Sigfox Network Emulator

User guide

November 2019

Contents

1. General information

1.1. System overview

Sigfox is the first and only network operator fully dedicated to low-throughput communication for connected objects. With an extremely cost effective and very low energy consuming out-of-the-box connectivity offer, Sigfox brings a revolution to the world of Internet of Things and M2M. The network, which already connects millions of objects, is being rolled out worldwide.
To facilitate the development of IOT applications, from the connected object itself to the application that processes the data transmitted by the objects, SIGFOX has developed a network emulator. This emulator is composed of an USB device called Sigfox SDR Dongle, associated to a software package that emulates the SIGFOX network. It is intended solely for product or software designers for use in a research and development setting.

The emulator runs in conducted mode with the object under development and is compatible with all European and FCC bands, allowing to develop applications without concern for network coverage issues. The software package is downloadable from a website and potential protocol evolutions or new features can be upgraded by downloading the last versions of the software package.

1.2. Technical characteristics

The Sigfox SDR dongle is designed for use connected to a USB 2.0 port of a laptop, solely in industrial and laboratory environments.
Refer to support.sigfox.com/products/SDR-dongle for the Sigfox SDR Dongle datasheet and notice.

RADIO CHARACTERISTICS	
Operating frequency	865-870 MHz, 902-928 MHz software configurable
Monitored spectrum	192kHz
Receiver Sensitivity	-64dBm @ 100bps
Transmit Power	14dBm
SOFTWARE PACKAGE FEATURES	
Version	2.0.0
System requirement	Linux Ubuntu / Windows 7 and 10
Supported browsers	Chrome, Firefox
Demodulation capacity	1 simultaneous demodulation signal
Supported protocol	Sigfox V1 protocol
Software update	Automatic check via internet connection
Language	English
Uplink Bit rate	100b/s and 600b/s software configurable

2. Requirements

2.1. Network emulator software

The software can be downloaded from support.sigfox.com/products/SDR-dongle and runs on:

- Linux platform: Ubuntu distribution 16.04 and later
- Windows platform: Windows 7 and 10 (64bits versions).

Choose the installation for the appropriate platform: Windows or Linux, and follow the instructions.

Note: For Linux users, make sure you are in the “plugdev” group.

The software will launch with the default browser. The supported browsers are Google Chrome and Mozilla Firefox.

The emulator must be configured in terms of central frequencies for uplink and downlink transactions and uplink bit rate according to the radio zone to emulate. The values for countries where the Sigfox network is already operated are provided by the software. For new countries or where the network is not yet operated, please contact Sigfox at support.sigfox.com to get the right values.

2.2. SDR dongle

The Sigfox SDR dongle is available for purchase online. The kit should include:

- 1 SDR dongle
- 1 SMA/SMA cable
- 1 SMA/UFL cable
- 1 attenuator (40dB)

The SDR dongle must be connected to an USB 2.0 High speed port on the machine on which the software was installed.

The object under development must be connected via one of the cable provided to the dongle’s SMA connector.

The emulator can receive signals whose conducted power at SMA connector is between -64dBm (@100b/s) and -8dBm. If the signal power is higher than -8dBm, the emulator will saturate and it won’t demodulate the signal. Thus, adequate attenuators must be inserted between the object and the emulator in order to ensure that the signals power is within the given range. Typically, the 40dB attenuator provided with the kit should be used for class 0U objects whatever the radio zone.

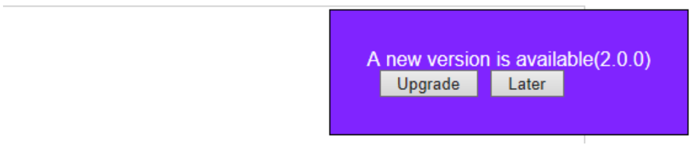
2.3. Device

The object under development must be built from a transceiver which is “Sigfox Ready” certified. **The transceiver must integrate the “emulation mode” command, to switch from private key to public key**.
A list of such transceivers is available on:
resources.sigfox.com/document/sigfox-ready-modules-compatibility.
This mode enables the authentication by the emulator of the messages transmitted by the transceiver and, conversely, the authentication by the transceiver of the messages transmitted by the emulator during downlink transactions.
The emulator can run in a mode where the authentication is disabled, however, as the device under test uses its private key, it will not be able to authenticate the downlink messages sent by the emulator during bidirectional transactions.

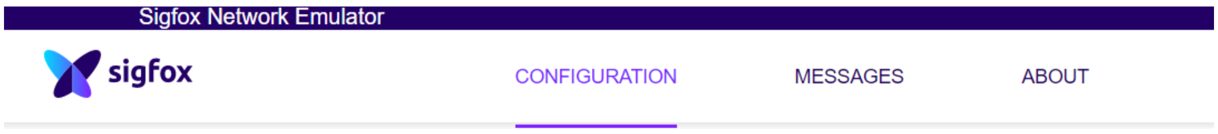
3. Launching the application

Once the software package is installed, connect the dongle to an USB port on the computer and launch the Sigfox Network Emulator. On first use, the drivers may not be detected straight away. If there is any issue, unplug and re-plug the dongle.

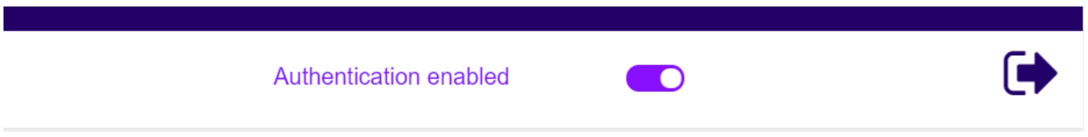
The application automatically checks for updates and will prompt if a new version is available.



The menu bar contains three items: a **Configuration** menu, a **Messages** view and **About** the application. The active menu is displayed in puple.



On the right hand side, there is the authentication toggle button and the exit button.



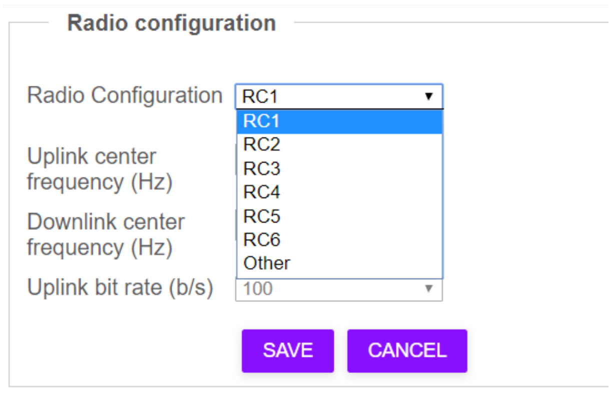
4. Emulator configuration

The Configuration menu contains four sub-menus allowing to configure:

- The radio zone where the object under development will operate (§4.1)
- The declared devices (§4.2)
- The callbacks triggered on messages reception or network events (§4.3)
- The connection in case of proxy configuration (§4.4)

4.1. Radio configuration page

The first displayed page when launching the emulator software is the radio configuration page. You can choose a radio configuration from the RC list.



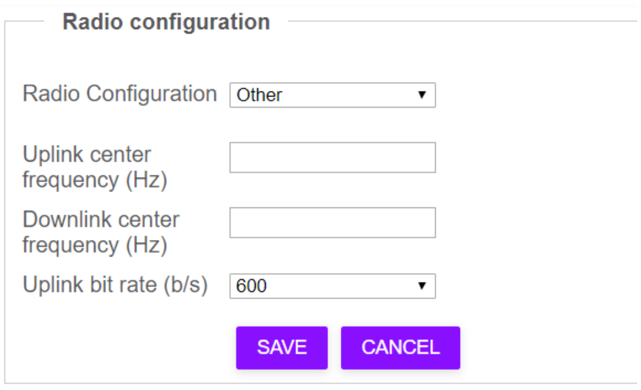
Radio configuration

Radio Configuration: RC1

Uplink center frequency (Hz):

Downlink center frequency (Hz):

Uplink bit rate (b/s): 100



Radio configuration

Radio Configuration: Other

Uplink center frequency (Hz):

Downlink center frequency (Hz):

Uplink bit rate (b/s): 600

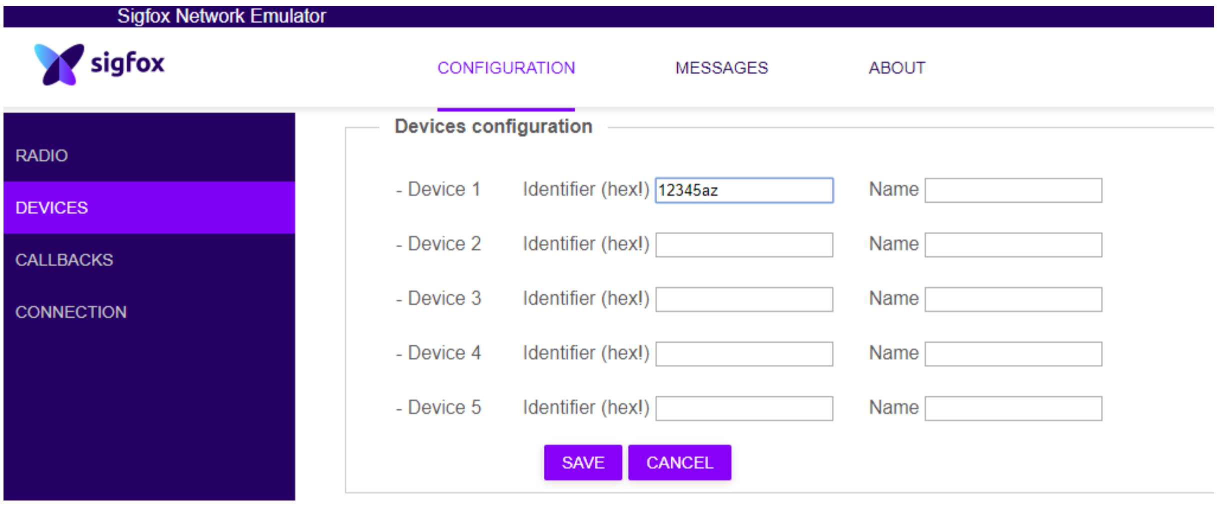
If you need another radio zone to develop your application, select “Other” and enter the desired values for the bit rate and Uplink and Downlink frequencies. Be aware that the transceiver of the object you are developing must be configured with the same values. Save your configuration before leaving the page.

For example, for RC7 (Russia), please input the following parameters:

	Radio Configuration	***Uplink center frequency (Hz)***	***Downlink center frequency (Hz)***	***Uplink bit rate (b/s)***
RC7	Other	868800000	869100000	100

4.2. Device registration page

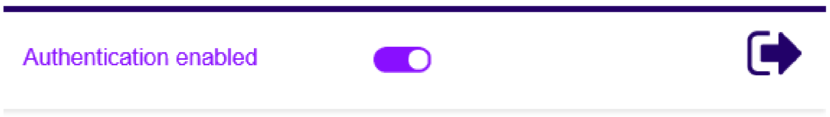
Up to 5 devices can be registered. The <Name> field is optional.
The device ID must be entered in hexadecimal, uppercase or lowercase. All messages coming from a non-identified device will be ignored.



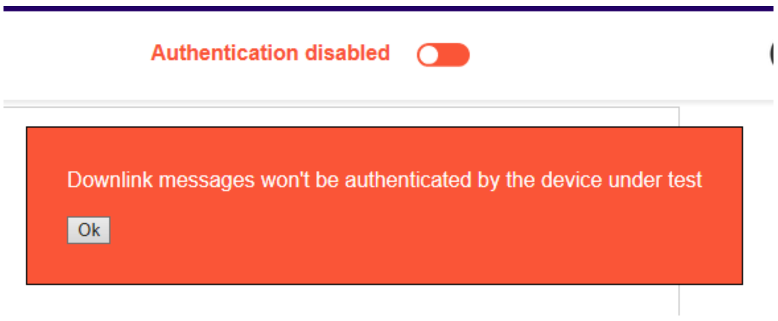
Devices configuration

- Device 1	Identifier (hex!) 12345az	Name	
- Device 2	Identifier (hex!)	Name	
- Device 3	Identifier (hex!)	Name	
- Device 4	Identifier (hex!)	Name	
- Device 5	Identifier (hex!)	Name	

To be able to authenticate the objects messages, the devices under test must run in “emulator mode”, that is public key usage. Refer to the user guide of your transceiver to set on this mode. It usually consists of an AT command or a SPI command.
Any message from a non-declared device or from a device not set in emulator mode shall be ignored by the emulator.
Nevertheless, you can deactivate the authentication with the toggle button at the right-hand side of the menu bar.



Warning: If authentication is disabled, messages from declared devices will be displayed on the network emulator and callbacks will be activated, but, in case of bidirectional transactions, the device under test will not be able to authenticate the downlink message sent by the emulator.



Downlink messages won't be authenticated by the device under test

4.3 Callbacks configuration

The emulator can automatically forward some events using the «callback» system.

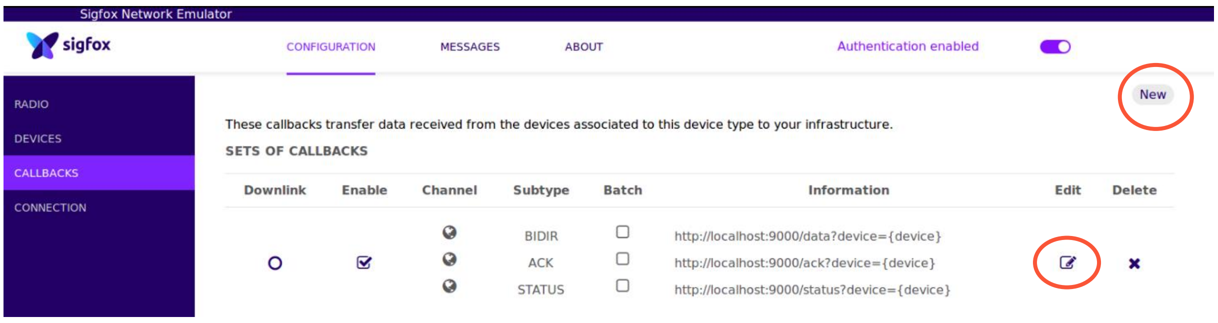
A callback is a custom http request containing the device(s) data, along with other variables, sent to a given server/platform.

On Sigfox Backend, callbacks are triggered when a new device message is received, when a location has been computed, or when a device communication loss has been detected for example.

The emulator provides 3 types of callbacks:

- Data: for messages events
- Service acknowledge: for confirmation of downlink message reception by the device
- Service status: for status of the device

You have a set of available variables for each type of callback.
These variables are replaced by their value when a callback is called.
When receiving a callback, the client system must return an HTTP 2xx code within 10 seconds. If the client system fails to process the callback during this time, an automatic retry will be scheduled 1 minute later.

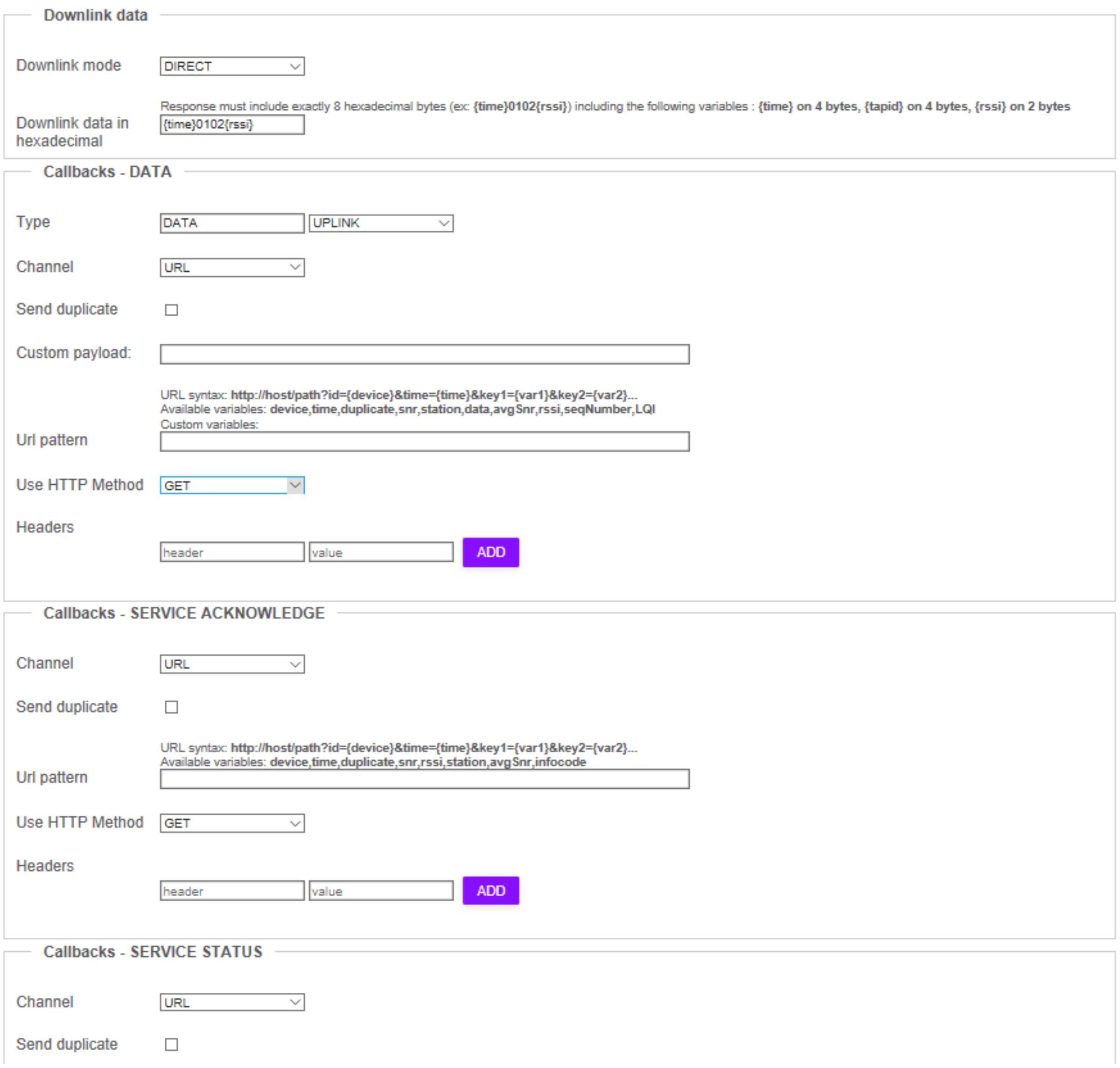


These callbacks transfer data received from the devices associated to this device type to your infrastructure.

SETS OF CALLBACKS

Downlink	Enable	Channel	Subtype	Batch	Information	Edit	Delete
○	✓	●	BIDIR	☐	http://localhost:9000/data?device={device}	✎	✖
		●	ACK	☐	http://localhost:9000/ack?device={device}		
		●	STATUS	☐	http://localhost:9000/status?device={device}		

Click on the “New” button or on the “Edit” icon to configure calbacks.



Downlink data

Downlink mode: DIRECT

Downlink data in hexadecimal: {time}0102{rssi}

Response must include exactly 8 hexadecimal bytes (ex: **{time}0102{rssi}**) including the following variables : **{time} on 4 bytes, {tapid} on 4 bytes, {rssi} on 2 bytes**

Callbacks - DATA

Type: DATA UPLINK

Channel: URL

Send duplicate: ☐

Custom payload:

URL syntax: **http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...**
Available variables: **device,time,duplicate,snr,station,data,avgSnr,rssi,seqNumber,LQI**
Custom variables:

Url pattern:

Use HTTP Method: GET

Headers: header value ADD

Callbacks - SERVICE ACKNOWLEDGE

Channel: URL

Send duplicate: ☐

URL syntax: **http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...**
Available variables: **device,time,duplicate,snr,rssi,station,avgSnr,infocode**

Url pattern:

Use HTTP Method: GET

Headers: header value ADD

Callbacks - SERVICE STATUS

Channel: URL

Send duplicate: ☐

The three callback types share a set of common variables:

- `time (int):` the event timestamp (in seconds since the Unix Epoch)
- `device (string):` device identifier (in hexadecimal – up to 8 characters <=> 4 bytes)
- `duplicate (bool):` «true» if the message is a duplicate one, meaning that the backend has already processed this message from a different base station, «false» otherwise. To receive duplicate messages, you must check the «send duplicate» box in the callback configuration page. This variable will always be set to false by the emulator
- `snr (float):` the signal to noise ratio (in dB – Float value with two maximum fraction digits)
- `rssi (float):` the RSSI (in dBm – Float value with two maximum fraction digits). If there is no data to be returned, then the value is null.

- `avgSnr (float):` the average signal to noise ratio computed from the last 25 messages (in dB – Float value with two maximum fraction digits) or «N/A». The device must have sent at least 15 messages.
- `station (string):` the base station identifier (in hexadecimal – 4 characters <=> 2 bytes). This variable will always be set to 0x0000 by the emulator
- `data (string):` the user data (in hexadecimal)
- `seqNumber (int):` the sequence number of the message if available

The **Channel** of the callback can be selected to choose if the callback has to be sent through an **URL** (in development phase), or **batch URL** (for production).
In the simple mode (URL), each message is directly forwarded in a single HTTP request. You can use HTTP GET, POST or PUT methods, although POST method is recommended.

GET Method:
The variables are automatically replaced in query string.

```
GET http://hostname/path?id={device}&time={time}&key1={data}&key2={signal}
```

POST/PUT methods:
POST or PUT method allows you to set the content-type and the body of your request You can choose the content-type between 3:

- application/x-www-form-urlencoded

This is the default content type for POST or PUT request. When using this content-type, you can set body in form encoded format :

```
    device={device}&data={data}&time={time}
```

Note that if you put some variables in the query part of the URL, these parameters will be appended to the body, whether it is empty or not. E.g.:

```
    POST
http://hostname/path?id={device}&time={time}&key1={data}&key2={signal}

    with a body template:
    device={device}&data={data}&time={time}

    will result in the following body:


id={device}&time={time}&key1={data}&key2={signal}&device={device}&data=
{data}&time={time}
```

- application/json

You can set a JSON object in the body:

```
    {
        "device" : "{device}",
        "data" : "{data}",
        "time" : {time}
    }
```
Please refer to JSON specification for more details

- text/plain

This content type is a free.

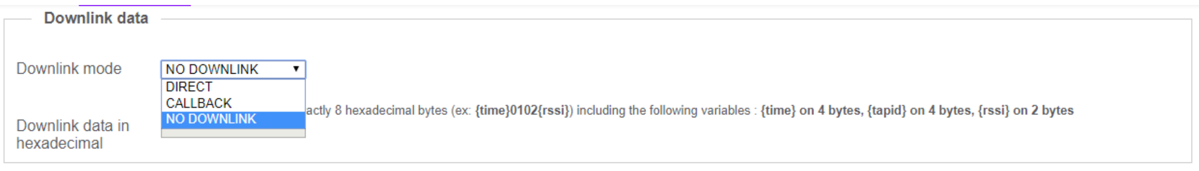
```
    device => {device}
    data => {data}
    time => {time}
```

Batch:
In the batch mode, messages are gathered together per callbacks, each line representing a message, then is sent in batch using a single HTTP request every second. In production mode, this avoids a possible peak load that your server can not handle. As the payload contains multiple messages, only HTTP POST method are supported. When selecting the batch URL mode on the emulator, one single event will be sent in each batch (as the emulator can treat only one message at a time).

```
POST http://hostname/path?batch={batch} where
batch={device};{data};{signal};...
```

4.3.1. Downlink modes

The downlink mode can be chosen in the Downlink data frame.


Downlink data

Downlink mode: NO DOWNLINK ▾ (DIRECT / CALLBACK / NO DOWNLINK)

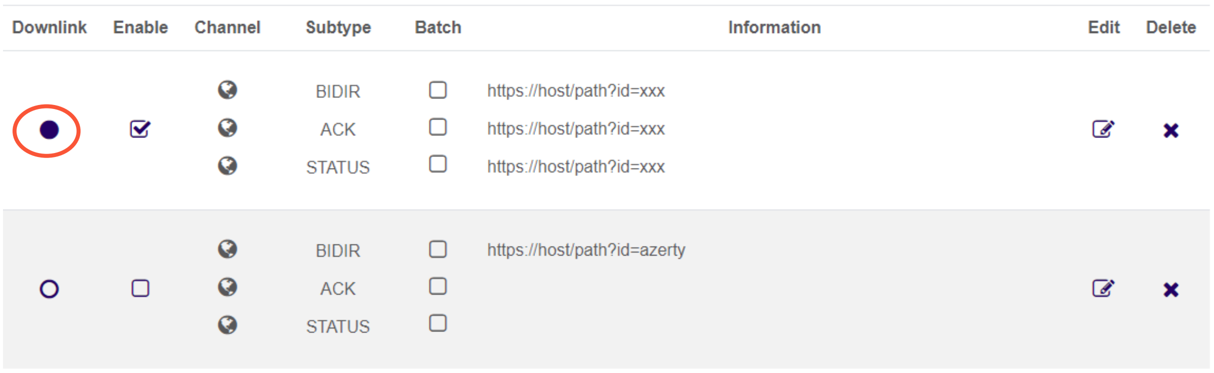
Downlink data in hexadecimal: actly 8 hexadecimal bytes (ex: **{time}0102{rssi}**) including the following variables : **{time} on 4 bytes, {tapid} on 4 bytes, {rssi} on 2 bytes**

Choose the NO DOWNLINK mode if your application is uplink only. The Data callback type will be set to **uplink.**

In the DIRECT downlink mode, the same 8 bytes data defined in the <downlink data > field will be sent to any declared device requiring a downlink transmission. In this mode, the Data callback type is set to **uplink**. This mode is mainly a support to the development of the application embedded on the object as it allows to develop it without having to provide any callback.

In the CALLBACK Downlink mode, the Data callback is set to **bidirectional** mode.

Note that by default the downlink settings on the configuration page is not selected. You need to activate it by clicking on it.



Downlink	Enable	Channel	Subtype	Batch	Information	Edit	Delete
●	✓	🌐	BIDIR	☐	https://host/path?id=xxx	✎	✖
		🌐	ACK	☐	https://host/path?id=xxx		
		🌐	STATUS	☐	https://host/path?id=xxx		
○	☐	🌐	BIDIR	☐	https://host/path?id=azerty	✎	✖
		🌐	ACK	☐			
		🌐	STATUS	☐			

If you configure a callback DATA or SERVICE, you'll need to implement a RESTful, web-facing service.

4.3.2. Data Callback

This callback type defines the reception of a device message. Common variable available for each subtype are:

- UPLINK: This subtype does not define any additional variable.
- BIDIR: Bi directional so uplink and downlink

`ack (bool):` true if this message needs to be acknowledged, false else.

The client can decide not to send any answer to the device. There are 2 ways to do so:

- respond to the callback with the HTTP NO_CONTENT code (204).
- respond with a json data containing the noData field. ex:

```
{ "0CB3" :
    {
        "noData" : true
    }
}
```

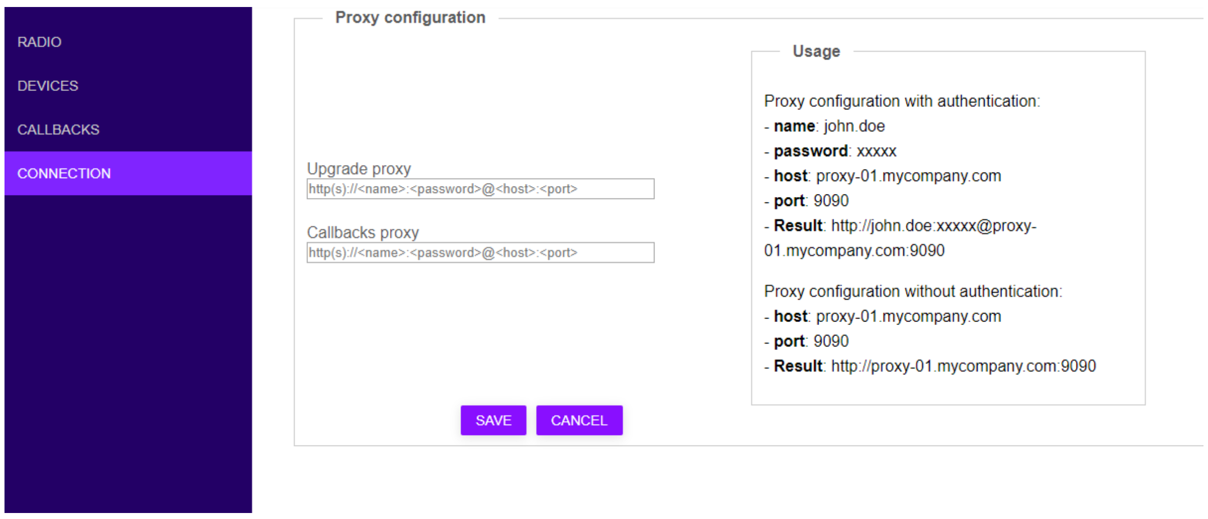
4.3.3. Service Callback

There are two types of Service Callbacks: ACKNOWLEDGE (ACK) and STATUS

- `infoCode (int):` this is the status code of the downlink:
  - `0 (ACKED)` the station emitted the answer,
  - `1 (NO_ANSWER)` the client did not give any answer,
  - `2 (INVALID_PAYLOAD)` the data to send to the device is invalid,
  - `3 (OVERRUN_ERROR)` the device exceeded its daily downlink quota, so it was blocked because of a lower priority than transmissions for devices that did not exceed their quota,
  - `4 (NETWORK_ERROR)` it was not possible to transmit the answer,
  - `5 (PENDING)` not technically a code that is sent in the callback because it is a transient state before the answer is sent,
  - `6 (NO_DOWNLINK_CONTRACT)` the device asked for an answer but its BSS order does not allow downlink,
  - `7 (TOO_MANY_REQUESTS)` the device asked for an answer before the expiration of the listening time,
  - `8 (INVALID_CONFIGURATION)` the device type is configured to get data by callback, but no BIDIR callback was defined
- `infoMessage (string):` a message associated to the code. Not implemented on the SNE.
- `downlinkAck (bool):` true if the station acknowledged the transmission, false else.
- `downlinkOverusage (bool):` true if the device exceeded its daily quota, false else.

4.4. Proxy configuration

You may need to configure the “Upgrade proxy” and “Callbacks proxy” environment variables used by your proxy servers. The two following examples show you how to populate those variables with or without authentication mechanism. Save your configuration before leaving the page.



Upgrade proxy
http(s)://<name>:<password>@<host>:<port>

Callbacks proxy
http(s)://<name>:<password>@<host>:<port>

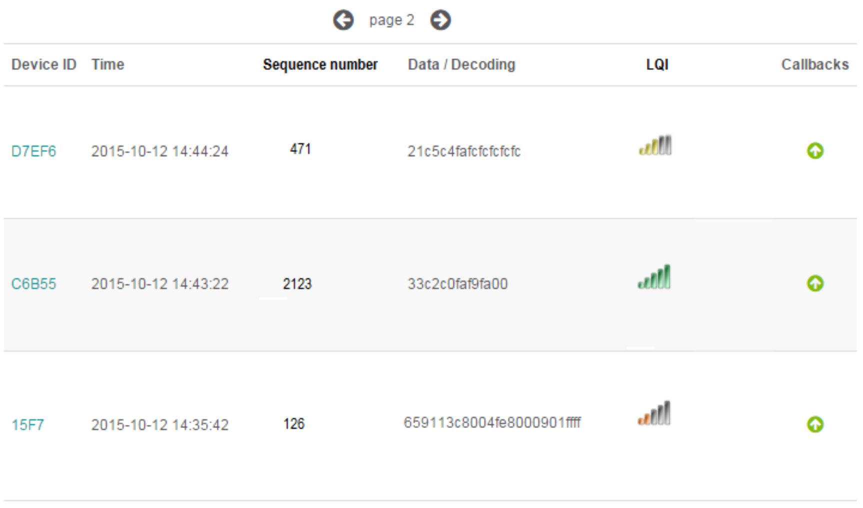
Usage

Proxy configuration with authentication:
- **name**: john.doe
- **password**: xxxxx
- **host**: proxy-01.mycompany.com
- **port**: 9090
- **Result**: http://john.doe:xxxxx@proxy-01.mycompany.com:9090

Proxy configuration without authentication:
- **host**: proxy-01.mycompany.com
- **port**: 9090
- **Result**: http://proxy-01.mycompany.com:9090

5. Messages display

Each message received from a declared device is displayed on the Messages page, listed from newest to oldest. The maximum capacity is 100 messages, oldest messages being deleted when the limit of 100 is reached.



page 2

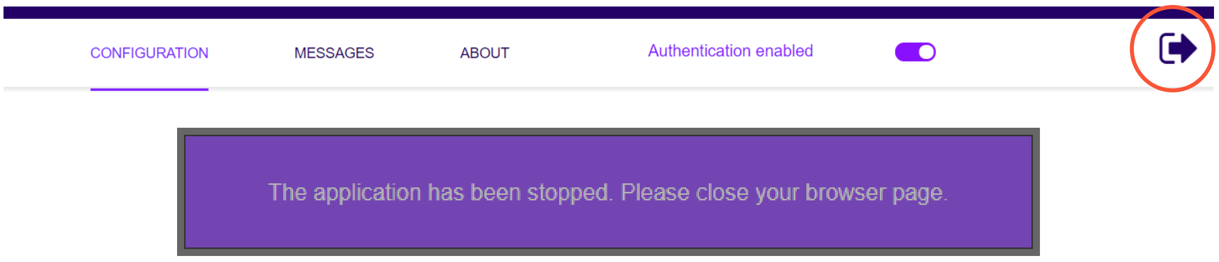
Device ID	Time	Sequence number	Data / Decoding	LQI	Callbacks
D7EF6	2015-10-12 14:44:24	471	21c5c4fafcfcfcfc		●
C6B55	2015-10-12 14:43:22	2123	33c2c0faf9fa00		●
15F7	2015-10-12 14:35:42	126	659113c8004fe8000901ffff		●

For each message, the information displayed is:

- The device identifier
- Timestamp of reception
- The sequence number: should be incremented by one for every new message from a given device if no message has been lost
- The payload in hexadecimal format
- The Link Quality Indicator
- The Callback status:
  - Upwards arrow for an uplink transaction:
    - grey background if no callback set
    - orange background between callback activation and receiving return code
    - red background if the callback returned an Error code
    - green background if the callback returned OK
  - Upwards and downwards arrows side by side for a bidirectional transaction:
    - same colour code as above for the upwards arrow
    - downward arrow on orange background: transitional state, waiting for the transmission of the response
    - red background: failure of transmission of response, the reason for failure is specified in the variable Infocode of the service callback ACKNOWLEDGE.
    - green background: response transmitted by the emulator

6. Closing the application

To close the application, click on the exit button on the right-hand side of the menu bar. This will stop the application server and deactivate the web page of the emulator.



The application has been stopped. Please close your browser page.

If you simply close the web page of the emulator without stopping, the application server will go on running and the emulator will go on receiving messages and triggering callbacks.

7. Uninstalling / reinstalling

To uninstall Sigfox Network Emulator, go the “Add/Remove program” view on the Windows or Ubuntu Software Center on Linux, and select Uninstall.

Note: the folder Snek in the user folder won’t be deleted. If you need to re-install the application, your configurations (devices, callbacks…) will be saved.
If you wish to uninstall fully, delete manually the folder after removing the application.

Once the application embedded on the object has been developed and tested with the emulator, simply exit the emulator mode to operate on the real Sigfox network. Please note that the operation of IoT Application (connected object only) on the real Sigfox network is subject to its prior certification by Sigfox.

8. Troubleshooting

Sigfox Network Emulator is a tool to validate end to end message process, it will decode the first frame received and may not process the following ones.
To test the radio aspect please use the Radio Signal Analyzer software.

You will find information about events that occurred on the system in the snek.log file located in your user’s directory in the “Snek” folder, especially about reception of messages from devices that are not declared or reception of non-authenticated messages (e.g. from a device that does not use public key).

If you experience some issues with your emulator, try the following solutions.

Nothing happens when you launch the emulator application.

- Check that your platform is compliant with the application. See §2. Upgrade your operating system if necessary.
- If you work on Linux platform, please check that you are in the plugdev group

Error message: Sigfox Network Emulator cannot be launched

- Connect the emulator USB dongle to your platform prior launching the application
- Unplug and replug the dongle
- Wait for the OS to detect the drivers prior launching the software

Windows prevents the download and/or the run of the snek.exe file

- Select yes to download
- Right click on the file and select run anyway

You can't see any message on the messages page

- Check that the device connected to the emulator is declared on the Configuration/Devices page. If you are not sure of the device ID, you can check in the snek.log file if a message has been received~~.~~
- If the message is not notified in the snek.log file, check that the radio configuration of the emulator is the same as the one of your device.
- **Check that you turned your device into emulation mode (public key).** Please refer to the user guide of your device. You can check it in the snek.log file or disable authentication with the button at the right top of the screen and send another message with your device.

The device under test does not receive downlink messages

Check that you turned your device into emulation mode (public key). Please refer to the user guide of your device.

Possible error on the callbacks page:

- Invalid response data: empty or incorrect "downlink data" field in direct mode.
- Wrong values: one or more variable in the callbacks URL are incorrect

Message "Saturation Detected":

Make sure to use the attenuator provided. The device should not be in direct contact with the dongle.

If the issue cannot be solved with the steps above, please check the support web site: https://support.sigfox.com/

9. Disclaimer

The Sigfox Network Emulator is intended solely for product or software designers for use in a research and development setting. Sigfox disclaims all express and implied warranties, including, without limitation, the warranties of merchantability, satisfactory quality, fitness for a particular purpose and non-infringement.