

# Node-Red Time Server

## Sentrius RS1xx LoRa Sensor

Application Note

v1.0

## 1 INTRODUCTION

The purpose of this document is to describe the procedure to setup and use node-red to connect to the MQTT broker provided in The Things Enterprise Stack (v3). It is assumed that the reader of this document is already able to register gateways and create applications on their TTS v3 server. If not, please consult [this application note](#).

## 2 MQTT SETUP IN TTS V3

In your application, on The Things Enterprise Stack (v3), create an API key to use for the MQTT integration. Under **Integration > MQTT**, click **Generate new API Key**. Make a note of all the values for address, username and password as they will be needed in node-red configuration. Password is not visible anymore once you leave this page.

For more general information, refer to The Things Enterprise Stack instructions for generating API keys. [Click here](#)

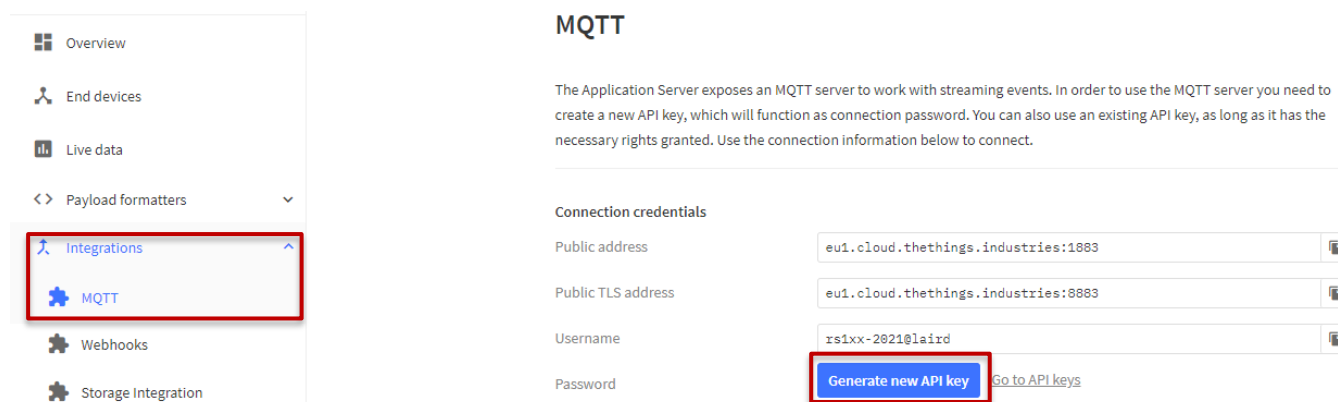


Figure 1: MQTT integration in TTS v3

## 3 NODE RED INSTALLATION

node-RED is a flow-based programming environment which is used to wire devices models, APIs, and online services together. Flow-based programming is a way of describing an application's behavior as a network of *nodes*. Each node receives data, does something with that data, and then passes the data onward. This network of nodes is set up using the node-RED palette web interface.

### 3.1 Pre-Requisites

Check if these ports are open by using the following links.

<http://portquiz.net:1880/>

<http://portquiz.net:1883/>

If they can't be reached, contact your IT department to open these ports on your network.

The port 1880 is used for the node-red user interface. It can be disabled again when you are done configuring your flow.

The port 1883 is used by node-red for the connection to the MQTT broker.

## 3.2 Installation

node-RED uses *node.js* to host the flow-based programming palette and to create a user web interface. [Click here](#) for more information on the node-RED tool.

### 3.2.1 Install Node.js

For Ubuntu, [click here](#) to see instructions for downloading and installing Node.js.

For Windows, [click here](#) to see instructions to install the *node.js* package manager (NPM) and *node.js* locally on your PC.

### 3.2.2 Install Node-Red

For Ubuntu, [click here](#) to see instructions for installing Node-Red. Follow the installation instructions for “Installing with npm”. Then follow the instructions for running Node-Red.

For Windows, [click here](#).

## 3.3 Configuring Node-Red application

[Click here](#) to download **timeserver\_tts\_v3.json**. Then run it on terminal using the following command.

```
node-red timeserver tts v3.json
```

If the node-red application is running from your local computer, the GUI can be accessed by browsing to <http://localhost:1880/> on that computer.

Note that a node is not connected to TTS v3 server yet.

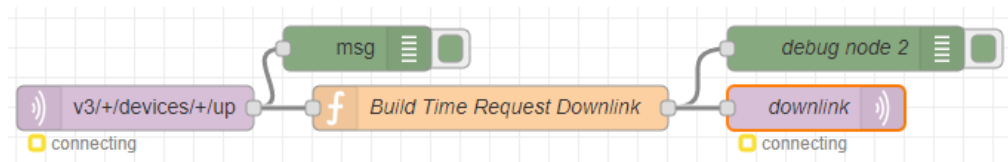


Figure 2: Node-red app

### 3.3.1 Configure TTS v3 server in node-red app

Go to the hamburger menu (☰) at the top-right corner and select **Configuration nodes**. Then, double-click mqtt-broker node. The URL for your MQTT broker may be different than the one in the example.



Figure 3: mqtt-broker node

On Connection tab, enter the **Public address** (Figure 1) acquired when creating the API key into the Server field.

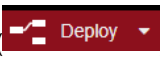
The screenshot shows the 'Connection' tab of a Node-Red configuration. The 'Server' field is set to 'eu1.cloud.thethings.industries' and the 'Port' field is set to '1883'. Both fields are highlighted with red boxes. Below these fields, there are several checkboxes: 'Enable secure (SSL/TLS) connection' (unchecked), 'Client ID' (set to 'Leave blank for auto generated'), 'Keep alive time (s)' (set to '60'), 'Use clean session' (checked), and 'Use legacy MQTT 3.1 support' (unchecked).

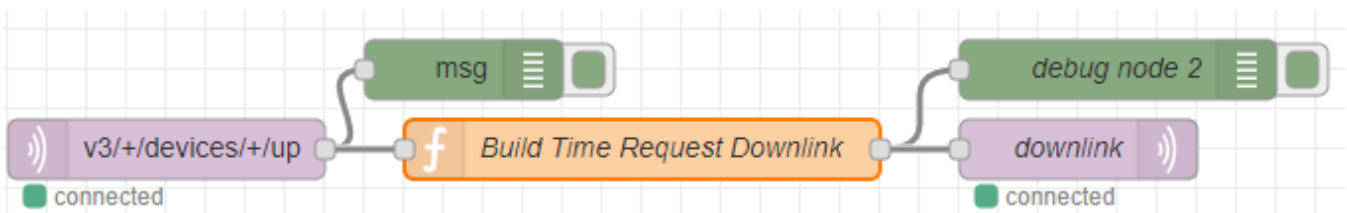
**Figure 4: Connection tab**

On Security tab, enter username and password acquired when creating API key.

The screenshot shows the 'Security' tab of a Node-Red configuration. The 'Username' and 'Password' fields are both empty and highlighted with red boxes. The 'Connection' and 'Messages' tabs are also visible above the input fields.

**Figure 5: Security tab**

Click deploy button (  ) to apply changes. Now, the status changed to connected.



**Figure 6: Node-red app connected to MQTT broker configured in TTS v3**

At this point, it is ready to receive uplinks and generate downlinks if the RTC time is requested by the sensor. The RS1xx sensors will only request the time once after power cycle. This happens on the first uplink after the join accept message. To force an RS1xx device to request the time for testing purposes, remove and replace the batteries in the unit. That will force the RS1xx to join the LoRaWAN network again and request the time in the first uplink after the join accept.

---

**Note:** To utilize backloging feature, set RS1xx's operating mode to **Laird** or **Laird2** via Sentrius mobile app.

---

## 4 MONITORING DATA

You can monitor downlink timestamp data on TTS v3. The below screenshot shows that TTS v3 sent down timestamp after receiving first uplink packet from RS1xx.



**Figure 7: Incoming/outgoing data on TTS v3**

Also, the node-red debug tab shows the downlink payload in base64 format under debug node 2.

```
6/7/2021, 10:00:29 AM node: debug node 2
v3/rs1xx-test@laird/devices/0025ca0a000001bc/down/push : msg.payload : string[77]
{"downlinks":
[{"f_port":1,"frm_payload":"AgEVBgcBAB0=", "priority":"NORMAL"}]}
```

**Figure 8: Generated downlink data in node-red app**

## 5 REVISION HISTORY

Version	Date	Notes	Approver
1.0	11 June 2021	Initial Release	Chris Boorman