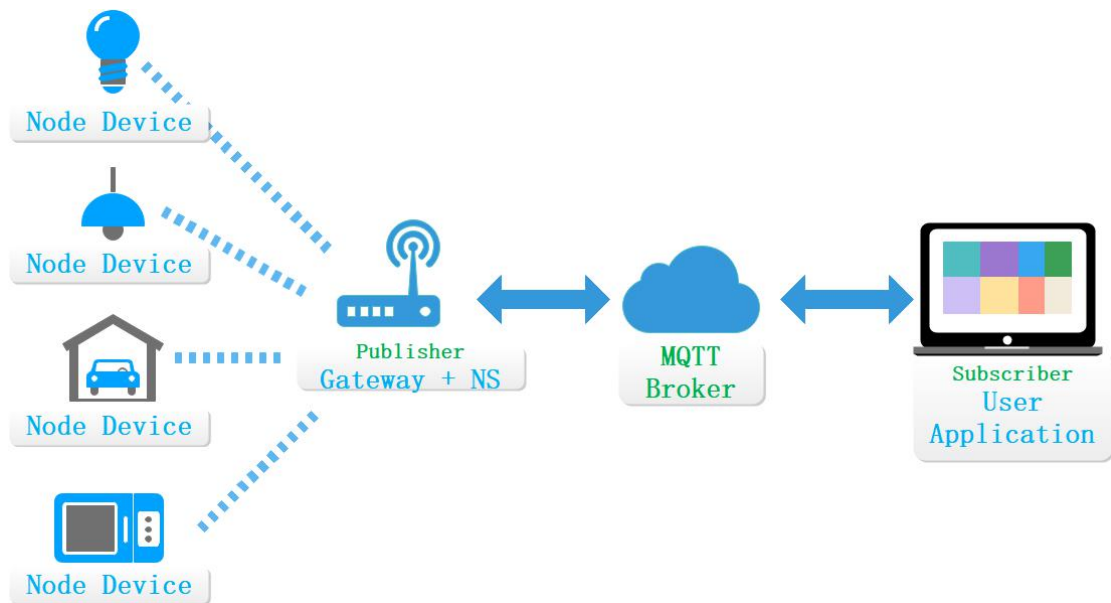


图 3

实际应用 1：

使用 RAK 商业网关内置 MQTT Broker 功能。

RAK 商业网关获得数据并发送给内置 NS，内置 NS 将数据发布至内置 MQTT Broker，用户通过第三程序订阅数据。

RAK 商业网关即为发布者，又为 MQTT Broker。

**注意：1.使用网关内置的 MQTT Broker，无法通过公网订阅或者发布数据；**

**2.网关内置的 MQTT Broker 仅适合于项目研发、测试阶段使用，请勿做生产部署使用。**

实际应用 2 :

不使用 RAK 商业网关内置 MQTT Broker 功能。

RAK 商业网关获得数据并发送给内置 NS ,内置 NS 将数据发布至第三方 MQTT Broker。

用户通过第三程序订阅数据。

RAK 商业网关仅为发布者。

## 3. 订阅节点数据

本章以 RAK 的商业网关为基础，学习如何通过 mqtt.fx 工具订阅节点上报的数据。

所需工具：mqtt.fx 工具（MQTT 客户端）。

下载地址：<https://mqttfx.jensd.de/index.php/download>。

### 3.1. 商业网关使用内置 MQTT 服务器

#### 3.1.1. 商业网关配置内置 MQTT 服务器

将商业网关设置为 Network Server 模式（在该模式下，商业网关既是网关角色，又是 NS 服务器角色），并在商业网关中添加一个节点。

在浏览器中输入商业网关的 IP 地址，进入商业网关的 web 界面：





图 4

默认用户名和密码都为 root。输入密码，点击 Login，进入如下界面：

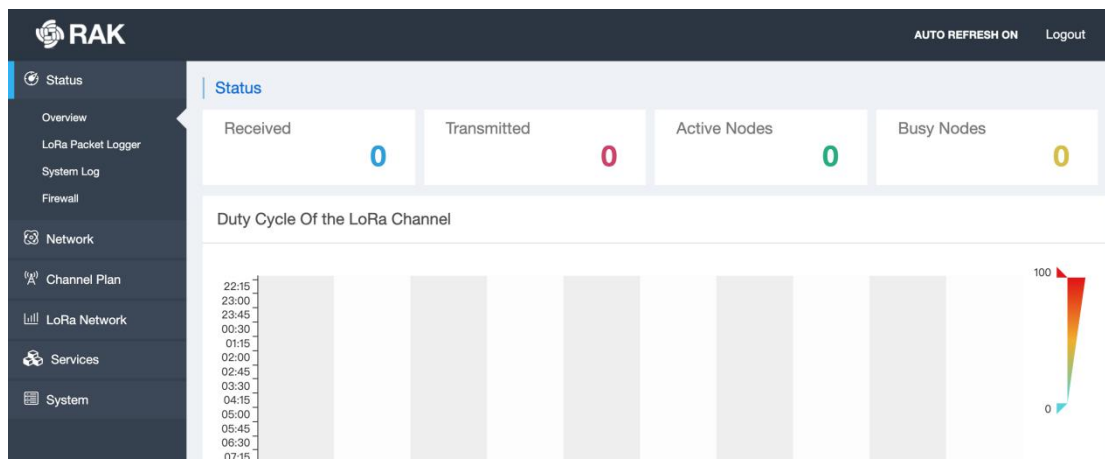


图 5

参考图 6，依次点击 LoRa Network，点击 Network Settings，在 Mode 中选择 Network Server 模式

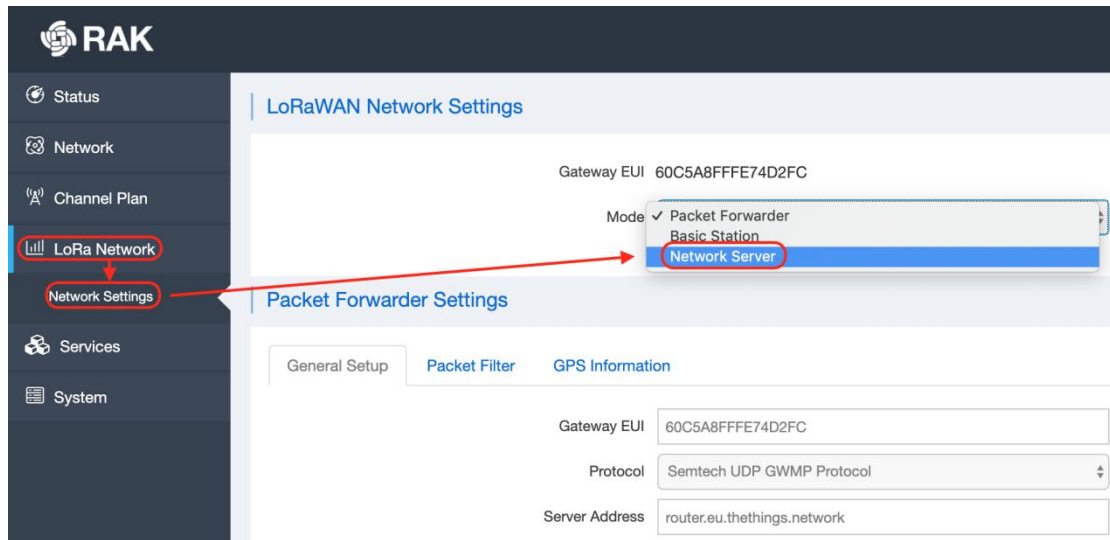


图 6

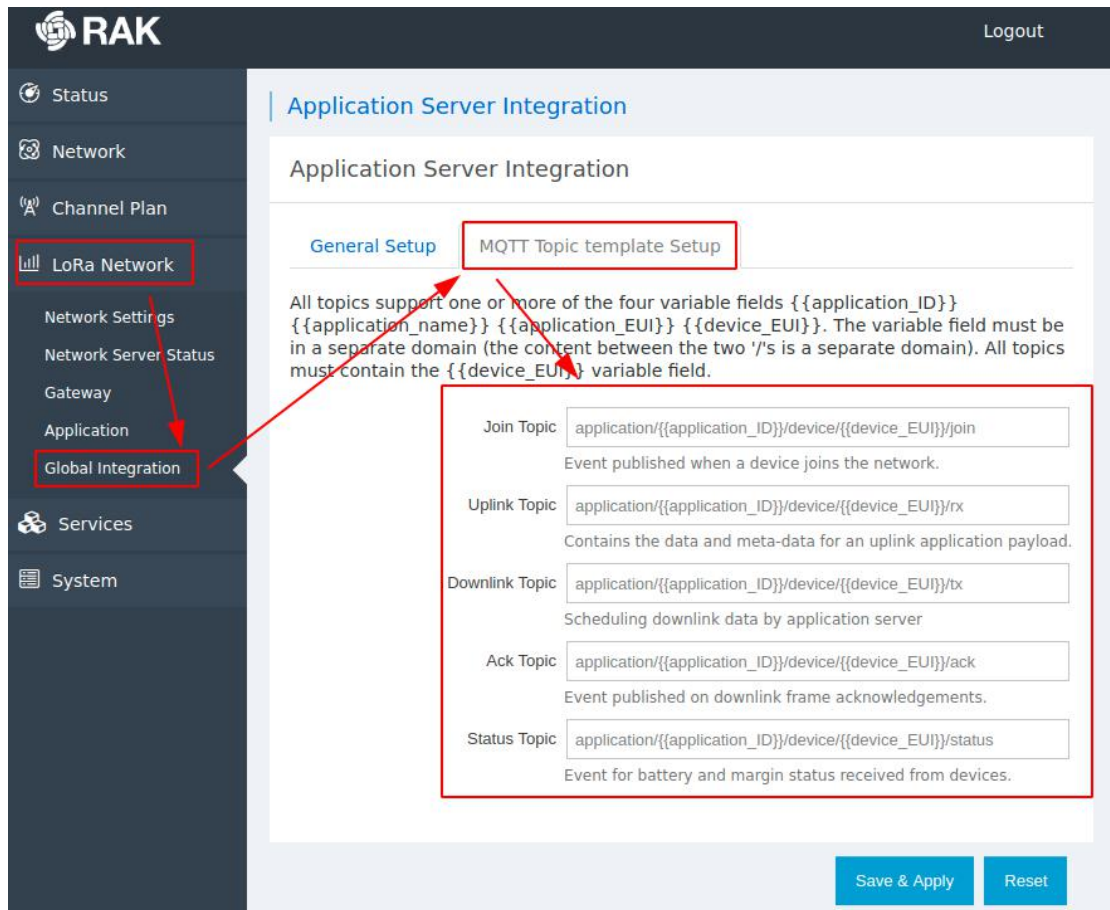
选择 Network Server 模式之后，点击图 7 所示 Switch mode，等待 15 秒钟左右，网关即可切换到 Network Server 模式



图 7

商业网关模式切换成功之后，我们需要在商业网关中增加 Application 和节点。

参考图 8 获取订阅的 topic。



**Application Server Integration**

Application Server Integration

General Setup | MQTT Topic template Setup

All topics support one or more of the four variable fields `{{application_ID}}`, `{{application_name}}`, `{{application_EUI}}`, `{{device_EUI}}`. The variable field must be in a separate domain (the content between the two '/'s is a separate domain). All topics must contain the `{{device_EUI}}` variable field.

Join Topic:   
Event published when a device joins the network.

Uplink Topic:   
Contains the data and meta-data for an uplink application payload.

Downlink Topic:   
Scheduling downlink data by application server

Ack Topic:   
Event published on downlink frame acknowledgements.

Status Topic:   
Event for battery and margin status received from devices.

Save & Apply | Reset

图 8

使用 Uplink Topic 订阅节点上报的数据，要求如下：

“`application/{{application_ID}}/device/{{device_EUI}}/rx`”，其中，`{{application_ID}}`需要替换成我们实际的 application ID，application ID 参考图 9 获取。`{{device_EUI}}`需要替换成我们节点的 device\_EUI，device\_EUI 参考图 10 获取。

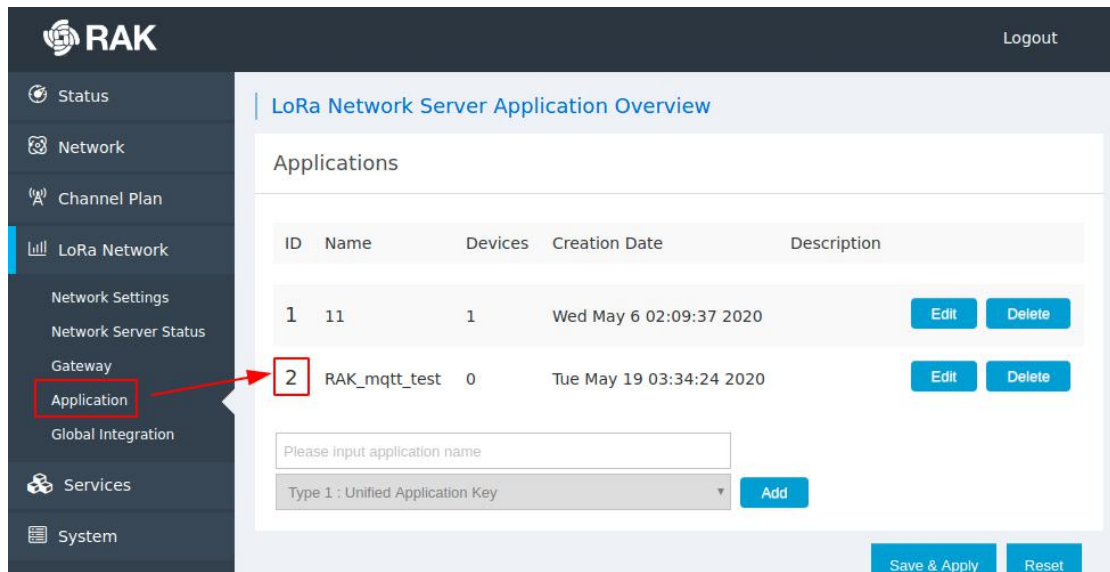


图 9

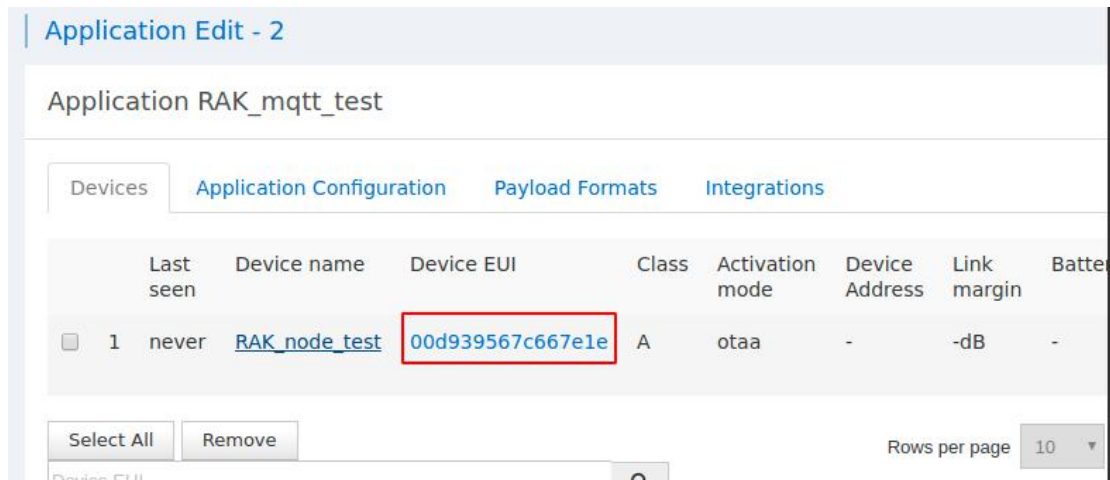


图 10

替换之后的 topic 如下：

application/2/device/00d939567c667e1e/rx

如果我们要订阅一个应用下的所有节点数据，比如应用 2 下的所有节点数据，可以使用以下 topic：

application/2/device/+/rx

如果我们要订阅所有应用下的节点数据，可以使用以下 topic：

application/+/device/+/rx

### 3.1.2. 通过 mqtt.fx 订阅节点信息

打开 mqtt.fx 工具，主界面如图 11 所示：

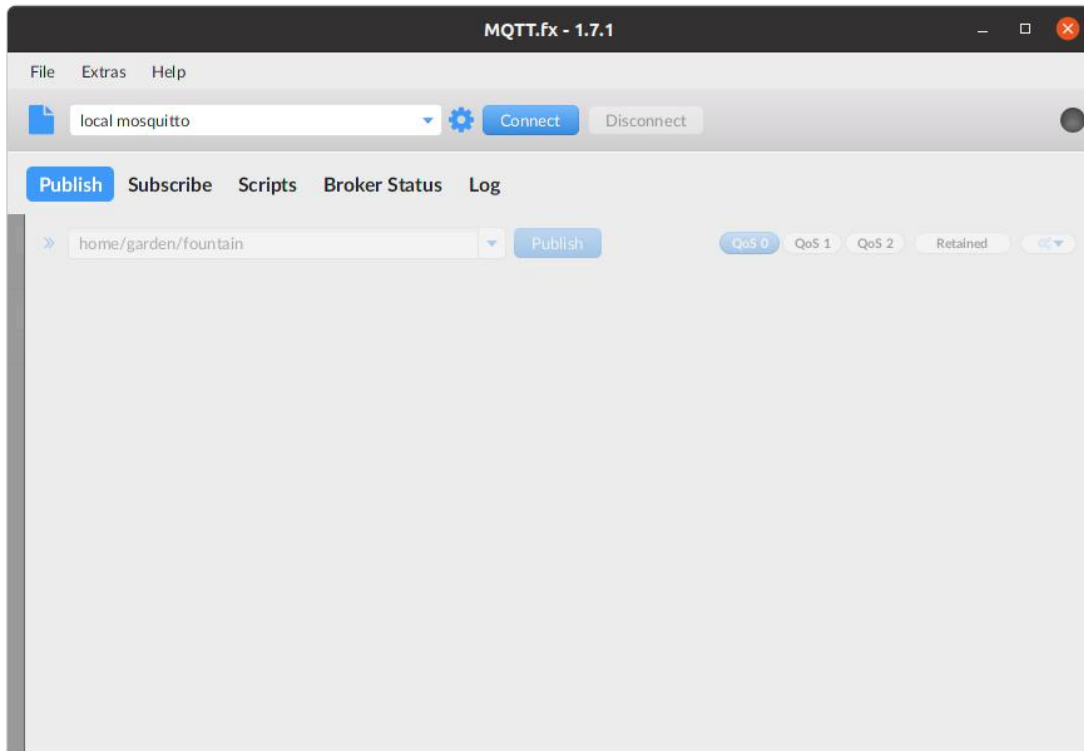


图 11

点击图 12 左上角的新建按钮：

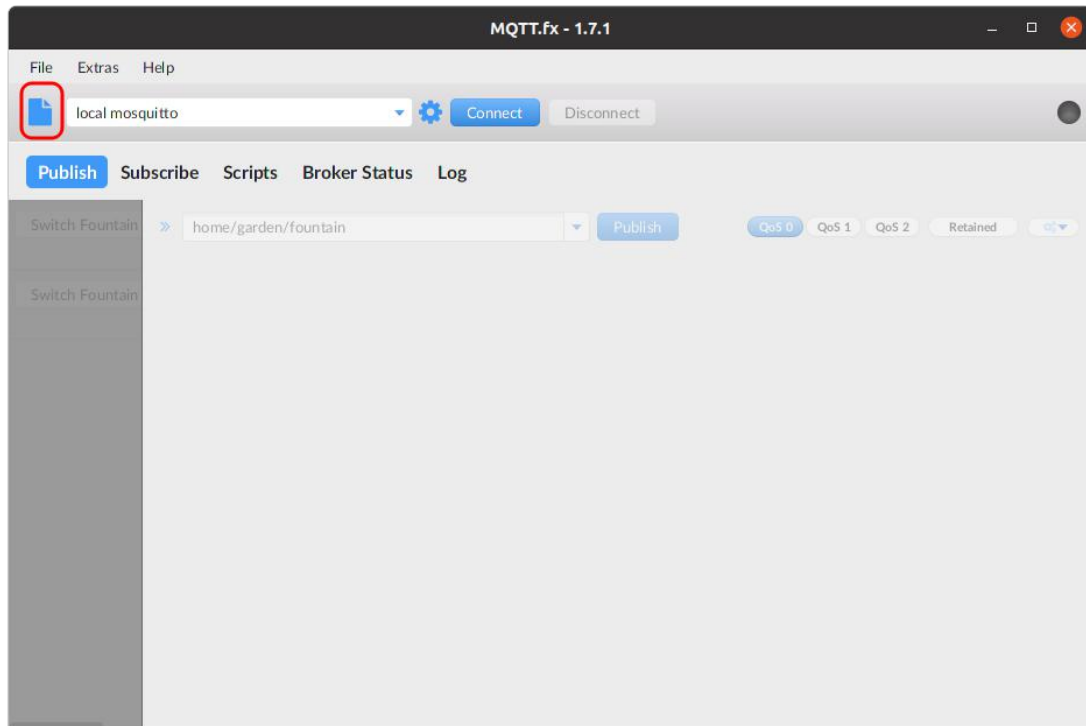


图 12

在图 13 中红框处输入网关 IP 地址：

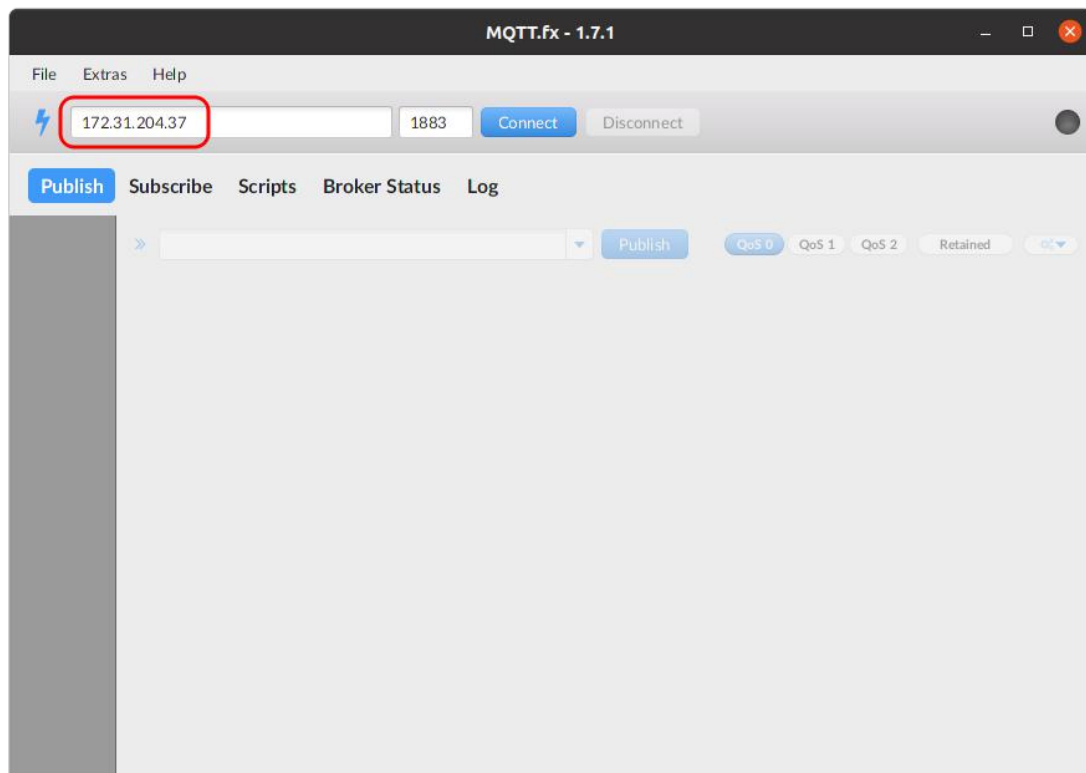


图 13

输入完网关的 IP 之后，点击图 14 中的 Connect 按钮，最右侧的黑色圆形变成绿色，说明我们已经成功连接到网关内置的 MQTT 服务器：

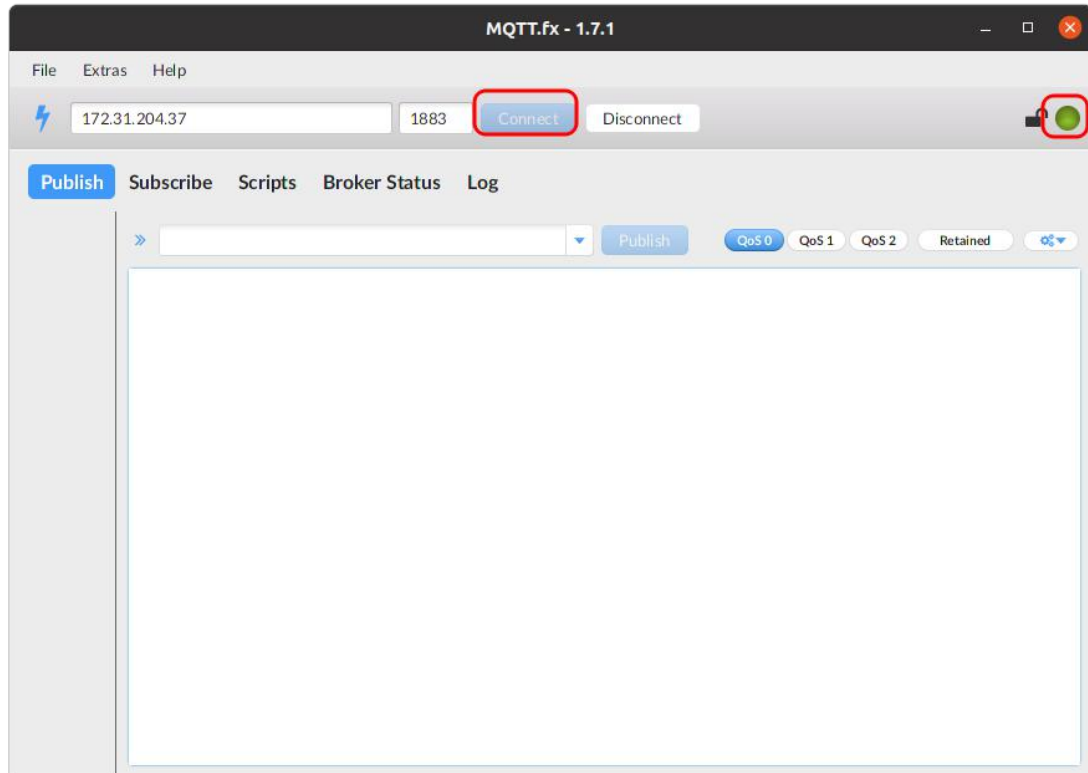


图 14

点击图 15 的 Subscribe 订阅按钮，在输入框中输入订阅的 topic，再点击输入框右侧 Subscribe 按钮，开始订阅节点上报的数据：

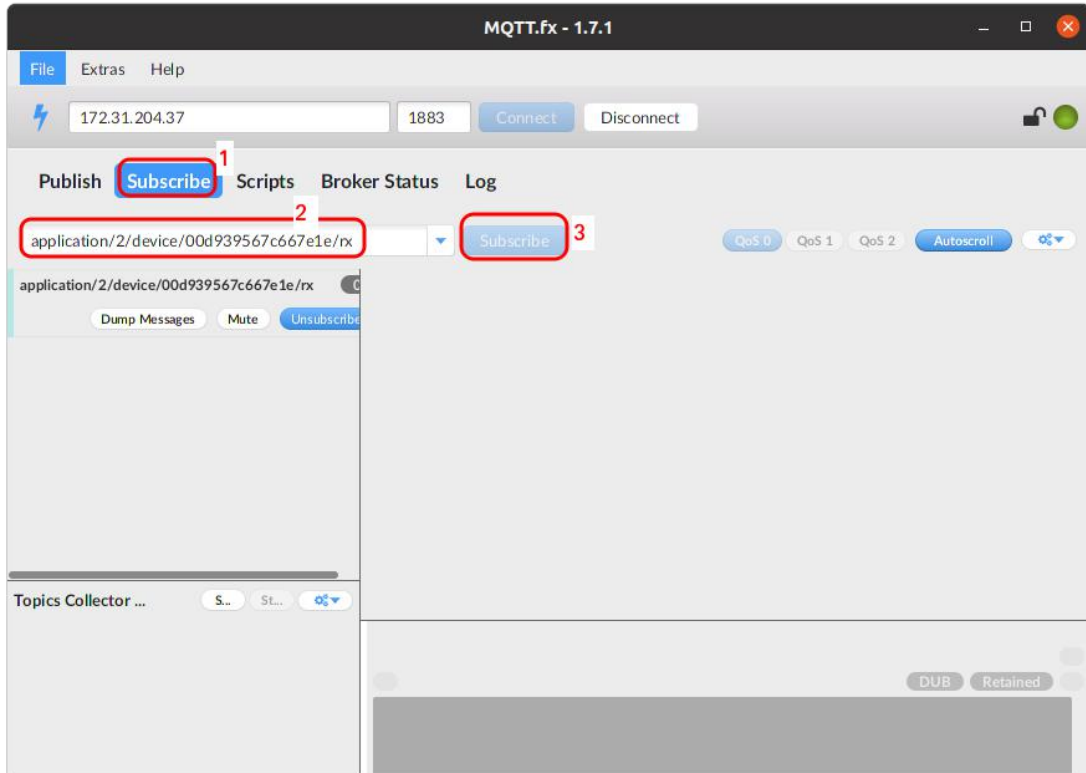


图 15

使用节点成功 join 之后发送一条数据。我们在节点测发送一条“HelloRakwireless”。

节点需要接收 16 进制的数据，我们将“Hello Rakwireless”转换为 16 进制就是“48656c6c6f52616b776972656c657373”。



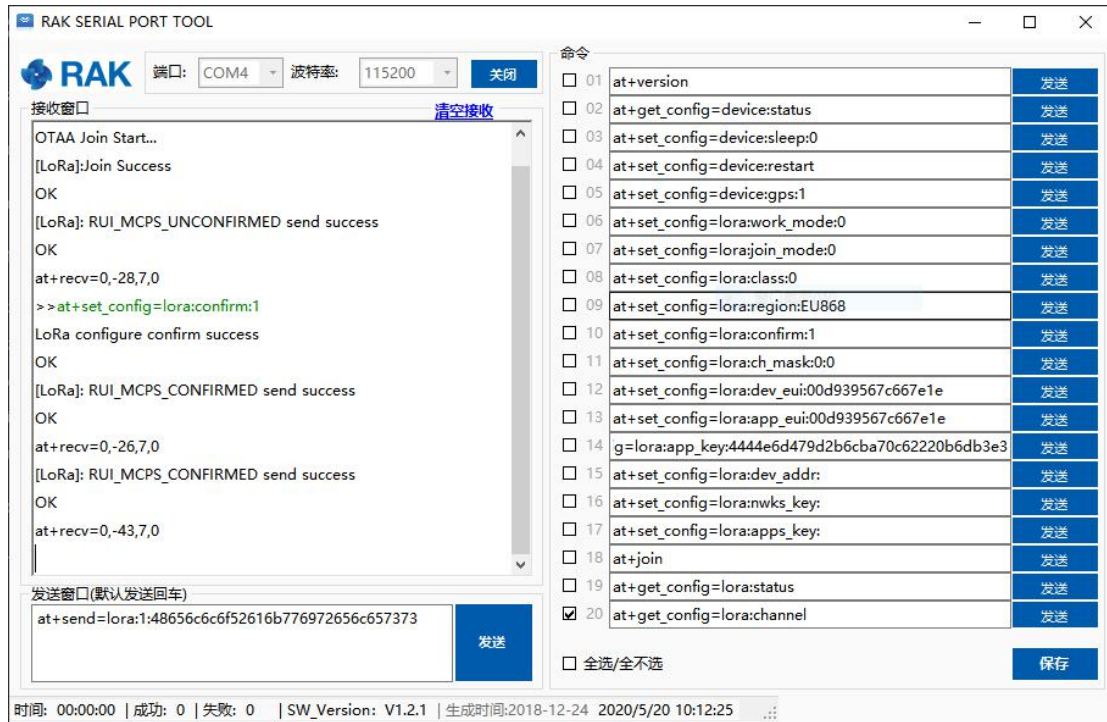


图 16

在 mqtt.fx 界面可以看到我们订阅到的节点数据，如图 17 所示：

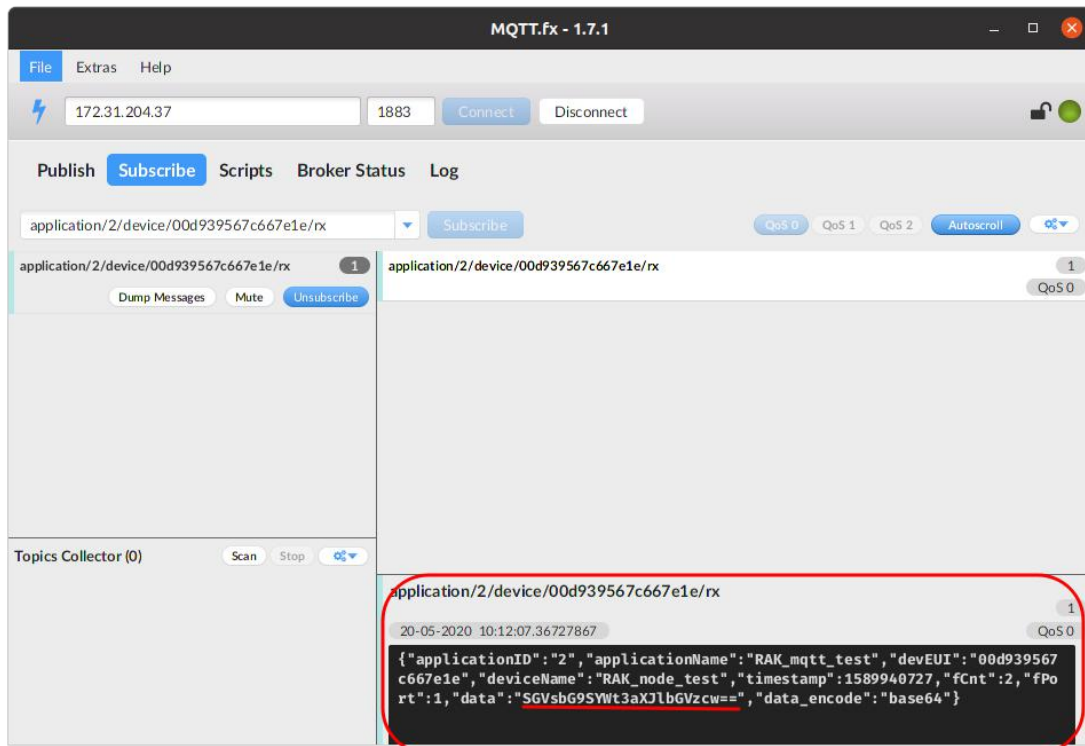


图 17

data 部分就是节点发送的数据。Data 部分的内容是对节点发送的数据进行了 base64 编码，我们只需要对 data 部分进行 base64 解码，即可看到原始数据。

查看商业网关 web 管理页面，看到的数据与 mqtt.fx 一致。

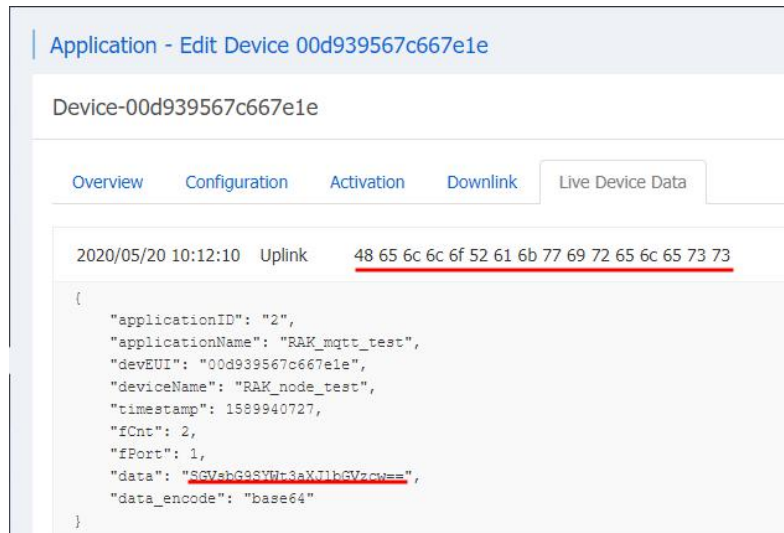


图 18

### 3.1.3. 通过 mqtt.fx 向节点发送信息

通过 mqtt.fx 给节点发送数据，需要使用 Downlink Topic。

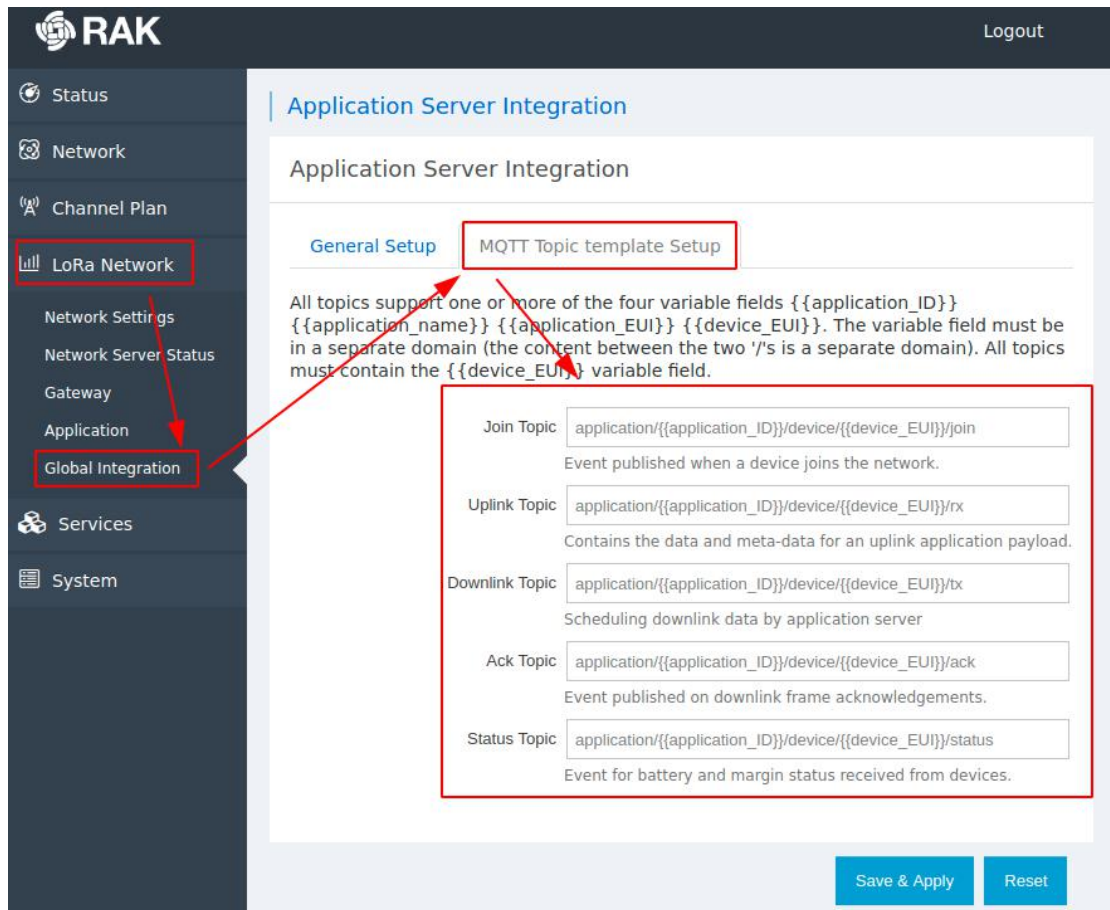


图 19

我们需要将 Downlink Topic 中的 `{{application_ID}}` 和 `{{device_EUI}}` 更换为节点对应的 Application ID 和节点的 Device EUI。Application ID 的获取参考图 20，Device EUI 的获取参考图 21。

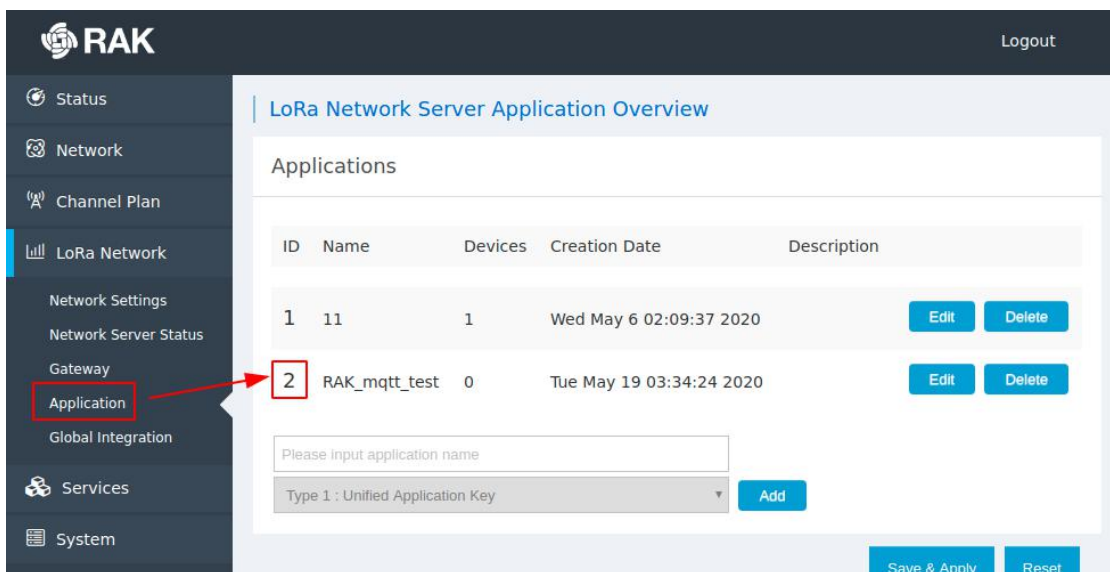


图 20

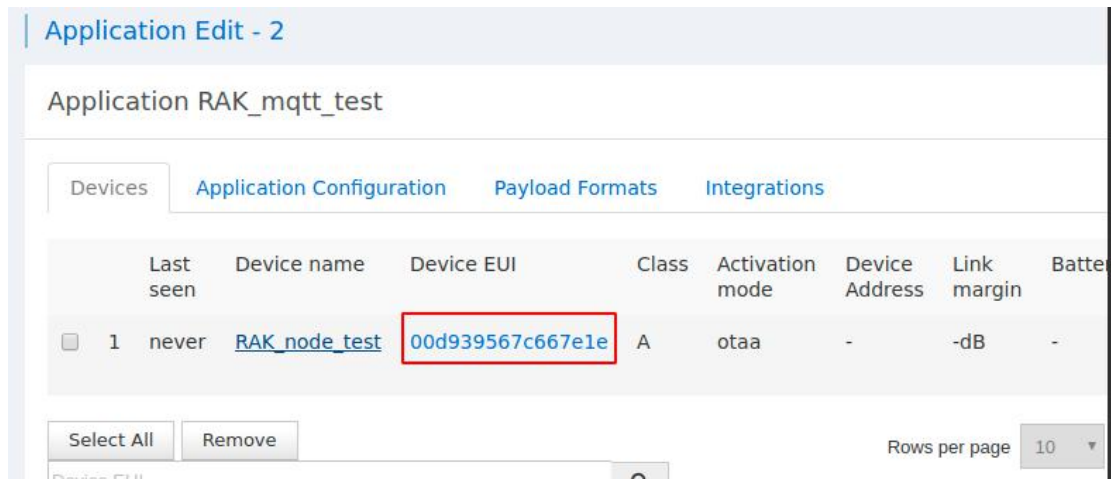


图 21

如图 22 所示，点击 mqtt.fx 左上角的 Publish 标签，在输入框 2 的位置输入 Downlink Topic，在输入框 3 的位置输入 `{"confirmed": true, "data": "SGVsbG8=", "fPort": 10}`，点击按钮 4 的 Publish 按钮，即可将数据发送到节点（注：class c 模式节点会立刻收到 mqtt.fx 下发的数据；class a 模式节点会在下一次上发数据之后收到 mqtt.fx 下发的数据）。

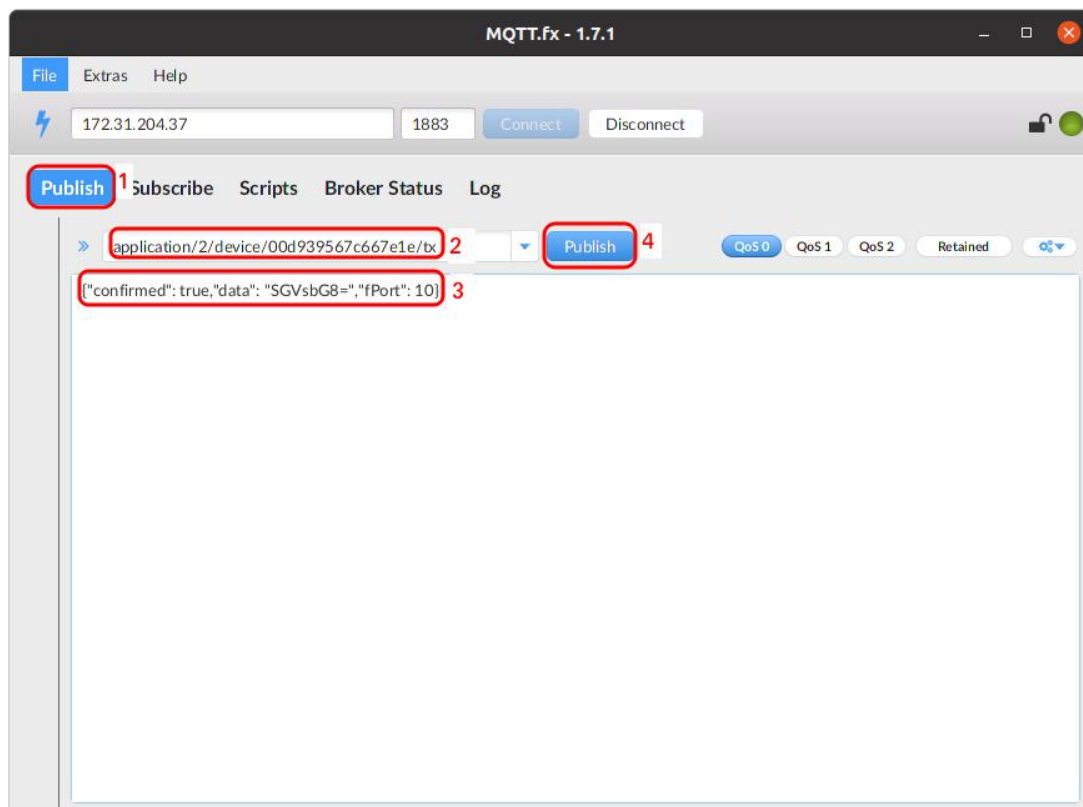


图 22

{`"confirmed": true,"data": "SGVsbG8=","fPort": 10}`} 格式说明

- a. Confirmed 可选值为 true 或者 false。
- b. data 的内容就是我们要发送的数据，需要对数据进行 base64 编码。
- c. fPort 是要发送的端口号，有效端口号为 1-255。

如图 23 所示，可以在节点端看到接收到的数据：

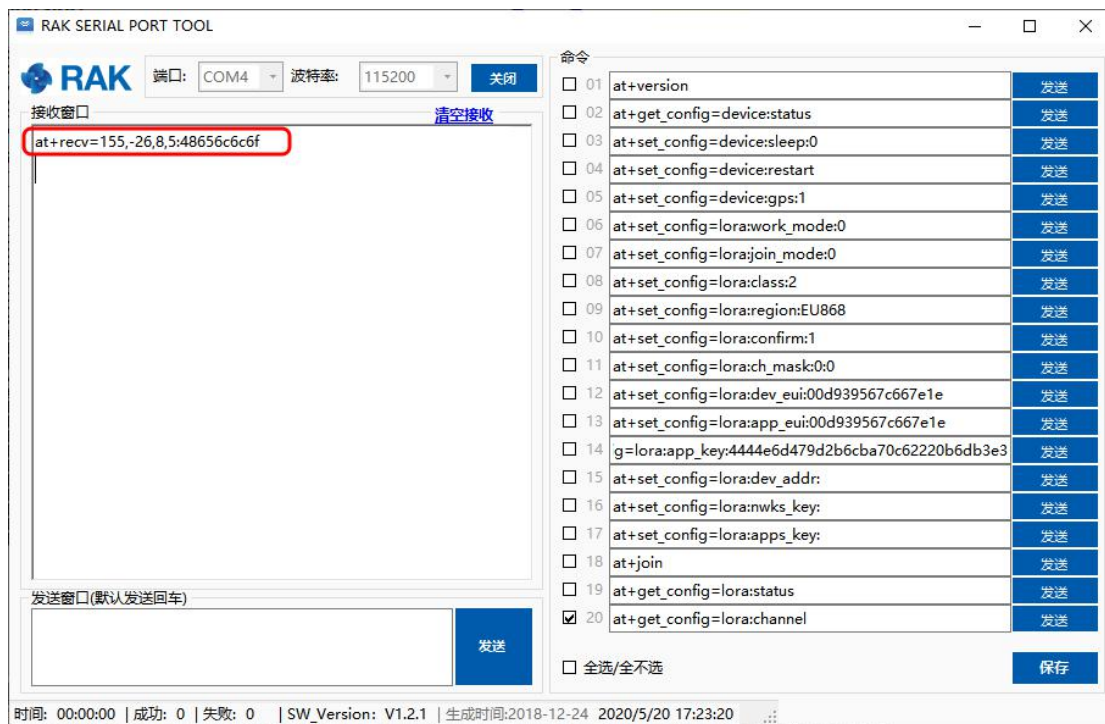


图 23

## 3.2. 商业网关使用私有 MQTT 服务器

本节将说明如何在网关配置私有的 MQTT 服务器，如何通过 mqtt.fx 从私有 MQTT 服务器订阅信息。

**本节讲述以用户名/密码方式访问私有 MQTT。**

### 3.2.1. 商业网关配置私有的 MQTT 服务器

在浏览器打开商业网关的 web 管理页面，参考图 24 配置：

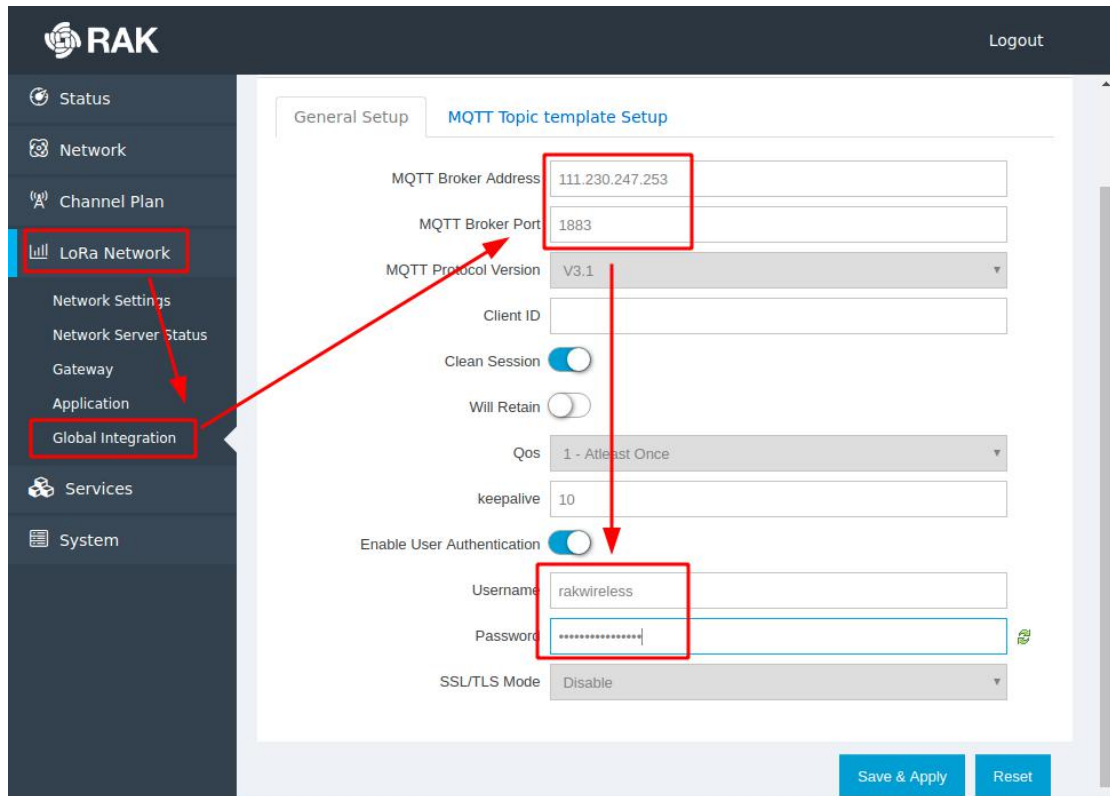


图 24

- MQTT Broker Address 处填写用户 MQTT 服务器的 IP 地址
- MQTT Broker Port 处填写 MQTT 服务的端口号，该端口号如果用户没有更改的话默认为 1883
- 打开 Enable User Authentication 开关
- 输入访问 MQTT 服务的用户名和密码

配置完成之后，点击右下角的 Save&Apply 按钮保存更改。



### 3.2.2. 通过 mqtt.fx 连接私有的 MQTT 服务器

打开 mqtt.fx，点击图 25 所示设置图标：

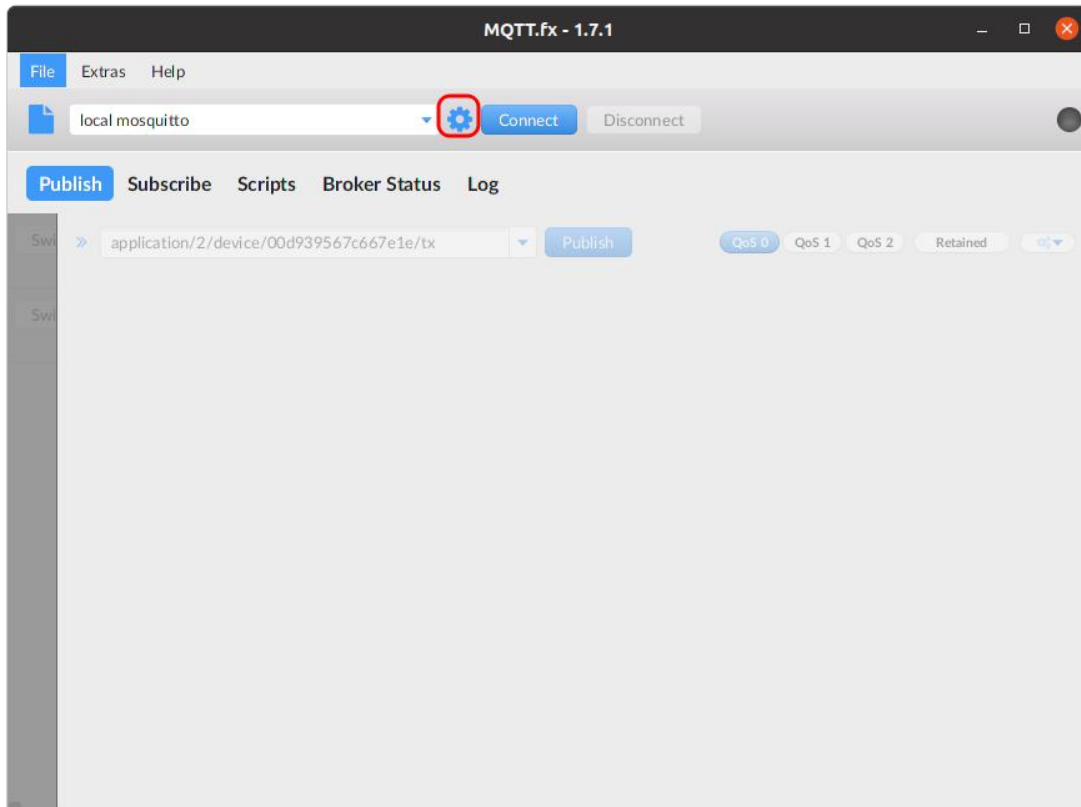


图 25

点击图 26 左下角的加号，新建一个 Profile，输入 Profile Name，配置 MQTT 服务器的 IP 地址和端口，在 User Credentials 标签下输入用户名和密码。配置完成之后点击右下角的 OK 按钮：

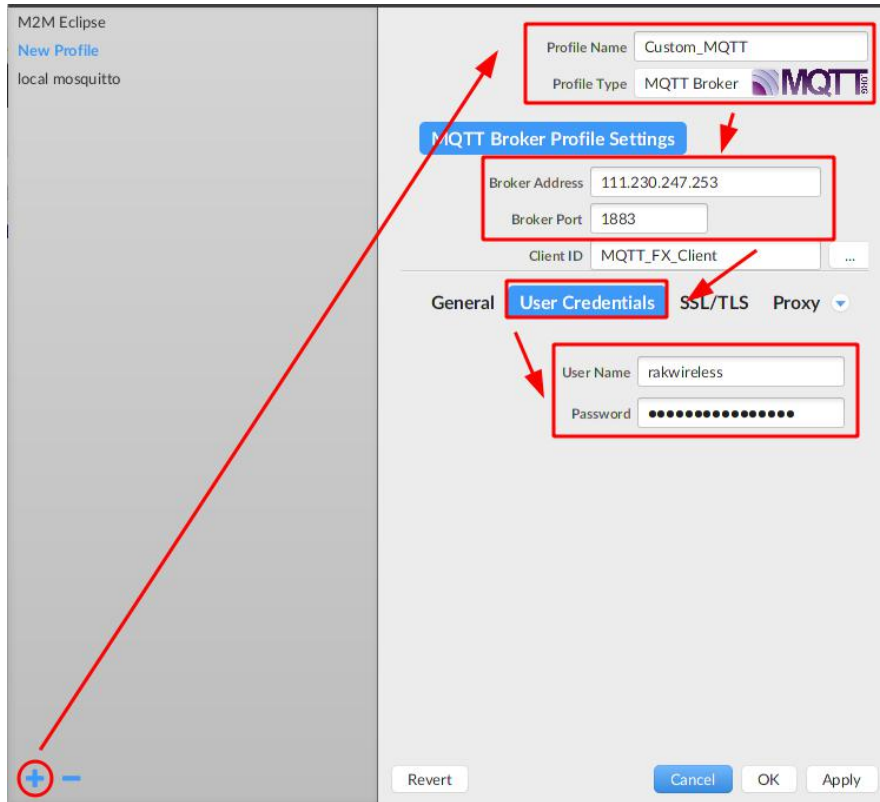


图 26

如图 27 所示，选择我们刚刚创建的 Profile，点击 Connect 按钮，即可成功连接到私有的 MQTT 服务器上。订阅与发布可参考上一小节 [3.1 商业网关内置 MQTT 服务器](#)

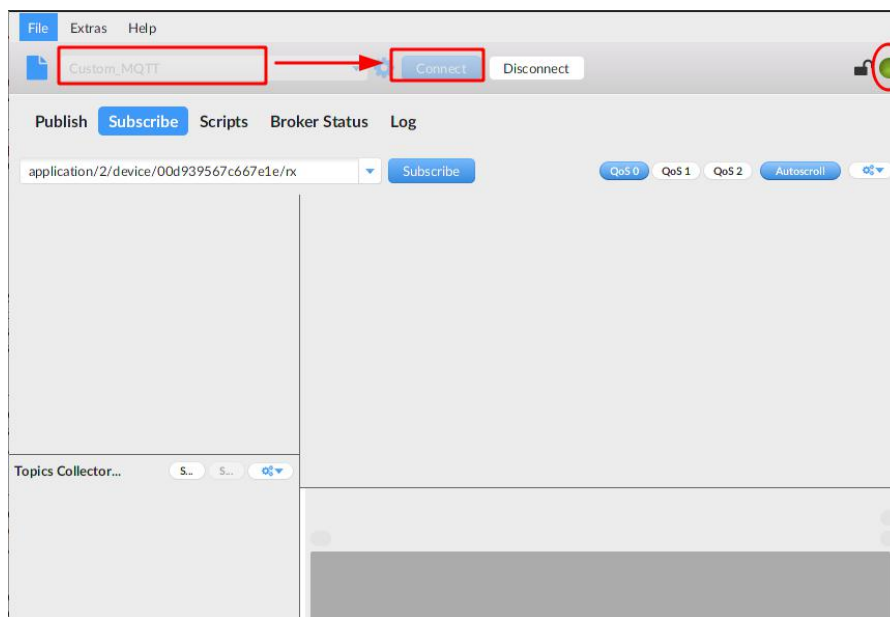


图 27



## 4. 通过 MQTT 获取的节点数据格式定义

在前面章节我们了解了通过 MQTT 如何订阅节点的上行数据 Uplink 以及如何向节点下发数据 Downlink，节点还有其他三种类型的数据，分别是 Join、Ack 和 Status。其中 Join 就是节点入网时的信息，Ack 是向节点下发数据之后节点回复的确认信息，Status 是节点的电池电量信息。

下面我们将分别介绍这五种数据的具体格式以及含义。

### 4.1. Uplink

```
{
  "applicationID": "1", // 节点所属应用的 id
  "applicationName": "test-app", // 节点所属应用的名称
  "devEUI": "3637343457387e11", // 节点的 devEUI
  "deviceName": "dev-5205", // 节点名称
  "timestamp": 1592730721, // 接收到节点数据的 Unix 时间戳
  "fCnt": 6,
  "fPort": 2,
  "data": "AQIDBA==", // base64 编码后的数据，解码之后就是节点实际上发的数据
  "data_encode": "base64", // 数据的编码类型
  "adr": true, // 节点是否开启了 adr
  "rxInfo": [ // 所有接收到节点数据的网关信息
```

```
{  
  
  "gatewayID": "d896e0fff010611e", // 网关的 gateway_id  
  
  "loRaSNR": 13.3, // 当前网关的信噪比  
  
  "rssi": -71, // 当前网关的 RSSI  
  
  "location": { // 对应网关的经纬度以及海拔信息  
  
    "latitude": 0,  
    "longitude": 0,  
    "altitude": 0  
  }  
}  
  
],  
  
"txInfo": {  
  
  "frequency": 486300000, // 节点发送数据使用的频率  
  
  "dr": 2 // 节点当前的 data rate  
  
}  
}
```

## 4.2. Downlink

```
{  
  
  "devEUI": "3637343457387e11", // 节点的 devEUI 信息  
  
  "confirmed": true, // This dl pkt need confirm or not.  
  
  "fPort": 2, // The port will be used for sending this packet  
  
  "data": "AgAAAA==" // 发送给节点的数据，经过 base64 编码
```

```
}
```

### 4.3. Join

```
{  
  "applicationID": "1",  
  "applicationName": "test-app",  
  "deviceName": "dev-5205",  
  "devEUI": "3637343457387e11",  
  "devAddr": "02000001" // Join 成功之后分配给节点的短地址  
}
```

### 4.4. Ack

注意：服务器只有给节点下发了 confirmed 类型的数据后，节点才会回复 Ack（注意：节点不一定立即回复 Ack，Ack 可能会在节点下一次发送上行数据是携带。）。

```
{  
  "applicationID": "1",  
  "applicationName": "test-app",  
  "deviceName": "dev-5205",  
  "devEUI": "3637343457387e11",  
  "acknowledged": true,  
  "fCnt": 7  
}
```

### 4.5. Device Status

```
{  
  "applicationID": "1",  
  "applicationName": "test-app",
```

```
"deviceName": "dev-5205",
```

```
"devEUI": "3637343457387e11",
```

"battery": 254, // 电池剩于电量的分级。254 表示电源满电状态，1 表示电池电量即将耗尽。

```
"margin": 8, // 是最近一次成功接收 DevStatusReq 命令的解调信噪比
```

```
"externalPowerSource": false, // 是否使用了额外的电源
```

```
"batteryLevelUnavailable": false, // 节点的电量级别是否有效
```

"batteryLevel": 100 // batteryLevelUnavailable 为 true 的情况下，batteryLevel 表示电量百分比

```
}
```

## 5. 程序示例

以下是使用 python 代码调用 mqtt 订阅节点的上发数据并将对应的内容打印出来，每收到一条上行数据会，程序会主动向节点发送一个下行数据，内容是“Hello rak”。使用代码前请仔细阅读代码注释。

以下源码基于 python3 运行环境，在运行代码前，需要使用命令 `pip3 install paho-mqtt`

安装依赖库。

```
#!/usr/bin/env python
```

```
import json
import base64
import paho.mqtt.client as mqtt
from datetime import datetime
```

```
# mqtt 服务器 IP
mqtt_ip = '111.230.247.253'

# mqtt 服务器端口
mqtt_port = 1883

# mqtt 用户名
mqtt_username = 'rakwireless'

# mqtt 密码
mqtt_password = 'rakwireless.com'

# mqtt 订阅 topic。该 topic 可以订阅所有节点信息
mqtt_rx_topic = 'application/+/device/+/rx'

# 将字符串转换为 16 进制
def str_to_hex(s):
    return r"\x"+r"\x'.join([hex(ord(c)).replace('0x', '') for c in s])

# 一旦订阅到消息，回调此方法
def on_message(mqttc, obj, msg):
    on_print_rak_node_info(msg.payload)

# 打印订阅到的节点信息
def on_print_node_rx_info(json_rx):
    try:
        devEUI      = json_rx['devEUI']
        applicationID      = json_rx['applicationID']
        applicationName = json_rx['applicationName']
        deviceName      = json_rx['deviceName']
        timestamp      = json_rx['timestamp']
        fCnt           = json_rx['fCnt']
        fPort          = json_rx['fPort']
        data           = json_rx['data']
        data_hex       = str_to_hex(base64.b64decode(data).decode("utf-8"))

        # 将时间戳转换为本地时间

        str_local_time = datetime.fromtimestamp(timestamp)

        print('----- devEUI:[%s] rxpk info -----' % devEUI)
```



```
print('+t applicationName:\t%s' % applicationName)
print('+t applicationID:\t\t%s' % applicationID)
print('+t deviceName:\t\t%s' % deviceName)
print('+t datetime:\t\t%s' % str_local_time)
print('+t fCnt:\t\t\t%d' % fCnt)
print('+t fPort:\t\t\t%d' % fPort)
print('+t data:\t\t\t%s' % data)
print('+t data_hex:\t\t%s' % data_hex)
print('-----')
```

```
except Exception as e:
    print(e)
finally:
    pass
```

# 订阅到节点的数据之后，向节点发送“Hello rak”字符串

```
def on_print_rak_node_info(payload):
```

```
    json_str = payload.decode()
```

```
    try:
```

```
        json_rx = json.loads(json_str)
```

```
        on_print_node_rx_info(json_rx)
```

```
        dev_eui = json_rx['devEUI']
```

```
        app_id = json_rx['applicationID']
```

```
    # 商业网关默认的 tx topic
```

```
    tx_topic = 'application/%s/device/%s/tx' % (app_id, dev_eui)
```

```
    str_hello = "Hello Rak"
```

```
    tx_msg = '{"confirmed":true,"fPort":10,"data":"%s"}' %
str(base64.b64encode(str_hello.encode("utf-8")), "utf-8")
```

```
    # 发布消息
```

```
    mqttc.publish(tx_topic, tx_msg, qos=0, retain=False)
```

```
    print('Send \'Hello rak\' to node %s' % dev_eui)
```

```
except Exception as e:
```

```
    raise e
```

```
finally:
```

```
    pass
```

```
mqttc = mqtt.Client()
mqttc.on_message = on_message

# 如果没有用户名和密码，请注释改行
mqttc.username_pw_set(mqtt_username, password=mqtt_password)

# 连接 mqtt broker，心跳时间为 60s
mqttc.connect(mqtt_ip, mqtt_port, 60)

mqttc.subscribe(mqtt_rx_topic, 0)

mqttc.loop_forever()
```

## 6. 修订历史

版本	描述	日期
1.0	创建文档	2020-07-03



### 关于瑞科慧联:

RAK 是一家专注于 IoT 领域以产品为驱动型的公司，凭借团队深厚的无线通讯技术领域的积累，采用创新的商业模式高效地为全球中小型的网络运营商(Network Operator)，系统集成商(System Integrator)和服务提供商(Service Provider)等提供高性能的 IoT 产品与应用方案。