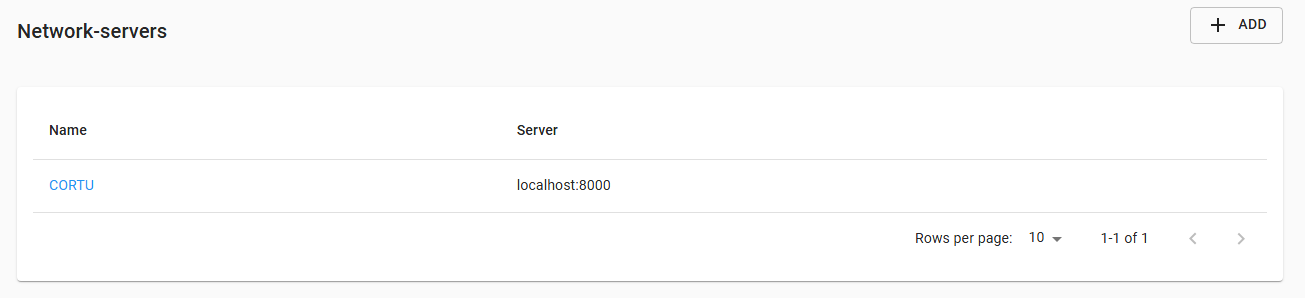
# LORAWAN GATEWAY CONFIGURATIONS

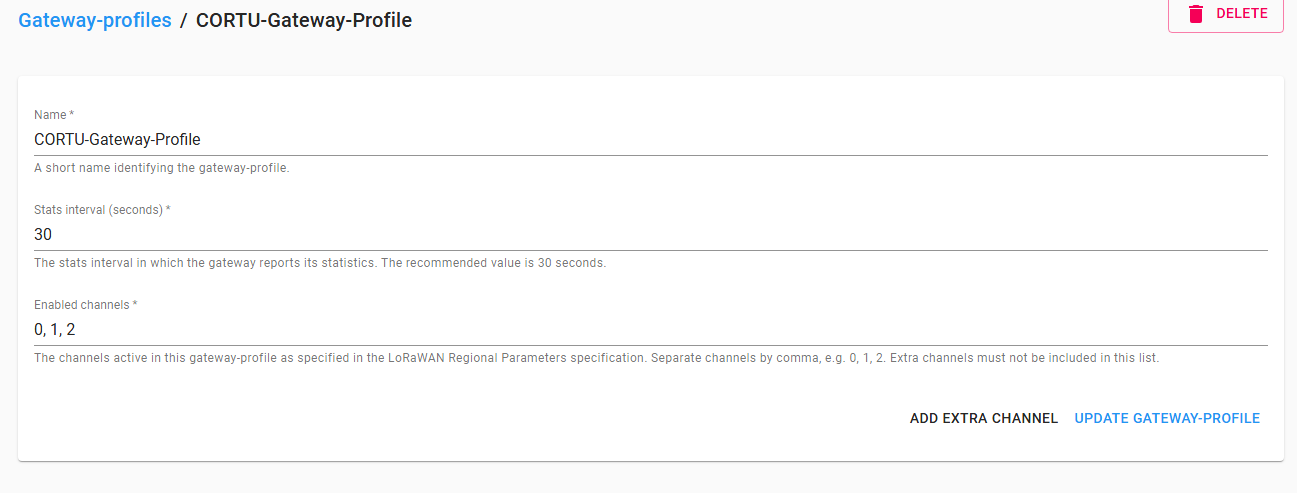
* It is connected to my phone hotspot in client mode
* I type in the new IP address it gets from my phone to putty, since it is now a client it will have a new client IP address
* I then sign in via ssh using the default parameters

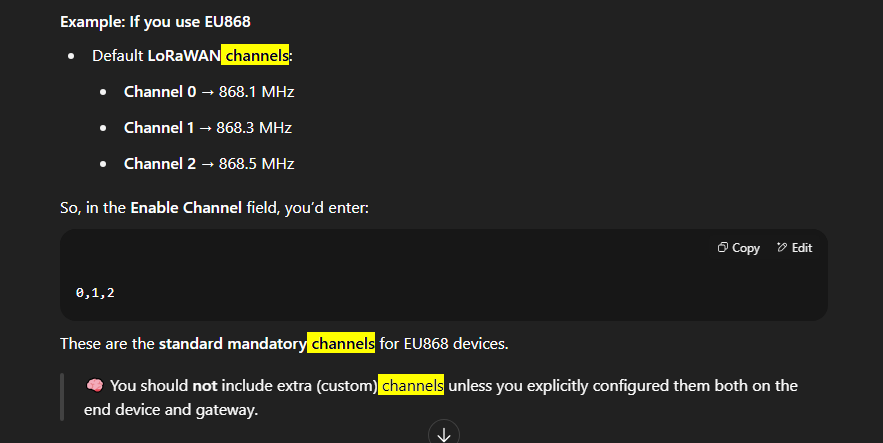
## GATEWAY CONFIG PARAMTERS

## Chripstack server

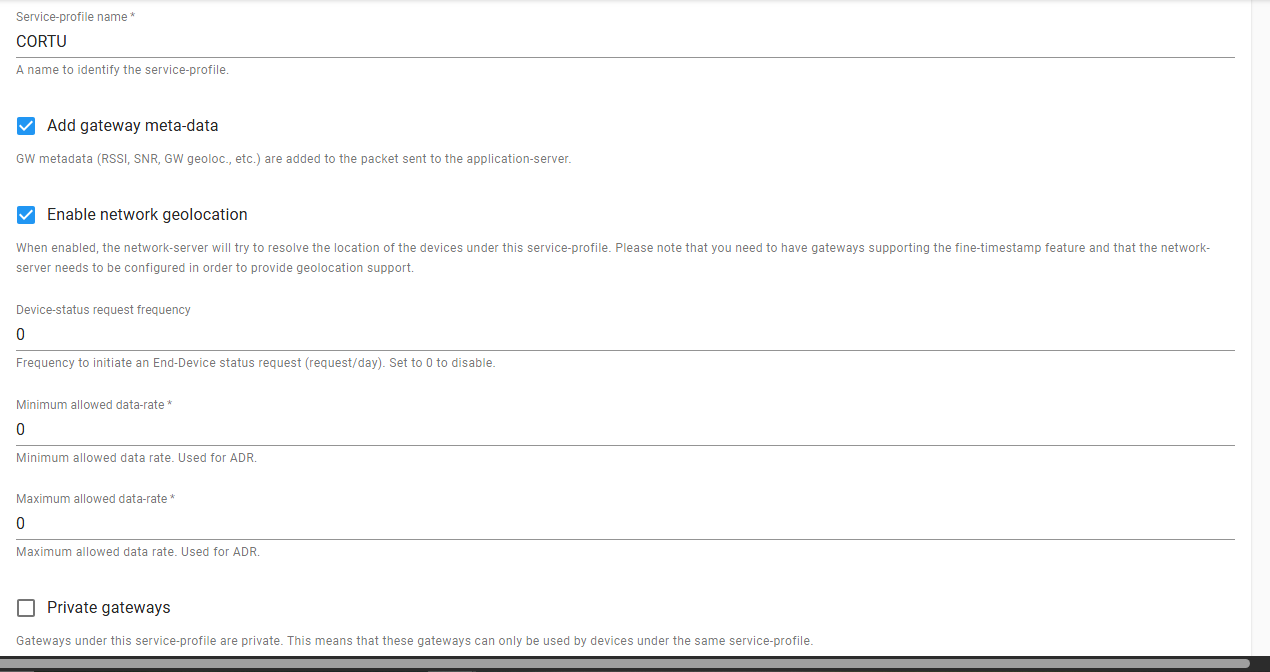


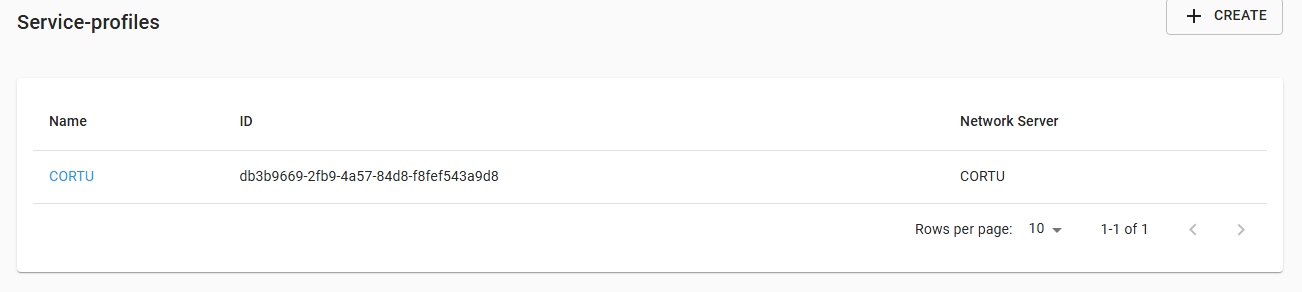
## CREATING A GATEWAY PROFILE





## Setting up Service Profile



You should have something like this:  


### Notes from GPT

Great question!

**🔍 What is Device Status Request Frequency?**

This is the number of times **per day** that **ChirpStack will send a MAC command** to your device requesting **status info** like battery level, margin, etc.

**🤔 What Should You Choose?**

Here are your options depending on your situation:

**✅ Best Option for RAK3172 + ABP (Manual/Testing Use)**

* **Set it to 0**  
  → This disables automatic status requests.
  + ✅ Saves airtime
  + ✅ Prevents unnecessary downlink packets
  + ✅ Best for testing or if you don’t handle DevStatusReq in firmware

**🛠 If You Want Periodic Status (e.g., in Production)**

* Set it to 1 or 2 if:
  + You want once/twice a day health checks from devices
  + Your firmware handles DevStatusReq MAC command
  + You’re monitoring a fleet of deployed nodes

But this only works well with **OTAA**, because ABP devices don’t always respond well unless specifically programmed to do so.

**💡 Recommendation for You**

Since you're:

* Using **ABP**
* Testing/just trying to get messages through
* Likely not parsing MAC commands on the RAK3172 yet

→ **Set it to 0**

You can always increase it later when you want to explore more advanced monitoring features.

Let me know when you're at the “Add Device” screen and I can walk you through filling in DevAddr, NwkSKey, and AppSKey to match your sketch.

I later switched to OTAA, meaning I set up the LORAWAN End Device [RAK3172] to work using OTAA. This did not affect anything. If you read what Device Status Frequency above, you would realize why it wont affect communication with end device.

**ADR (Adaptive Data Rate)**

Great question again — those **ADR (Adaptive Data Rate)** settings are important for fine-tuning LoRaWAN communication, but they can be safely **left at zero** in most testing or manual ABP scenarios.

Let’s break it down:

**🔧 What Do These Settings Mean?**

**1. Min Allowed Data Rate**

* The **lowest (slowest, longest range)** data rate the Network Server is allowed to assign to the device.
* Lower DR = lower bandwidth but better range.

**2. Max Allowed Data Rate**

* The **highest (fastest, shorter range)** data rate allowed.

Data rates vary by region. For example, in EU868:

* DR0 = SF12 (long range, slow)
* DR5 = SF7 (shorter range, faster)

In ChirpStack:

* These settings limit the range of rates used by **ADR** (which adjusts the rate based on link quality).

**✅ What to Set for Testing**

If:

* You're using **ABP** (no join procedure)
* Your RAK3172 sketch has a **fixed DR**
* You **don’t need ADR right now**

Then:

**Leave both values as 0**, or don’t set them at all.

→ This effectively means "don’t limit it" — ChirpStack will just go with what the device uses.

**🧠 Later on (Optional)**

Once your messages are going through and you want:

* Efficient communication
* Better battery life
* Dynamic data rate switching

You can:

* **Enable ADR in your node’s firmware**
* Set min = 0 (SF12), max = 5 (SF7), or a range suited to your environment

**⚠️ Important**

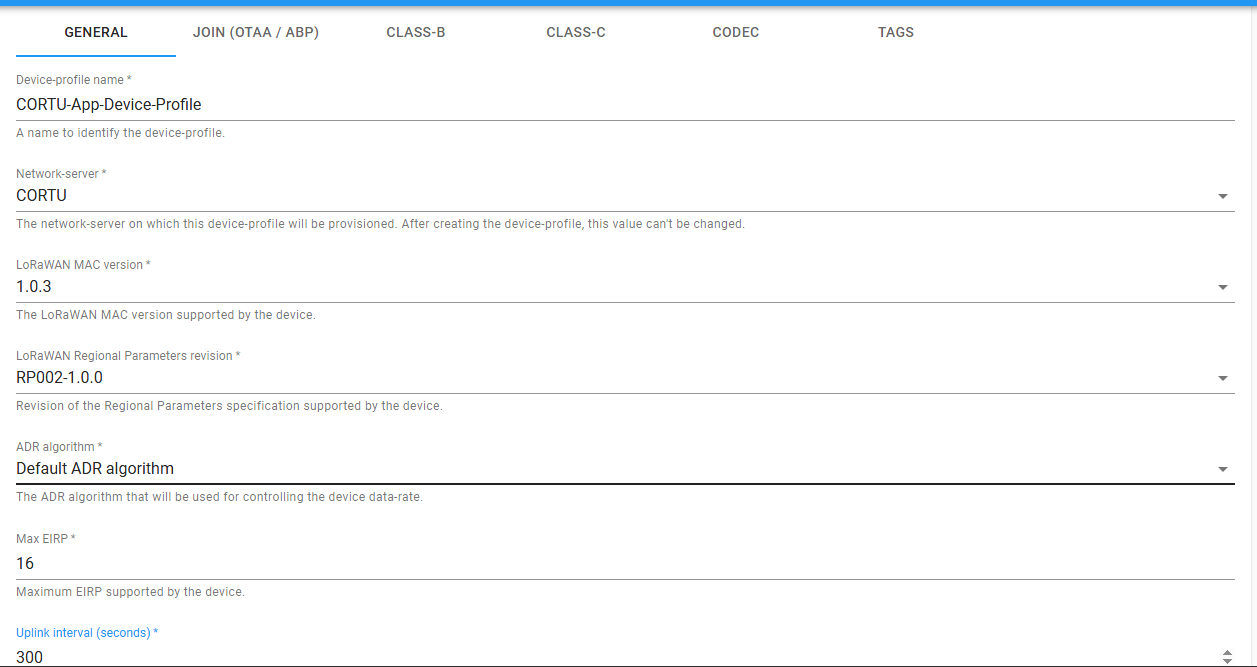
If your **device doesn't handle ADR**, ChirpStack may try to send MAC commands it can’t interpret. So unless your firmware supports ADR:

✅ **Disable ADR on the device side and in ChirpStack. Leave those DR fields at 0.**

Would you like help matching your RAK3172 sketch's data rate to your ChirpStack config?

## Device Setup After Creating Application

Before Setting up device you have to create an Application. Click on Applications in the bottom left corner of chripstack web page



### Notes

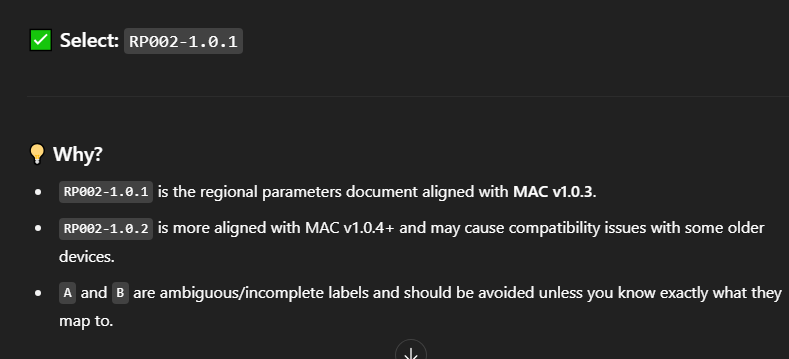
Device Profile name: any name you want

Network server: Already created when creating network server

**LoraWAN MAC Version**:   
 **📌 Why 1.0.3?**

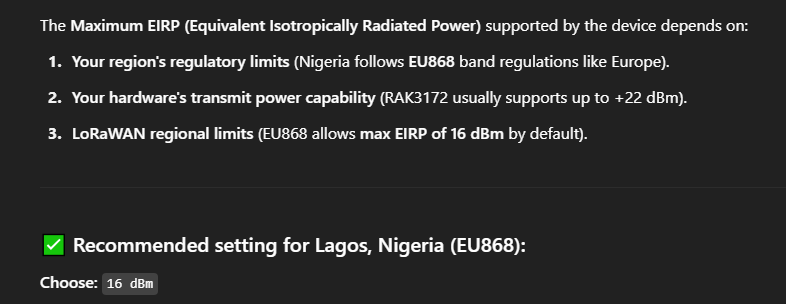
* RAK3172 uses the STM32WLE5 chip and by default supports LoRaWAN **1.0.3**.
* This version is commonly used in LoRaWAN deployments and is **well supported** in ChirpStack.
* If you're using Arduino-based firmware, it usually targets 1.0.2 or 1.0.3.

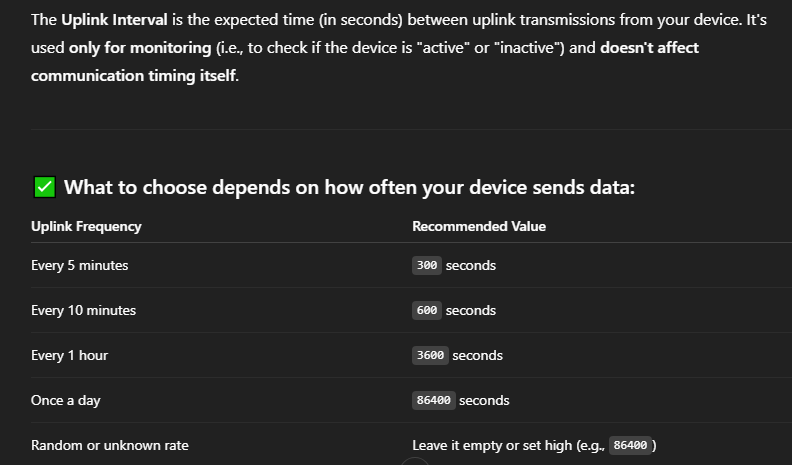
**LORAWAN Regional Parameters Revision:**

****

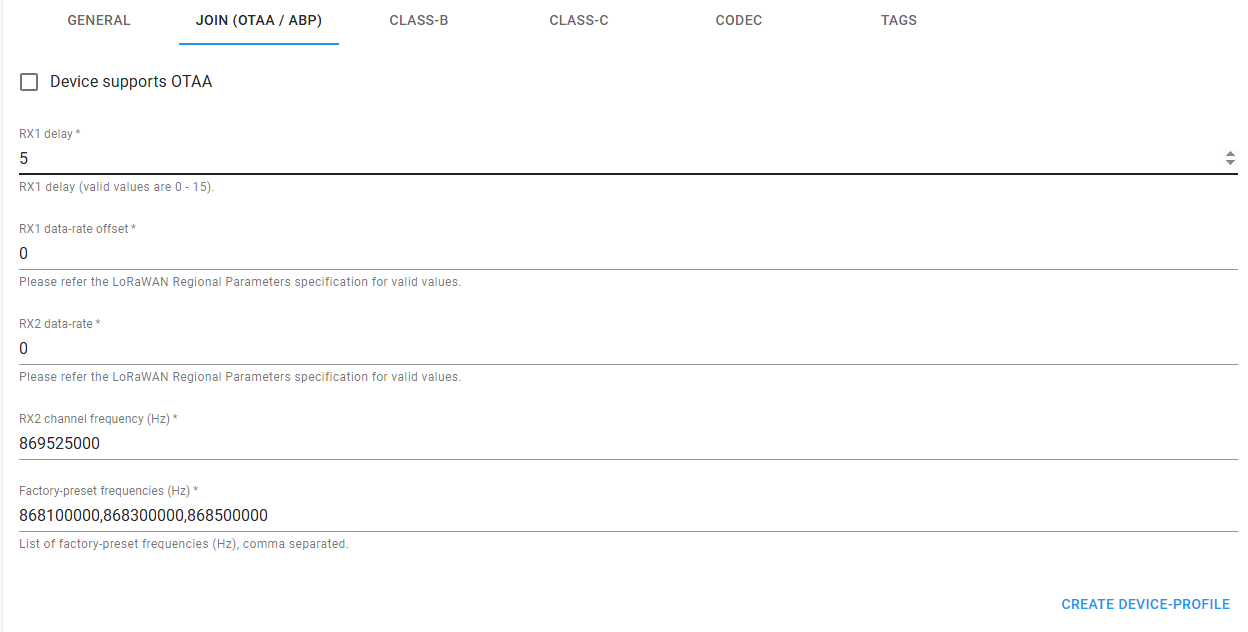
**ADR Algorithm:** Just choose the default

**MAX EIRP:**

****

**Uplink Interval:  
**

You can adjust it later



**Here’s what you should choose for each of those parameters based on your location (Lagos, Nigeria, which falls under EU868 / IN865 LoRaWAN band) and general best practices for LoRaWAN Class A devices:**

**✅ 1. RX1 Delay**

* **What it is: Time between the uplink transmission and the start of the first receive window (RX1).**
* **Recommended value: 5**
  + **This is the default for most LoRaWAN networks.**
  + **Settable from 0–15 seconds, but 5 is standard.**

**✅ 2. RX1 Data-Rate Offset**

* **What it is: Offset applied to the uplink data rate to determine the RX1 downlink data rate.**
* **Recommended value: 0**
  + **This keeps RX1 data rate the same as the uplink, which is safest for beginners.**
  + **Valid values: 0–5 typically, but depends on the regional band.**

**✅ 3. RX2 Data-Rate**

* **What it is: Data rate used for the second receive window (RX2).**
* **Recommended value (EU868): 0**
  + **For EU868, DR0 = SF12 / 125 kHz is commonly used for RX2.**

**✅ 4. RX2 Channel Frequency (Hz)**

* **What it is: Frequency used for RX2 receive window.**
* **Recommended value (EU868): 869525000**
  + **This is the default RX2 frequency for EU868.**
  + **Confirm it matches your ChirpStack’s regional setup.**

**✅ 5. Factory Preset Frequencies**

* **What it is: The frequencies pre-programmed into your device, usually matching the default channels of the band.**
* **Recommended value (EU868):**
* **868100000,868300000,868500000**
  + **These are the standard 3 default channels for EU868 and most compliant gateways expect these.**

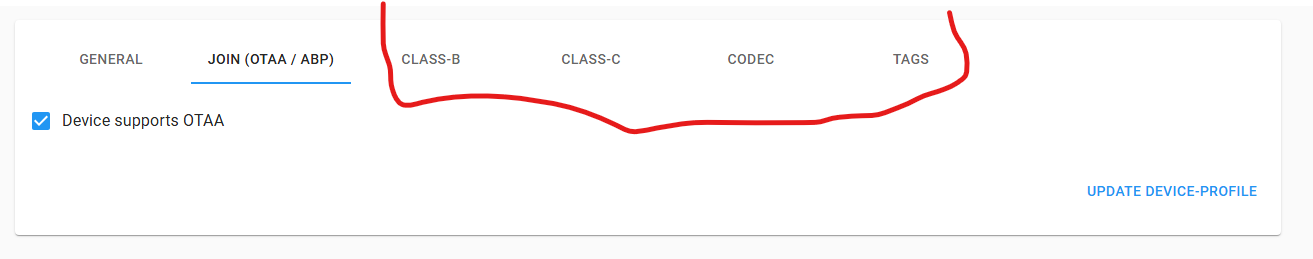
**📌 Summary Table for EU868 (Lagos):**

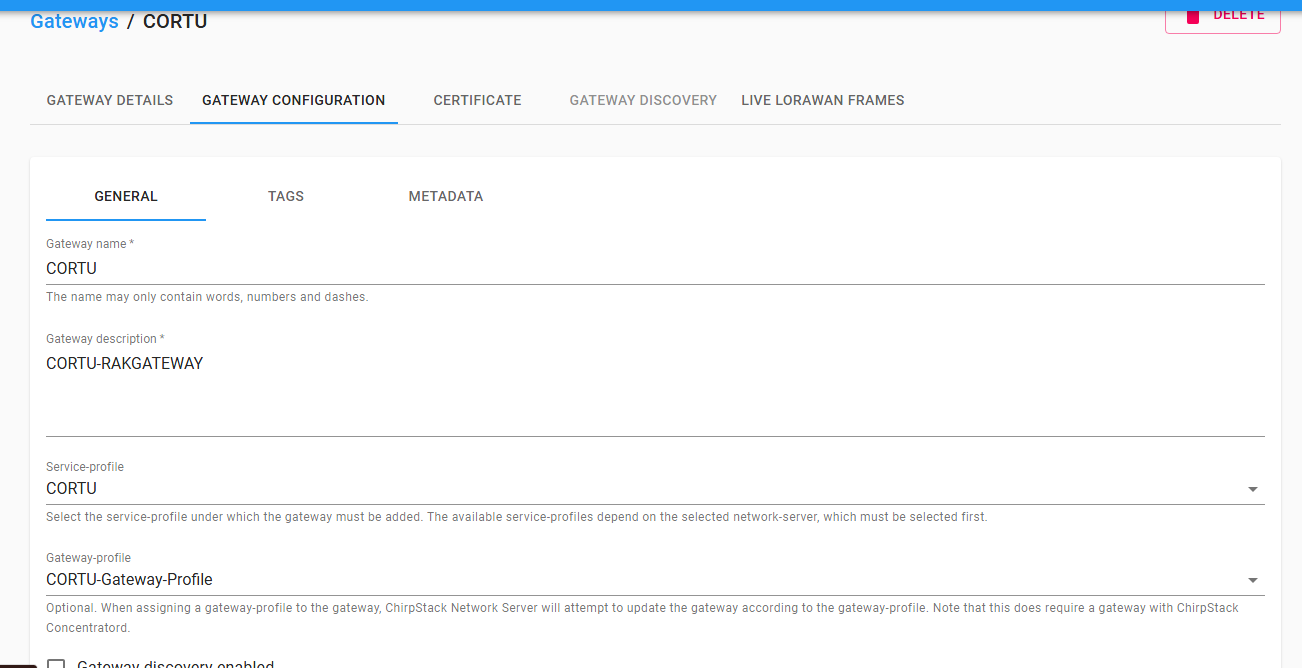
| **Field** | **Value** |
| --- | --- |
| **RX1 delay** | **5** |
| **RX1 data-rate offset** | **0** |
| **RX2 data-rate** | **0** |
| **RX2 channel frequency** | **869525000** |
| **Factory preset frequencies** | **868100000,868300000,868500000** |

**Let me know if you're using IN865 instead of EU868, and I’ll give you updated values.**

**if you check the box Device Supports OTAA, you won’t need the above setup**

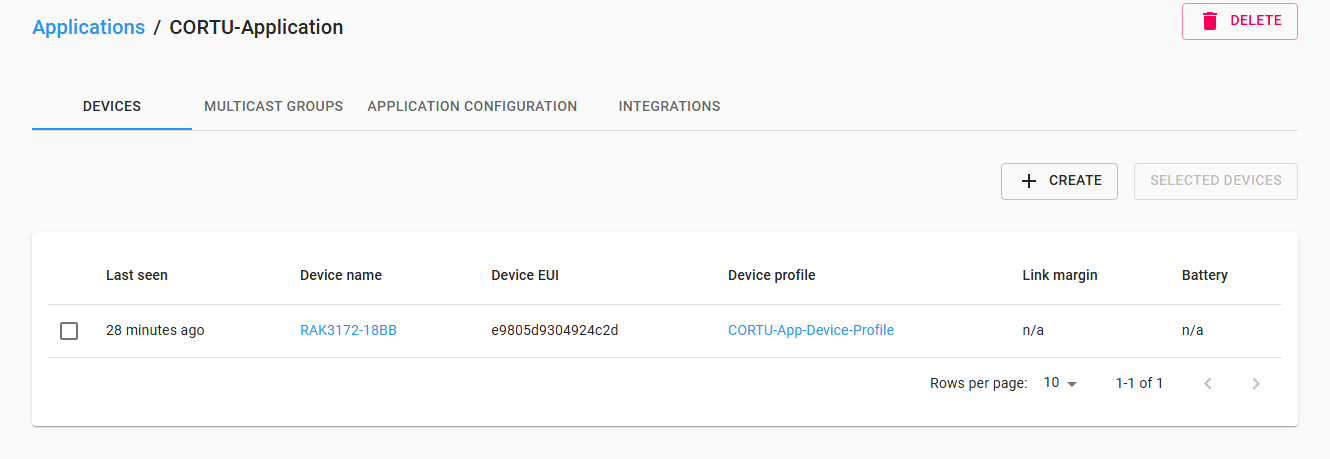
I did not touch the guys in the red box.





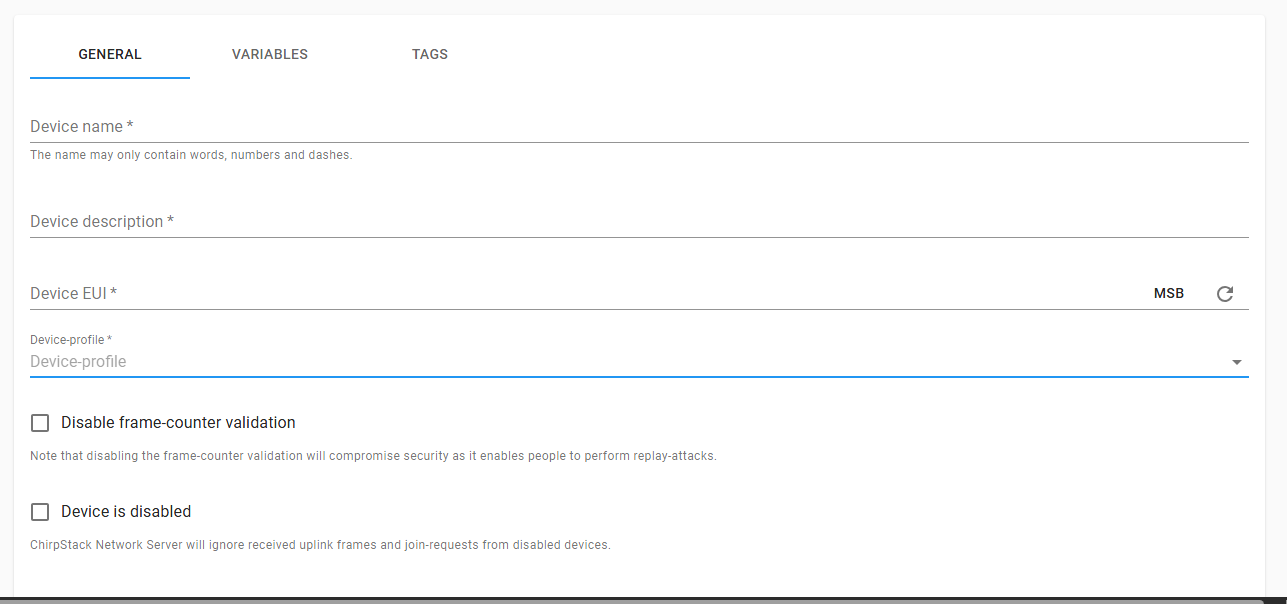
## Adding A Device

This is under Applications.

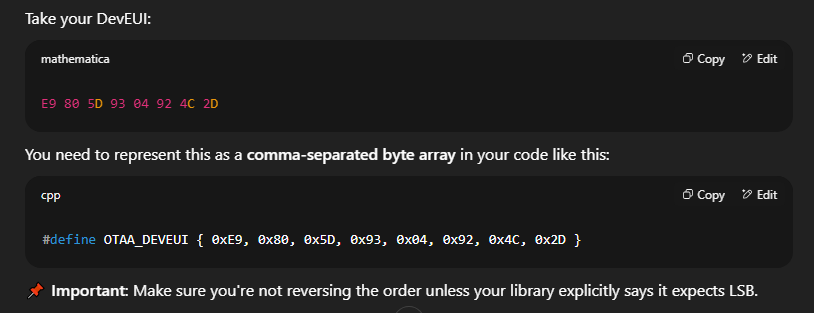


I have already created one, but to create yours click create. I named the Device using the module name and last 4 characters of the DEUI written on the module.

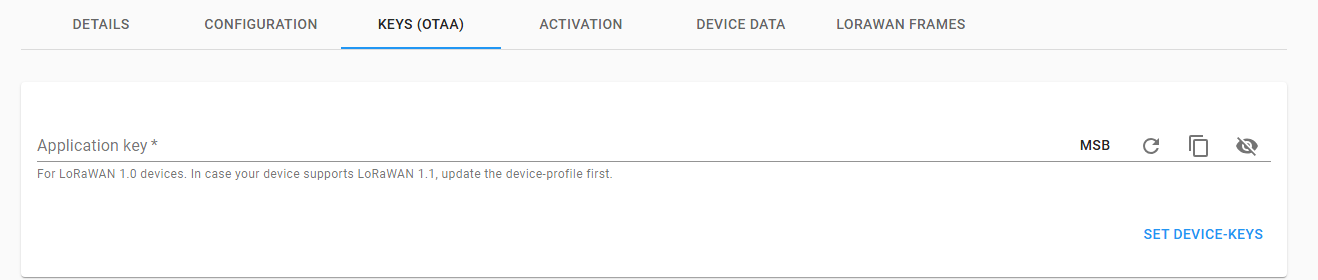
**To add a device:**



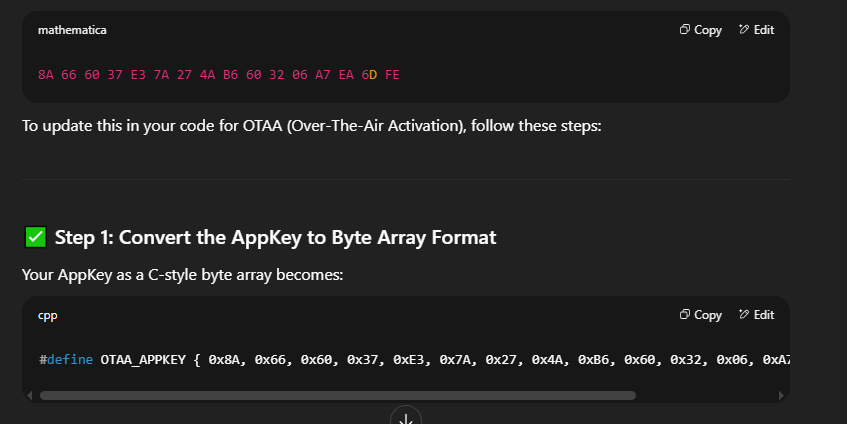
* The device name and description can be anything you want.
* The device profile is what we created earlier
* Generate a Device EUI by clicking the refresh button beside MSB and leave it as MSB as that is the format our Arduino code uses.
* Now copy that Device EUI and add it to your Arduino code



Once you have created it, a page will appear asking for Application Key.



You can generate this using the same process as above and also include it in your code.



### APPEUI

