EUROPEAN UNION

Interreg
Austria-Czech Republic
European Regional Development Fund

LoRa(WAN) Webinar

ChirpStack

Author: Harald Eigner (TU Wien)

These slides show how to setup a LoRaWAN test environment. The presentation focuses on a setup based on the open source software ChirpStack, a Raspberry Pi, and an IMST iC880A concentrator board. It contains all steps starting from scratch until a successful OTAA join of a RN2483 based evaluation board, including the required commands for the RN2483.

TU WIEN
TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

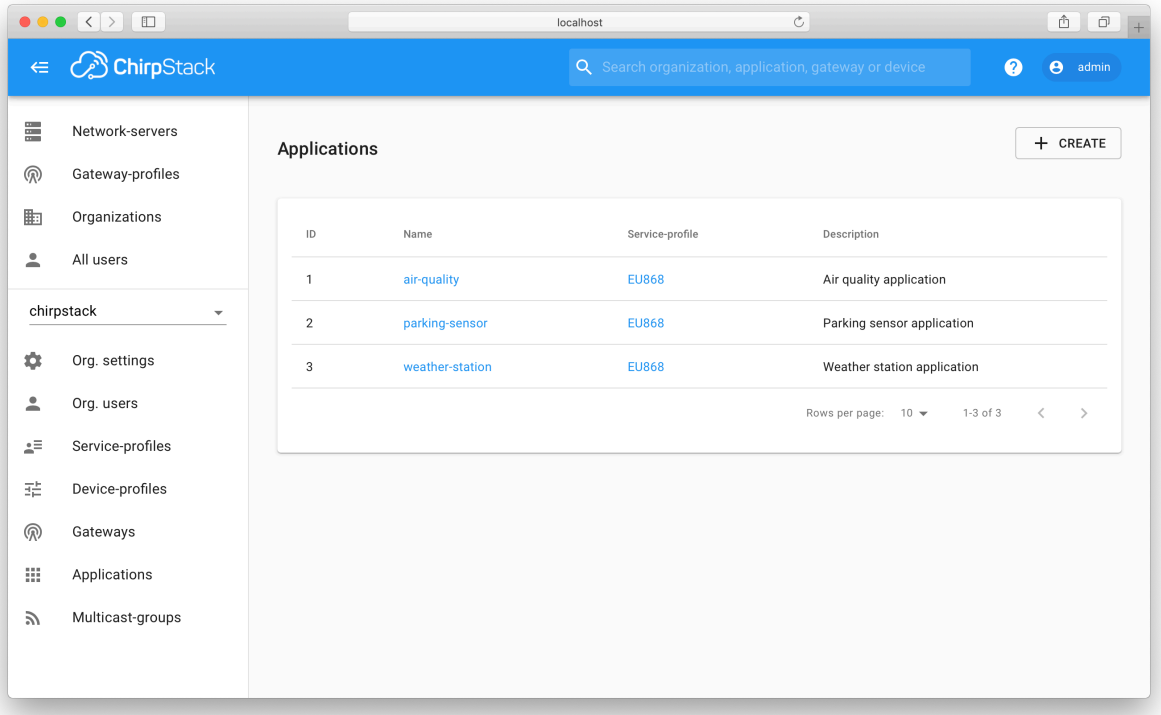
EMCE

Contents

Contents

ChirpStack

- https://www.chirpstack.io/
- Open-source components for LoRaWAN networks
- Web interface for device management
- APIs for integration



ChirpStack web interface

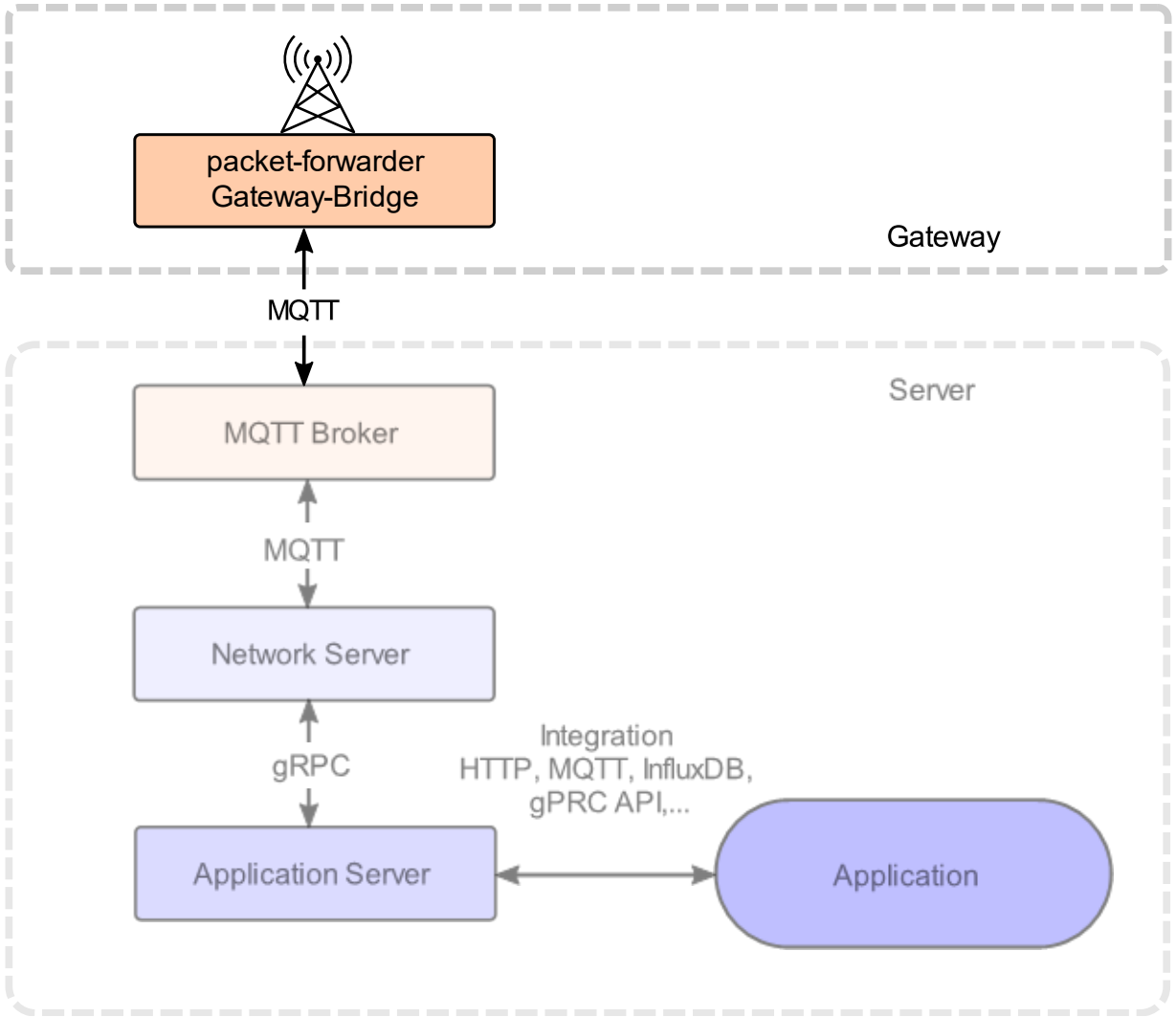
Gateway

Packet-forwarder:

- Forwards LoRa Packets to gateway-bridge

Gateway Bridge:

- Converts Packet Forwarder protocol into JSON or Protobuf data format
- Can be integrated on gateway or server



Server

MQTT Broker:

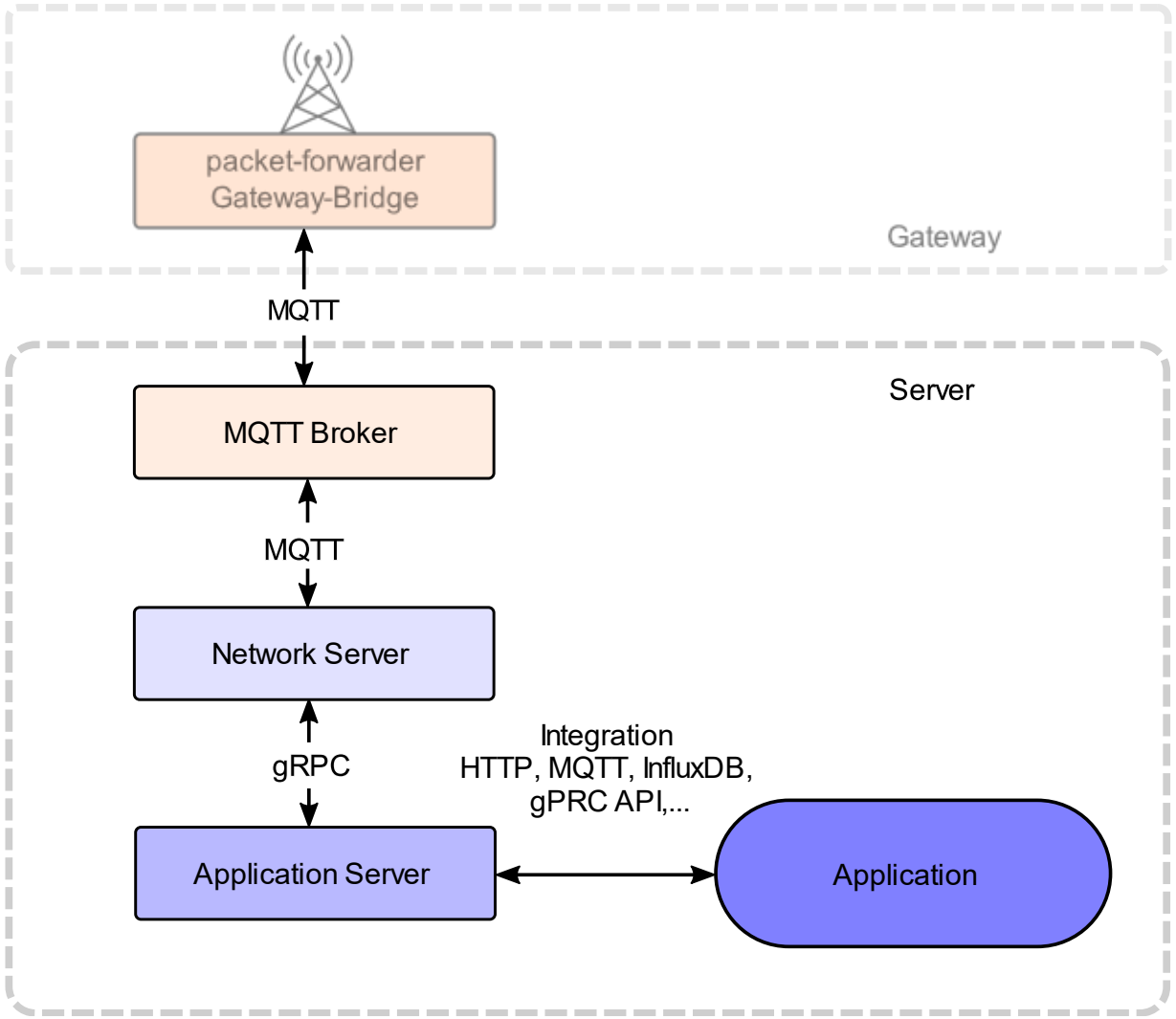
- Communication between gateway and network server via MQTT using MQTT broker like Eclipse Mosquitto

Network Server:

- Communication with application server via gPRC

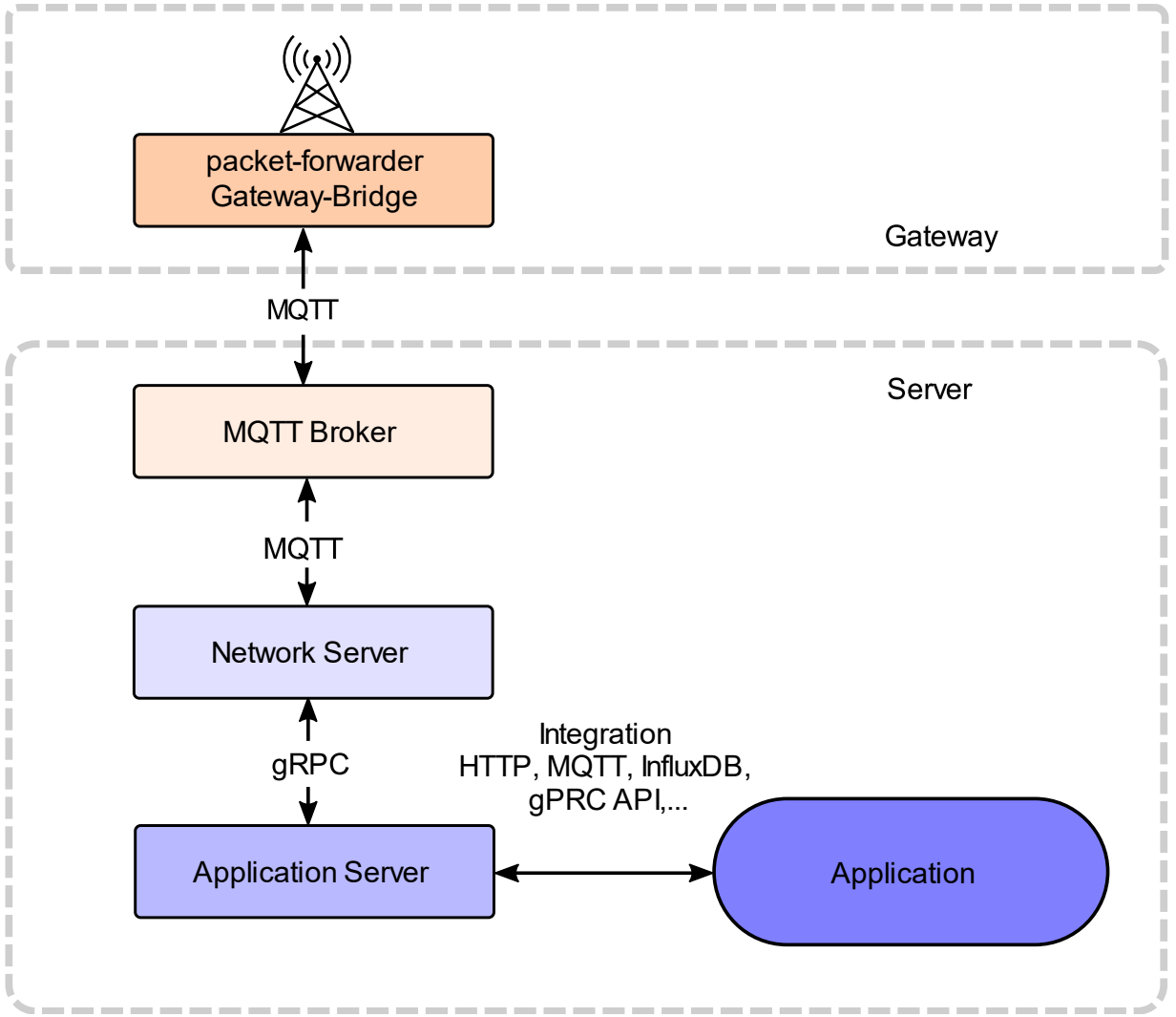
Application Server:

- Several implementations available to connect with application



Gateway Operating System

- Gateway OS is an open source Linux based embedded operating system
- Two systems available:
  - Base OS system: provides Packet Forwarder and Gateway Bridge
  - Full OS system: provides full server environment



Installation Guide for Debian or Ubuntu

- Tutorial of steps needed to setup the ChirpStack server stack

Many configurations are possible. In this tutorial, the following assumptions of the deployment architecture are made:

- All ChirpStack components and their dependencies will be installed on a single server instance.
- The ChirpStack Gateway Bridge component will be installed on gateway but can also be installed on the server itself.
- No firewall rules are setup.

Contents

Installation Guide for Debian or Ubuntu

Full installation guide for:

- ChirpStack Server stack
  - Installation guide for Debian or Ubuntu
- Gateway
  - Installation guide for
    - Raspberry Pi 3
    - IMST – iC880A Concentrator shield
- Connecting a Device
  - Installation guide for including a Microchip RN2483 wireless module to the network

Contents

Installation Guide for Debian or Ubuntu

Install dependencies:

- MQTT broker: A publish/subscribe protocol that allows users to publish information under topics that others can subscribe to.
- Redis: An in-memory database used to store relatively transient data
- PostgreSQL: The long-term storage database used by the open source packages

Command to install the dependencies:

```
sudo apt install mosquitto mosquitto-clients redis-server redis-tools postgresql
```

Installation Guide for Debian or Ubuntu

Setup PostgreSQL databases and users:

- To enter the command line utility for PostgreSQL:

```
sudo -u postgres psql
```

- Inside the prompt, the databases are set up used by the ChirpStack components.

```
-- set up the users and the passwords
-- (note that it is important to use single quotes and a semicolon at the end!)
create role chirpstack_as with login password 'dbpassword';
create role chirpstack_ns with login password 'dbpassword';

-- create the database for the servers
create database chirpstack_as with owner chirpstack_as;
create database chirpstack_ns with owner chirpstack_ns;

-- change to the ChirpStack Application Server database
\c chirpstack_as

-- enable the pq_trgm and hstore extensions
-- (this is needed to facilitate the search feature)
create extension pg_trgm;
-- (this is needed to store additional k/v meta-data)
create extension hstore;
-- exit psql
\q
```

Installation Guide for Debian or Ubuntu

Setup ChirpStack software repository:

ChirpStack provides a repository that is compatible with the Ubuntu apt package system.

- Installation of *dirmngr* and *apt-transport-https*:

```
sudo apt install apt-transport-https dirmngr
```

- Set up the key for this new repository:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
```

- Add the repository to the repository list by creating a new file:

```
sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable main" | sudo tee /etc/apt/sources.list.d/chirpstack.list
```

- Update the apt package cache:

```
sudo apt update
```

Installation Guide for Debian or Ubuntu

Installing the ChirpStack Network Server:

- Installing the package using apt:

```
sudo apt install chirpstack-network-server
```

- Next step is to update the configuration file to match the database and band configuration.
  - The configuration is located at:

```
/etc/chirpstack-network-server/chirpstack-network-server.toml
```

- Example configuration files can be found in:

https://www.chirpstack.io/network-server/install/config/

- After updating the configuration, the ChirpStack Network Server needs to be restarted and validated that there are no errors:

```
# start chirpstack-network-server
sudo systemctl start chirpstack-network-server

# start chirpstack-network-server on boot
sudo systemctl enable chirpstack-network-server
```

- Print the ChirpStack Network Server log-output:

```
sudo journalctl -f -n 100 -u chirpstack-network-server
```

Installation Guide for Debian or Ubuntu

Installing the ChirpStack Application Server:

- Installing the package using apt:

```
sudo apt install chirpstack-application-server
```

- Next step is to update the configuration file to match the database and band configuration.
  - The configuration is located at:

```
/etc/chirpstack-network-server/chirpstack-application-server.toml
```

- Example configuration files can be found in:

https://www.chirpstack.io/application-server/install/config/

- After updating the configuration, the ChirpStack Application Server needs to be restarted and validated that there are no errors:

```
# start chirpstack-network-server
sudo systemctl start chirpstack-application-server

# start chirpstack-network-server on boot
sudo systemctl enable chirpstack-application-server
```

- Print the ChirpStack Application Server log-output:

```
sudo journalctl -f -n 100 -u chirpstack-application-server
```

Installation Guide for Debian or Ubuntu

Setting up Mossquitto MQTT Broker:

- First, the MQTT authentication and authorization must be set up. At the server, we change to the mosquito directory and create a new password file for authentication:

```
cd /etc/mosquito

# Create a password file, with users chirpstack_gw, chirpstack_ns, chirpstack_as # and bob.
sudo mosquitto_passwd -c /etc/mosquitto/passwd chirpstack_gw
sudo mosquitto_passwd /etc/mosquitto/passwd chirpstack_ns
sudo mosquitto_passwd /etc/mosquitto/passwd chirpstack_as
sudo mosquitto_passwd /etc/mosquitto/passwd bob

 # Secure the password file
sudo chmod 600 /etc/mosquitto/passwd
```

- Next, we add permissions for the Chirpstack components to the access control list file */etc/mosquitto/acls*:

```
user chirpstack_gw
topic write gateway/+/event/+
topic read gateway/+/command/+

user chirpstack_ns
topic read gateway/+/event/+
topic write gateway/+/command/+

user chirpstack_as
topic write application/+/device/+/event/+
topic read application/+/device/+/command/+

user bob
topic read application/123/device/+/event/+
topic write application/123/device/+/command/+
```

- The authentication for the subscription to the MQTT broker must be updated in the configuration files of the Network and Application Server

Contents

Gateway Installation Guide

Gateway Operatins System (OS):

- ChirpStack Gateway OS is an open-source Linux based embedded OS which can run on various LoRa gateway models.

- Two different image types:
  - chirpstack-gateway-os-base: Provides the ChirpStack packet forwarder and ChirpStack Gateway Bridge pre-installed including a CLI utility for gateway configuration
  - chirpstack-gateway-os-full: Provides a full ChirpStack Network Server and ChirpStack Application Server environment running on the gateway, on top of all the features that are provided by the chirpstack-gateway-os-base image.

Gateway Installation Guide

Gateway components:

Several gateway models are supported by Chirpstack. Two components are needed to run a gateway:

- Embedded Linux board: Raspberry Pi
- Concentrator shield: IMST – iC880A, IMST – iC980A, RAK2245, …

A list of supported hardware can be found at https://www.chirpstack.io/gateway-os/

The following installation guide is suited for:

- Raspberry Pi 3 +
- IMST – iC880A

Gateway Installation Guide for Raspberry Pi 3 +IMST – iC880A

Installing Gateway OS for Raspberry 3:

- Downloading the image for the base OS of type *.wic.gz* at:

https://www.chirpstack.io/gateway-os/install/raspberrypi/

- Flashing the image on a SD card using Balena Etcher or Win32DiskImager for example
- Once connected to the ethernet, you can log in using ssh with IP address allocated to the gateway:

```
ssh admin@xxx.xxx.xxx.xxx
```

default username: *admin*

default password: *admin*

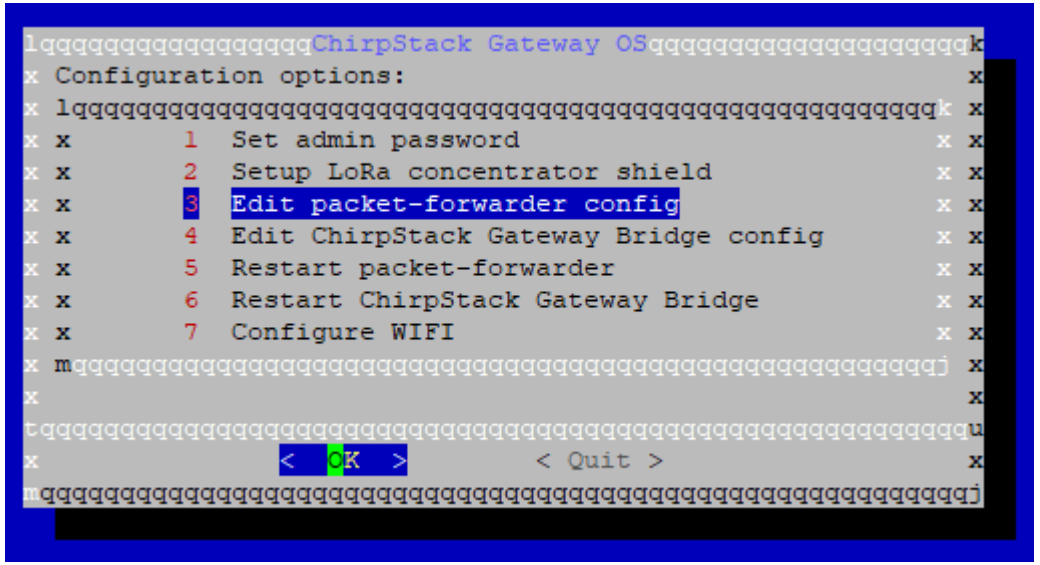
- The following message will be prompted after the login:

```
  ________    _           _____ __             __     _
 / ____/ /_  (_)_______  / ___// /_____ ______/ /__  (_)___
/ /   / __ \/ / ___/ __ \\__ \/ __/ __ `/ ___/ //_/ / / __ \
/ /___/ / / / / /  / /_/ /__/ / /_/ /_/ / /__/ ,< _ / / /_/ /
\____/_/ /_/_/_/  / .___/____/\__/\__,_/\___/_/|_(_)_/\____/
                 /_/

Documentation and copyright information:
> www.chirpstack.io

Commands:
> sudo gateway-config  - configure the gateway
> sudo monit status    - display service monitor
```

Gateway Installation Guide

Connecting the gateway with ChirpStack:

- Next step is to connect the Gateway-Bridge with the MQTT broker.
  By executing *sudo gateway-config* the following screen is prompted:



- In the packet-forwarder config, the IP address pointing to the MQTT broker must be inserted at the MQTT authentication section:
- Additionally, the authentication for the MQTT subscription must be updated

```
# Generic MQTT authentication.
[integration.mqtt.auth.generic]
# MQTT servers.
#
# Configure one or multiple MQTT server to connect to. Each item
must be in
# the following format: scheme://host:port where scheme is tcp, ssl or
ws.
servers=[
    "tcp://xxx.xxx.xxx.xxx:1883",
]
```

Gateway Installation Guide

Connecting the gateway with ChirpStack:

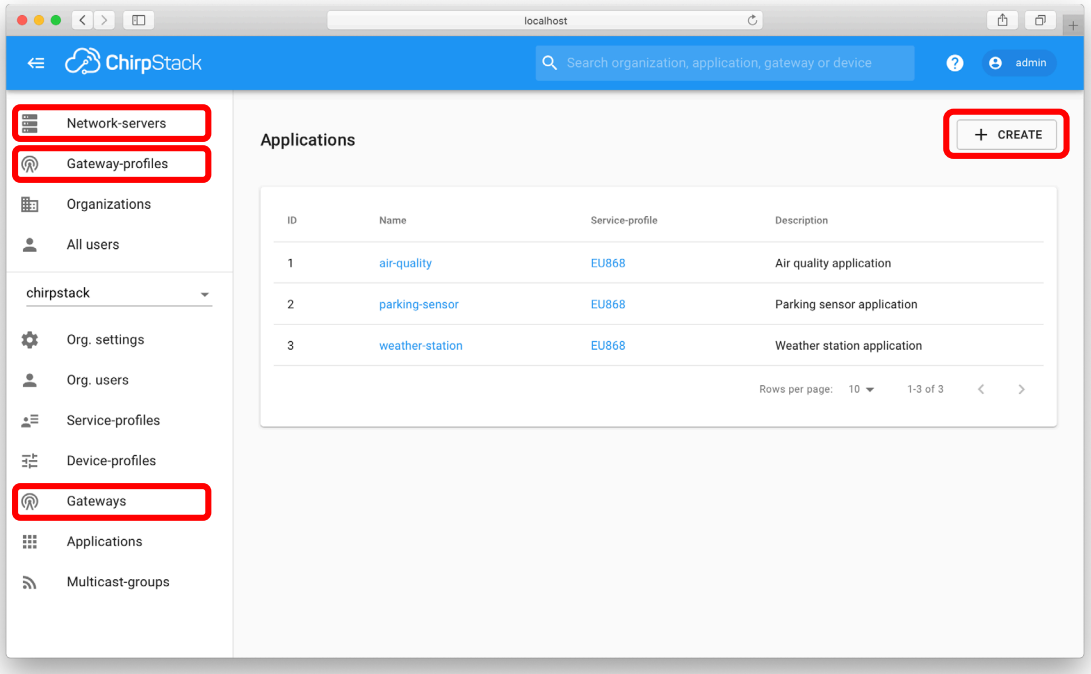
- Next step is to add the gateway to ChirpStack. To do this we must log into the web-interface of the Application Server:

http://localhost:8080/

default username: *admin*

default password: *admin*

- Three steps are needed to add a Gateway to the ChirpStack server:
  - Adding the Network server in tab “Network-servers”
  - Adding a Gateway-profile in tab “Gateway-profiles”
  - Adding a Gateway in tab “Gateways”

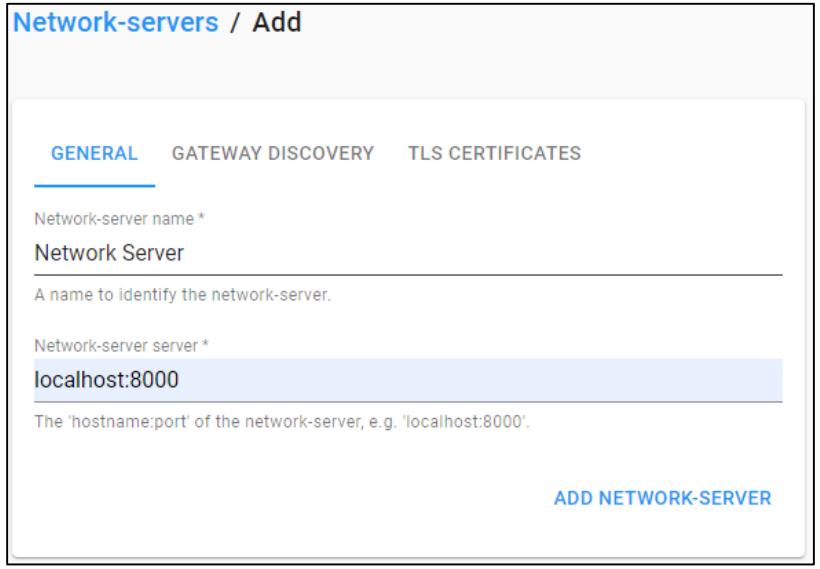


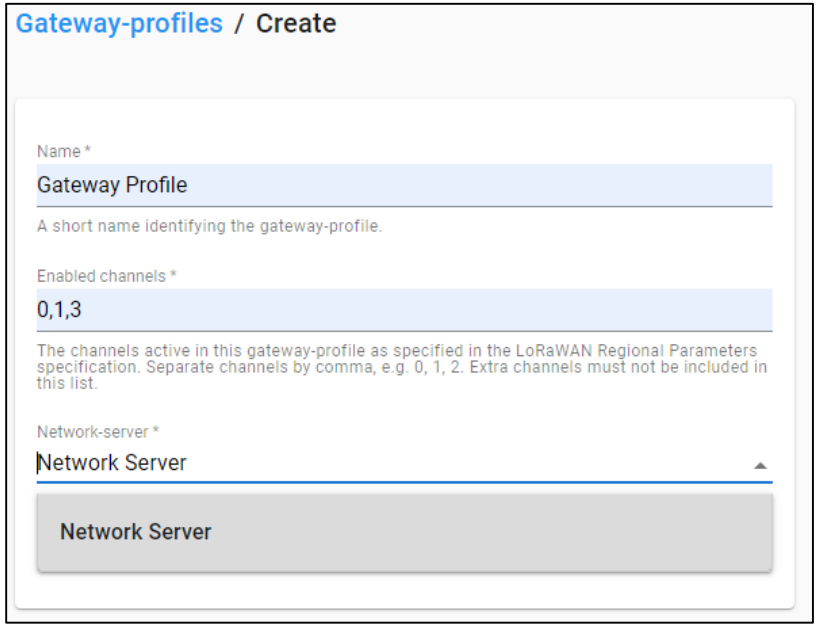
Gateway Installation Guide

Connecting the gateway with ChirpStack:

- Adding the Network Server:
  - Network Server name
  - Network Server server

- Adding a Gateway Profile
  - Gateway Profile name
  - Enabled channels according to LoRaWAN Regional Parameters
  - Network Server

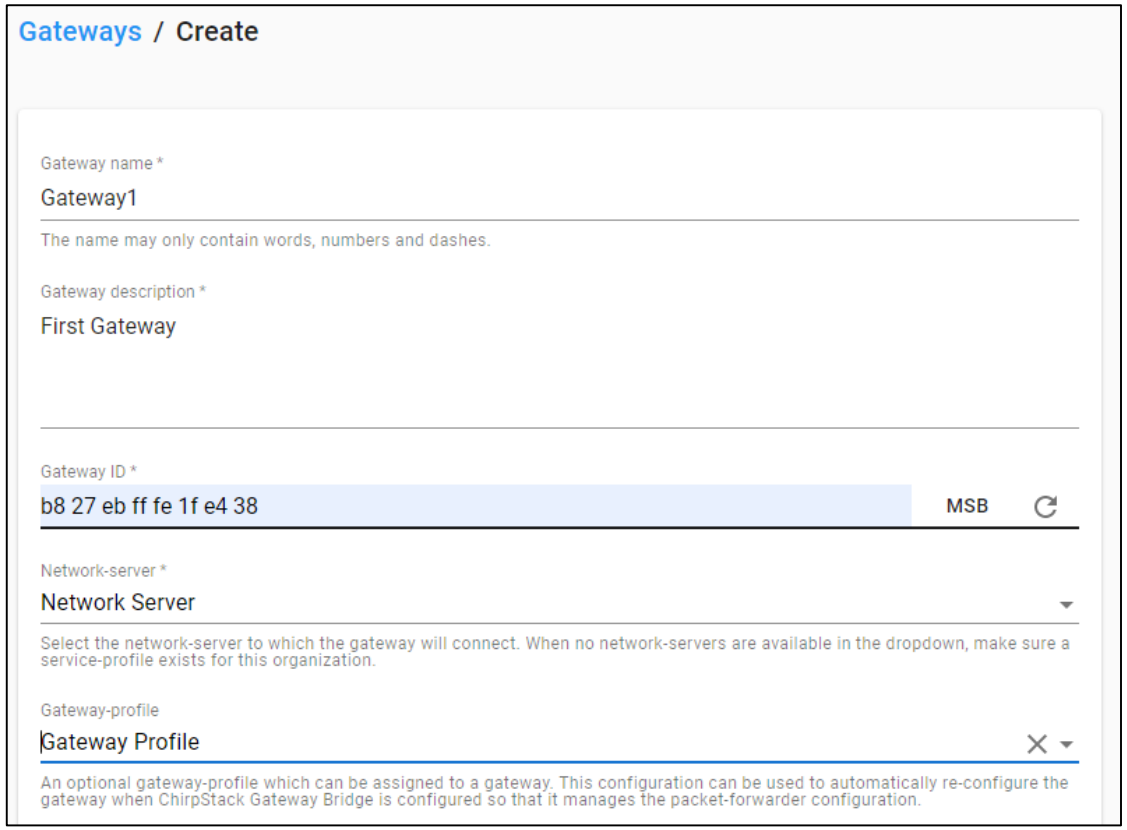




Gateway Installation Guide

Connecting the gateway with ChirpStack:

- Adding the Gateway:
  - Gateway Name
  - Gateway Description
  - Gateway ID
  - Network Server
  - Gateway Profile



Connecting a Device

Requirements:

Before the device is connected to the system, the following information about the device needs to be known:

- DevEUI: Identifier assigned by the manufacturer
- LoRaWAN MAC version implemented by the device
- Regional Parameters revision implemented by the device

Additionally, the following information is needed specifically for each activation process:

ABP:

- Device address
- Session Keys

OTAA:

- Device root-keys

Contents

Connecting a Device

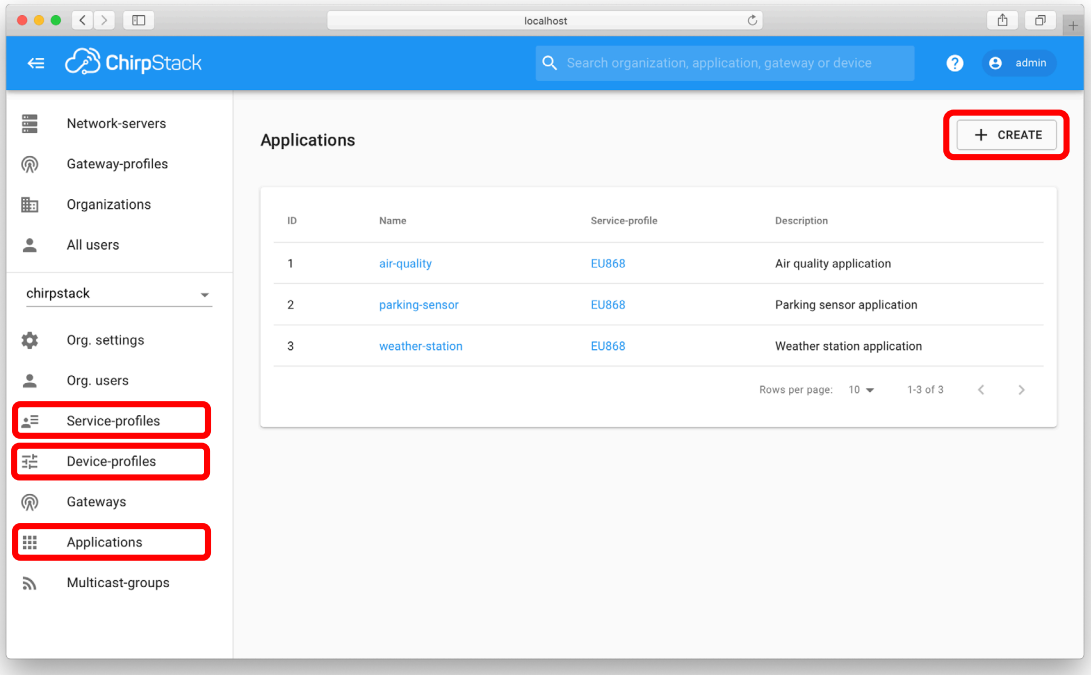
- A new device can be added in the web-interface of the Application Server:

http://localhost:8080/

default username: *admin*

default password: *admin*

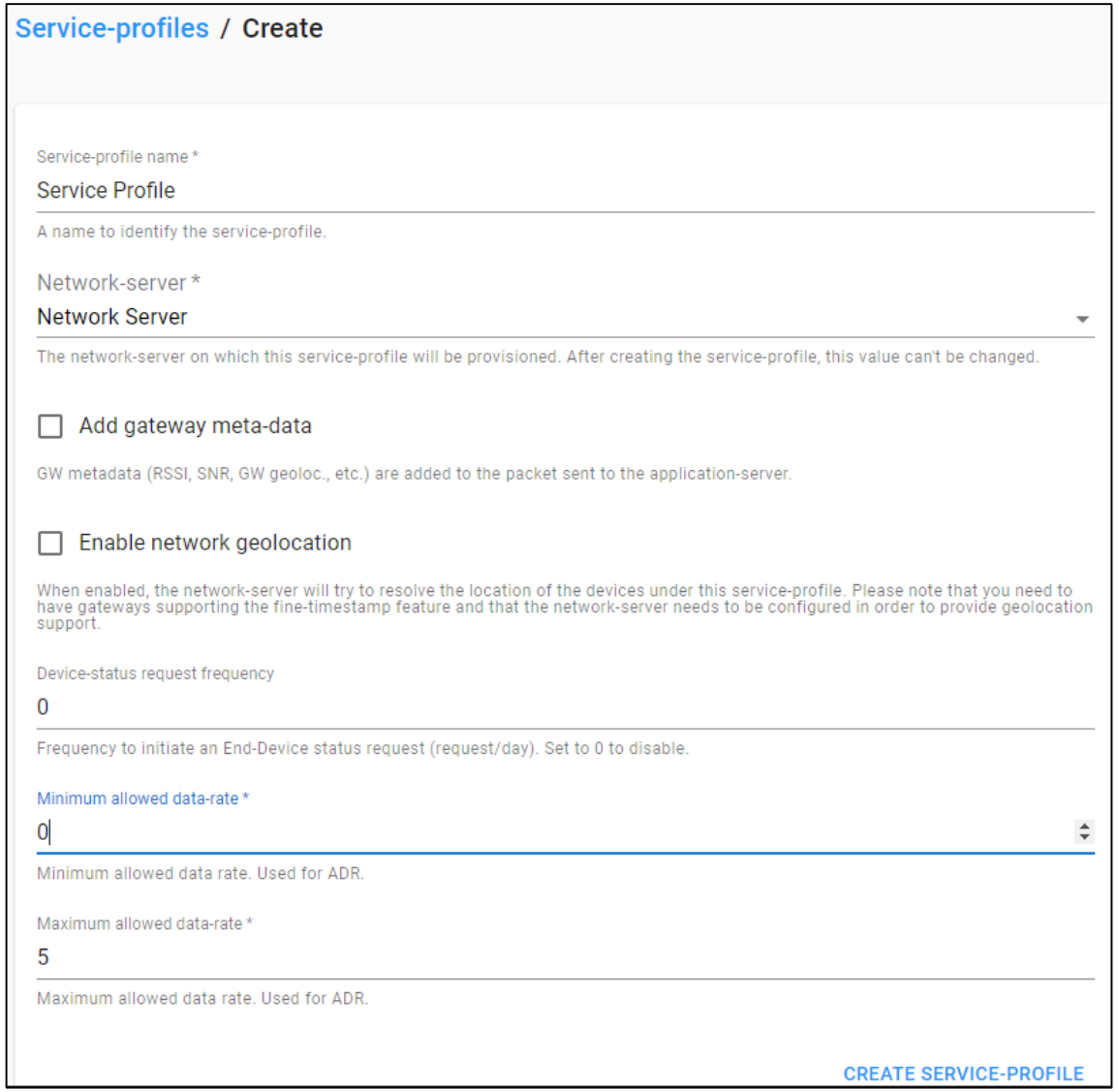
- Three steps are needed to add a Device to the ChirpStack server:
  - Adding a service profile in tab “Service-profiles”
  - Adding a device profile in tab “Device-profiles”
  - Adding an application in tab “Applications”
  - Adding a device inside the created application



Connecting a Device

Adding a Service Profile:

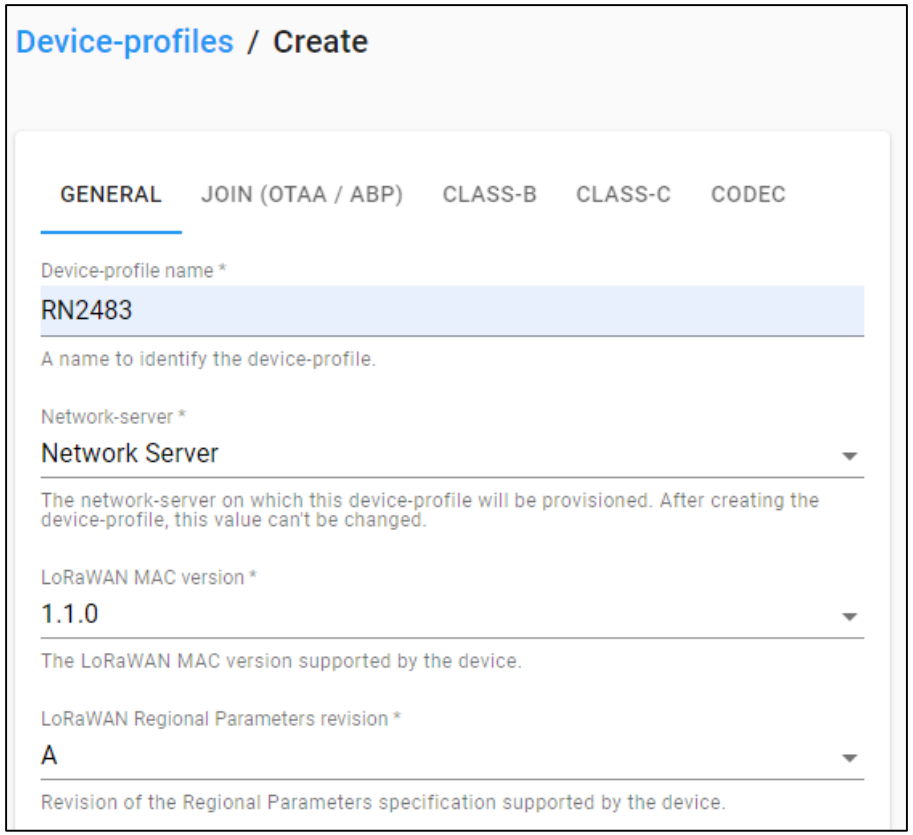
The Service Profile is the "contract" between a user and the network. It describes the features that are enabled for the user(s) of the Service Profile and the rate of messages that can be sent over the network.



Connecting a Device

Adding a Device Profile:

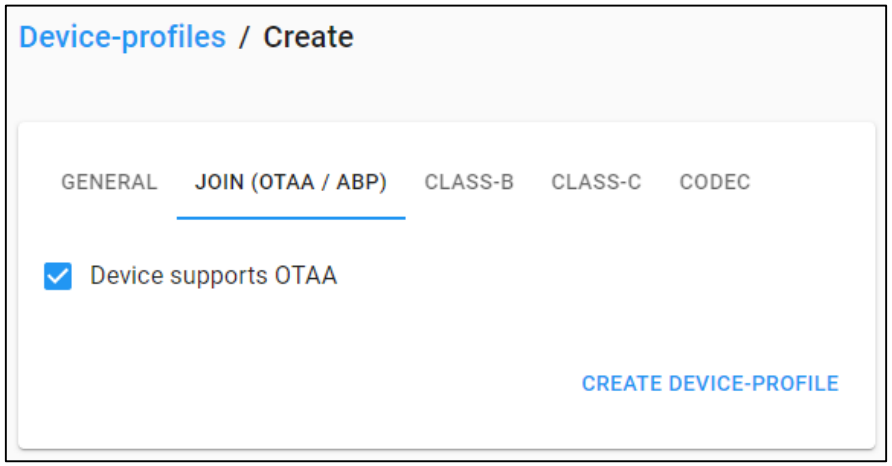
A Device Profile defines the device capabilities and boot parameters that are needed by the Network Server for setting the LoRaWAN radio access service. These information elements shall be provided by the end-device manufacturer.



Connecting a Device

Adding a Device Profile:

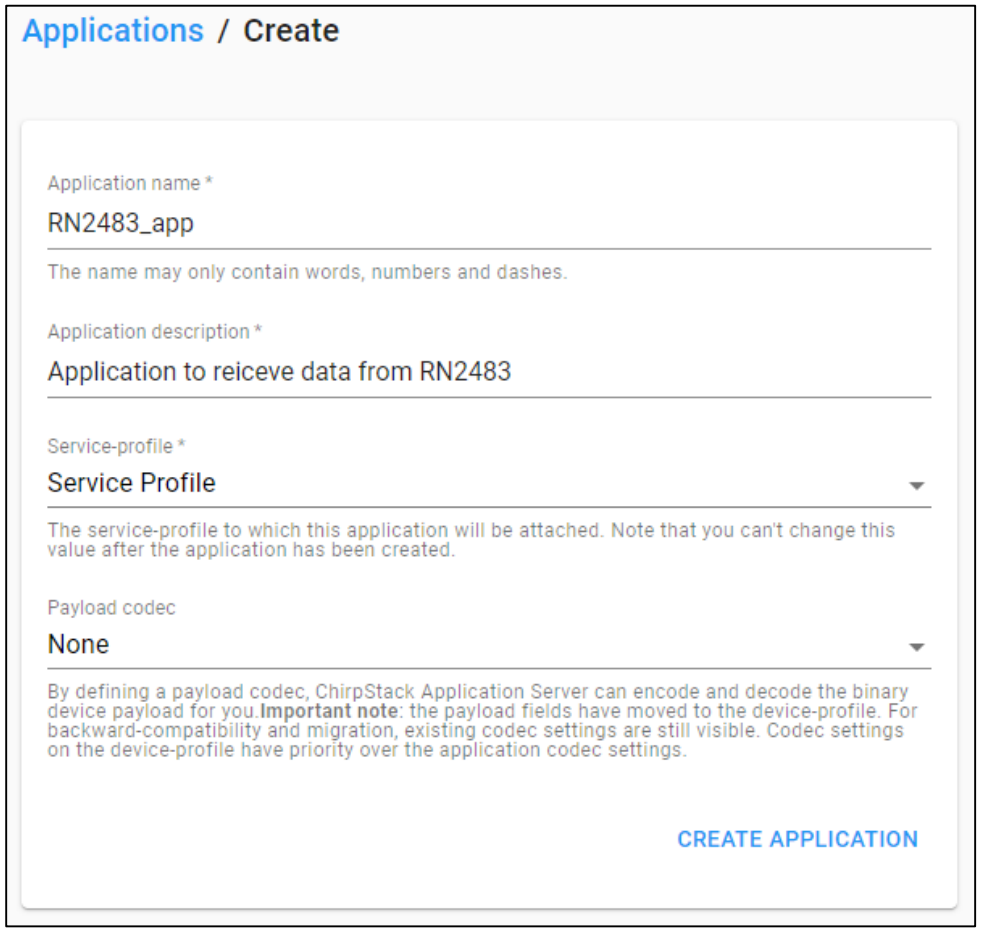
- Definition of the activation process in “JOIN”-tab
- Settings of the downlink windows RX1 and RX2
- List of factory preset frequency channels by the device
- Settings for Class-B if supported
- Settings for Class-C if supported



Connecting a Device

Adding an Application:

An application is a collection of devices with the same purpose / of the same type.

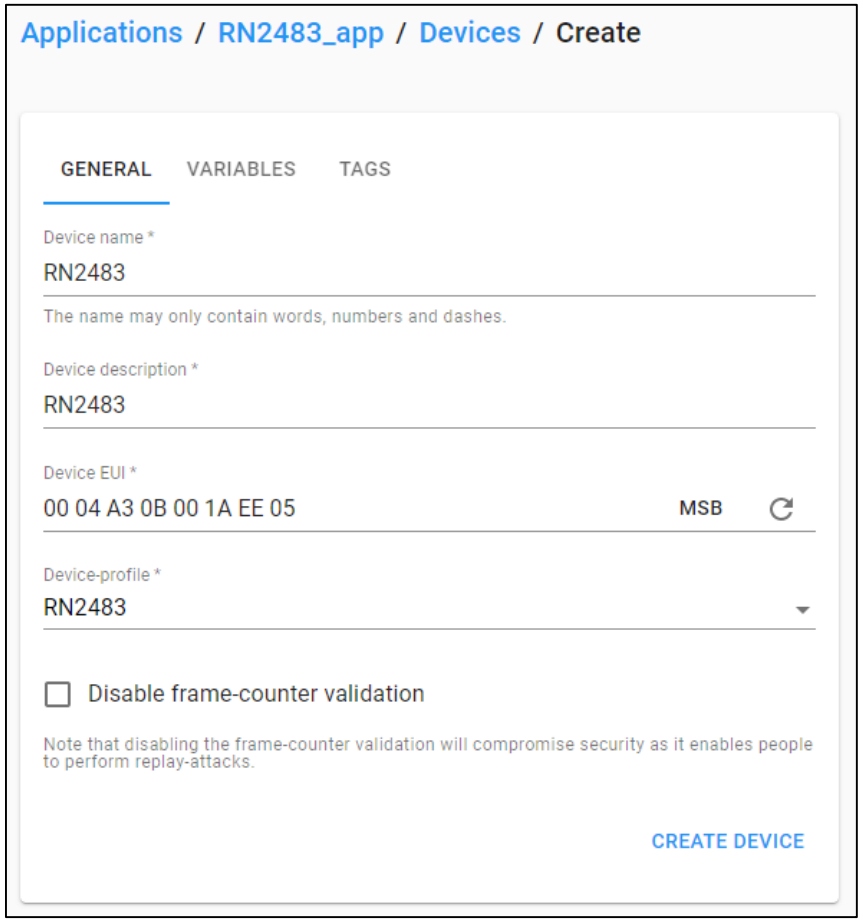


Connecting a Device

Adding a Device:

Inside the application you can add devices

- Device EUI: Identifier of the device by the manufacturer.
- The device EUI of our RN2483 is *0004A30B001AEE05*



Connecting a Device

After these steps, it is now possible to join the network with our device:

- Setting up the connection with the RN2483 via a RS232-terminal and the following settings:

bitrate: *57600*

Data bits: *8*

Stop bits: *1*

Pairity: *none*

Flow control: *none*

terminated with <CR><LF>

- All commands to operate the device can be found in the RN2483 LoRa® Technology Module Command Reference User's Guide by Microchip. The commands for performing a OTAA join and an uplink message are:

OTAA join: **mac join otaa**

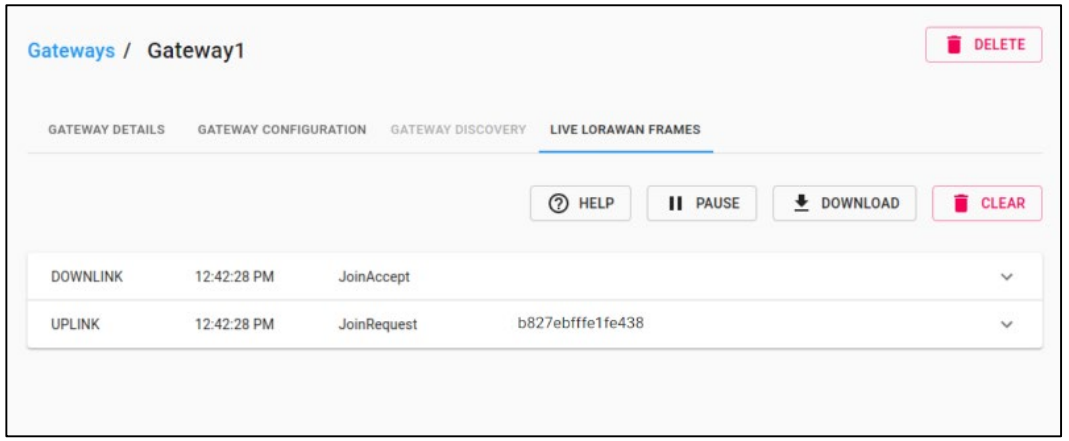
TX message: **mac tx <type> <portno> <data>**

- <type>: cnf (confirmed) or uncnf (unconfirmed)
- <portno>: Port number (1 to 223)
- <data>: data message

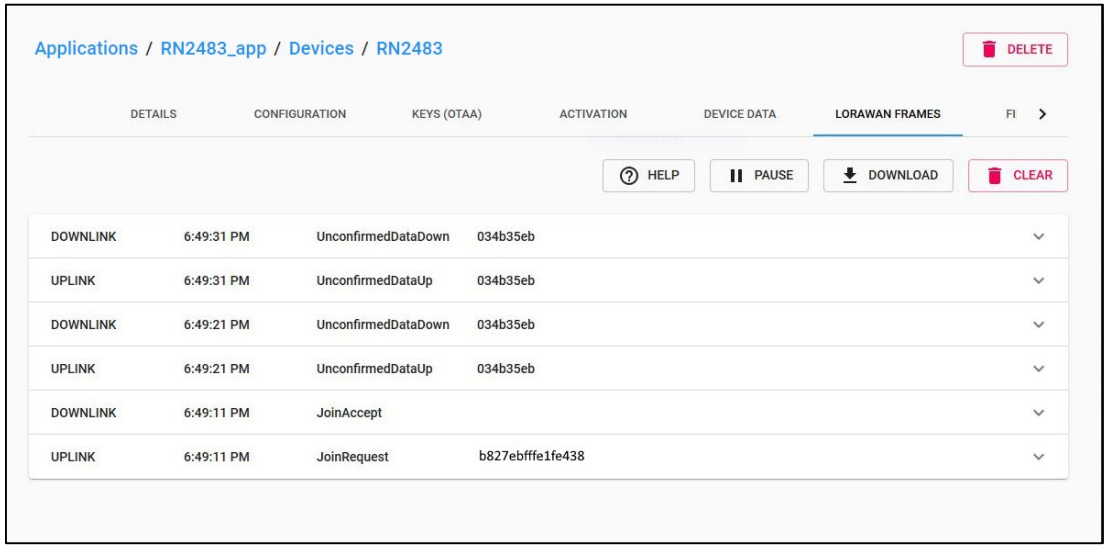
Connecting a Device

Message Flow:

- The message flow of the LoRa messages received and transmitted by the gateway can be found in the gateway section in the web-interface.



- The LoRaWAN message flow can be found in the application section of the web-interface.

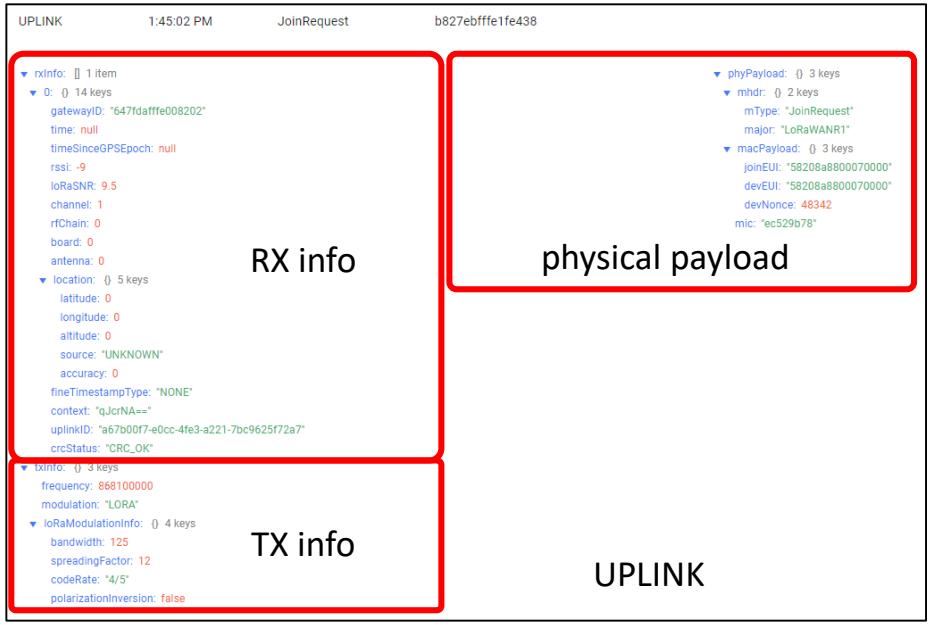


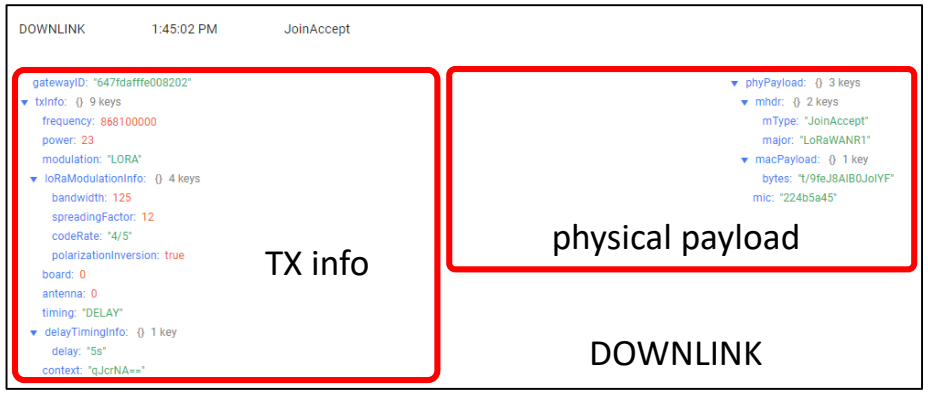
Connecting a Device

Message Flow:

An LoRaWAN message is divided in up to 3 parts

- RX info: contains information about the quality of the received signal like RSSI and SNR
- TX info: includes information of the LoRa parameters like spreading factor, code rate, bandwidth and frequency channel
- Physical payload





Connecting a Device

Message Flow OTAA join :

- Join Request
  - Contains joinEUI, devEUI, and devNonce for the calculation of the session keys
- Join Accept
  - Contains encrypted data like JoinNonce, NetID, and DevAddr

DOWNLINK	6:49:31 PM	UnconfirmedDataDown	26011a87
UPLINK	6:49:31 PM	UnconfirmedDataUp	26011a87
DOWNLINK	6:49:21 PM	UnconfirmedDataDown	26011a87
UPLINK	6:49:21 PM	UnconfirmedDataUp	26011a87
DOWNLINK	6:49:11 PM	JoinAccept	
UPLINK	6:49:11 PM	JoinRequest	b827ebfffe1fe438

JoinRequest:


```
▼ phyPayload: {} 3 keys
  ▼ mhdr: {} 2 keys
      mType: "JoinRequest"
      major: "LoRaWANR1"
  ▼ macPayload: {} 3 keys
      joinEUI: "58208a8800070000"
      devEUI: "58208a8800070000"
      devNonce: 48342
    mic: "ec529b78"
```


JoinAccept:


```
▼ phyPayload: {} 3 keys
  ▼ mhdr: {} 2 keys
      mType: "JoinAccept"
      major: "LoRaWANR1"
  ▼ macPayload: {} 1 key
      bytes: "t/9feJ8AIB0JolYF"
    mic: "224b5a45"
```

Connecting a Device

Message Flow after join procedure:

- In this example, several unconfirmed uplink messages were transmitted after the join procedure
- Every uplink is followed by a downlink with network parameters the device must adapt. In this case:
  - “RX ParamSetupReq”: sets the frequency and data rate of the second receive window.
  - “RX TimingSetupReq”: sets the delay between the end of the reception of an uplink and the opening of the first reception slot
- Further network parameters like maximum duty cycle or adding new channels
- A list of MAC Commands exchanged between receiver and transmitter are documented in the LoRaWAN specification by the LoRa Alliance

DOWNLINK	6:49:31 PM	UnconfirmedDataDown	26011a87
UPLINK	6:49:31 PM	UnconfirmedDataUp	26011a87
DOWNLINK	6:49:21 PM	UnconfirmedDataDown	26011a87
UPLINK	6:49:21 PM	UnconfirmedDataUp	26011a87
DOWNLINK	6:49:11 PM	JoinAccept	
UPLINK	6:49:11 PM	JoinRequest	b827ebfffe1fe438

Uplink:


```
▾ phyPayload: {} 3 keys
  ▾ mhdr: {} 2 keys
      mType: "UnconfirmedDataUp"
      major: "LoRaWANR1"
  ▾ macPayload: {} 3 keys
    ▾ fhdr: {} 4 keys
        devAddr: "26011a87"
      ▾ fCtrl: {} 5 keys
          adr: true
          adrAckReq: false
          ack: false
          fPending: false
          classB: false
        fCnt: 1
        fOpts: null
      fPort: 1
    ▾ frmPayload: [] 1 item
      ▾ 0: {} 1 key
          bytes: "b2Y3IX9O/1/mhRIabQ=="
  mic: "cee0d2e2"
```


Downlink:


```
▾ phyPayload: {} 3 keys
  ▾ mhdr: {} 2 keys
      mType: "UnconfirmedDataDown"
      major: "LoRaWANR1"
  ▾ macPayload: {} 3 keys
    ▾ fhdr: {} 4 keys
        devAddr: "26011a87"
      ▾ fCtrl: {} 5 keys
          adr: true
          adrAckReq: false
          ack: false
          fPending: false
          classB: false
        fCnt: 1
      ▾ fOpts: [] 2 items
        ▾ 0: {} 2 keys
            cid: "RXParamSetupReq"
          ▾ payload: {} 2 keys
              frequency: 868300000
              dlSettings: "08"
        ▾ 1: {} 2 keys
            cid: "RXTimingSetupReq"
          ▾ payload: {} 1 key
              delay: 1
      fPort: null
      frmPayload: null
  mic: "579fcb05"
```

Connecting a Device

Message Flow after join procedure:

- There are two different frame counters that keep track of uplink and downlink frames:
  - FcntUp: Number of uplink messages, transmitted and incremented by the device via “fCnt” field
  - FcntDown: Number of downlink messages, transmitted and incremented by the network server via “fCnt” field
- If either the device or the network receives a message with a frame counter that is lower than the last one, the message is ignored

DOWNLINK	6:49:31 PM	UnconfirmedDataDown	26011a87
UPLINK	6:49:31 PM	UnconfirmedDataUp	26011a87
DOWNLINK	6:49:21 PM	UnconfirmedDataDown	26011a87
UPLINK	6:49:21 PM	UnconfirmedDataUp	26011a87
DOWNLINK	6:49:11 PM	JoinAccept	
UPLINK	6:49:11 PM	JoinRequest	b827ebfffe1fe438

Uplink:


```
▾ phyPayload: {} 3 keys
  ▾ mhdr: {} 2 keys
      mType: "UnconfirmedDataUp"
      major: "LoRaWANR1"
  ▾ macPayload: {} 3 keys
    ▾ fhdr: {} 4 keys
        devAddr: "26011a87"
      ▾ fCtrl: {} 5 keys
          adr: true
          adrAckReq: false
          ack: false
          fPending: false
          classB: false
        fCnt: 1
        fOpts: null
      fPort: 1
    ▾ frmPayload: [] 1 item
      ▾ 0: {} 1 key
          bytes: "b2Y3IX9O/1/mhRIabQ=="
    mic: "cee0d2e2"
```


Downlink:


```
▾ phyPayload: {} 3 keys
  ▾ mhdr: {} 2 keys
      mType: "UnconfirmedDataDown"
      major: "LoRaWANR1"
  ▾ macPayload: {} 3 keys
    ▾ fhdr: {} 4 keys
        devAddr: "26011a87"
      ▾ fCtrl: {} 5 keys
          adr: true
          adrAckReq: false
          ack: false
          fPending: false
          classB: false
        fCnt: 1
      ▾ fOpts: [] 2 items
        ▾ 0: {} 2 keys
            cid: "RXParamSetupReq"
          ▾ payload: {} 2 keys
              frequency: 868300000
              dlSettings: "08"
        ▾ 1: {} 2 keys
            cid: "RXTimingSetupReq"
          ▾ payload: {} 1 key
              delay: 1
      fPort: null
      frmPayload: null
    mic: "579fcb05"
```

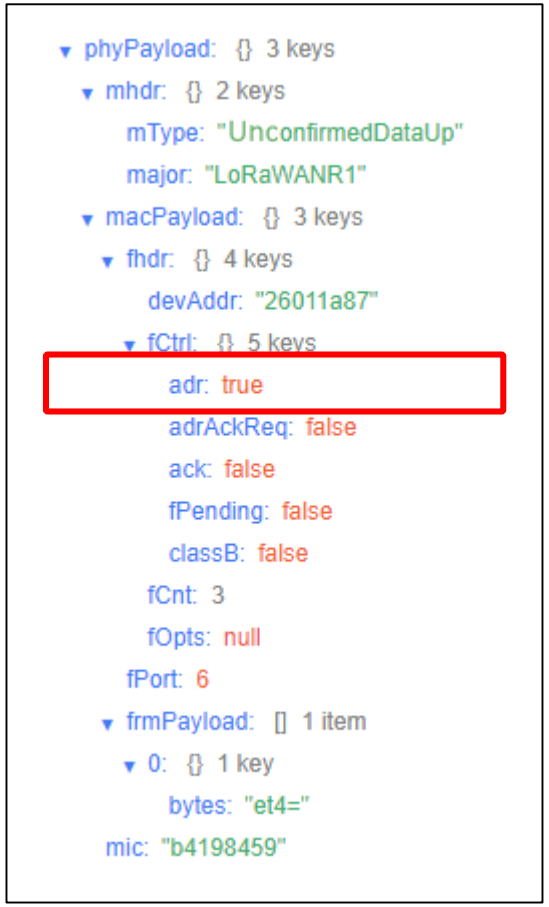
Connecting a Device

Message Flow ADR:

If the ADR flag is set to one, the network server can adapt parameters of spreading factor (data rate), power, and useable channels for an end device.

Uplink message:

- ADR bit is set to one -> ADR is activated
- Spreading factor 12



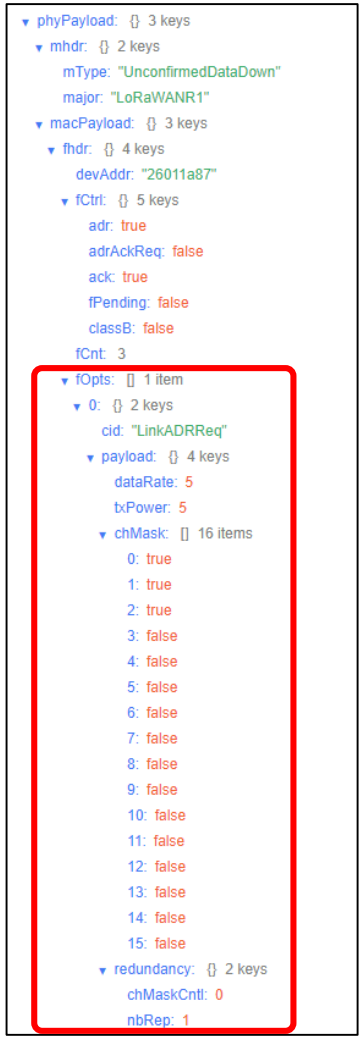
Connecting a Device

Message Flow ADR:

A downlink message follows wit a “LinkADRReq” command to set new parameters for the device:

- dataRate: 5
- txPower: 5
- chMask: enabled channels; 0, 1, and 2 are the mandatory channels

UPLINK	7:01:25 PM	UnconfirmedDataUp	26011a87
DOWNLINK	7:01:13 PM	UnconfirmedDataDown	26011a87
UPLINK	7:01:13 PM	UnconfirmedDataUp	26011a87

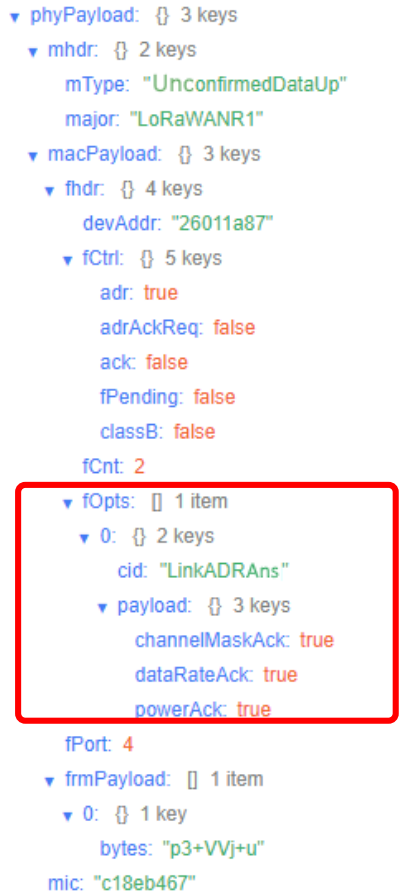


Connecting a Device

Message Flow ADR:

- Next uplink message adapted the parameters from the ADR request
- “LinkADRAns” includes acknowledgments of the changed power, data rate, and channel mask.
- The network server changes transmission parameters until suitable parameters in terms of connectivity and battery efficiency is achieved

UPLINK	7:01:25 PM	UnconfirmedDataUp	26011a87
DOWNLINK	7:01:13 PM	UnconfirmedDataDown	26011a87
UPLINK	7:01:13 PM	UnconfirmedDataUp	26011a87



Connecting a Device

Message Flow:

- After several responses to an uplink message with network and transmission parameter settings, there is no downlink response to an unconfirmed message.