

Product : Nemeus Gateway Quick user Guide

Doc : User Manual

Reference :

History : V1.2

Table of contents

Table of contents	2
Disclaimer	4
Document history	5
References	5
Revisions	5
1. Product overview.....	6
1.1 Product overview.....	6
1.2 Applications.....	6
2. Gateway I/O and mechanical description	7
2.1 Identification	7
2.2 HW Interfaces	7
2.3 Network Interfaces	8
2.4 Tools.....	8
3. Configuration tool.....	8
3.1 Prerequisite	8
3.2 Installation / Execution.....	8
3.3 Logs.....	8
3.4 Master window.....	9
3.5 Action Windows	10
3.6 Configuration Folders	12
3.6.1 DHCP Parameters	12
3.6.2 SNTPParameters	13
3.6.3 SNET Parameters	13
3.6.4 FWD Parameters.....	13
3.6.5 MAC Parameters.....	13
3.6.6 RADIO Parameters	14
3.6.7 RX FILTER Parameters	15
3.6.8 UPDATE Parameters.....	15
3.7 Device management.....	15
3.7.1 Main device window	16
3.7.1.1 Main device window action.....	16
3.7.1.1.1 Top window action	16
3.7.1.1.2 Contextual menu	16
3.7.1.1.3 Context action button.....	17

3.7.1.2	Add/Set device window.....	17
3.7.1.2.1	Button action	17
3.7.1.2.2	Yaml file	18
3.7.1.3	Device context window	20
3.8	Installation / Running	21
4.	Interface with Data server	21
5.	Packet forwarder	22

Disclaimer

1. This document is provided for reference purposes only so that Nemeus customers may select the appropriate products for their use. Nemeus neither makes warranties or representation with respect to the accuracy or completeness of the information contained in this document, nor grants any license to any intellectual property rights or any other rights of Nemeus or any third party with respect to the information in this document.
2. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
3. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Nemeus products listed in this document, please confirm the latest product information with Nemeus company.
4. Nemeus has used reasonable care in compiling the information included in this document, but Nemeus assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
5. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Nemeus makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Nemeus products.
6. Nemeus products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. Use of Nemeus products for such application is under customer responsibility.
7. You should use the products described herein within the range specified by Nemeus, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Nemeus shall have no liability for malfunctions or damages arising out of the use of Nemeus products beyond such specified ranges.
8. In case Nemeus products listed in this document are detached from the products to which the Nemeus products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Nemeus products may not be easily detached from your products. Nemeus shall have no liability for damages arising out of such detachment.
9. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Nemeus.
10. Please contact Nemeus company (contact@nemeus.fr) if you have any questions regarding the information contained in this document.

Document history

Version	Date	Author	Comments
V0.1	11/02/16	Alan Kreutz	Initial
V0.2	22/02/16	Gilles Ronco	Update
V1.1	01/08/16	Gilles Ronco	Add LoRa WAN server
V1.2	22/08/16	Alan Kreutz	Add information

Table 1 : Document versions

References

[1] Nemeus MG003-L-EU-gateway

[2] LoRaWAN specification v1.0.1

Revisions

This document is dedicated to Nemeus Pico-Gateway.

1. Product overview

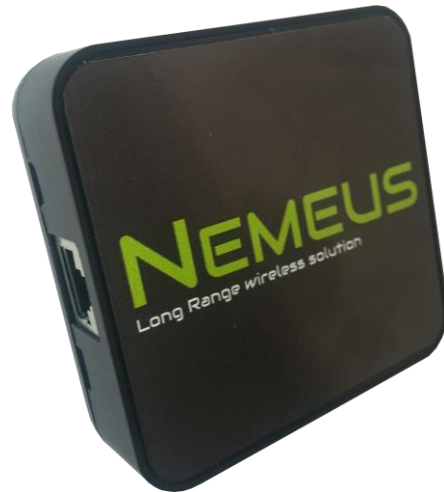
The **MG003-L-EU** is a **pico-gateway dedicated to long-range indoor communication** for low power wireless devices operating on ISM 868Mhz unlicensed band (European band). With interferers' robustness and low power consumption, it is a good solution for autonomous gateway.

This gateway implements LoRa & FSK modulations and can be used in two modes :

- Packet forwarder mode
- LoRa WAN 1.0.1 EU server

1.1 Product overview

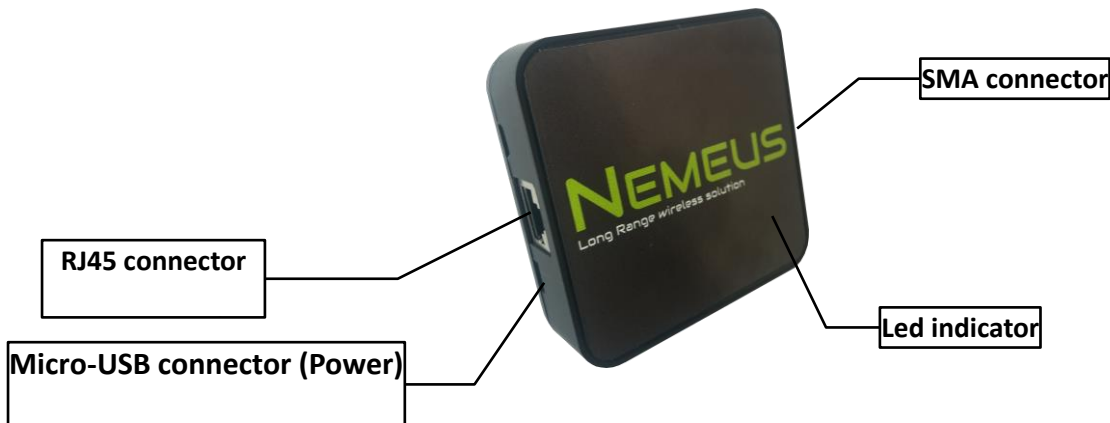
- Size 75mm x 75mm x 25mm
- Modulation : LoRa™ / FSK
- Maximum output power +26 dBm
- 8 channels Freq/SF solution
- Powered by 5v USB interface
- Communication through Ethernet interface



1.2 Applications

- Automated Meter reading
- Home and Building automation
- Industrial monitoring and control
- IOT (Internet of Things)

2. Gateway I/O and mechanical description



2.1 Identification

The gateway Identification is provided on a label on the gateway bottom. Information format is as follow :

HW version: MG003-EU-1.1
HW id: <HW ID>
MAC address: <aa:bb:cc:dd:ee:ff>
Default host name: <Nemeus-gw-aabbccdeeff>
SW: <SW version>

Note that if the hardware version ends with *FRAM*, the gateway is equipped of a FRAM. After power down, devices information are kept.

2.2 HW Interfaces

The gateway has following interfaces :

- A micro-USB connector which is used to power the gateway. The consumption of the gateway is compliant with USB power constraints and can be connected to a computer or a plug. This connector is not usable to control or configure the gateway.
- A RJ45 connector used to connect the gateway directly to a LAN. This interface is used for the Ethernet communication of the gateway with an external server (LoRa WAN 1.0.1 EU server in case of the gateway is used in Packet mode or Applicative server in case of use of LoRa WAN 1.0.1 EU embedded network server). This interface is also used for a remote configuration and control of the gateway through a LAN with a JAVA application.
- A SMA connector to plus the antenna. **The gateway must not be powered without antenna to avoid electronic damage.**
- A LED indicator giving a status on the current state of the gateway :
 - Blue (at power-ON) : Waiting dynamic IP address from a DHCP server (Dynamic mode)
 - Green : IDLE mode / Connected external server / no processing ongoing
 - Blue : Processing ongoing (DL frame emission to device) OR Ethernet disconnection

- Red : Tx issue / Conflict on Tx transmission for several devices

2.3 Network Interfaces

The Ethernet communication is based on UDP protocol. The UL/DL communication with external server (Applicative server in case of LoRa WAN 1.0.1 embedded server or external LoRa WAN server in case of packet server mode) is done through JSON. JSON format description is provided at the end of this document.

2.4 Tools

The gateway is provided with a JAVA application used to configure it through a LAN network.

3. Configuration tool

3.1 Prerequisite

Nemeus Java application to control the gateway requests that the JAVA Runtime Evaluation 8 kit is installed onto the computer. This one can be found on

<http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>)

3.2 Installation / Execution

Nemeus JAVA Application is made of only one file : o_NemeusCustomerGateway.jar. This one can be put in any folder.

To execute the JAVA application, the following command must be executed in a Command DOS windows in the folder in which the file is available.

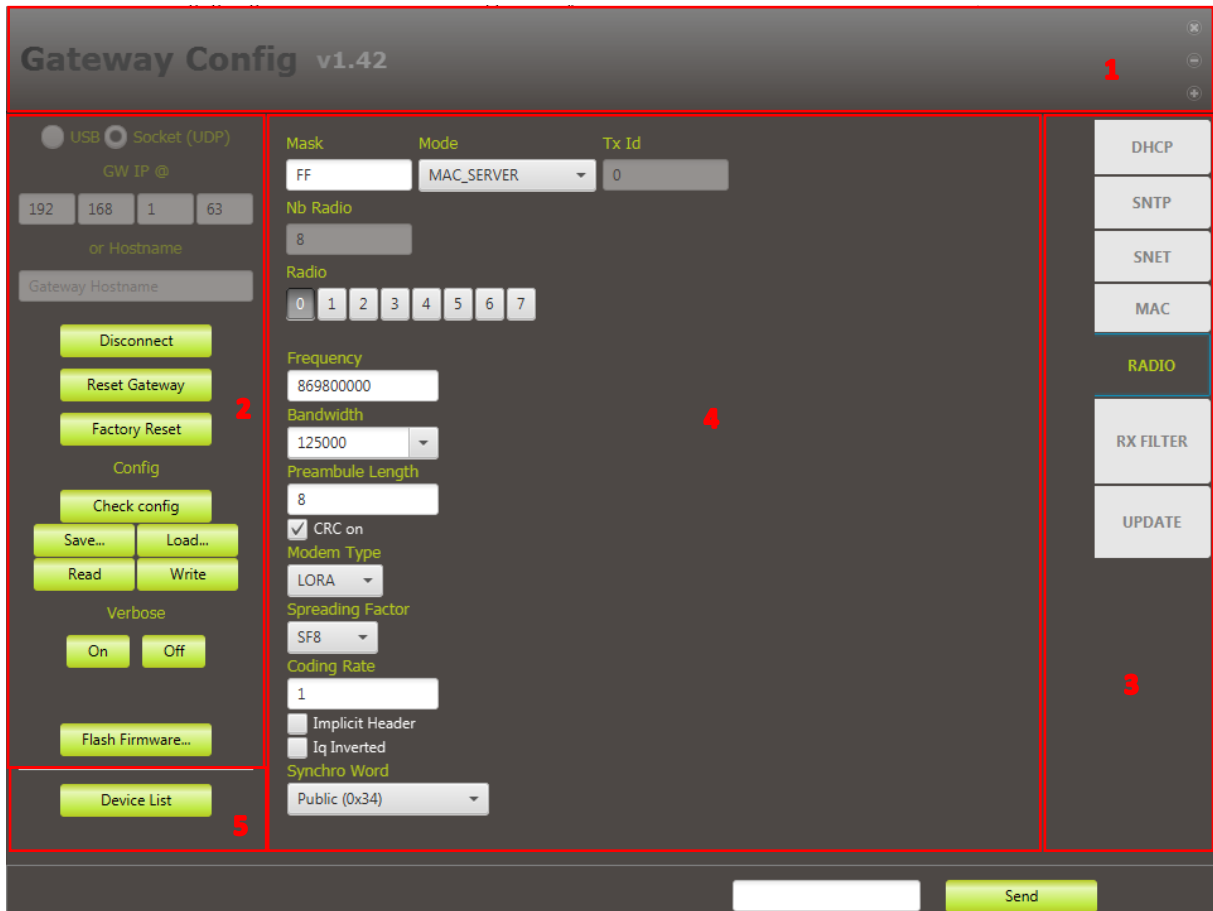
```
java -jar o_NemeusCustomerGateway.jar
```

3.3 Logs

During execution, the JAVA application records information in a log file which is created in same folder of the .jar file at the time this one is started. The generated log file stays the same during all the JAVA application execution. The file format is log-mgw-<xxxx> where xxxx corresponds to the date and time of the JAVA application start.

3.4 Master window

The following figure gives an overview of the JAVA application.



This one is based on 5 sub-windows:

- Sub-window 1 : The application title and the version of application
- Sub-window 2 : The Action window
- Sub-window 3 : The configuration folder
- Sub-window 4 : The configuration parameters
- Sub-window 5 : The devices configuration (LoRa WAN server configuration)

3.5 Action Windows



The following figure shows the Action window and the different control available.

USB/Socket(UDP) parameters :

By default the configuration of the gateway is done through the LAN. As option and for debug purpose the gateway can be configured/controlled through a Custom I/F inside the enclosure (called USB). This interface is not dedicated to be used by Customers.

The Socket(UDP) must be always activated.

GW IP @ or Hostname parameters:

These two parameters are exclusives and correspond to two ways to identify the gateway on the LAN. The Hostname parameter can be used to connect to the gateway through its hostname <Nemeus-gw-aabbccddeeff>. This solution is only possible if a DHCP server is reachable to resolve the name (**preferred solution**). The GW IP @ is used in case there is no DHCP server available. In such a case the IP @ must previously forced on the gateway through a LAN having DHCP server available. (**This mode is only for LAN without DHCP server**)

Connect button :

As soon as the Gateway Hostname or GW IP has been set, the “Connect” button is used to create a socket between the JAVA application and the Gateway. **Notice : The Gateway must disconnected before closing the JAVA application or trying to**

connect to another gateway.

Reset gateway :

As soon as the gateway is connected to the JAVA application, the Gateway can be reset through the “Reset Gateway” button. This action is needed in order that the gateway takes into account the new configuration written in the filesystem of the gateway.

Factory Reset :

Reset the gateway and recover the factory settings. This action deletes all user configuration and could be used to recover a stable state.

Configuration action:

The action “Check config”, “Save”, “Load”, “Read”, “Write” deals about the gateway parameters displayed on the application. Gateway configuration is stored on the gateway filesystem. It could be read from & write to the gateway file system to view or change the gateway parameters. The application allows to save to a .cfg file or load a .cfg file.

Check Config: Check the configuration set in the application view and check if the format is correct.

Save: Save gateway configuration displayed in application in a .cfg file on the computer. The configuration is stored in binary format.

Load: Load gateway configuration from a .cfg file. The application checks the file content. If the configuration is correct, the parameters are set in the application view.

Read: send an action to read the gateway configuration file and display parameters in application view if the format is correct.

Write: send an action to write the gateway configuration file.

Verbose:

This action simply enable(ON)/disable(OFF) the verbose trace on java console. This is helpful in case of DEBUG.

Flash firmware:

As soon as the gateway is connected to the JAVA application, the Gateway can be flashed with a new firmware through the “Flash firmware” button.

You have to select a binary file provided by Nemeus company. This binary file contains the new firmware to flash with CRC. During the flash firmware, you could see a progress indicator window. Do not turn off power of gateway. Do not unplug the USB cable or ethernet cable. The flash firmware action is done by sending binary segment on USB or UDP interface.

After flash update success, the gateway reset. You must wait a few minutes during filesystem creation. It is important to not stop the process in order to have a correct filesystem.

Device list:

As soon as the gateway is connected to the JAVA application, the Gateway, configured as LoRaWAN server, can manage the associated devices through the “Device List” button.

3.6 Configuration Folders

8 configuration folders are available. Each one is selected through the corresponding button on right side:

- **DHCP folder** : DHCP configuration and parameters
- **SNTP folder** : Simple Network Time Protocol configuration
- **SNET folder** : Data & Secu Server connection parameters
- **MAC folder**: MAC parameters. This tab is displayed only if gateway is in LoraWAN router mode.
- **FWD folder**: Packet Forwarder parameters. This tab is displayed only if gateway is in packet forwarder mode.
- **RADIO folder**: LoRa/FsK Radio(s) configuration
- **RX FILTER folder** : Rx Frame Filter
- **UPDATE folder** : Gateway information

These folders allows to display different parts of gateway configuration. Click on the corresponding button to display the parameters on the center part of application.

3.6.1 DHCP Parameters

The gateway IP address can be selected by two ways:

- Static IP address – Static mode
- Dynamic IP address selection – Dynamic mode

The mode parameter allows to select the configuration between Static or Dynamic.

Source IP@, Subnet Mask, router IP@ and DNS are editable only in static mode

DCHP parameters are following:

- Source MAC @: Nemeus Gateway MAC address (Always Read only parameter)
- Period: DHCP client scheduling (**must not be modified**)
- Ignore Ping checkbox (to not answer to ICMP)
- Source IP @: Nemeus Gateway IP address
- Subnet Mask: subnet mask in your local network
- Router IP @: IP address of your local network router (internet box...)
- DNS 1 IP @: IP address of DNS 1
- DNS 2 IP @: IP address of DNS 2

-

3.6.2 SNTP Parameters

The SNTP parameters are used to select a SNTP server from which the gateway will pick the current time periodically to update its internal time base. The SNTP server can be configured either by SNTP name (SNTP URL) or by IP address. If SNTP name is not empty, the IP address can't be edited. The parameters are:

- SNTP server name
- SNTP IP address
- Period : SNTP server polling period in minutes
- Repeat: number of repeat in case of failure

3.6.3 SNET Parameters

The SNET parameters are used to configure the Data server & Security server. The data server is the Customer server to exchange data from/to the gateway in UL and DL. By default, the security server is a Nemeus server used for Gateway FW update. The parameters are following :

- Server name or IP address for Data server & Security server
- Server Port communication
- Data Server period : PULL DATA periodicity
- Security server period: security key update period
- Security Key: Original security key for ciphering

3.6.4 FWD Parameters

DO NOT TOUCH these parameters

- Downlink Frequency: Force the downlink frequency to use (**keep 0**)
- Downlink Timestamp offset: timestamp server offset to add (**keep current value**)
- Downlink SF: Force downlink Spreading Factor (**keep 0**)
- Stat Period: Statistic period sent to data server (**for debug purpose / Do not touch**)
- Gateway EUI: Unique ID for gateway. This ID is contained in the frames sent by gateway. By default, this is based on the MAC address.

3.6.5 MAC Parameters

- Network ID: Network identifier

- Router port: LoRaWAN port used by Chat application
- Rx packet mode: Uplink frame destination.
 - Local mode means the uplink frame is forwarded to Java application
 - Remote mode means the uplink frame is forwarded to data server
- Rx packet format for local mode: Format of uplink data frame forwarded to Java application (local mode): JSON, BASE64 in AT unsol, hexadecimal format in AT unsol.

3.6.6 RADIO Parameters

These parameters are used to configure the Rx mode of the 8 transceivers. Each transceiver can be enabled/disabled by Mask parameter (in hexadecimal). FF means 8 SX to use.

Mode indicates the gateway mode. Simple packet forwarder mode to an external LoRa WAN server or internal LoRa WAN server mode.

TX ID indicates which transceiver is used for TX transmission from the gateway to the devices. It is not modifiable. This transceiver will switch to Tx as soon as the gateway has to transmit a frame to a device. During this period, the transceiver is not able to receive any frame on Rx.

The radio buttons (0 to 7) are used to select a transceiver to configure with Rx parameters. The parameters below are the radio parameters of this transceiver:

- The RX frequency
- The Bandwidth
- The Preamble length
- The CRC
- The modulation type : LoRa or FSK

For LoRa modulation, the parameters are following:

- Spreading Factor (SF7 to SF12)
- Coding Rate
- Implicit Header (boolean)
- Iq inverted (boolean)
- Synchro word

For FSK modulation, the parameters are following:

- Bandwidth AFC
- Baudrate
- Fixed lenbgh
- Synchro word length

- Synchro word

3.6.7 RX FILTER Parameters

Rx packet filter configures the accepted upcoming packet.

If a packet doesn't match with this rule, the packet is dropped. The parameters are following:

- Offset: byte offset for data checking
- Mask Number: number of masks
- Mask length: mask size in bytes
- Mask: Mask value compared to the Rx Frame at specified offset.

3.6.8 UPDATE Parameters

These parameters give information on FW version and Check period for new version. These parameters are:

- Version: CRC of file (not editable)
- Period: firmware update checking period

Furthermore, since the Gateway does not have an embedded GPS, a position of gateway can be introduced manually through following parameters:

- GW latitude,
- GW longitude
- GW altitude

This position is transferred to server.

3.7 Device management

In LoRaWAN server, the java application offers a user interface to manage the devices registered to the gateway.

Devices are registered in the filesystem. This part presents the user interface to display, read and write information in device gateway file.

At the bottom of the action window, the "Device List" button displays a new window to manage devices.

3.7.1 Main device window

The device list displays all registered devices with the file index.

When user connects to gateway, the application automatically reads the device file content. If some devices are registered, the window will display in the device list window.

Index	Config				Context
	Device EUI	App Key	Device Type	Rx W2 Frequency	
0	70B3D5326000000	00010203040506070809...	FF11180E	869525000	Context
1	333738390F8D3469	8F18F3207163194DBE59...	FF11180F	869525000	Context

You will find one line by device. This line is composed of multiple cells with an index (index in the device file), device EUI, Application key, device type (device option represented in a bit field), Rx window 2 frequency and an action button to display more about device.

3.7.1.1 Main device window action

Different actions are possible.

3.7.1.1.1 Top window action

Top window button action:

- “Add Device”: open a window to add a new device
- “Get All Devices”: refresh the devices list by asking it to gateway
- “Load Yaml...”: load an external .yaml file to add a list of devices from a yaml format file

3.7.1.1.2 Contextual menu

A mouse right click on a cell displays a contextual menu:

- “Edit Device...”: to edit the device. This display the same window as the “Add Device” button. The window fields are set with the device value. It allows to see a device type bit fields representation
- “Delete Device” to delete the device from the gateway

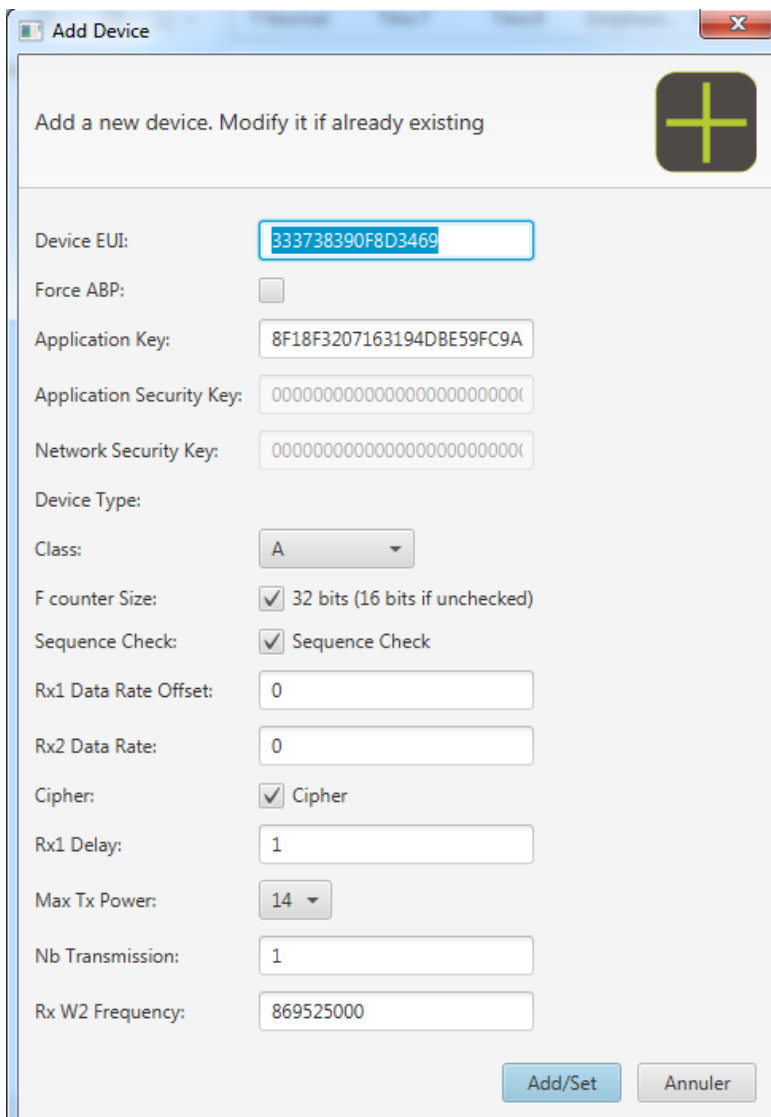
3.7.1.1.3 Context action button

The context action button in a device line displays context parameters from a device.

3.7.1.2 Add/Set device window

3.7.1.2.1 Button action

Using top bottom “Add device” or contextual menu “Edit device...” displays the window:



The screenshot shows a window titled "Add Device" with a close button (X) in the top right corner. The window contains the following fields and controls:

- Header: "Add a new device. Modify it if already existing" with a green plus icon in a black square.
- Device EUI: Text input field containing "333738390F8D3469".
- Force ABP: Unchecked checkbox.
- Application Key: Text input field containing "8F18F3207163194DBE59FC9A".
- Application Security Key: Text input field containing "00000000000000000000000000000000".
- Network Security Key: Text input field containing "00000000000000000000000000000000".
- Device Type:
 - Class: Dropdown menu showing "A".
 - F counter Size: Checked checkbox with text "32 bits (16 bits if unchecked)".
 - Sequence Check: Checked checkbox with text "Sequence Check".
 - Rx1 Data Rate Offset: Text input field containing "0".
 - Rx2 Data Rate: Text input field containing "0".
 - Cipher: Checked checkbox with text "Cipher".
 - Rx1 Delay: Text input field containing "1".
 - Max Tx Power: Dropdown menu showing "14".
 - Nb Transmission: Text input field containing "1".
 - Rx W2 Frequency: Text input field containing "869525000".
- Buttons: "Add/Set" and "Annuler" at the bottom right.

The parameters to configure are:

- Device EUI
- Application Key
- Application Security Key if Force ABP checkbox is checked
- Network Security Key if Force ABP checkbox is checked
- Device Type:
 - o LoRa WAN class (A or C)
 - o Frame counter size (counter on 16 or 32 bits)
 - o Sequence Check
 - o Rx window 1 Data Rate Offset
 - o Rx window 2 Data Rate
 - o Ciphering enable/disable
 - o Rx window 1 delay
 - o Max Tx power
 - o Number of transmission
 - o Rx window 2 frequency

3.7.1.2.2 Yaml file

To enter multiple devices, it is possible to load a yaml file with the device information.

The file begins with

And ends with:

...

Between every entry, the --- separator must be present

Example:

```
---  
# the device uid between quote (string)  
deviceUid: '454545454545'  
# the application key between quote (string)  
appKey: '00000000000000000000000000000000'  
# Force ABP true/false  
abp: true
```

```
# Application SKey if abp is true
appSKey: '00000000000000000000000000000000'
# Network SKey if abp is true
nwkSKey: '00000000000000000000000000000000'
# Device class. 0 means class A, 2 means class C
deviceClass: 0
# Frame counter size. true means 32 bits, false means 16 bits
fcounterSize: true
# Sequence check
sequenceCheck: true
# Rx1 Data rate offset
rx1DataRateOffset: 0
# Rx2 Data Rate according to the lorawan specification
rx2DataRate: 0
# ciphering enable or disable
ciphered: true
# rx1 Delay
rx1Delay: 1
# max Tx power
maxTxPower: 1
# Number of transmission
nbTransmission: 1
# Rx window 2 frequency in Hz
rxW2Frequency: 869525000
# Device Address
deviceAddress: '1212346'
# Last dev nonce
lastDevNonce: '0000'
# Frame counter Downlink
FCounterDownlink: 0
#Frame counter Uplink
FCounterUplink: 0
# device state. 0 UNREGISTERD, 1 REGISTERING, 2 REGISTERED
state: 0
```

number of downlink frames

nbDownlinkFrame: 0

deviceId: '123456789ABCDEF0'

appKey: '000102030405060708090A0B0C0D0E0F'

abp: false

deviceId: '123456789ABCDEF1'

appKey: '000102030405060708090A0B0C0D0E0F'

...

3.7.1.3 Device context window

The “Context” action button on each device entry displays a window with device context information:

The screenshot shows a window titled "Device index 333738390F8D3469". The window contains the following fields and controls:

- Device Address: D611B001
- Nwk SKey: 00000000000000000000000000000000
- App SKey: 00000000000000000000000000000000
- Last Dev Nonce: 0000
- FCNT Uplink: 0
- FCNT Downlink: 0
- State: UNREGISTERED (dropdown)
- Nb Downlink Frames: 0

	Payload	Port	Mode	Class	
Pending Frames 1	EMPTY		Unconf	A	Remove Frame
Pending Frames 2	EMPTY		Unconf	A	Remove Frame
Pending Frames 3	EMPTY		Unconf	A	Remove Frame
	Add		2	Unconf	A

At the bottom of the window, there are two buttons: "Get context" and "Set context".

The device context is composed of:

- Device Address (**not editable**)
- Network Security Key (**not editable**)

- Application Security Key (**not editable**)
- Last Dev Nonce
- Frame Counter Uplink
- Frame Counter Downlink
- Device State (*UNREGISTERED, REGISTERING, REGISTERED*)
- Number of pending downlink frames
- Pending frames information (payload, port, mode, class). For each pending frames, it is possible to remove the pending frame from the server.
- Possibility to add a pending frame (payload, port, mode, class)

There are 2 action buttons to get and set the device context.

3.8 Installation / Running

This application must be executed from the console with the following command:

```
java -jar NemeusGateway.jar (request JAVA 8 Update 65 minimum version)
```

Debug information are printed on the console. A log file is also generated in the folder in which the JAVA is executed.

The verbose "ON/OFF" is used to display on the console and write in the log file the debug information from JAVA application and gateway.

4. Interface with Data server

The interface with the data server depends on the gateway mode. Currently, Packet Forwarder mode is described.

5. Packet forwarder

The interface used for the packet forwarder is the same as the one defined by Semtech (protocol version 1).

Git: https://github.com/Lora-net/packet_forwarder

You must use a release inferior to v3.0.0

The last PROTOCOL.TXT file (version 1) is:



PROTOCOL.TXT

See Appendix A.

5.1 LoRaWAN frames description

5.1.1.1 Uplink frames

An example of JSON frame with LoRaWAN packet:

```
{ "lwpk":  
  [{"time": "2016-07-19T10:23:36.000000Z",  
    "freq": 868.100000,  
    "datr": "DR4",  
    "rssi": -96,  
    "lsnr": 7.0,  
    "deui": "70B3D53260000000",  
    "mode": "UNCONF",  
    "dadd": "D629A000",  
    "adrb": 0,  
    "aarb": 0,  
    "ackb": 0,  
    "cntu": 0,  
    "cntd": 0,  
    "port": 2,  
    "lmic": "811F1DC6",  
    "size": 2,  
    "data": "yv4="}]}
```

The associated metadata description:

"freq": central frequency in MHz

"datr": LoRa data rate identifier

"rssi": RSSI in dBm

"lsnr": LoRa SNR ratio in dB

"deui": device EUI

"mode": frame mode

"dadd": device Address

"adr": Adaptative Data Rate bit value

"aarb": ADR ACK Request bit

"ackb": ACK bit

"cntu": frame counter uplink

"cntd": frame counter downlink

"port": application port

"lmic": MIC

"size": packet payload size in bytes

"data": Base64 encoded packet payload

5.1.1.2 Downlink frames

A downlink frames is added in the device FIFO list. If list is full, the oldest is deleted and the new frame is added.

```

{"lwpk":
  {"deui": "70B3D53260000000",
   "mode": "UNCONF",
   "port": 2,
   "clas": "A",
   "data": "yv7eyg==",
   "size": 4}}

```

The associated metadata description:

"deui": device Unique Identifier

"mode": Acknowledgement mode

"port": application port

"clas": frame class

"data": Base64 encoded packet payload

"size": packet payload size in bytes

To delete the first FIFO frame:

```

{"lwpk":
  {"deui": "70B3D53260000000",
   "size": 0,
   "frid": 0}}

```

OR

```

{"lwpk":
  {"deui": "70B3D53260000000",
   "size": 0}}

```

To delete the second frame of FIFO list:

```
  {"lwpk":  
    {"deui":"70B3D53260000000",  
     "size":0,  
     "frid":1}}
```

To delete the third frame of FIFO list:

```
  {"lwpk":  
    {"deui":"70B3D53260000000",  
     "size":0,  
     "frid":2}}
```

“frid”: frame ID in the FIFO list

Annex A.

_____ -
/____) _ ||
((____ _ _ | | _ _ _ | | _
 _ \ | _ | (_) _ | / _) _ \
____)) _ | | | | | _ | _ ((_ | | | |
(____ / | ____) _ | _ | \) ____) \ ____) _ | | |
(C)2013 Semtech-Cycleo

Basic communication protocol between Lora gateway and server

=====

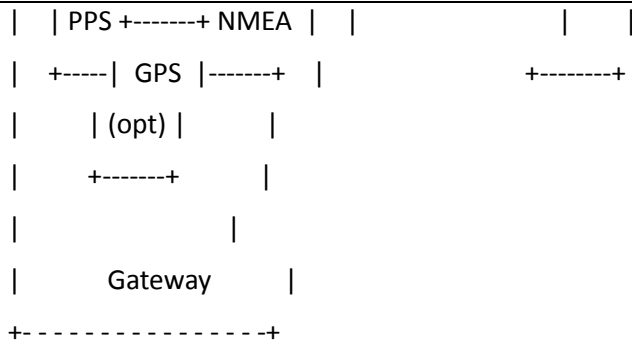
1. Introduction

The protocol between the gateway and the server is purposefully very basic and for demonstration purpose only, or for use on private and reliable networks.

There is no authentication of the gateway or the server, and the acknowledges are only used for network quality assessment, not to correct UDP datagrams losses (no retries).

2. System schematic and definitions

(((Y)))
|
|
+---|-----+ xxxxxxxxxxxx +-----+
+-+-----+ +-----+	xx x x xxx					
					xx Internet xx	
		SPI			xx Intranet xx	Server
+-----+ +-----+	xxxx x xxxx					
^ ^	xxxxxxxx					



__Concentrator__: radio RX/TX board, based on Semtech multichannel modems (SX130x), transceivers (SX135x) and/or low-power stand-alone modems (SX127x).

__Host__: embedded computer on which the packet forwarder is run. Drives the concentrator through a SPI link.

__GPS__: GNSS (GPS, Galileo, GLONASS, etc) receiver with a "1 Pulse Per Second" output and a serial link to the host to send NMEA frames containing time and geographical coordinates data. Optional.

__Gateway__: a device composed of at least one radio concentrator, a host, some network connection to the internet or a private network (Ethernet, 3G, Wifi, microwave link), and optionally a GPS receiver for synchronization.

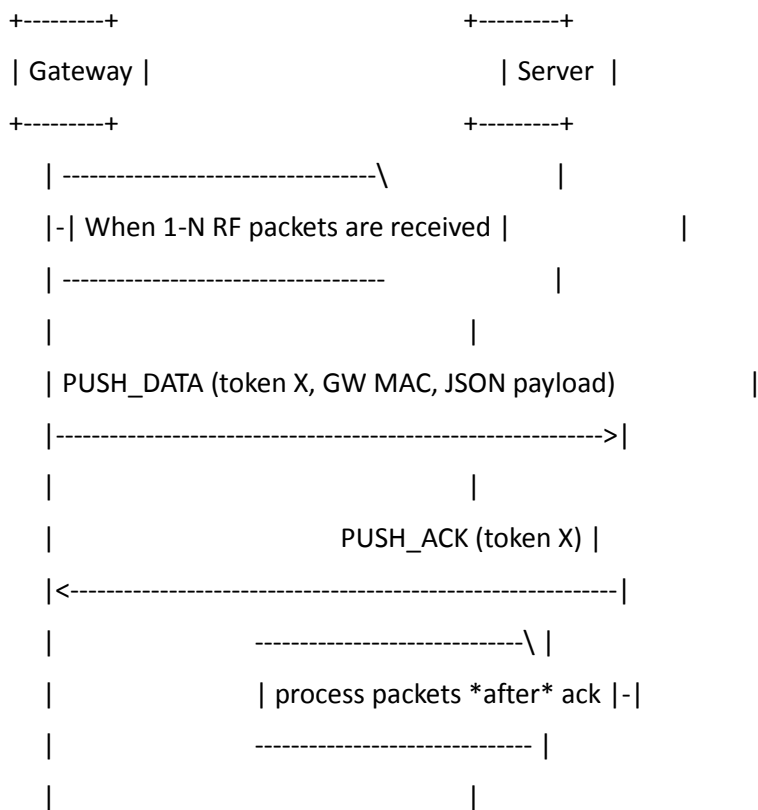
__Server__: an abstract computer that will process the RF packets received and forwarded by the gateway, and issue RF packets in response that the gateway will have to emit.

It is assumed that the gateway can be behind a NAT or a firewall stopping any incoming connection.

It is assumed that the server has an static IP address (or an address solvable through a DNS service) and is able to receive incoming connections on a specific port.

3. Upstream protocol

3.1. Sequence diagram



3.2. PUSH_DATA packet

That packet type is used by the gateway mainly to forward the RF packets received, and associated metadata, to the server.

Bytes | Function

```

:-----:|-----
0   | protocol version = 1
1-2 | random token
3   | PUSH_DATA identifier 0x00
4-11 | Gateway unique identifier (MAC address)
12-end | JSON object, starting with {, ending with }, see section 4

```

3.3. PUSH_ACK packet

That packet type is used by the server to acknowledge immediately all the PUSH_DATA packets received.

Bytes | Function

```

:-----:|-----:
0 | protocol version = 1
1-2 | same token as the PUSH_DATA packet to acknowledge
3 | PUSH_ACK identifier 0x01

```

4. Upstream JSON data structure

The root object can contain an array named "rxpk":

```

``` json
{
 "rxpk":[{...}, ...]
}
```

```

That array contains at least one JSON object, each object contain a RF packet and associated metadata with the following fields:

Name | Type | Function

```

:----:|:-----:|-----:
time | string | UTC time of pkt RX, us precision, ISO 8601 'compact' format
tmst | number | Internal timestamp of "RX finished" event (32b unsigned)
freq | number | RX central frequency in MHz (unsigned float, Hz precision)
chan | number | Concentrator "IF" channel used for RX (unsigned integer)
rfch | number | Concentrator "RF chain" used for RX (unsigned integer)
stat | number | CRC status: 1 = OK, -1 = fail, 0 = no CRC
modu | string | Modulation identifier "LORA" or "FSK"
datr | string | LoRa datarate identifier (eg. SF12BW500)
datr | number | FSK datarate (unsigned, in bits per second)
codr | string | LoRa ECC coding rate identifier
rssi | number | RSSI in dBm (signed integer, 1 dB precision)
lsnr | number | Lora SNR ratio in dB (signed float, 0.1 dB precision)
size | number | RF packet payload size in bytes (unsigned integer)
data | string | Base64 encoded RF packet payload, padded

```

Example (white-spaces, indentation and newlines added for readability):

```
``` json
{"rxpk":[
 {
 "time":"2013-03-31T16:21:17.528002Z",
 "tmst":3512348611,
 "chan":2,
 "rfch":0,
 "freq":866.349812,
 "stat":1,
 "modu":"LORA",
 "datr":"SF7BW125",
 "codr":"4/6",
 "rssi":-35,
 "lsnr":5.1,
 "size":32,
 "data":"-DS4CGaDCdG+48eJNM3Vai-zDpsR71Pn9CPA9uCON84"
 },{
 "time":"2013-03-31T16:21:17.530974Z",
 "tmst":3512348514,
 "chan":9,
 "rfch":1,
 "freq":869.1,
 "stat":1,
 "modu":"FSK",
 "datr":50000,
 "rssi":-75,
 "size":16,
 "data":"VEVTVF9QQUNLRVRfMTIzNA=="
 },{
 "time":"2013-03-31T16:21:17.532038Z",
 "tmst":3316387610,
 "chan":0,
 "rfch":0,
 "freq":863.00981,
 "stat":1,
```

```

 "modu":"LORA",
 "datr":"SF10BW125",
 "codr":"4/7",
 "rssi":-38,
 "lsnr":5.5,
 "size":32,
 "data":"ysgRI452xNLep9S1NTIglomKDxUgn3DJ7DE+b00Ass"
 }
]}
 ...

```

The root object can also contain an object named "stat" :

```

 `` json
 {
 "rxpk":[{...}, ...],
 "stat":{...}
 }
 ...

```

It is possible for a packet to contain no "rxpk" array but a "stat" object.

```

 `` json
 {
 "stat":{...}
 }
 ...

```

That object contains the status of the gateway, with the following fields:

Name	Type	Function
time	string	UTC 'system' time of the gateway, ISO 8601 'expanded' format
lati	number	GPS latitude of the gateway in degree (float, N is +)
long	number	GPS longitude of the gateway in degree (float, E is +)
alti	number	GPS altitude of the gateway in meter RX (integer)
rxnb	number	Number of radio packets received (unsigned integer)

rxok | number | Number of radio packets received with a valid PHY CRC  
 rxfw | number | Number of radio packets forwarded (unsigned integer)  
 ackr | number | Percentage of upstream datagrams that were acknowledged  
 dwnb | number | Number of downlink datagrams received (unsigned integer)  
 txnb | number | Number of packets emitted (unsigned integer)

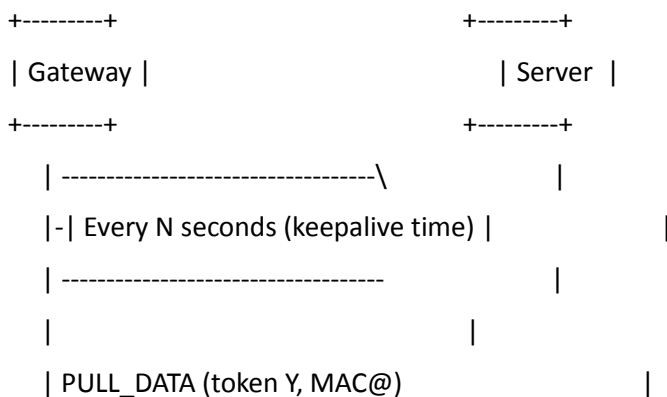
Example (white-spaces, indentation and newlines added for readability):

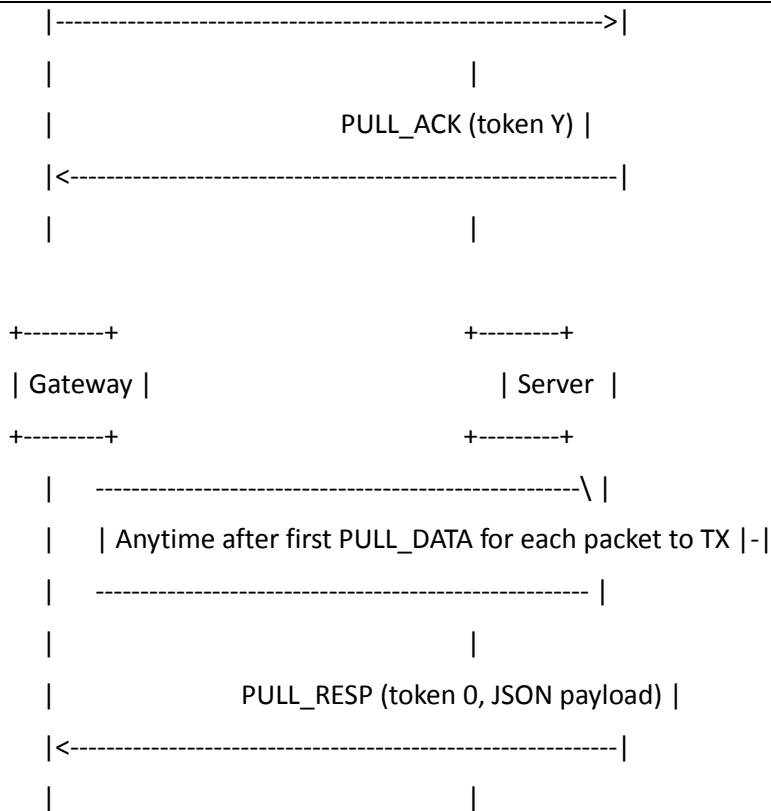
```
``` json
{"stat":{
  "time":"2014-01-12 08:59:28 GMT",
  "lati":46.24000,
  "long":3.25230,
  "alti":145,
  "rxnb":2,
  "rxok":2,
  "rxfw":2,
  "ackr":100.0,
  "dwnb":2,
  "txnb":2
}}
```
```

## 5. Downstream protocol

-----

### ### 5.1. Sequence diagram ###





### 5.2. PULL\_DATA packet ###

That packet type is used by the gateway to poll data from the server.

This data exchange is initialized by the gateway because it might be impossible for the server to send packets to the gateway if the gateway is behind a NAT.

When the gateway initialize the exchange, the network route towards the server will open and will allow for packets to flow both directions.

The gateway must periodically send PULL\_DATA packets to be sure the network route stays open for the server to be used at any time.

| Bytes | Function                                |
|-------|-----------------------------------------|
| 0     | protocol version = 1                    |
| 1-2   | random token                            |
| 3     | PULL_DATA identifier 0x02               |
| 4-11  | Gateway unique identifier (MAC address) |



### ### 5.3. PULL\_ACK packet ###

That packet type is used by the server to confirm that the network route is open and that the server can send PULL\_RESP packets at any time.

Bytes | Function

:-----:|-----

- 0 | protocol version = 1
- 1-2 | same token as the PULL\_DATA packet to acknowledge
- 3 | PULL\_ACK identifier 0x04

### ### 5.4. PULL\_RESP packet ###

That packet type is used by the server to send RF packets and associated metadata that will have to be emitted by the gateway.

Bytes | Function

:-----:|-----

- 0 | protocol version = 1
- 1-2 | unused bytes
- 3 | PULL\_RESP identifier 0x03
- 4-end | JSON object, starting with {, ending with }, see section 6

## 6. Downstream JSON data structure

-----

The root object must contain an object named "txpk":

```
``` json
{
  "txpk": {...}
}
```
```

That object contain a RF packet to be emitted and associated metadata with the following fields:

Name | Type | Function

:----:|:-----:|-----

imme | bool | Send packet immediately (will ignore tmst & time)  
tmst | number | Send packet on a certain timestamp value (will ignore time)  
time | string | Send packet at a certain time (GPS synchronization required)  
freq | number | TX central frequency in MHz (unsigned float, Hz precision)  
rfch | number | Concentrator "RF chain" used for TX (unsigned integer)  
powe | number | TX output power in dBm (unsigned integer, dBm precision)  
modu | string | Modulation identifier "LORA" or "FSK"  
datr | string | LoRa datarate identifier (eg. SF12BW500)  
datr | number | FSK datarate (unsigned, in bits per second)  
codr | string | LoRa ECC coding rate identifier  
fdev | number | FSK frequency deviation (unsigned integer, in Hz)  
ipol | bool | Lora modulation polarization inversion  
prea | number | RF preamble size (unsigned integer)  
size | number | RF packet payload size in bytes (unsigned integer)  
data | string | Base64 encoded RF packet payload, padding optional  
ncrc | bool | If true, disable the CRC of the physical layer (optional)

Most fields are optional.

If a field is omitted, default parameters will be used.

Examples (white-spaces, indentation and newlines added for readability):

```
``` json
{"txpk":{
  "imme":true,
  "freq":864.123456,
  "rfch":0,
  "powe":14,
  "modu":"LORA",
  "datr":"SF11BW125",
  "codr":"4/6",
  "ipol":false,
  "size":32,
  "data":"H3P3N2i9qc4yt7rK7ldqoeCVJGBybzPY5h1Dd7P7p8v"
}}
```

```
...  
  
`` json  
{ "txpk": {  
    "imme": true,  
    "freq": 861.3,  
    "rfch": 0,  
    "powe": 12,  
    "modu": "FSK",  
    "datr": 50000,  
    "fdev": 3000,  
    "size": 32,  
    "data": "H3P3N2i9qc4yt7rK7ldqoeCVJGBybzPY5h1Dd7P7p8v"  
  }  
}  
...
```

7. Revisions

v1.2

- * Added value of FSK bitrate for upstream.
- * Added parameters for FSK bitrate and frequency deviation for downstream.

v1.1

- * Added syntax for status report JSON object on upstream.

v1.0

- * Initial version.

