Live Objects LPWA - complete guide

Orange Live Objects team

1.14.9,

Table of Contents

Chapter 1. Introduction

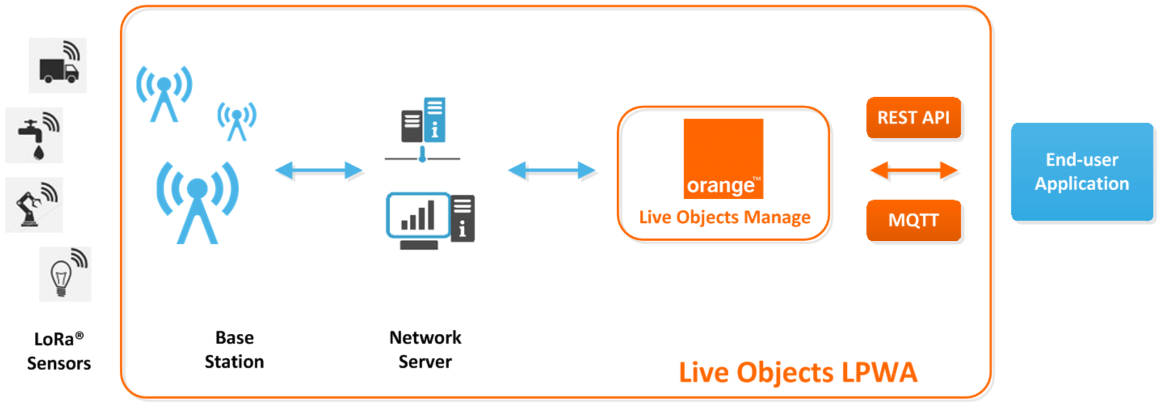
This document is a guide for Orange Live Objects LPWA solution, with the following chapters:

- Overview,
- Getting started,
- REST Api,
- MQTT interface,

Chapter 2. Overview

2.1. Live Objects LPWA Architecture

The Live Objects LPWA architecture is the following



2.2. What is Live Objects ?

Live Objects is a SaaS providing a set of tools for IoT / M2M solution integrators that want to interconnect **devices** or **connected « things »** and **business applications**.

The main features provided are:

- **connectivity interfaces** to collect data, send command or notification from/to IoT/M2M devices,
- **device management** (supervision, configuration, ressources, firmware, etc.),
- **message routing** between devices and business applications,
- and **data storage** with **advanced search** features.

2.3. REST / HTTPS interface

The REST / HTTPS interface provides the following features : The public interfaces are secured with API keys.

Device management :

- register device : register a new device in Live Objects LPWA
- list devices : get the list of your registered devices, and their information
- device information : get detailed information of one device

Command management :

- register command : register a command to be sent to a registered device

- list commands : get the list of sent commands

Message management :

- list messages : get the list of the received messages from the registered devices

2.4. Security

2.4.1. Web portal Users management

An account can be associated to various users. A user is also associated to a list of roles. These users can connect to the Live Objects LPWA web portal.

2.4.2. Roles

Administrator

An **Administrator** of the Tenant has full access to the fleet and can manage web portal users

User

A **User** has limited rights on the account and cannot manage other users.

Table 1. Administrator vs User rights

Functionality	Administrator	User
Access data messages	Yes	Yes
Access data commands	Yes	Yes
Send command	Yes	Yes
Access devices information	Yes	Yes
Register Devices	Yes	No
Update Devices	Yes	No
Delete Devices	Yes	No
Manage Users	Yes	No

2.4.3. API keys

API keys are used to control the applications accessing to the Live Objects LPWA platform.

An account is associated to a Master Key. Additional API Keys can be generated with specific roles. You can easily configure your API key by following the Getting started guide.

It is the first step required to use the REST and / or MQTT interfaces with your business applications.

Chapter 3. Getting started

This chapter is a step-by-step manual for new users of Live Objects LPWA giving instructions covering the basic use cases of the product.

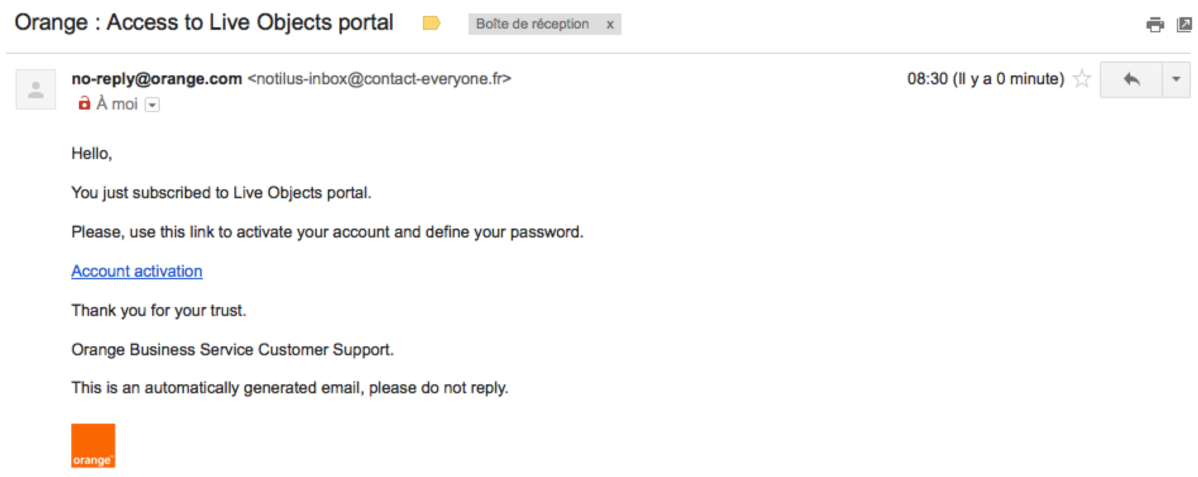
3.1. Account creation

In order to use Live Objects LPWA, you need to have a dedicated account on the service.

Please contact Orange to request an account, for this you must provide a valid email address and billing information.

Once the account is created, you will receive an email with a link to activate your account:

account activation email



Orange : Access to Live Objects portal

no-reply@orange.com <notilus-inbox@contact-everyone.fr>
À moi

08:30 (Il y a 0 minute)

Hello,

You just subscribed to Live Objects portal.

Please, use this link to activate your account and define your password.

Account activation

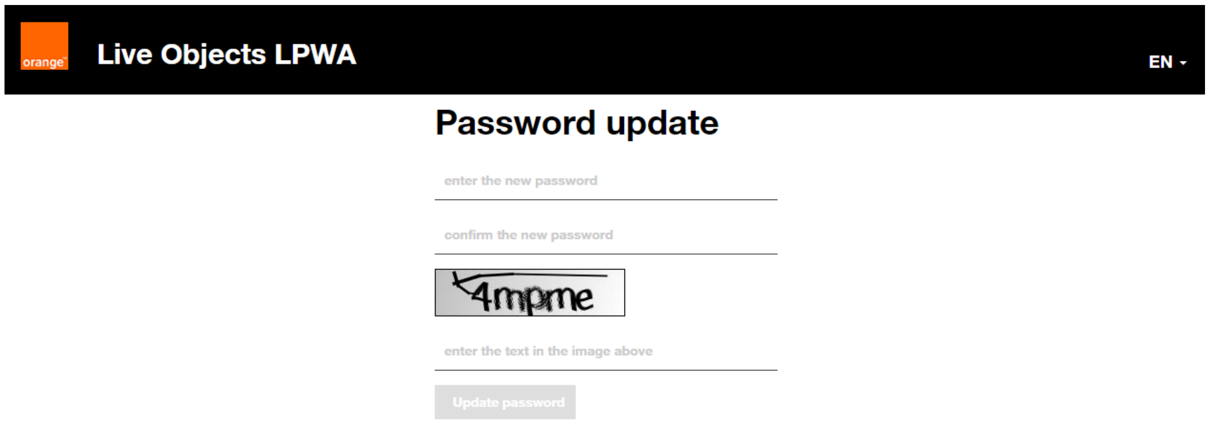
Thank you for your trust.

Orange Business Service Customer Support.

This is an automatically generated email, please do not reply.

by clicking on "*Activate Account*" link in this email, you are redirected to a web page where you can choose the password of your user account:

account activation web page



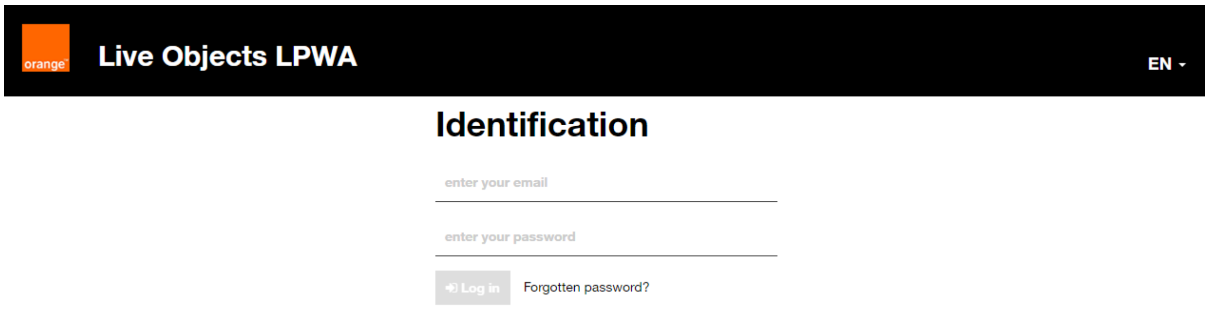
Once you entered twice your password and a correct "captcha", and clicked on "update password", you are redirected to the Live Objects LPWA home page where you can now log into your newly created user account.

3.2. Log-in

To log on to Live Objects LPWA web portal:

- connects to lpwa.liveobjects.orange-business.com using your web-browser:

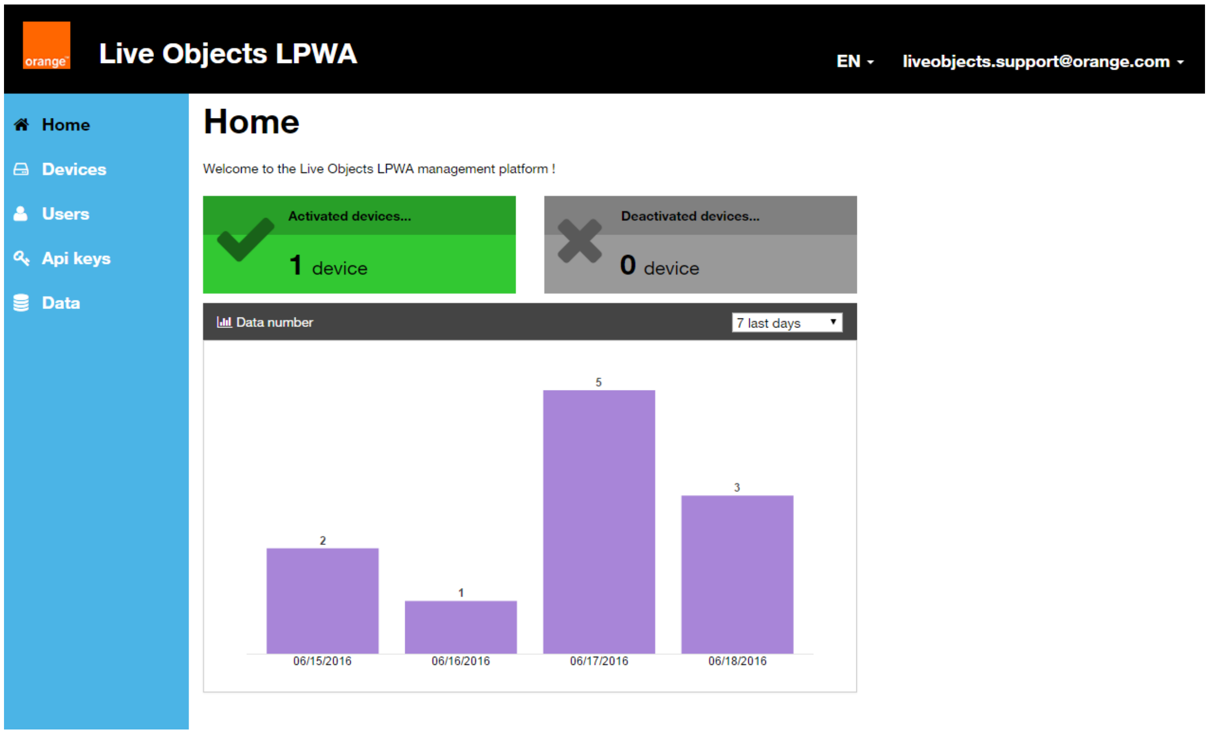
Live Objects LPWA public landing page



- Fill the “Log in” form with your credentials:
  - your email address,
  - the password configured during the account activation phase,
- then click on “Log in” button.

If the credentials are correct, a success message is displayed and you are redirected to the home page:

Home page



Home

Welcome to the Live Objects LPWA management platform !

Activated devices... 1 device

Deactivated devices... 0 device

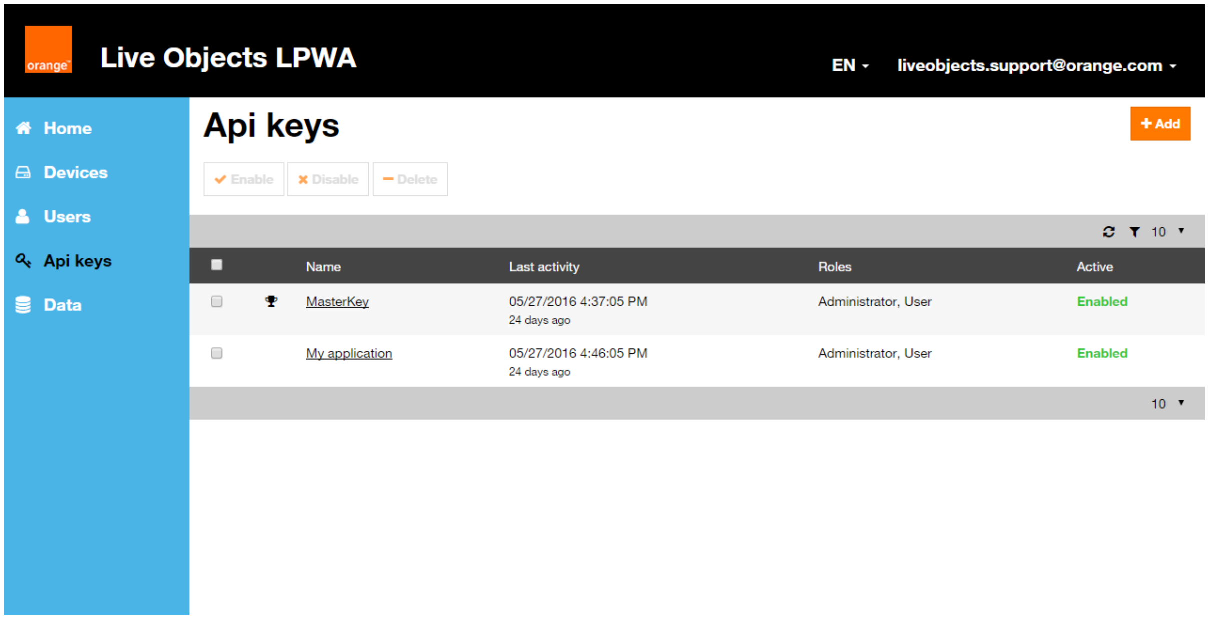
Data number — 7 last days

06/15/2016	06/16/2016	06/17/2016	06/18/2016
2	1	5	3

3.3. Add an API key

To configure a new API key, go to "Api keys" :

Api Keys page

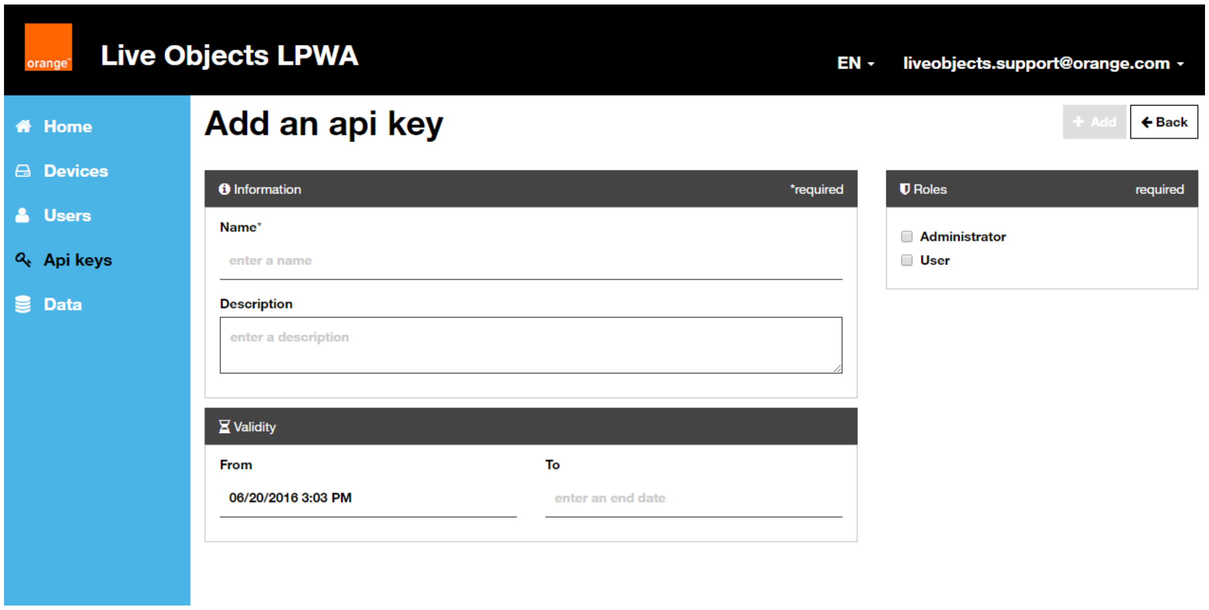


Api keys

	Name	Last activity	Roles	Active
	MasterKey	05/27/2016 4:37:05 PM 24 days ago	Administrator, User	Enabled
	My application	05/27/2016 4:46:05 PM 24 days ago	Administrator, User	Enabled

Click Add and complete the following fields :

Api Key form



Add an api key

Information *required

Name*

enter a name

Description

enter a description

Validity

From: 06/20/2016 3:03 PM

To: enter an end date

Roles required

Administrator

User

Then you will be able to use this API key with your application.

[landing]

Chapter 4. REST API

4.1. Principles

4.1.1. URLs

All URLs of the API share a common “<base URL>” :

```
http(s)://<base URL>/api/
```

Right after the base URL is placed a version number. The current version is version “v0”.

```
http(s)://<base URL>/api/v0
```

As a consequence all methods described in this document are available on URLs starting by: **http(s)://liveobjects.orange-business.com/api/v0**

4.1.2. Endpoints

4.1.3. Content

By default all methods that consume or return content only accept one format: JSON (cf. http://json.org).

As a consequence, for those methods the use of HTTP headers “*Content-Type*” or “*Accept*” with value “*application/json*” is optional.

4.1.4. API-key authentication

The API key must be added as a HTTP header named “**X-API-Key**” into the request.

Example (HTTP request to the API)

```
GET /api/v0/vendors/lora/devices HTTP/1.1
Host: <base URL>
X-API-Key: <API key>
```

If you don’t provide such an API Key, or if you use an invalid API key, *Live Objects LPWA* responds with the standard HTTP Status code **403 Forbidden**.

4.1.5. Paging

Some methods that return a list of items allow paging: the method doesn't return the full list of items, but only a subset of the complete list matching your request.

You need to use two standard query parameters (i.e. that must be added at the end of the URL, after a "?", separated by a "&", and defined like this: "<param>=<value>"):

- "**size**": maximum number of items to return (i.e. number of items per "page"),
- "**page**": number of the page to display (starts at 0).

Those parameters are not mandatory: by default "page" will be set to **0** and "size" to **20**.

Example: If size=10 and page=0 then item number 0 to 9 (at most) will be returned. If size=20 and page=1, then items number 20 to 39 (at most) will be returned.

Example (HTTP request to the API)

```
GET /api/v0/vendors/lora/devices?page=100&size=20 HTTP/1.1
Host: <base URL>
X-API-Key: <API key>
```

The responses to such methods are a "page" of items - a JSON object with the following attributes:

- **totalCount**: total number of items matching request in service (only part of them are returned),
- **size**: the value for "size" taken into account (can be different of the one in request if the value was invalid),
- **page**: the value for "page" taken into account (can be different of the one in request if the value was invalid),
- **data**: list of returned items.

4.2. Device management

4.2.1. List devices

4.2.1.1. Request

Url

```
GET /api/v0/vendors/lora/devices
```

Table 2. Query parameters

Name	Description
page	*Optional.* page selection. 0 by default.
size	*Optional.* size selection. 20 by default.
devEUI	*Optional.* regexp on devEUI
name	*Optional.* regexp on device name
status	*Optional.* device status : ACTIVATED/DEACTIVATED
sort	*Optional.* sorting selection. Prefix with '-' for descending order

HTTP Headers

```
X-API-Key: <your API key>
Accept: application/json
```

exemple:

```
GET /api/v0/vendors/lora/devices?name=DeviceTest&status=ACTIVATED&sort=-name
```

4.2.1.2. Response

HTTP Code

200 OK

Table 3. Body

JSON Params	Description
page	page selection.
size	size selection.
totalCount	list size
data	contains the device list
devEUI	device EUI (cf. LoRaWan)
name	name of the device
activationType	**OTAA**: Over The Air Activation
profile	profile of the Device which represent the Device Class (A or C). Can be a specific for a Device (*ex. LoRaMote devices*) or generic (*ex. LoRaWAN/DemonstratorClasseA or LoRaWAN/DemonstratorClasseC*). Please refer to Orange LoRa® device reference.

JSON Params	Description
deviceStatus	ACTIVATED: The device is authorized to communicate on the LPWA network. DEACTIVATED: The device is not authorized to communicate on the LPWA network, upcoming messages are dropped.
tags	*Optional.* tags registered during device registration/edition process.
lastActivationTs	last activation date of the device.
lastDeactivationTs	*Optional.* last deactivation date of the device.
lastCommunicationTs	*Optional.* last communication date of the device.
creationTs	registration date of the device
updateTs	*Optional.* last update date of the device

example:

```
{
  "page" : 0,
  "size" : 20,
  "totalCount" : 2,
  "data" : [ {
    "devEUI" : "0018B20000000272",
    "name" : "DeviceTest2",
    "activationType" : "OTAA",
    "profile" : "SMTC/LoRaMoteClassA.2",
    "deviceStatus" : "ACTIVATED",
    "tags" : [ "Lyon", "Test" ],
    "lastActivationTs" : "2016-06-09T08:04:37.971Z",
    "lastCommunicationTs" : "2016-06-03T15:55:36.944Z",
    "creationTs" : "2016-06-03T15:20:53.803Z",
    "updateTs" : "2016-06-09T08:04:37.971Z"
  },
  {
    "devEUI" : "0018B20000000274",
    "name" : "DeviceTest1",
    "activationType" : "OTAA",
    "profile" : "LoRaWAN/DemonstratorClasseA",
    "deviceStatus" : "ACTIVATED",
    "tags" : [ "Lyon", "Test" ],
    "lastActivationTs" : "2016-06-09T08:04:37.971Z",
    "lastCommunicationTs" : "2016-06-03T15:55:36.944Z",
    "creationTs" : "2016-06-03T15:20:53.803Z",
    "updateTs" : "2016-06-09T08:04:37.971Z"
  } ]
}
```

4.2.2. Device information

4.2.2.1. Request

Url

```
GET /api/v0/vendors/lora/devices/<devEUI>
```

HTTP Headers

```
X-API-Key: <your API key>
Accept: application/json
```

exemple:

```
GET /api/v0/vendors/lora/devices/0018B20000000272
```

4.2.2.2. Response

HTTP Code

200 OK

Table 4. Body

JSON Params	Description
devEUI	device EUI (cf. LoRaWan)
name	name of the device
activationType	**OTAA**: Activation Over The Air.
profile	profile of the Device which represent the Device Class (A or C). Can be a specific for a Device (*ex. LoRaMote devices*) or generic (*ex. LoRaWAN/DemonstratorClasseA or LoRaWAN/DemonstratorClasseC*). Please refer to Orange LoRa® device reference.
deviceStatus	ACTIVATED: The device is authorized to communicate on the LPWA network. DEACTIVATED: The device is not authorized to communicate on the LPWA network, upcoming messages are dropped.
appEUI	appEUI of the device (cf. LoRaWan)
tags	tags registered during device registration process.
lastActivationTs	last activation date of the device.
lastDeactivationTs	*Optional.* last deactivation date of the device.
lastCommunicationTs	*Optional.* last communication date of the device.
lastBatteryLevel	*Optional.* battery level (0: External power source, 1..254: 1=min / 254 = max, 255: Not able to measure the level).
lastDlFcnt	*Optional.* last downlink frame counter used by the platform.
lastUlFnct	*Optional.* last uplink frame counter used by the device.
creationTs	registration date of the device
updateTs	*Optional.* last update date of the device

Table 5. Error case

HTTP Code	Error code	message
400	**4001**	The device EUI validation has failed (must be an hexadecimal string of size 16)
404	**40411**	The device was not found

example:

```
{
  "devEUI" : "0018B20000000272",
  "name" : "DeviceTest2",
  "activationType" : "OTAA",
  "profile" : "SMTC/LoRaMoteClassA.2",
  "deviceStatus" : "ACTIVATED",
  "appEUI" : "0000000000000000",
  "tags" : [ "Lyon", "Test" ],
  "lastActivationTs" : "2016-06-09T08:04:37.971Z",
  "lastCommunicationTs" : "2016-06-03T15:55:36.944Z",
  "lastDlFcnt" : 1,
  "lastUlFnct" : 42,
  "lastBatteryLevel" : 127,
  "creationTs" : "2016-06-03T15:20:53.803Z",
  "updateTs" : "2016-06-09T08:04:37.971Z"
}
```

4.2.3. Unregister device

Url

```
DELETE /api/v0/vendors/lora/devices/<devEUI>
```

HTTP Headers

```
X-API-Key: <your API key>
Accept: application/json
```

exemple:

```
DELETE /api/v0/vendors/lora/devices/0018B20000000272
```

4.2.3.1. Response

HTTP Code

200 OK

Table 6. Error case

HTTP Code	Error code	message
400	**4001**	The device EUI validation has failed (must be an hexadecimal string of size 16)
404	**40411**	The device was not found
500	**5002**	Internal error. Please, contact the assistance.
500	**5003**	Internal error. Please, contact the assistance.
500	**5004**	Internal error. Please, contact the assistance.

4.2.4. Register device

4.2.4.1. Over The Air

4.2.4.1.1. Request

Url

```
POST /api/v0/vendors/lora/devices
```

HTTP Headers

```
X-API-Key: <your API key>
Content-Type: application/json
Accept: application/json
```

Table 7. Body

JSON Params	Description
deviceStatus	ACTIVATED: The device is authorized to communicate on the LPWA network. DEACTIVATED: The device is not authorized to communicate on the LPWA network, upcoming messages are dropped.
profile	profile of the Device which represent the Device Class (A or C). Can be a specific for a Device (*ex. LoRaMote devices*) or generic (*ex. LoRaWAN/DemonstratorClasseA or LoRaWAN/DemonstratorClasseC*). Please refer to Orange LoRa® device reference.
activationType	**OAA**: Activation Over The Air.
name	name of the device
tags	list of additional information used to tag the uplink messages of the device
devEUI	device EUI (cf. LoRaWan)

JSON Params	Description
appEUI	appEUI of the device (cf. LoRaWan)
appKey	appKey of the device (cf. LoRaWan)

example:

```
POST /api/v0/vendors/lora/devices
```

```
{
  "deviceStatus": "ACTIVATED",
  "profile": "LoRaWAN/DemonstratorClasseA",
  "activationType": "OTAA",
  "tags" : [ "Lyon", "Test" ],
  "name": "DeviceTest3",
  "devEUI": "0018B20000000272",
  "appEUI": "0000000000000000",
  "appKey": "D6C84412B3153C0FE26CA88CA54231F1"
}
```

4.2.4.2. Response

HTTP Code

201 CREATED

Table 8. Body

JSON Params	Description
devEUI	device EUI (cf. LoRaWan)
name	name of the device
activationType	**OTAA**: Activation Over The Air.
profile	profile of the Device which represent the Device Class (A or C). Can be a specific for a Device (*ex. LoRaMote devices*) or generic (*ex. LoRaWAN/DemonstratorClasseA or LoRaWAN/DemonstratorClasseC*). Please refer to Orange LoRa® device reference.
deviceStatus	ACTIVATED: The device is authorize to communicate on the LPWA network. DEACTIVATED: The device is not authorize to communicate on the LPWA network, upcoming messages are dropped.
appEUI	appEUI of the device (cf. LoRaWan)
tags	*Optional.* tags registered during device registration process.
lastActivationTs	last activation date of the device.

JSON Params	Description
creationTs	registration date of the device

Table 9. Error case

HTTP Code	Error code	message
400	**4001**	The device EUI validation has failed (must be an hexadecimal string of size 16)
404	**40413**	Bad account configuration. Please, contact the assistance.
409	**4096**	The device EUI is already registered
409	**40910**	The device network address is already registered
500	**5002**	Internal error. Please, contact the assistance.
500	**5003**	Internal error. Please, contact the assistance.
500	**5004**	Internal error. Please, contact the assistance.

example

```
{
  "devEUI" : "0018B20000000272",
  "name" : "DeviceTest3",
  "activationType" : "OTAA",
  "profile" : "LoRaWAN/DemonstratorClasseA",
  "deviceStatus" : "ACTIVATED",
  "appEUI" : "0000000000000000",
  "tags" : [ "Lyon", "Test" ],
  "lastActivationTs" : "2016-04-03T15:20:53.803Z",
  "creationTs" : "2016-04-03T16:22:16.301Z"
}
```

4.2.5. Update device

4.2.5.1. Over The Air

4.2.5.1.1. Request

Url

```
PATCH /api/v0/vendors/lora/devices/<devEUI>
```

HTTP Headers

```
X-API-Key: <your API key>
Content-Type: application/json
Accept: application/json
```

Table 10. Body

JSON Params	Description
deviceStatus	*Optional.* ACTIVATED: The device is authorized to communicate on the LPWA network. DEACTIVATED: The device is not authorized to communicate on the LPWA network, upcoming messages are dropped.
tags	*Optional.* List of additional information used to tag the uplink messages of the device
name	*Optional.* Name of the device
appEUI	*Optional.* AppEUI of the device (cf. LoRaWan)
appKey	*Optional.* AppKey of the device (cf. LoRaWan)

example:

```
PATCH /api/v0/vendors/lora/devices/0018B20000000272
```

```
{
  "deviceStatus": "DEACTIVATED",
  "tags" : [ "Lyon" ],
  "name": "DeviceTest3",
  "appEUI": "0000000000000000",
  "appKey": "D6C84412B3153C0FE26CA88CA54231F1"
}
```

4.2.5.2. Response

HTTP Code

200 OK

Table 11. Body

JSON Params	Description
devEUI	device EUI (cf. LoRaWan)
name	name of the device
activationType	**OTAA**: Activation Over The Air.

JSON Params	Description
profile	profile of the Device which represent the Device Class (A or C). Can be a specific for a Device (*ex. LoRaMote devices*) or generic (*ex. LoRaWAN/DemonstratorClasseA or LoRaWAN/DemonstratorClasseC*). Please refer to Orange LoRa® device reference.
deviceStatus	ACTIVATED: The device is authorize to communicate on the LoRa network. DEACTIVATED: The device is not authorize to communicate on the LPWA network, upcoming messages are dropped.
appEUI	appEUI of the device (cf. LoRaWan)
tags	*Optional.* tags registered during device registration process.
lastActivationTs	last activation date of the device.
lastDeactivationTs	*Optional.* last deactivation date of the device.
creationTs	registration date of the device
updateTs	last update date of the device

Table 12. Error case

HTTP Code	Error code	message
400	**4001**	The device EUI validation has failed (must be an hexadecimal string of size 16)
404	**40411**	The device was not found
500	**5002**	Internal error. Please, contact the assistance.
500	**5003**	Internal error. Please, contact the assistance.
500	**5004**	Internal error. Please, contact the assistance.

example

```
{
  "devEUI" : "0018B20000000272",
  "name" : "DeviceTest3",
  "activationType" : "OTAA",
  "profile" : "LoRaWAN/DemonstratorClasseA",
  "deviceStatus" : "DEACTIVATED",
  "appEUI" : "0000000000000000",
  "tags" : [ "Lyon", "Test" ],
  "lastActivationTs" : "2016-06-09T08:04:37.971Z",
  "lastDeactivationTs" : "2016-06-09T08:04:37.971Z",
  "creationTs" : "2016-06-03T15:20:53.803Z",
  "updateTs" : "2016-06-09T08:04:37.971Z"
}
```

4.3. Command

4.3.1. List commands

4.3.1.1. Request

Url

```
GET /api/v0/vendors/lora/devices/<devEUI>/commands
```

Table 13. Query parameters

Name	Description
page	*Optional.* page selection. 0 by default.
size	*Optional.* size selection. 20 by default.
timeRange	*Optional.* filter data where timestamp is in timeRange "from,to"
sort	*Optional.* sorting selection. Prefix with '-' for descending order

HTTP Headers

```
X-API-Key: <your API key>
Accept: application/json
```

example

```
GET /api/v0/vendors/lora/devices/0018B20000000272/commands?page=0&size=20&timeRange=2016-
03-05T14:46:01.000Z,2016-05-05T14:00:01.000Z&sort=port
```

4.3.1.2. Response

HTTP Code

200 OK

Table 14. Body

JSON Params	Description
page	page selection.
size	size selection.
totalCount	list size
data	contains the command list
id	unique id of the command
data	hexadecimal raw data of the command
port	port of the device on which the command was sent (cf. LoRaWan)
confirmed	network ack confirmation
commandStatus	status of the command. SENT: The command was injected into LPWA network core. ERROR: The command could injected into LPWA network core.
creationTs	registration date of the command

Table 15. Error case

HTTP Code	Error code	message
400	**4001**	The device EUI validation has failed (must be an hexadecimal string of size 16)
400	**4002**	The parameter validation has failed for the specified field
404	**40411**	The device was not found

example

```
{
    "page": 0,
    "size": 20,
    "totalCount": 2,
    "data": [
        {
          "id": "5703cfa9e4b0b24cd6862865",
          "data": "01",
          "port": 1,
          "confirmed": true,
          "commandStatus": SENT,
          "creationTs": "2016-06-03T15:50:39.669Z"
        },
        {
          "id": "5703cfa9e4b0b24cd6862866",
          "data": "01",
          "port": 1,
          "confirmed": true,
          "commandStatus": SENT,
          "creationTs": "2016-06-03T15:50:39.669Z"
        }
    ]
}
```

4.3.2. Register command

Note: The hexadecimal payload will be encrypted by the network server. The downlink frame counter will be automatically incremented based on the last "lastDlFcnt" Cf. "Device information" chapter.

4.3.2.1. Request

Url

```
POST /api/v0/vendors/lora/devices/<devEUI>/commands
```

HTTP Headers

```
X-API-Key: <your API key>
Content-Type: application/json
Accept: application/json
```

Table 16. Body

JSON Params	Description
data	hexadecimal raw data of the command
port	port of the device on which the command was sent (cf. LoRaWan)
confirmed	*Optional.* network ack confirmation

example

```
POST /api/v0/vendors/lora/devices/0018B20000000272/commands
```

```
{
  "data": "01",
  "port": 1,
  "confirmed": true
}
```

4.3.2.2. Response

HTTP Code

201 CREATED

Table 17. Body

JSON Params	Description
id	unique id of the command
data	hexadecimal raw data of the command.
port	port of the device on which the command was sent (cf. LoRaWan)
confirmed	network ack confirmation
commandStatus	status of the command. SENT: The command was sent to the Device. ERROR: The command could not be sent to the Device.
creationTs	registration date of the command

Table 18. Error case

HTTP Code	Error code	message
400	**4001**	The device EUI validation has failed (must be an hexadecimal string of size 16)
400	**4002**	The command validation has failed

example

```
{
  "id": "5703cfa9e4b0b24cd6862866",
  "data": "1324",
  "port": 1,
  "confirmed": true,
  "commandStatus": SENT,
  "creationTs": "2016-06-03T15:50:39.669Z"
}
```

4.4. UL messages

4.4.1. List messages

4.4.1.1. Request

Url

```
GET /api/v0/data/streams/urn:lora:<devEUI>!uplink
```

Table 19. Query parameters

JSON Params	Description
limit	*Optional.* max number of data to return, value is limited to 100
timeRange	*Optional.* filter data where timestamp is in timeRange "from,to"
bookmarkId	*Optional.* id of the last document retrieved that can be used to paginate

HTTP Headers

```
X-API-Key: <your API key>
Accept: application/json
```

example

```
GET /api/v0/data/streams/urn:lora:0018B20000000272!uplink?timeRange=2016-03-
05T14:46:01.000Z,2016-05-05T14:00:01.000Z&limit=100
```

4.4.1.2. Response

HTTP Code

200 OK

Table 20. Body

JSON Params	Description
id	unique id of the value
streamId	id of the message device stream (ex: urn:lora:<devEUI>!uplink)
timestamp	timestamp of the message when received by the network platform
model	data model of the field "value"
payload	hexadecimal raw data of the message. *The payload is already decrypted.*
tags	list of tags that was set on the device when the payload was received by the platform. *Tags of a value cannot be changed.*
metadata/source	source of the payload : urn:lora:<devEUI>
metadata/connector	entry point of the payload.
metadata/network/lora/devEUI	device EUI (cf. LoRaWan)
metadata/network/lora/port	port of the device on which the command was sent (cf. LoRaWan)
metadata/network/lora/fcnt	uplink frame counter of the message. (cf. LoRaWan)
metadata/network/lora/rssi	received signal strength indication measured by the best gateway.
metadata/network/lora/snr	signal noise ratio measured by the best gateway.
metadata/network/lora/sf	spreading factor used by the device.
metadata/network/lora/signalLevel	signal quality indicator from 1 to 5.
created	stored date of the payload

example

```
[
    {
      "id" : "5743000f0cf25e30a712e83c",
      "streamId" : "urn:lora:0018B20000000272!uplink",
      "timestamp" : "2016-05-23T13:05:18.307Z",
      "model" : "lora_v0",
      "value" : {
        "payload" : "ae2109000cf3"
      },
      "tags" : [ "Lyon", "Test" ],
      "metadata" : {
        "source" : "urn:lora:0018B20000000272",
        "connector": "lora",
        "network": {
          "lora": {
            "devEUI": "0018B20000000272",
            "port": 1,
            "fcnt": 3,
            "rssi": -36.0,
            "snr": 10.25,
            "sf": 7,
            "signalLevel": 2
          }
        }
      },
      "created" : "2016-05-23T13:05:19.617Z"
    },
    {
      "id" : "5742ff440cf25e30a712e836",
      "streamId" : "urn:lora:0018B20000000272!uplink",
      "timestamp" : "2016-05-23T13:01:55.334Z",
      "model" : "lora_v0",
      "value" : {
        "port" : 1,
        "fcnt" : 2,
        "signalLevel" : 2,
        "payload" : "ae1f03000cf3"
      },
      "tags" : [ "Lyon", "Test" ],
      "metadata" : {
        "source" : "urn:lora:0018B20000000272",
        "connector": "lora",
        "network": {
          "lora": {
            "devEUI": "0018B20000000272",
            "port": 1,
            "fcnt": 2,
            "rssi": -36.0,
            "snr": 10.25,
            "sf": 7,
            "signalLevel": 2
          }
        }
      },
      "created" : "2016-05-23T13:01:56.639Z"
    }
  ]
```

Chapter 5. MQTT interface

5.1. Endpoints

MQTT endpoints:

- **mqtt://liveobjects.orange-business.com:1883** for non SSL connection
- **mqtts://liveobjects.orange-business.com:8883** for SSL connection

MQTT over Websocket endpoints:

- **ws://liveobjects.orange-business.com:80/mqtt**
- **wss://liveobjects.orange-business.com:443/mqtt**

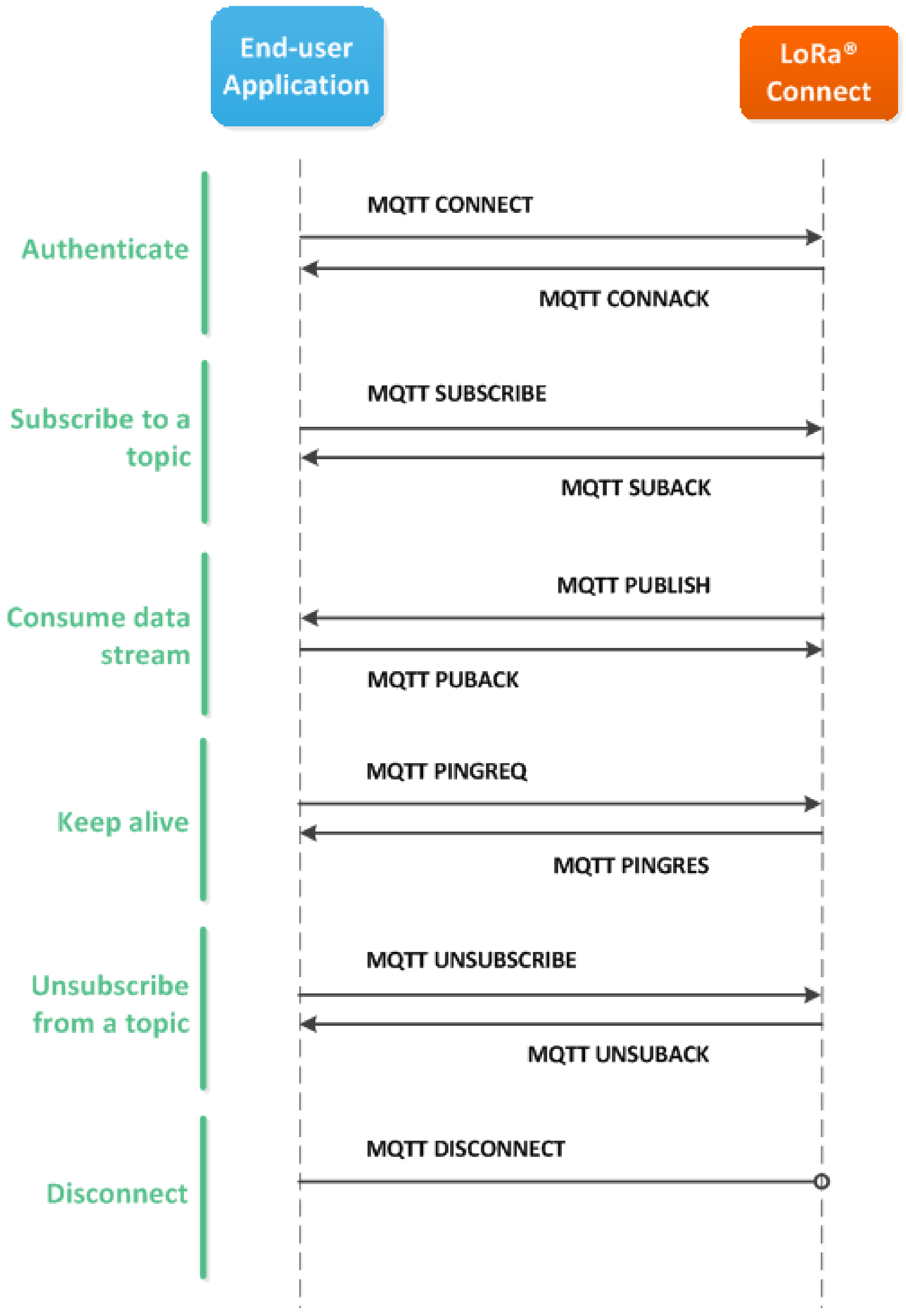
It is recommended to use the MQTTS endpoint for your production environment, otherwise your communication with Live Objects will not be secured.

The certificate presented by the MQTT server is signed by VeriSign. The public root certificate to import is the following:

```
-----BEGIN CERTIFICATE-----
MIIE0zCCA7ugAwIBAgIQGNrRniZ96LtKIVjNzGs7SjANBgkqhkiG9w0BAQUFADCB
yjELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQL
ExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMTowOAYDVQQLEzEoYykgMjAwNiBWZXJp
U2lnbiwgSW5jLiAtIEZvciBhdXRob3JpemVkIHVzZSBvbmx5MUUwQwYDVQQDEzxW
ZXJpU2lnbiBDbGFzcyAzIFB1YmxpYyBQcmltYXJ5IENlcnRpZmljYXRpb24gQXV0
aG9yaXR5IC0gRzUwHhcNMDYxMTA4MDAwMDAwWhcNMzYwNzE2MjM1OTU5WjCByjEL
MAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQLExZW
ZXJpU2lnbiBUcnVzdCBOZXR3b3JrMTowOAYDVQQLEzEoYykgMjAwNiBWZXJpU2ln
biwgSW5jLiAtIEZvciBhdXRob3JpemVkIHVzZSBvbmx5MUUwQwYDVQQDEzxWZXJp
U2lnbiBDbGFzcyAzIFB1YmxpYyBQcmltYXJ5IENlcnRpZmljYXRpb24gQXV0aG9y
aXR5IC0gRzUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCvJAgIKXo1
nmAMqudLO07cfLw8RRy7K+D+KQL5VwijZIUVJ/XxrcgxiV0i6CqqpkKzj/i5Vbex
t0uz/o9+B1fs70PbZmIVYc9gDaTY3vjgw2IIPVQT60nKWVSFJuUrjxuf6/WhkcIz
SdhDY2pSS9KP6HBRTdGJaXvHcPaz3BJ023tdS1bTlr8Vd6Gw9KIl8q8ckmcY5fQG
BO+QueQA5N06tRn/Arr0PO7gi+s3i+z016zy9vA9r911kTMZHRxAy3QkGSGT2RT+
rCpSx4/VBEnkjWNHiDxpg8v+R70rfk/Fla4OndTRQ8Bnc+MUCH7lP59zuDMKz10/
NIeWiu5T6CUVAgMBAAGjgbIwga8wDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8E
BAMCAQYwbQYIKwYBBQUHAQwEYTBfoV2gWzBZMFcwVRYJaW1hZ2UvZ2lmMCEwHzAH
BgUrDgMCGgQUj+XTGoasjY5rw8+AatRIGCx7GS4wJRYjaHR0cDovL2xvZ28udmVy
aXNpZ24uY29tL3ZzbG9nby5naWYwHQYDVR0OBBYEFH/TZafC3ey78DAJ80M5+gKv
MzEzMA0GCSqGSIb3DQEBBQUAA4IBAQCTJEowX2LP2BqYLz3q3JktvXf2pXkiOOzE
p6B4Eq1iDkVwZMXnl2YtmAl+X6/WzChl8gGqCBpH3vn5fJJaCGkgDdk+bW48DW7Y
5gaRQBi5+MHt39tBquCWIMnNZBU4gcmU7qKEKQsTb47bDN0lAtukixlE0kF6BWlK
WE9gyn6CagsCqiUXObXbf+eEZSqVir2G3l6BFoMtEMze/aiCKm0oHw0LxOXnGiYZ
4fQRbxC1lfznQgUy286dUV4otp6F01vvpX1FQHKOtw5rDgb7MzVIcbidJ4vEZV8N
hnacRHr2lVz2XTIIM6RUthg/aFzyQkqFOFSDX9HoLPKsEdao7WNq
-----END CERTIFICATE-----
```

5.2. Principles

The MQTT bridge acts as a standard MQTT v3.1 message broker (cf. MQTT Protocol Specification 3.1), with some limitations:


End-user Application
LoRa® Connect
Authenticate
MQTT CONNECT
MQTT CONNACK
Subscribe to a topic
MQTT SUBSCRIBE
MQTT SUBACK
Consume data stream
MQTT PUBLISH
MQTT PUBACK
Keep alive
MQTT PINGREQ
MQTT PINGRES
Unsubscribe from a topic
MQTT UNSUBSCRIBE
MQTT UNSUBACK
Disconnect
MQTT DISCONNECT

5.2.1. Authenticate

In order to access the uplink message stream the MQTT Agent/Codec needs to authenticate through the MQTT interface.

5.2.1.1. MQTT Connect

The first packet exchanged should be a **MQTT Connect** packet, sent from the client to the Bridge.

This packet must contain:

- **clientId**: free usage (Not taken into account),
- **username**: used to specify the format of messages : "*payload*"
- **password**: the API Key (provided on web portal)
- **willRetain**, **willQoS**, **willFlag**, **willTopic**, **willMessage**: *Not taken into account*,
- **keepAlive**: recommended: 30 seconds

On reception, the MQTT bridge validates the API Key.

- If the API Key is valid, then MQTT Bridge returns a **MQTT CONNACK** message with return code **0x00 Connection Accepted**.
- If the API Key is not valid, then MQTT Bridge returns a **MQTT CONNACK** message with return code **0x04 Connection Refused: bad user name or password**, and closes the TCP connection.

5.2.2. Subscribe to a topic

The Agent/Codec can subscribe to one or multiple device uplink message stream by configuring the MQTT **Topic**.

5.2.2.1. MQTT Subscribe

Once authenticated (cf. Authenticate), the client can at any time subscribe and unsubscribe to/from topics. MQTT Bridge answers with a **MQTT SUBACK** packet only once all subscriptions could be resolved:

MQTT specification enforce that a **SUBACK** is returned even if actual subscription is impossible / forbidden. As a consequence the MQTT client cannot be informed that it subscribed to an non existing Topic.

MQTT Bridge answers to **UNSUBSCRIBE** packet with a **UNSUBACK** packet only once existing subscriptions have been properly closed.

Available topics :

- **router/~event/v1/data/new/urn/lora/<devEUI>/uplink** to subscribe to one device uplink message data stream

- **router/~event/v1/data/new/urn/lora/#** to subscribe to all devices uplink message data streams

Using a FIFO :

- **fifo/<fifo_name>** to subscribe your persisted queue. Restrictions can be applied to API keys so each key can access only queues specified in its restriction list.

5.2.3. Consume data stream

5.2.3.1. Message Delivery

When a message is published on a topic the MQTT client subscribed to, the MQTT Bridge will deliver the message to the MQTT client by sending a **MQTT PUBLISH** message to the client, with the qos matching the client subscription.

Table 21. Message structure

JSON Params	Description
streamId	messages associated to urn:lora:<devEUI>!uplink
timestamp	timestamp of the message when received by the network platform
model	data model of the field "value"
value	contains the payload and its associated network information.
value/payload	hexadecimal raw data of the message
tags	list of tags that was set on the device when the payload was received by the platform. *Tags of a value cannot be changed.*
metadata/source	source of the payload : urn:lora:<devEUI>
metadata/connector	entry point of the payload.
metadata/network/lora/devEUI	device EUI (cf. LoRaWan)
metadata/network/lora/port	port of the device on which the command was sent (cf. LoRaWan)
metadata/network/lora/fcnt	uplink frame counter of the message. (cf. LoRaWan)
metadata/network/lora/rssi	received signal strength indication measured by the best gateway.
metadata/network/lora/snr	signal noise ratio measured by the best gateway.

JSON Params	Description
metadata/network/lora/sf	spreading factor used by the device.
metadata/network/lora/signalLevel	signal quality indicator from 1 to 5.

example

```
{
  "streamId": "urn:lora:0018B20000000272!uplink",
  "timestamp": "2016-05-23T13:05:18.307Z",
  "model": "lora_v0",
  "value": {
    "payload": "ae2109000cf3"
  },
  "tags": [
    "Lyon",
    "Test"
  ],
  "metadata": {
    "source": "urn:lora:0018B20000000272",
    "connector": "lora",
    "network": {
      "lora": {
        "devEUI": "0018B20000000272",
        "port": 2,
        "fcnt": 8,
        "rssi": -36.0,
        "snr": 10.25,
        "sf": 7,
        "signalLevel": 2
      }
    }
  }
}
```

5.2.4. Keep alive

5.2.4.1. MQTT Ping Req/Res

MQTT Bridge answers to **PINGREQ** packets with **PINGRES** packets: this is a way for the MQTT client to avoid connection timeouts. (recommended: 30 seconds).

MQTT message "qos" 0, 1 and 2 are supported, but don't offer any guarantee here: currently subscribed client to this PubSub topic may or may not receive the message.

5.2.5. Disconnect

5.2.5.1. MQTT Disconnect

MQTT Bridge closes the MQTT / TCP connection when receiving a **MQTT DISCONNECT** message.

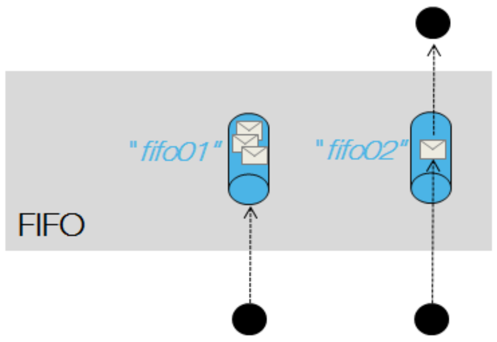
5.2.5.2. TCP Disconnect

When the TCP connection closes (by client or MQTT bridge), the MQTT bridge will close itself the currently active subscriptions, etc.

5.3. Persisted queue

In order to prevent data loss during real time data consumption over MQTT, Live Objects provide a configurable **FIFO mode**, to store the unacknowledged data. Messages published on a FIFO topic are persisted until a subscriber is available and acknowledges the handling of the message. Publication to and consumption from a FIFO topic use acknowledgement.

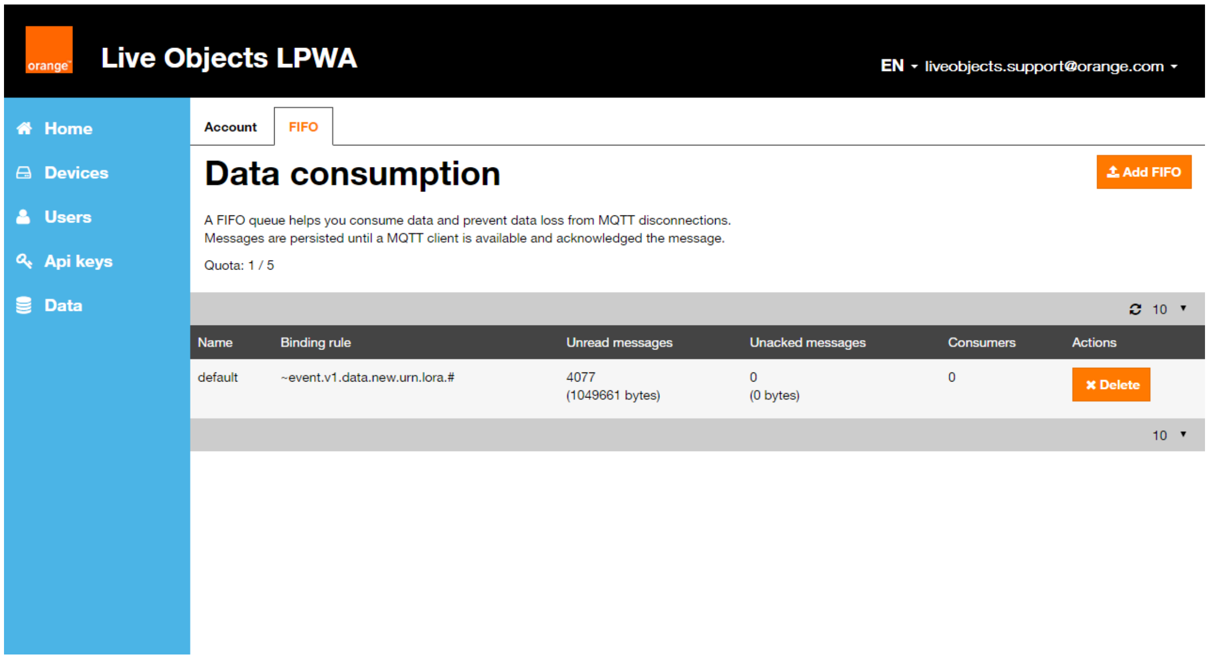
Concept



If multiple subscribers consume from the same FIFO topic, messages are load balanced between them. For more information please refer to LiveObjects MQTT Documentation.

To create a FIFO click on **your mail address** ⇒ **Settings** ⇒ **FIFO**

Fifo page



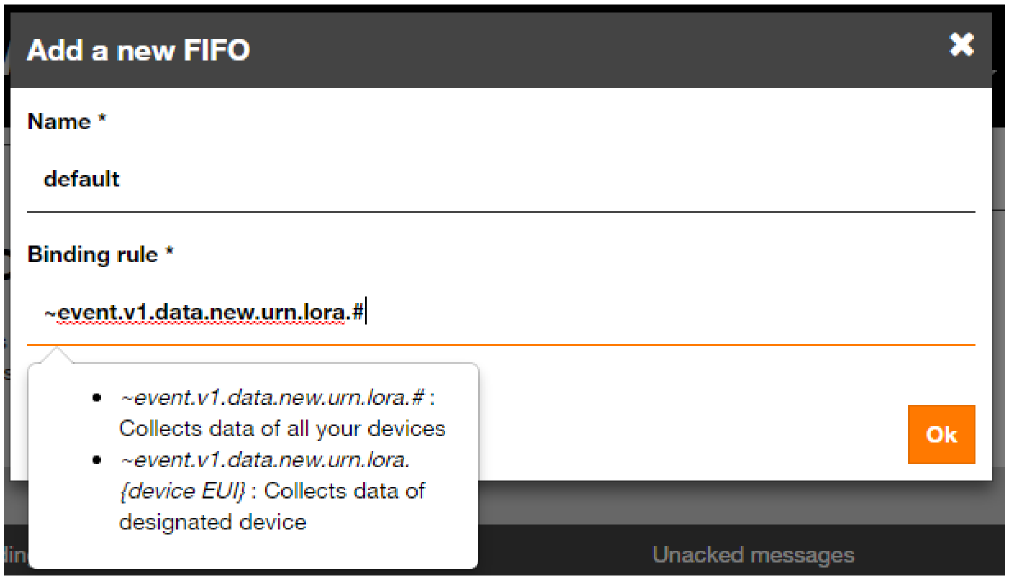
Data consumption

A FIFO queue helps you consume data and prevent data loss from MQTT disconnections.
Messages are persisted until a MQTT client is available and acknowledged the message.

Quota: 1 / 5

Name	Binding rule	Unread messages	Unacked messages	Consumers	Actions
default	~event.v1.data.new.urn.lora.#	4077 (1049661 bytes)	0 (0 bytes)	0	Delete

Add FIFO form



Name *

default

Binding rule *

~event.v1.data.new.urn.lora.#

- *~event.v1.data.new.urn.lora.#* : Collects data of all your devices
- *~event.v1.data.new.urn.lora.{device EUI}* : Collects data of designated device

5.4. Quick start using HiveMQ

HiveMq is an online tool that help you use your web browser as a MQTT client for testing purpose. Go to : hivemq.com

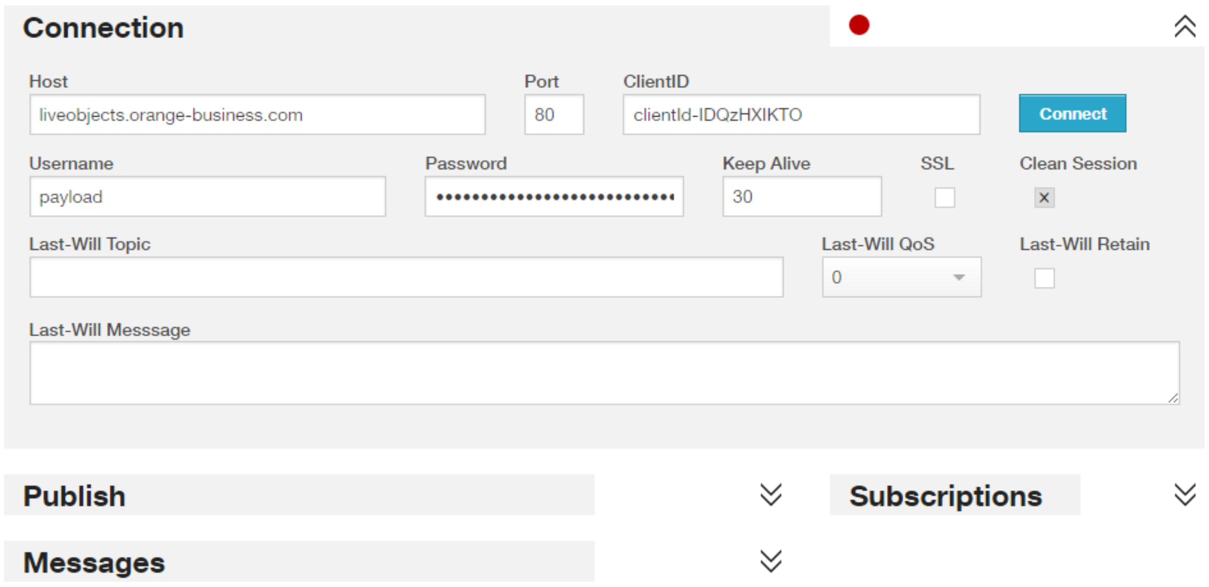
5.4.1. Connect

- **Host** : liveobjects.orange-business.com
- **Port** : *80 for websocket*
- **ClientID** : *<anything>*
- **Username** : *payload*

- **Password** : a valid API Key you registered on the web portal (Cf. Getting started)

Then click "Connect"

example



Host	liveobjects.orange-business.com	Port	80
ClientID	clientId-IDQzHXIKTO		
Username	payload	Password	••••••••••••••••••••••••••
Keep Alive	30	SSL	☐
Clean Session	☒		
Last-Will Topic		Last-Will QoS	0
Last-Will Retain	☐		
Last-Will Messsage			

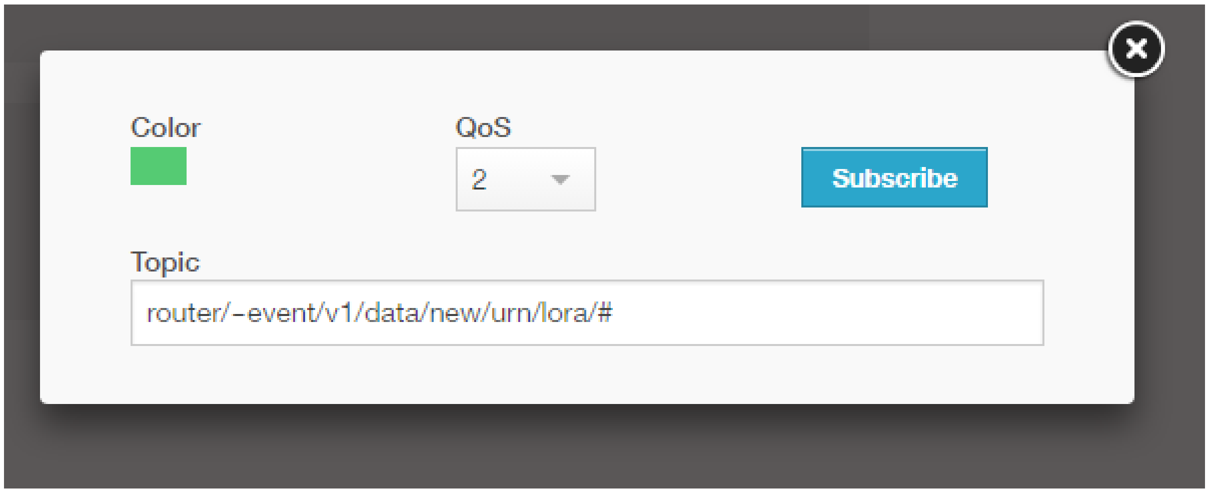
5.4.2. Subscribe

Click on "Add New Topic Subscription"

- **Topic** : the topic. *example:* **router/~event/v1/data/new/urn/lora/#** for all devices uplink message data streams. Or **fifo/<fifo_name>** to consume from a fifo.

Then click "Subscribe"

example

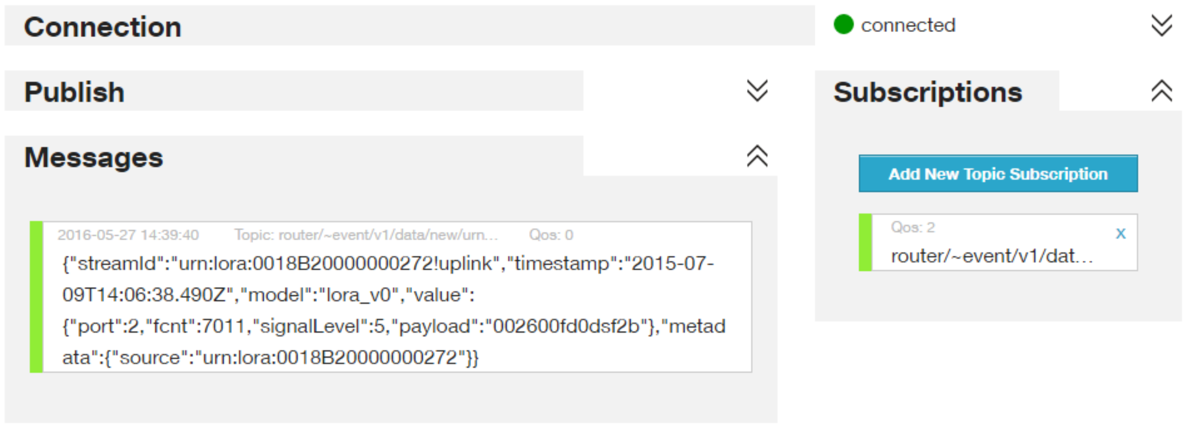


Color	🟩
QoS	2
Topic	router/~event/v1/data/new/urn/lora/#

5.4.3. Consume

For each payload sent by one of your device, you should see an uplink message on the "Messages" tab

example



2016-05-27 14:39:40 Topic: router/~event/v1/data/new/urn... Qos: 0

```
{"streamId":"urn:lora:0018B20000000272!uplink","timestamp":"2015-07-09T14:06:38.490Z","model":"lora_v0","value":{"port":2,"fcnt":7011,"signalLevel":5,"payload":"002600fd0dsf2b"},"metadata":{"source":"urn:lora:0018B20000000272"}}
```