RAK7248 Supported LoRa Network Servers

AWS IoT Core for LoRaWAN

Execute the following steps to set up your AWS account and permissions:

Set Up Your AWS Account and Permissions

If you don't have an AWS account, refer to the instructions in the guide here. The relevant sections are **Sign up for an AWS account** and **Create a user and grant permissions**.

Overview

The high-level steps to get started with AWS IoT Core for LoRaWAN are as follows:

1. Set up Roles and Policies in IAM
2. Add a Gateway (see section Add the Gateway to AWS IoT)
3. Add Device(s) (see section Add a LoRaWAN Device to AWS IoT)
    - Verify device and service profiles
    - Set up a Destination to which device traffic will be routed and processed by a rule.

These steps are discussed as you browse through this guide. For additional details, refer to the AWS LoRaWAN developer guide.

Add the Gateway to AWS IoT

Preparation

Refer to the online guide for steps required prior to onboarding your gateway. For more details, check the software section of the datasheet.

Frequency Band Selection and Role Setup

Refer to the online guide for information on selecting an appropriate frequency band.

📝 **NOTE**

LoRa® Frequency bands supported by RAK7248 are as follows:

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

You can select an appropriate frequency band from the RAK store .

Add the LoRaWAN Gateway

To register the gateway with AWS IoT Core for LoRaWAN, follow the steps in the online guide under the **Add a gateway using the console** section.

Add a LoRaWAN Device to AWS IoT

Preparation

- Go to the datasheet to learn more about the RAK4631 WisBlock LPWAN Module.
- Follow the steps in the online guide under the **Before onboarding your wireless device** section, then proceed to the **Add your wireless device to AWS IoT Core for LoRaWAN** section.

Verify Profiles

AWS IoT Core for LoRaWAN supports device profiles and service profiles. Device profiles contain the communication and protocol parameter values the device needs to communicate with the network server. Service profiles describe the communication parameters the device needs to communicate with the application server.

Some pre-defined profiles are available for device and service profiles. Before proceeding, verify that these profile settings match the devices you will be setting up to work with AWS IoT Core for LoRaWAN. For more details, refer to the online guide under the **Add profiles to AWS IoT Core for LoRaWAN** section.

> 📝 **NOTE**
>
> Proceed only if you have a device and service profile that will work for you.

Set Up a Destination for Device Traffic

Because most LoRaWAN devices don't send data to AWS IoT Core for LoRaWAN in a format that can be consumed by AWS services, traffic must first be sent to a Destination. A Destination represents the AWS IoT rule that processes a device's data for use by AWS services. This AWS IoT rule contains the SQL statement that selects the device's data and the topic rule actions that send the result of the SQL statement to the services that will use it.

For more information, refer to the online guide under sections **Add a destination using the console** and **Create an IAM role for your destinations**. Also, refer to **Create rules to process LoRaWAN device messages** section in the online guide .

Set Up the Gateway

- **Set Up the Gateway Hardware**
- **Set Up the Gateway Software**

For Additional Software References, check the following links:

- **FAQ**
- **Community Forum**

Configure the Gateway Device

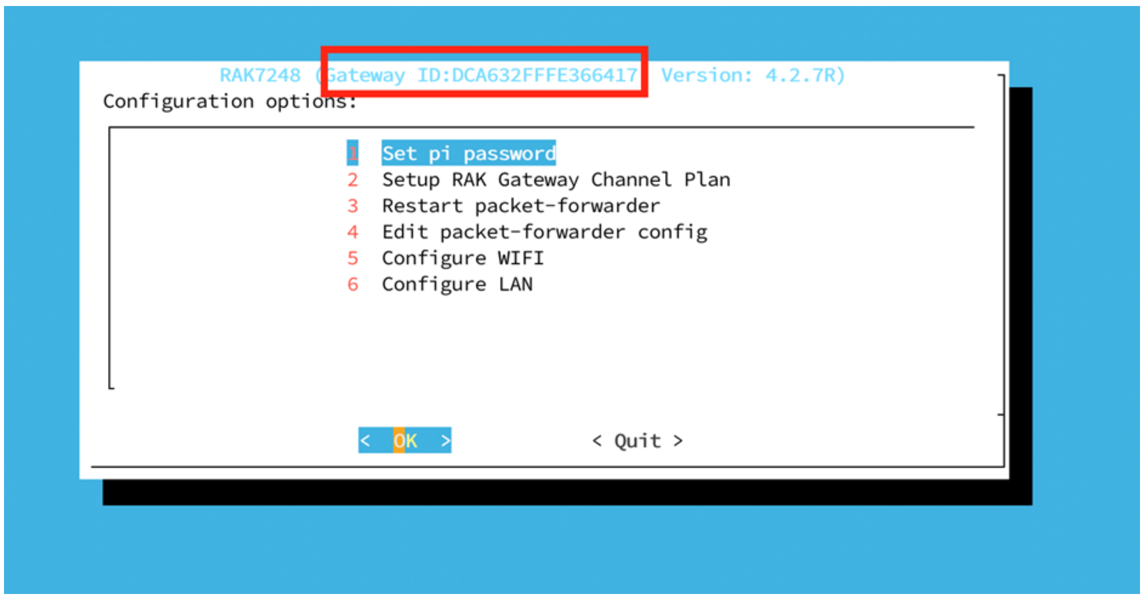
1. To connect to the AWS IoT Core for LoRaWAN, you need to download and compile **BasicsStation**. Assuming that you have successfully accessed the gateway via SSH, it is a good policy to update/upgrade. Doing so, it will update all the packages:

```sh
sudo apt update
```

```sh
sudo apt upgrade
```

2. Now, register the gateway in AWS IoT Core for LoRaWAN. To register your gateway, you will need the device's EUI. To find the gateway's EUI, run the command below to open the Graphics User Interface (GUI). The EUI can be found on the top of the GUI:

```sh
sudo gateway-config
```



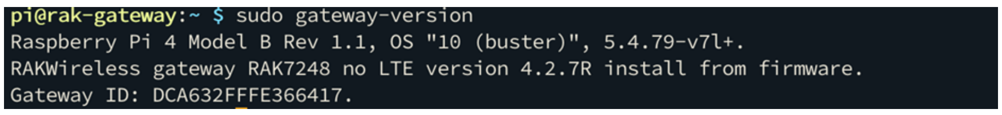
```
RAK7248 (Gateway ID:DCA632FFFE366417  Version: 4.2.7R)
Configuration options:

            1  Set pi password
            2  Setup RAK Gateway Channel Plan
            3  Restart packet-forwarder
            4  Edit packet-forwarder config
            5  Configure WIFI
            6  Configure LAN

        <  OK  >              < Quit >
```

Figure 1: RAK7248 Configuration Options

You can also run the following command:



```
pi@rak-gateway:~ $ sudo gateway-version
Raspberry Pi 4 Model B Rev 1.1, OS "10 (buster)", 5.4.79-v7l+.
RAKWireless gateway RAK7248 no LTE version 4.2.7R install from firmware.
Gateway ID: DCA632FFFE366417.
```

Figure 2: Gateway Version

```sh
pi@rak-gateway:~ $ sudo gateway-version
Raspberry Pi 4 Model B Rev 1.1, OS "10 (buster)", 5.4.79-v7l+.
RAKwireless gateway RAK7248 no LTE version 4.2.7R install from firmware
Gateway ID: DCA632FFFE366417
```

3. Only the configuration of the Basics station remains. Clone the Basics station repository and enter the downloaded folder:

```sh
git clone https://github.com/lorabasics/basicstation.git
cd basicstation
```

📝 **NOTE:**

Run the following commands in the basicstaion directory.

4. Install all the dependencies by running the following command:

```sh
make platform=corecell variant=std
```

5. Replace the file `loragw_stts751.c` file using following command:

```sh
sudo cp -f /opt/ttn-gateway/sx1302_hal/libloragw/src/loragw_stts751.c deps/lgw1302/platform-corec
```

6. Make a clean build in the basicstation directory, and then do a `make` to install all dependencies with the changes.

```sh
make clean
make platform=corecell variant=std
```

7. As the installation is complete, you need to configure the Basics Station protocol. Enter the folder with the given examples by Semtech and specifically corecell folder as the build is for SX1302.

```sh
cd examples/corecell
```

8. First, change the reset pin in the `reset_gw.sh` to 17 in order to successfully start the concentrator.

```sh
#!/bin/sh

# This script is intended to be used on SX1302 CoreCell platform, it performs the following actio
#         - export/unexport GPIO23 and GPIO18 used to reset the SX1302 chip and to enable the LDOs
#
# Usage examples:
#        ./reset_lgw.sh stop
#        ./reset_lgw.sh start
#
# GPIO mapping has to be adopted with HW
#

SX1302_RESET_PIN=17
SX1302_POWER_EN_PIN=18

WAIT_GPIO() {
    sleep 0.1
}
```

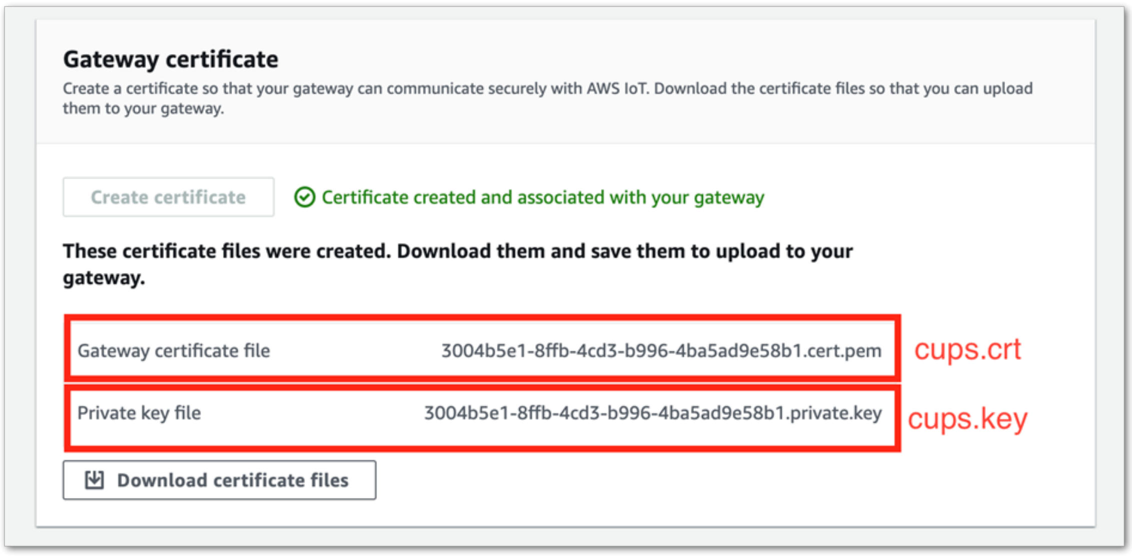
9. Create a new folder to store all the configuration files for basics station to connect to AWS IoT Core for LoRaWAN.

```sh
mkdir lns-aws
```

10. Copy `station.conf` into this folder.

```sh
cp lns-ttn/station.conf lns-aws/
```

11. Add the certificates that you downloaded earlier to **lns-aws**. (Refer to Add the LoRaWAN Gateway.)



Gateway certificate

Create a certificate so that your gateway can communicate securely with AWS IoT. Download the certificate files so that you can upload them to your gateway.

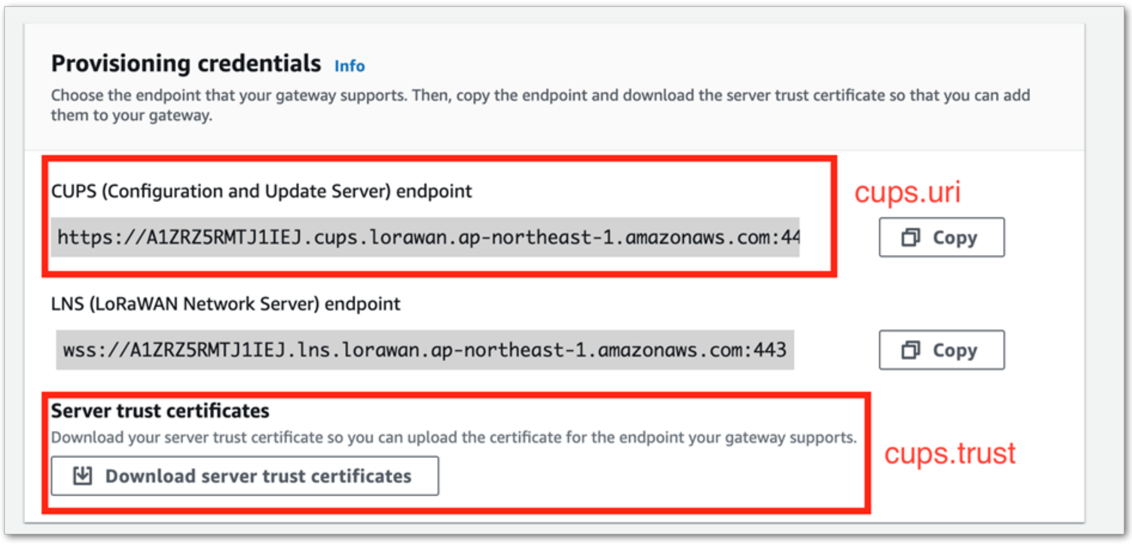
Create certificate Certificate created and associated with your gateway

These certificate files were created. Download them and save them to upload to your gateway.

Gateway certificate file 3004b5e1-8ffb-4cd3-b996-4ba5ad9e58b1.cert.pem cups.crt

Private key file 3004b5e1-8ffb-4cd3-b996-4ba5ad9e58b1.private.key cups.key

Figure 3: Gateway Certificate



Provisioning credentials Info

Choose the endpoint that your gateway supports. Then, copy the endpoint and download the server trust certificate so that you can add them to your gateway.

CUPS (Configuration and Update Server) endpoint cups.uri

https://A1ZRZ5RMTJ1IEJ.cups.lorawan.ap-northeast-1.amazonaws.com:44

LNS (LoRaWAN Network Server) endpoint

wss://A1ZRZ5RMTJ1IEJ.lns.lorawan.ap-northeast-1.amazonaws.com:443

Server trust certificates

Download your server trust certificate so you can upload the certificate for the endpoint your gateway supports. cups.trust

Figure 4: Provisioning Credentials

12. All the configurations about basics stations are ready. You should now have the files as follows:

```sh
pi@rak-gateway:~/basicstation/examples/corecell/lns-aws $ ls -al
total 28
drwxr-xr-x 2 pi pi 4096 Oct 28 17:57 .
drwxr-xr-x 4 pi pi 4096 Oct 28 17:44 ..
-rw-r--r-- 1 pi pi 1225 Oct 28 17:57 cups.crt
-rw-r--r-- 1 pi pi 1680 Oct 28 17:57 cups.key
-rw-r--r-- 1 pi pi 1607 Oct 28 17:57 cups.trust
-rw-r--r-- 1 pi pi   69 Oct 28 17:57 cups.uri
-rw-r--r-- 1 pi pi 2460 Oct 28 17:46 station.conf
```

13. Before you start using basics station, disable the packet forwarder. Use the command `sudo systemctl edit ttn-gateway.service` to edit the service of packet forwarder, and add the following codes to the file.

> 📝 **NOTE:**
>
> If you want to re-activate the packet forwarder, you can just set the value of **Restart** to **always**.

```sh
[Unit]
Description=The Things Network Gateway

[Service]
WorkingDirectory=/opt/ttn-gateway/packet_forwarder/lora_pkt_fwd/
ExecStart=/opt/ttn-gateway/packet_forwarder/lora_pkt_fwd/start.sh
SyslogIdentifier=ttn-gateway
Restart=no
RestartSec=5

[Install]
WantedBy=multi-user.target
```

14. To apply the changes made to the unit, execute the following command:

```sh
sudo systemctl daemon-reload
```

15. Restart the ttn-gateway service to load the new service configuration. Unit file must be restated if you modify the running unit file.

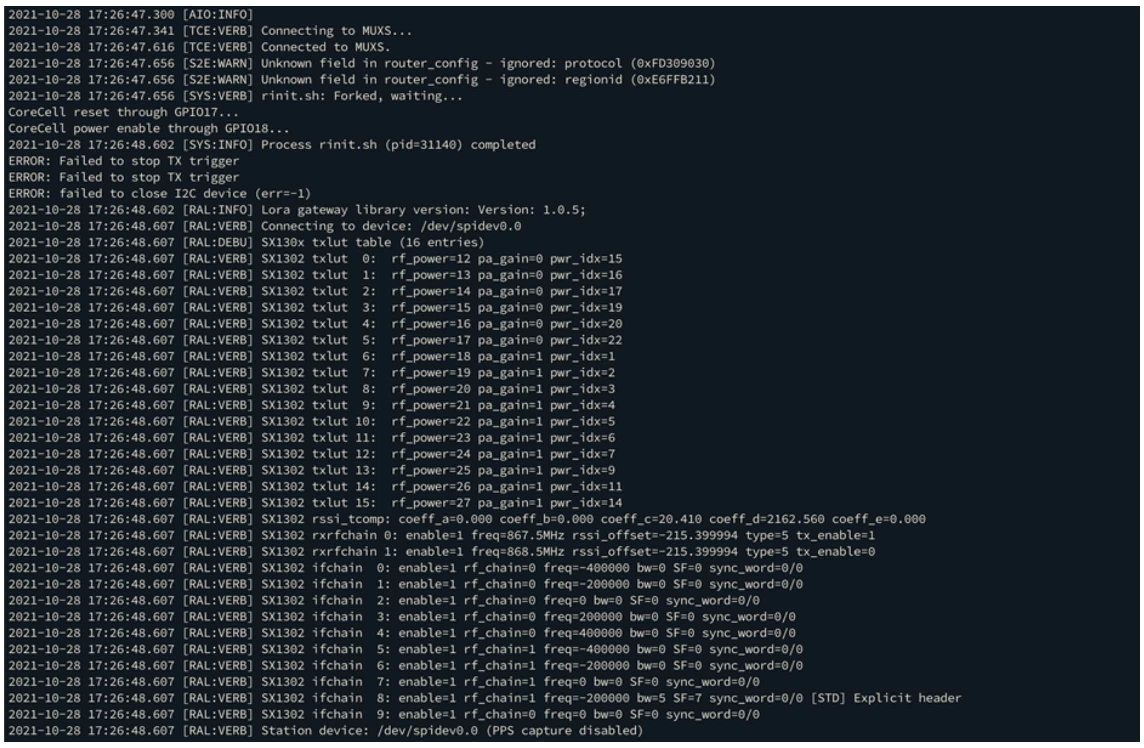
```sh
sudo systemctl restart ttn-gateway.service
```

16. Find the ttn-gateway process PID and `kill` this process:

```sh
pi@rak-gateway:~/basicstation/example/corecell/lns-aws $ px aux | grep ttn-gateway
pi      28165   0.2     0.1     7680        2780    pts/0   S+  18:20   0:00 /bin/bash/opt/ttn-ga
pi      28236   0.0     0.0     7348         488    pts/1   S+  18:20   0:00 grep --color=auto tt
pi@rak-gateway:~/basicstation/example/corecell/lns-aws $ sudo kill 28165
```

17. Now, you can start basics station. Exit the configuration folder and start basics station.

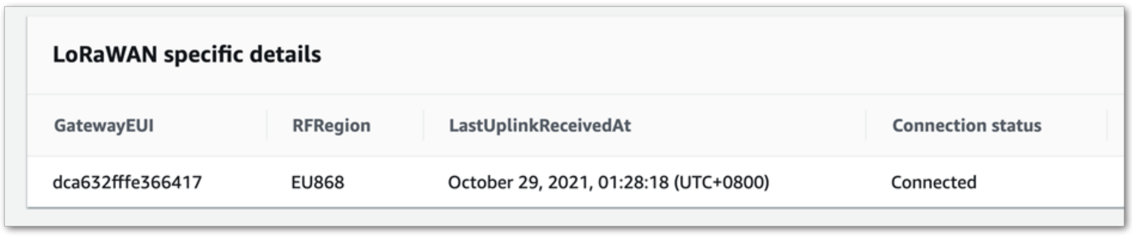
```sh
cd ..
./start-station.sh –l ./lns-aws
```



```
2021-10-28 17:26:47.300 [AIO:INFO]
2021-10-28 17:26:47.341 [TCE:VERB] Connecting to MUXS...
2021-10-28 17:26:47.616 [TCE:VERB] Connected to MUXS.
2021-10-28 17:26:47.656 [S2E:WARN] Unknown field in router_config - ignored: protocol (0xFD309030)
2021-10-28 17:26:47.656 [S2E:WARN] Unknown field in router_config - ignored: regionid (0xE6FFB211)
2021-10-28 17:26:47.656 [SYS:VERB] rinit.sh: Forked, waiting...
CoreCell reset through GPIO17...
CoreCell power enable through GPIO18...
2021-10-28 17:26:48.602 [SYS:INFO] Process rinit.sh (pid=31140) completed
ERROR: Failed to stop TX trigger
ERROR: Failed to stop TX trigger
ERROR: failed to close I2C device (err=-1)
2021-10-28 17:26:48.602 [RAL:INFO] Lora gateway library version: Version: 1.0.5;
2021-10-28 17:26:48.607 [RAL:VERB] Connecting to device: /dev/spidev0.0
2021-10-28 17:26:48.607 [RAL:DEBU] SX130x txlut table (16 entries)
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  0:  rf_power=12 pa_gain=0 pwr_idx=15
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  1:  rf_power=13 pa_gain=0 pwr_idx=16
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  2:  rf_power=14 pa_gain=0 pwr_idx=17
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  3:  rf_power=15 pa_gain=0 pwr_idx=19
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  4:  rf_power=16 pa_gain=0 pwr_idx=20
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  5:  rf_power=17 pa_gain=0 pwr_idx=22
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  6:  rf_power=18 pa_gain=1 pwr_idx=1
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  7:  rf_power=19 pa_gain=1 pwr_idx=2
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  8:  rf_power=20 pa_gain=1 pwr_idx=3
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut  9:  rf_power=21 pa_gain=1 pwr_idx=4
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut 10:  rf_power=22 pa_gain=1 pwr_idx=5
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut 11:  rf_power=23 pa_gain=1 pwr_idx=6
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut 12:  rf_power=24 pa_gain=1 pwr_idx=7
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut 13:  rf_power=25 pa_gain=1 pwr_idx=9
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut 14:  rf_power=26 pa_gain=1 pwr_idx=11
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 txlut 15:  rf_power=27 pa_gain=1 pwr_idx=14
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 rssi_tcomp: coeff_a=0.000 coeff_b=0.000 coeff_c=20.410 coeff_d=2162.560 coeff_e=0.000
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 rxrfchain 0: enable=1 freq=867.5MHz rssi_offset=-215.399994 type=5 tx_enable=1
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 rxrfchain 1: enable=1 freq=868.5MHz rssi_offset=-215.399994 type=5 tx_enable=0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  0: enable=1 rf_chain=0 freq=-400000 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  1: enable=1 rf_chain=0 freq=-200000 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  2: enable=1 rf_chain=0 freq=0 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  3: enable=1 rf_chain=0 freq=200000 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  4: enable=1 rf_chain=0 freq=400000 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  5: enable=1 rf_chain=1 freq=-400000 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  6: enable=1 rf_chain=1 freq=-200000 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  7: enable=1 rf_chain=1 freq=0 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  8: enable=1 rf_chain=1 freq=-200000 bw=5 SF=7 sync_word=0/0 [STD] Explicit header
2021-10-28 17:26:48.607 [RAL:VERB] SX1302 ifchain  9: enable=1 rf_chain=0 freq=0 bw=0 SF=0 sync_word=0/0
2021-10-28 17:26:48.607 [RAL:VERB] Station device: /dev/spidev0.0 (PPS capture disabled)
```

Figure 5: Configuring the Basics Station

If everything is configured properly, your gateway should be online in the AWS IoT Core for LoRaWAN console:



LoRaWAN specific details

GatewayEUI	RFRegion	LastUplinkReceivedAt	Connection status
dca632fffe366417	EU868	October 29, 2021, 01:28:18 (UTC+0800)	Connected

Figure 6: Connected Status in the LoRaWAN Console

Add End-Devices

Refer to RAK4631 Quick Start Guide to enable communication with the gateway.

Connect the Device and Verify the Connection Status

Follow the instructions in the online guide to connect your device to AWS IoT Core for LoRaWAN. To verify the connection status, refer to the instructions in the **Check device connection status using the console** section. You can also check **View format of uplink messages sent from LoRaWAN devices** .

Verifying Operation

Once setup is completed, provisioned OTAA devices can join the network and start to send messages. Messages from devices can then be received by AWS IoT Core for LoRaWAN and forwarded to the IoT Rules Engine.

Instructions for a sample Hello World application are given below, assuming that the device has joined and is capable of sending uplink traffic.

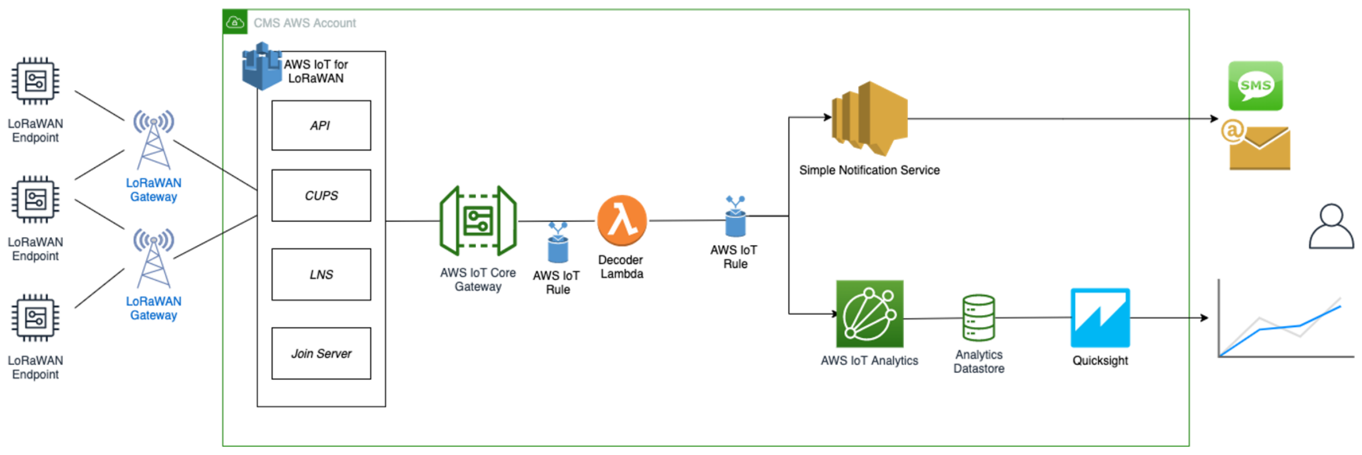


Figure 7: Sending Uplink Architecture

Create a Lambda Function for Destination Rule

Create the lambda function to process device messages processed by the destination rule.

1. Go to the AWS Lambda console .
2. In the navigation pane, click on **Functions**, then **Create function**.
3. Select **Author from scratch**.
4. Under **Basic Information**, enter the function name and choose ***Runtime Python 3.8***. from the drop-down under **Runtime**.
5. Click on **Create function**.
6. Under **Function code**, paste the copied code into the editor under the ***lambda_function.py*** tab.

```py
import base64
import json
import logging
import ctypes
import boto3

# define function name

FUNCTION_NAME = 'RAK-HelloWorld'

# Second Byte in Payload represents Data Types
# Low Power Payload Reference: https://developers.mydevices.com/cayenne/docs/lora/

DATA_TYPES = 1

# Type Temperature

TYPE_TEMP = 0x67

# setup iot-data client for boto3

client = boto3.client('iot-data')

# setup logger

logger = logging.getLogger(FUNCTION_NAME)
logger.setLevel(logging.INFO)

def decode(event):
    data_base64 = event.get('PayloadData')
    data_decoded = base64.b64decode(data_base64)
    result = {
        'devEui': event.get('WirelessMetadata').get('LoRaWAN'
                ).get('DevEui'),
        'fPort': event.get('WirelessMetadata').get('LoRaWAN'
                ).get('FPort'),
        'freq': event.get('WirelessMetadata').get('LoRaWAN'
                ).get('Frequency'),
        'timestamp': event.get('WirelessMetadata').get('LoRaWAN'
                ).get('Timestamp'),
        }

    if data_decoded[DATA_TYPES] == TYPE_TEMP:
        temp = data_decoded[DATA_TYPES + 1] << 8 \
            | data_decoded[DATA_TYPES + 2]
        temp = ctypes.c_int16(temp).value
        result['temperature'] = temp / 10

    return result


def lambda_handler(event, context):
    data = decode(event)
    logger.info('Data: %s' % json.dumps(data))
    response = client.publish(topic=event.get('WirelessMetadata'
                            ).get('LoRaWAN').get('DevEui')
                            + '/project/sensor/decoded', qos=0,
                            payload=json.dumps(data))
    return response
```

7. Once the code has been pasted, choose **Deploy** to deploy the lambda code.

8. Click on the **Permissions** tab of the lambda function.

9. Change the **Lambda Role Policy** permission.

   - Under **Execution role**, click on the hyperlink under **Role name**.
   - On the **Permissions tab**, find the policy name and select it.
   - Choose **Edit policy**, and choose the **JSON** tab.
   - Append the following to the Statement section of the policy to allow publishing to AWS IoT.

   ```json
   {
   "Effect":"Allow",
   "Action":[
       "iot:Publish"
   ],
   "Resource":[
       "*"
   ]
   }
   ```

- Choose **Review Policy**, then Save changes.

11. Create a test event that will allow you to test the functionality of the lambda function.
    - In the drop-down, for the ***Select a test event***, choose **Configure test events**.
    - Enter a name for the test event under the **Event name**.
    - Paste the following sample payload in the area under Event name:

```json
    {
    "WirelessDeviceId": "65d128ab-90dd-4668-9556-fe47c589610b",
    "PayloadData": "AWf/1w==",
    "WirelessMetadata": {
    "LoRaWAN": {
    "DataRate": "4",
    "DevEui": "0000000000000088",
    "FPort": 1,
    "Frequency": "868100000",
    "Gateways": [
            {
    "GatewayEui": "80029cffXXXXXXXX",
    "Rssi": -109,
    "Snr": 5
            }
        ],
    "Timestamp": "2021-02-08T04:00:40Z"
        }
    }
    }
```

12. Choose **Create** to save the event.
13. Navigate to the AWS IoT console, choose **Test** on the navigation pane, and select **MQTT client**.
14. Configure the MQTT client to subscribe to "#" (all topics).
15. Click on **Test** in the Lambda function page to generate the test event you just created.
16. Verify the published data in the AWS IoT Core MQTT Test client:

- Open another window. Go to **AWS IoT Console**, select **Test** under Subscription Topic, **enter #** and select to **Subscribe to topic**.
- The output should look similar to this:

```json
0000000000000000088/project/sensor/decoded      February 09, 2021, 14:45:29 (UTC+0800)
{
    "devEui": "0000000000000000088",
    "fPort": 1,
    "freq": "868100000",
    "timestamp": "2021-02-08T04:00:40Z",
    "temperature": -4.1
}
```

Create the Destination Rule

In this section, you create the IoT rule that forwards the device payload to your application. This rule is associated with the destination created earlier in Set up a Destination for Device Traffic section.

1. Navigate to the AWS IoT console .
2. In the navigation pane, choose **Act**, then select **Rules**.
3. On the Rules page, choose **Create**.
4. On the Create a rule page, enter as follows:
    - Name: **LoRaWANRouting**
    - Description: **Any description of your choice**.

> 📝 **NOTE:**
>
> The **Name of your Rule** is the information needed when you provision devices to run on AWS IoT Core for LoRaWAN.

5. Leave the default Rule query statement '**SELECT * FROM 'iot/topic**' unchanged. This query has no effect at this time, as traffic is currently forwarded to the rules engine based on the destination.
6. Under **Set one or more actions**, choose **Add action**.
7. On the Select an action page, choose **Republish a message to an AWS IoT topic**. Scroll down, and choose **Configure action**.
8. On the Configure action page, for Topic, enter ***project/sensor/decoded***.The AWS IoT Rules Engine will forward messages to this topic.
9. Under **Choose or create a role to grant AWS IoT access to perform this action**, select **Create Role**.
10. For Name, enter a name of your choice.
11. Choose **Create role** to complete the role creation. You will see a "**Policy Attached**" tag next to the role name, indicating that the Rules Engine has been permitted to execute the action.
12. Choose **Add action**.
13. Add one more action to invoke the Lambda function. Under **Set one or more actions**, choose **Add action**.

14. Choose **Send a message to a Lambda function**.

15. Choose **Configure action**.

16. Select the Lambda function created earlier and choose **Add action**.

17. Then, choose **Create rule**.

18. A "**Success**" message will be displayed at the top of the panel, and the destination has a rule bound to it.

You can now check that the decoded data is received and republished by AWS by triggering a condition or event on the device itself.

- Go to the AWS IoT console. In the navigation pane, select **Test**, and choose **MQTT client**.
- Subscribe to the wildcard topic # to receive messages from all topics.
- Send message from endDevice using AT command: `at+send:lora:1:01670110`.
- You should see traffic similar as shown below:

```json
    393331375d387505/project/sensor/decoded           February 09, 2021, 14:47:21 (UTC+0800)
    {
    "devEui": "393331375d387505",
    "fPort": 1,
    "freq": "867100000",
    "timestamp": "2021-02-09T06:47:20Z",
    "temperature": 27.2
    }
```

```json
project/sensor/decoded    February 09, 2021, 14:47:21 (UTC+0800)
    {
        "WirelessDeviceID": "6477ec22-9570-31d5981da021",
        "PayloadData": "AWcBEA==",
        "WirelessMetadata": {
            "LoRaWAN": {
                "DataRate": "4",
                "DevEui": "393331375d387505",
                "FPort": 1,
                "Frequency": "867100000",
                "Gateways": [
                    {
                        "GatewayEui": "ac1ff09fffe014bd5",
                        "Rssi": -103,
                        "Snr": 8.5
                    }
                ],
                "Timestamp": "2021-02-09T06:47:20Z"
            }
        }
    }
```

Configuring Amazon SNS

You will be using the Amazon Simple Notification Service to send text messages (SMS) when certain conditions are met.

1. Go to the Amazon SNS console .
2. Click on the menu to open the navigation pane.
3. Select **Text Messaging** (SMS) and choose **Publish text message**.
4. Under the Message type, select **Promotional**.
5. Enter your phone number (phone number that will receive text alerts).
6. Enter **Test message** for the Message and choose **Publish** message.
7. If the phone number you entered is valid, you will receive a text message and your phone number will be confirmed.
8. Create an Amazon SNS Topic as follows:
   - In the navigation pane, choose **Topics**.
   - Select **Create topic**.
   - Under Details, select **Standard**.
   - Enter a name of your choice. Here, you will use "***text_topic***".
   - Choose **Create topic**.
9. Create a subscription for this topic:
   - On the page for the newly created text_topic, choose the **Subscriptions** tab.
   - Choose **Create subscription**.
   - Select **Protocol** as SMS from the drop-down.
   - Under Endpoint, enter the previously validated phone number to receive the SMS alerts.
   - Choose **Create subscription**. You should see a "***Subscription to text_topic created successfully***" message.

Add a Rule for Amazon SNS Notification

Now, add a new rule to send an Amazon SNS notification when certain conditions are met in a decoded message.

1. Navigate to the AWS IoT console .
2. In the navigation pane, choose **Act**, then **Rules**.
3. On the Rules page, choose **Create**.
4. Enter the Name as *text_alert* and provide an appropriate **Description**.
5. Under the **Rule query statement**, enter the following query:

```
    SELECT devEui as device_id, "Temperature exceeded 25" as message, temperature as temp, timest
```

6. Choose **Add action**.
7. Choose **Send a message as an SNS push notification**.
8. Choose **Configure action**.
9. Under SNS target, select *text_topic* from the drop-down.
10. Select **RAW** under **Message format**.
11. Under **Choose or create a role to grant AWS IoT access to perform this action**, choose **Create role**.
12. Enter a name for the role, and choose **Add action**.
13. Select **Create rule**. You should see a "**Success**" message, indicating that the rule has been created.

Test the Rule for Amazon SNS Notification

After adding the rule for Amazon SNS notification, you should receive a text message when hitting the event.

Send message from end-device using AT command: `at+send:lora:1:01670110` . Here is the message from mobile after sending an uplink message:

```json
{
    "device_id": "393331375d387505",
    "message": "Temperature exceeded 25",
    "temp": 27.2,
    "time": "2021-02-22T07:58:54Z"
}
```

Send Downlink Payload

This section shows how to send downlink payload from AWS IoT LoRaWAN Server to end-device.

Install the AWS SAM CLI

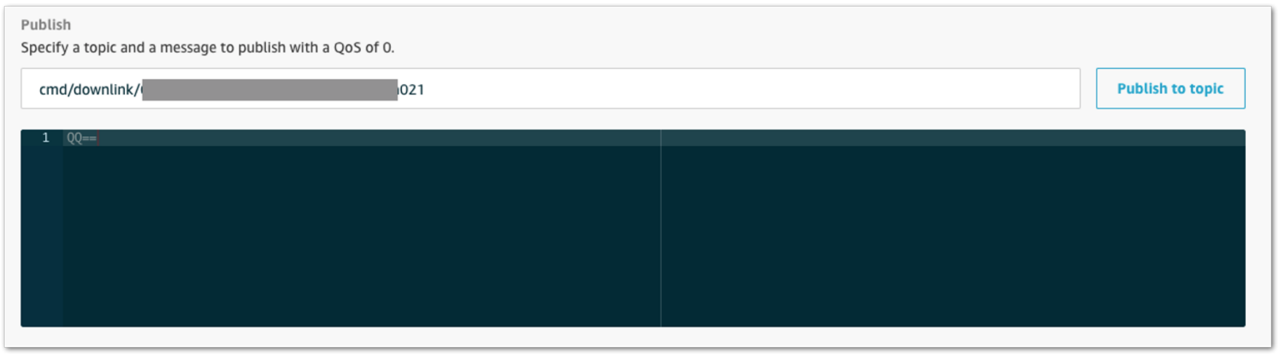
Follow the instruction in the online guide to install the [AWS SAM CLI.

Deploy the SAM Template to AWS

Follow the instruction in the Deploy SAM template to AWS GitHub.

Send Payload to End-Device

1. Send Payload to End-device.
    - Go to the AWS IoT console.
    - In the navigation pane, select **Test**, and choose **MQTT client**.
    - Subscribe to the wildcard topic # to receive messages from all topics.
    - Specify the topic to `cmd/downlink/{WirelessDeviceId}` and a base64-encoded message.


Publish

Specify a topic and a message to publish with a QoS of 0.

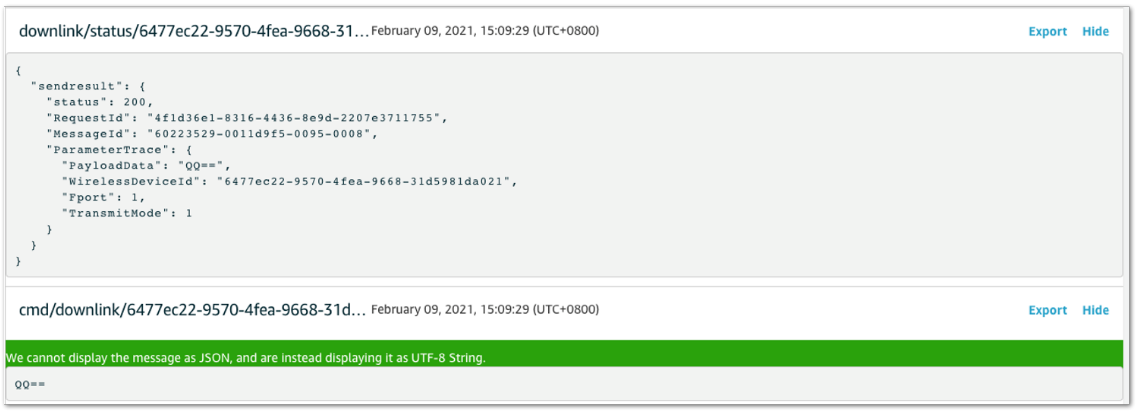
cmd/downlink/[redacted]021

```
QQ==
```

Figure 8: Specifying a topic

4. You should see traffic on AWS similar as shown below:

```json
downlink/status/6477ec22-9570-4fea-9668-31d5981da021   February 09, 2021, 15:09:29 (UTC+08
{
    "sendresult": {
        "status": 200,
        "RequestId": "4f1d36e1-8316-4436-8e9d-2207e3711755",
        "MessageId": "60223529-0011d9f5-0095-0008",
        "ParameterTrace": {
            "PayloadDate": "QQ==",
            "WirelessDeviceId": "6477ec22-9570-4fea-9668-31d5981da021",
            "Fport": 1,
            "TransmitMode": 1
        }
    }
}
```


downlink/status/6477ec22-9570-4fea-9668-31… February 09, 2021, 15:09:29 (UTC+0800)

```
{
  "sendresult": {
    "status": 200,
    "RequestId": "4f1d36e1-8316-4436-8e9d-2207e3711755",
    "MessageId": "60223529-0011d9f5-0095-0008",
    "ParameterTrace": {
      "PayloadData": "QQ==",
      "WirelessDeviceId": "6477ec22-9570-4fea-9668-31d5981da021",
      "Fport": 1,
      "TransmitMode": 1
    }
  }
}
```

cmd/downlink/6477ec22-9570-4fea-9668-31d… February 09, 2021, 15:09:29 (UTC+0800)

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

```
QQ==
```

Figure 9: Traffic on AWS

5. You should see traffic on your console of end-device similar as shown below:

```
SYSLOG:4:LoRa rX : 41 - 14
SYSLOG:4:LoRa Tx :
```

IoT Analytics

You will use IoT Analytics to visually display data via graphs if there is a need in the future to do further analysis.

Create an IoT Analytics Rule

Create a Rule First

1. Navigate to the AWS IoT console .
2. In the navigation pane, choose **Act** and then, choose **Rules**.
3. On the Rules page, choose **Create**.
4. Enter the Name as **Visualize** and provide an appropriate **Description**.
5. Under the Rule query statement, enter the following query:

```
SELECT * FROM 'project/sensor/decoded'
```

6. Choose **Add action**.

7. Select **Send a message to IoT Analytics**.
8. Choose **Configure Action**.
9. Choose **Quick Create IoT Analytics Resources**.
10. Under **Resource Prefix**, enter an appropriate prefix for your resources, such as *LoRa Choose Quick Create*.
11. Once the Quick Create Finished message is displayed, choose **Add action**.
12. Choose **Create rule**. You should see a Success message, indicating that the rule has been created.

Configure AWS IoT Analytics

Set Up AWS IoT Analytics

1. Go to the AWS IoT Analytics console .
2. In the navigation panel, choose **Data sets**.
3. Select the data set generated by the Quick Create in **Create an IoT Analytics Rule**.
4. In the Details section, edit the **SQL query**.
5. Replace the query with as follows:

```
SELECT devEui as device_id, temperature as temp, timestamp as time FROM LoRa_datastore
```

6. Under Schedule, choose **Add schedule**.
7. Under Frequency, choose **Every 1 minute**, and then click **Save**.

Configure Amazon QuickSight

Amazon QuickSight lets you easily create and publish interactive BI dashboards that include Machine Learning-powered insights.

1. Go to AWS Management console .
2. From the management console, enter **QuickSight** in the "*Search for services, features..*" search box.
3. Click on **QuickSight** in the search results.
4. If you haven't signed up for the service before, go ahead and sign up, as there is a free trial period.
5. Select the **Standard Edition**, and choose **Continue**.
6. Enter a unique name in the field **QuickSight account name**.
7. Fill in the **Notification email address**.
8. Review the other checkbox options and change them as necessary. The **AWS IoT Analytics** option must be selected.
9. Choose **Finish**. You will see a confirmation message.
10. Choose **Go to Amazon QuickSight**.
11. Select **Datasets**.
12. Select **New dataset**.
13. Select **AWS IoT Analytics**.
14. Under Select an AWS IoT Analytics data set to import, choose the data set created in **Create an IoT Analytics Rule**.
15. Choose **Create data source**, and then choose **Visualize**.
16. Select the dataset created, then select **Refresh** or **Schedule Refresh** for a periodic refresh of the dataset.

Testing Your "Hello Word" Application

Using your device, create a condition to generate an event such as a high-temperature condition. If the temperature is above the configured threshold, then you will receive a text alert on your phone. This alert will include key parameters about the alert.

You can also visualize the data set as follows:

1. Go to the AWS IoT Analytics console .
2. Choose **Data sets**.
3. Select the dataset created earlier.
4. Select **Content** and ensure there are at least few uplink entries available in the data set.
5. Go to the **QuickSight console** .
6. Choose **New analysis**.
7. Choose the dataset created in **Create an IoT Analytics Rule**.
8. To see a chart of your dataset, select the following values:
   - **Time** on the X-axis
   - **Value** as temp (Average)
   - **Color** as device_id.

Debugging

If you experience any issues, you can check the logs located in the `/var/log/` directory.

Troubleshooting

- Unable to see the web login:
  - Check that your wifi is connected to **RAK7Wireless_XXXX**.
  - Try ping **192.168.230.1**.

Last Updated: 12/3/2021, 2:11:13 AM