NOCXX
Data API

Changes history

Version	Date	Description
1.0	**2017.11.08**	**First edition issued**
1.1	**2017.11.14**	**Add user response**

Table of Contents

1. Introduction

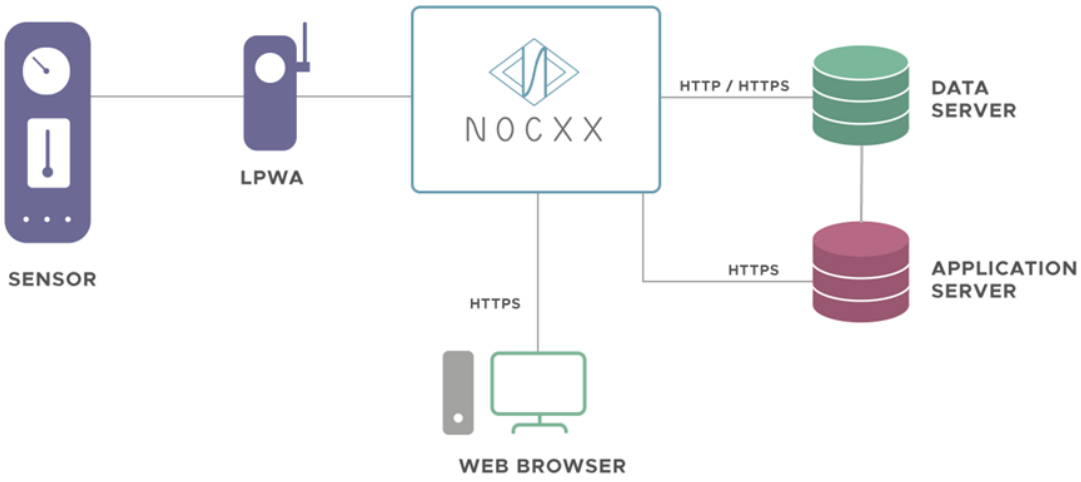
1.1. Aim of this document

This document defines the data HTTP/REST API of the NOCXX platform.
Two data APIs are defined :

- downlink data : from the application server (AS) to NOCXX (NS)
- uplink dtata : from the NOCXX(NS) to the data server (DS)

2. Network architecture

2.1. Global view of the network architecture



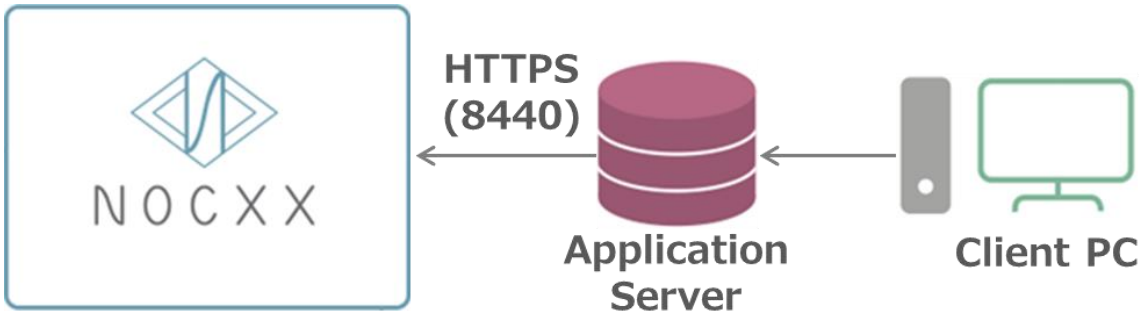
2.2. Description of the Network components

The sensors in LoRaWAN network exchange data with a Gateway using LoRa radio protocol.

- The sensors in LoRaWAN network exchange data with a Gateway using LoRa radio protocol.
- The LoRa Gateway makes the link between the LoRaWAN network and the Internet.
- The NOCXX (Network Server) manages the LoRaWAN protocol. The data from LoRaWAN network are sent to the Data Server. The users messages (example: actuator command) come to the Network Server through the Application Server.
- The Data Server saves the data from the LoRaWAN network. Those data can be accessed by the user through the Application Server.
- The Application Server provides network graphic interfaces to users.

3. Downlink API of the Network Server

This API allows to send downlink data to a sensor. It may be used by an application server.



3.1. Send downlink data frame

The data must be formatted with the application protocol used by the sensor.

This protocol depends on the sensor manufacturer.

URL	**https://users.nocxx.com/v1/frame/DEVADDR** or **https://users.nocxx.com/v1/frame/DEVEUI**
HEADER	"Content-Type : application/json"
	"Accept : application/json"
	basic http authentication with login/pwd
TYPE	POST
PORT	8440
JSON PARAMETER	**{** **"frame":"DATA",** **"port":PORT** **}**
RETURN	TODO
DEVADDR	DevAddr specified for ABP or OTAA sensors. It is formatted as a hexadecimal string, with a length of 4 bytes.
DEVEUI	DevEUI specified for OTAA sensors It is formatted as a hexadecimal string, with a length of 8 bytes.
DATA	the data frame to send to the sensor. This data will be encrypted by the network server with the LoRaWAN key. It is formatted as a hexadecimal string.
PORT	the LoRaWAN port used to send DL frame to the sensor. This field is optional, default used value is 1. It is formatted as an integer.

Response content:

STATUS	String “SUCCESS” or “FAILED”
message	Additional message (string)

Example of a curl command to send data “0123abcd” to a OTAA sensor “DEVEUI=AC000001AC000001 & DEVADDR=AC000001” on port 6:

```
curl -i -X POST -H "Accept:application/json" -H "Content-type: application/json" -u "user:userPwd"-d '{"frame":"0123abcd",”port”:6}' https://users.nocxx.com:8440/v1/frame/AC000001AC000001
```

```
curl -i -X POST -H "Accept:application/json" -H "Content-type: application/json" -u "user:userPwd"-d '{"frame":"0123abcd",”port”:6}' https://users.nocxx.com:8440/v1/frame/AC000001
```

3.2. Send a frame with confirmation message to user

If the user want to know the status of his downlink message, he can use the JSON optional field, called “response”.

In this object we can have 3 optional fields:

- **frameId** (interger): that user provides to identify the frame
- **ind** (json object): describes how to send response which indicates that message has been sent to sensor
- **ack** (json object): describes how to send response which indicates that an acknowledge has been received from sensor (confirmed messages only)

JSON
PARAMETER
	{
	“frame”:”DATA”,
	“port”:PORT,
	"response":{
	"frameId":fid,
	"ind":{
	"url":url_ind,
	"port":port_ind,
	"login":login_ind,
	"pwd":pwd_ind,
	"token":token_ind
	},
	"ack":{
	"url":url_ack,
	"port":port_ack,
	"login":login_ack,
	"pwd":pwd_ack,
	"token":token_ack
	}
	}
	}

Parameters:

fid	“Authorization : xxxxxxxxxxxxxxxxxxxx”
url_x	“keyId : xxxxxxxxxxxxxxxxxxxxxx”
port_x	[optional] response port (integer)
login_x (string)	[optional] response login (string)

Example

```
curl -k -i -X POST -H "Accept: application/json" -H "Content-type:
application/json" -u "usrlogin:usrP@55" -d '{
  "frame":"100600010002",
  "port":1,
  "response":{
    "frameId":5672,
    "ind":{
      "url":"https://resp.company.com/ind",
      "port":8463,
      "login":"resp",
      "pwd":"tstPwd"
    },
    "ack":{
      "url":"https://resp.company.com/ack",
      "port":8463,
      "login":"resp",
      "pwd":"tstPwd"
    }
  }
}' https://userns.nocxx.com:8440/v1/frame/AC000001/
```

3.3. Responses to user

If response parameter has been set in data downlink, the user will receive message according to his configuration.

JSON response	**{** **"DevAddr": devAddr,** **"DevEUI": devEUI,** **"respType": type,** **"frameId": frameId** **}**

Response content:

devAddr	Device address (string)
devEUI	Device EUI (string)
type	Message type (*ind* or *ack*), indicate the type of response if the same parameters are used for message sent and message acknowledge

	(string)
frameId	The frame Identifier provided by user in request. If user does not provide any frameId, value will be 0 (interger)

4. Uplink API of the Data Server

The NOCXX can push deciphered sensor frames to a client data server.
One or more data server can be be added in the NOCXX configuration, and each sensor indicates which data server is used.

4.1. Send uplink data frame

The data server should know the application protocol used by the sensor to extract the data values. This protocol depends on the sensor manufacturer.
A data server configuration can be customized with optional fields.

URL	**https://@DS_IP/{CLIENT_ID}/{DEVADDR}/{DEVEUI}** or **http://@DS_IP/{CLIENT_ID}/{DEVADDR}/{DEVEUI}** **{CLIENT_ID}, {DEVADDR}** and **{DEVEUI}** are optional fields that will be replaced by their values on NS post to data server (see example below)
HEADER	“Content-Type : application/json”
	“Accept : application/json”
Basic AUTH in the header (optional)	“Authorization : xxxxxxxxxxxxxxxxxxxxx”
Token in the header	“keyId : xxxxxxxxxxxxxxxxxxxxxxx”
TYPE	POST
PORT	xxxx

JSON PARAMETER	**{** **"frame":"DATA",** **"timestamp":TST,** **"gatewayID":GW_ID,** **"clientID":CLIENT_ID,** **"DevAddr":DEVADDR,** **"DevEUI":DEVEUI,** **"rxpk":RXPK** **}**
RETURN	Success with code « HTTP 200 OK »

Parameters:

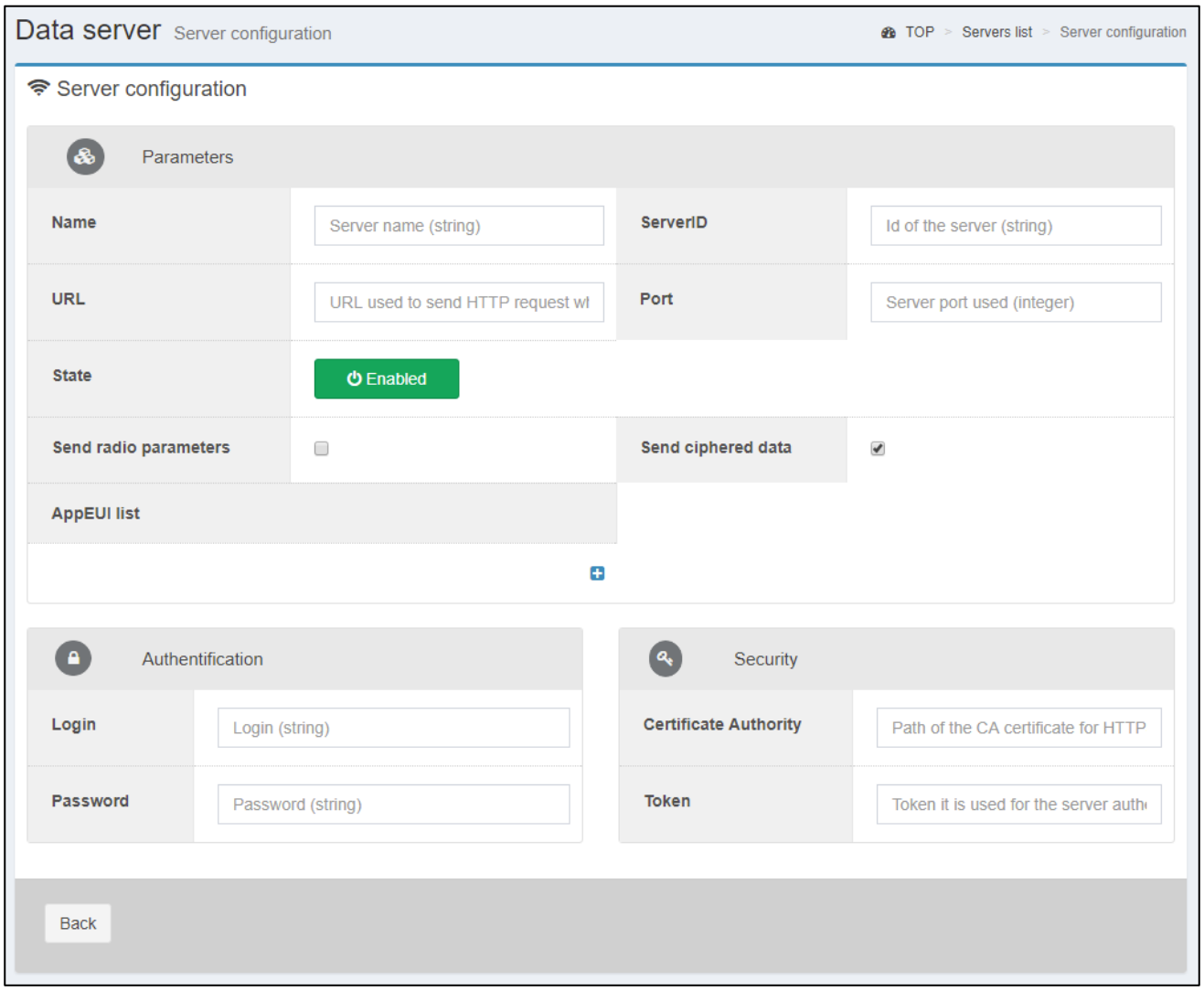
@DS_IP	IP address of the data serveur. Security protocol with https is available.
CLIENT_ID	Allow to identify a set of sensors in NOCXX. This field is always set in the json, and optional in the URL.
DEVADDR	DevAddr of ABP and OTAA sensors. It is formatted as a hexadecimal string, with a length of 4 bytes. This field is always set in the json, and optional in the URL.
DEVEUI	DevEUI specified only for OTAA sensors. It is formatted as a hexadecimal string, with a length of 8 bytes. This field is always set in the json, and optional in the URL.
DATA	The data frame sent by the sensor. This data is deciphered by the network server. It is formatted as a hexadecimal string.
TST	The time in seconds from the 1st January 1970. This value is set by the network server when it receives the uplink frame from a sensor.
GW_ID	The identifier of the gateway which received the frame
RXPK	Contains radio parameters received from the gateway. This field is optional, it is present only if the server is configured to send gateway's information. The section $4.2 describes the server configuration, where this option can be selected. The section $4.4 describes the content RXPK.

Example of a curl command to send data “0123abcd” from a OTAA sensor “DEVEUI=AC000001AC000001, DEVADDR=AC000001, CLIENT_ID=my_client, Data Server URL : https://dataserver.com/{CLIENT_ID}/{DEVEUI}”

```
curl -i -k -X POST -H "Accept: application/json" -H "Content-type:application/json" -u "user:userPwd" –d '{"frame":"0123abcd","timestamp":1490774095,"DevAddr":"AC000001","DevEUI":"AC000001AC000001","gatewayID":"AA555A0000000111"}' https://dataserver.com/my_client/AC000001AC000001
```

4.2. Configuration in the WEB interface

The web interface allows to configure one or more data server. Then, uplink data of a sensor can be forwarded to this data server.



Name: the name of your data server

ServerID: short name of the data server, without specials character.

ClientID:To select one of the clientID defined on the NS. This choice is available only for admin user.

URL: URL as described in $4.1

Port: port used by the data server

Login: used if a HTTP basic authentication is requested by data server (optional)

Password: used if a HTTP basic authentication is requested by data server (optional)

Certificate authority: used when the certificate of the server is not signed by a trust authority.

Token: a token can be requested by the data server, it will be added in the URL.
Send radio parameters : To send with the frame data the radio parameters and the gatewayID
Send ciphered data : To send the ciphered data, then DS will deciphered it
Server state : to enable or disable usage of this Data Server

4.3. Data bufferization in the network server

If the data server is not reachable when the network server tries to send it data, this data will be memorized for a time. Then the network server will try to resend this data later.
By default, the data is memorized for 7 days in the NS, and it will try to resend them each 15 minutes.

Note : The feature is only available if data server is not reachable. It the DS reply with an HTTP error, the data is discarded.

4.4. Radio parameters from base station

Radio parameters are received from the gateway. Depending on the type of gateway and its version, the content of radio parameters can be different.

Example:Kerlink LoRa IoT Station


```
"gatewayID":"AA555A0000000000",
"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"
}
```

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