

# MQTT Integration

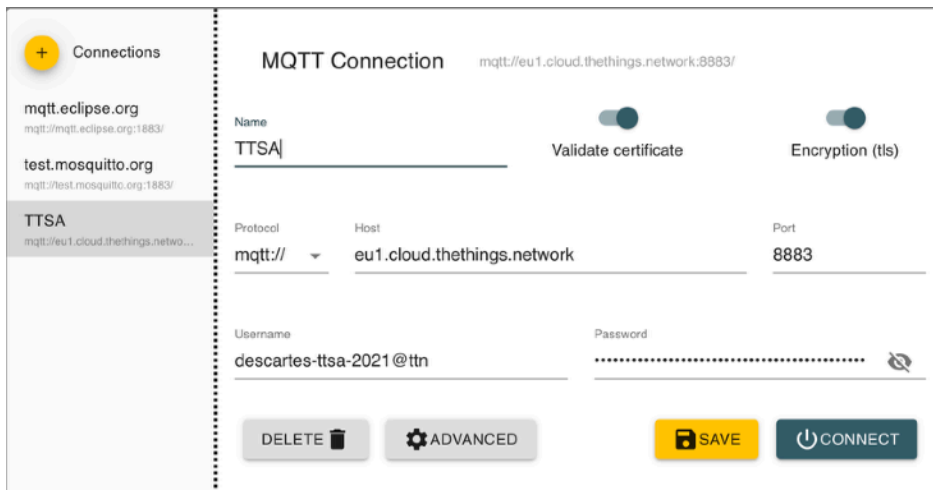
## 1. Get an API key

Go to the MQTT integration in the TTS console and click the [Generate new API key](#) button



Using the copy button, copy the two server addresses, Username and Password to a text document.

## 2. Create a connection in MQTT Explorer



The screenshot shows the MQTT Explorer interface. On the left, there's a 'Connections' sidebar with a yellow '+' button at the top. Below it, several connections are listed: 'mqtt.eclipse.org', 'test.mosquitto.org', and 'TTS' (selected). The main area is titled 'MQTT Connection' and shows the configuration for the selected 'TTS' connection. The URL bar shows 'mqtt://eu1.cloud.thethings.network:8883/'. The form includes fields for 'Name' (filled with 'TTS'), 'Protocol' (dropdown set to 'mqtt://'), 'Host' (filled with 'eu1.cloud.thethings.network'), 'Port' (filled with '8883'), 'Username' (filled with 'descartes-ttsa-2021@ttn'), and 'Password' (masked with dots). There are toggle switches for 'Validate certificate' and 'Encryption (tls)'. At the bottom, there are buttons for 'DELETE', 'ADVANCED', 'SAVE', and 'CONNECT'.



Click the + button and fill in the form with your information. You can name it as you wish. You should turn on Encryption. You will have to remove the :8883 from the end of the host that you copied.

Click Save and then Connect.

You will then be able to trigger an uplink on your device or from the console. This will show incoming information in the main window which you can drill down to the uplink and the JSON that TTS will have transmitted.

## 3. Do it in Python

From the GitHub zip file, open the MQTT-to-Tab-Python3 folder and open up the TTS.MQTT.Tab.py file in Idle (or other IDE of your choice).

Copy in your User and Password information and set the region to EU1, AU1 or NAM1 as appropriate.

Run the script!

Debugging and further activities on the next page ....

## Security / TLS / SSL / Certificate issues?

Different operating systems come with different security configurations. If you see any issues, sadly mostly from Windows, then you can either:

Change over to plain text transfers - acceptable for training & demo purposes:

- Comment out line 114, the `mqttc.tls_set()` line, with a `#` at the start of the line
- One line 122 change the 8883 in the connect parameters to 1883

OR you can download the certificate file `mqtt-ca.pem` from the link in the comments and put it next to the python script.

## TTS Documentation

<https://www.thethingsindustries.com/docs/integrations/mqtt/>

## Things to try

### Filtering the messages

Change the filter that MQTT uses when it requests information. You can copy this from MQTT Explorer, it will look something like this:

v3 / descartes-ttsa-2021@ttn / devices / ttsa-arduino-mkr-wan / up

`v3/descartes-ttsa-2021@ttn/devices/ttsa-arduino-mkr-wan/up`

Which says send a filter for:

- Application: `descartes-ttsa-2021@ttn`
- Device: `ttsa-arduino-mkr-wan`
- Messages: `up`

The Python is subscribing to `#` which is a wild card symbol that matches against everything from the place that it is used.

To filter on all devices for an application but just get the uplinks, you would use:

`v3/descartes-ttsa-2021@ttn/devices/+/up`

To filter on all messages for a device for an application:

v3/descartes-ttsa-2021@ttn/devices/ttsa-arduino-mkr-wan/#

## Sending a downlink

### Via MQTT Explorer

Instead of /up use /down/push with the following sample JSON in the Publish layout.

```
{
  "downlinks": [{
    "f_port": 15,
    "frm_payload": "vu8=",
    "priority": "NORMAL"
  }]
}
```

The frm\_payload is the bytes to send formatted as Base64. For a convertor tool, try <https://base64.guru/converter/decode/hex>

### Via Python

Store the JSON in a variable:

```
theDownlink = '{"downlinks":[{"f_port": 15,"frm_payload":"vu8=","priority":
"NORMAL"}]}'
```

Then use the publish command:

```
mqttc.publish("v3/descartes-ttsa-2021@ttn/devices/ttsa-adafruit-feather/
down/push", payload=theDownlink)
```