



Swisscom LPN Portal

Developer Guide for CMP LPN

Device Manager, Wireless Logger and Network Manager
LPN Core to Application Server Tunnel Interface
August 2023

NOTICE

This document contains proprietary material of Swisscom (Schweiz) Ltd. Any unauthorized reproduction, use, or disclosure of this material, or any part thereof, is strictly prohibited.

The material provided in this document is believed to be accurate and reliable. However, no responsibility is assumed by Swisscom (Schweiz) Ltd. for the use of this material. Swisscom (Schweiz) Ltd. reserves the right to make changes to the material at any time and without notice. This document is intended for information and operational purposes only. No part of this document shall constitute any contractual commitment by Swisscom (Schweiz) Ltd.

Portions of this documentation and of the software herein described are used by permission of their copyright owners.

Versions

Version	Date	Author	Details
1	31/08/16	Swisscom	Initial version
1.1	30/10/16	Swisscom	Revised initial version
1.2	31/01/17	Swisscom	New GUI features, new layout
1.3	04/05/17	Swisscom	DX API description
2	16/04/19	Swisscom	Merged with Application Development Guide
2.1	03/08/20	Swisscom	New features and uplink fields, valid with new platform release
2.2	02/10/20	Swisscom	Minor improvements, DX-API error codes added
2.3	25/06/21	Swisscom	IoT Flow Connector, Connectivity plan features and minor improvements
2.4	01/04/23	Swisscom	IoT Flow GUI, Tunnel API v2, Network Manager
2.5	01/08/23	Swisscom	MFA

Table of Contents

Definitions and acronyms	7
1. Available Documents and Scope	9
2. Swisscom LPN System	10
3. Device Manager	11
3.1. Devices Creation and Management	12
3.1.1 Device list	12
3.1.2 Device details	13
3.2. Device provisioning	16
3.2.1 Secure key provisioning	20
3.2.2 Connectivity plan (CP)	21
3.2.3 Removal of AS Routing Profile or CP	21
3.2.4 Billing of different message types	22
3.2.5 Connectivity Plan details	22
3.3. Application servers	26
3.3.1 Create a new application server	26
3.3.2 Edit an Application server	27
3.3.3 Activate UL/DL Security	28
3.3.4 Delete an Application server	28
3.4. AS routing profiles	29
3.4.1 Create an AS Routing Profile	30
3.4.2 Modify or Delete an AS Routing Profile	32
3.4.3 Assign or Remove an AS Routing Profile	32
3.5. Multi factor authentication (MFA) setup	33
4. Other LoRaWAN features	34
4.1. Roaming	34
4.2. IoT Flow network connectors	34
4.3. Multicasting	35
4.4. Network Geolocation	35
5. Upload Devices with CSV file	36
5.1. Guide on using the csv generator	36
6. Wireless Logger	38

6.1.	Metadata	38
6.2.	Expanding a message	40
7.	Network Manager	41
7.1.	Dashboard	41
7.2.	Base Station List	41
7.3.	Base Station View	42
7.4.	Managing Alarms	42
8.	Swisscom LPN API overview	44
9.	Swisscom LPN Portal Tunnel API	45
9.1.	Connectivity	45
9.2.	Parameters Format	46
9.2.1	ISO 8601 timestamps	46
9.2.2	Timestamp Encoding When UL/DL Security Is Activated	46
10.	Tunnel API (Uplink)	47
10.1.	Uplink Frame Report	47
10.2.	Sample of Uplink Frame HTTP Request	49
10.3.	LRC Authentication for UL Frame and DL Frame	51
10.4.	XML or JSON Encoding	52
10.4.1	Sample of Uplink JSON Payload	53
11.	Tunnel API (Downlink)	54
11.1.	Downlink Frame	54
11.1.1	Sample of Downlink Frame HTTP Request	55
11.1.2	Confirmed Downlink	56
11.1.3	LRC HTTP Response Codes (v1):	56
11.1.4	LRC HTTP Response Codes (v2):	57
11.2.	Downlink Multicast	59
11.3.	Application Server Authentication for Downlink Frame	59
12.	IoT Flow network connectors	61
12.1.	Set up the routing	61
12.2.	Add tags to your devices	61
12.3.	Set up your connection	62
12.4.	Set up your Flow	63
12.5.	Create custom drivers	64
12.6.	Queueing behavior	64

12.7.	Sample of Uplink JSON with decoded payload	64
12.8.	Uplink Transformation	66
12.9.	Sample of Downlink JSON payload	67
13.	DX API	68
13.1.	Authentication	68
13.1.1	DX Admin API	69
13.1.2	DX Core API	70
13.1.3	DX API versioning	70
13.1.4	DX API error codes	71
14.	LoRaWAN specification	72
14.1.	Globally unique EUI-64: DevEUI	73
14.2.	Over-The-Air Activation (OTAA)	74
14.2.1	128 bit application key: AppKey	74
14.2.2	64 bit application server identifier: JoinEUI (former AppEUI)	74
14.3.	Activation By Personalization (ABP)	75
14.3.1	Device address: DevAddr	75
14.3.2	128 bit network secret: NwkSKey	75
14.4.	Channel plans	76
14.4.1	ETSI EU 868	76
15.	Developing on LoRaWAN	77
15.1.	The LPN Developer Portal	77
15.1.1	Swisscom IoT Qualified for LoRaWAN devices	77

Definitions and acronyms

ABP	Activation by Personalization
ACK	Acknowledgement of an alarm
ADR	Adaptive Data Rate
AES	Advanced Encryption Standard
AS	Application Server
BPM	Business Process Management
BSS	Billing Support Systems
CP	Connectivity Plan
CSP	Communication Service Provider
End Device	A sensor or actuator
ESP	Estimated Signal Power
ETSI	European Telecommunications Standards Institute
FCtrl	Frame Control
GUI	Graphical User Interface
HSM	Hardware Security Module
HTTPS	Hypertext Transfer Protocol Secure
IEC	International Electrotechnical Commission
IoT	Internet of Things
ISM	Industrial Scientific Medical
JSON	JavaScript Object Notation
LAT	Latitude
LC	Logical Channel
LON	Longitude
LoRaWAN™	Long Range Wide Area Network
LPWAN	Low Power Wide Area Network
LRC	Long-Range Controller: Network Server
LRR	Long-Range Relay: software inside the gateway
M2M	Machine-to-Machine
MAC	Media Access Control
MIC	Message Integrity Code
NW	Network
OSS	Operations Support Systems
OTAA	Over-The-Air-Activation
PER	Packet Error Rate
REST	Representational State Transfer
RF	Radio Frequency
RFU	Reserved for Future Use

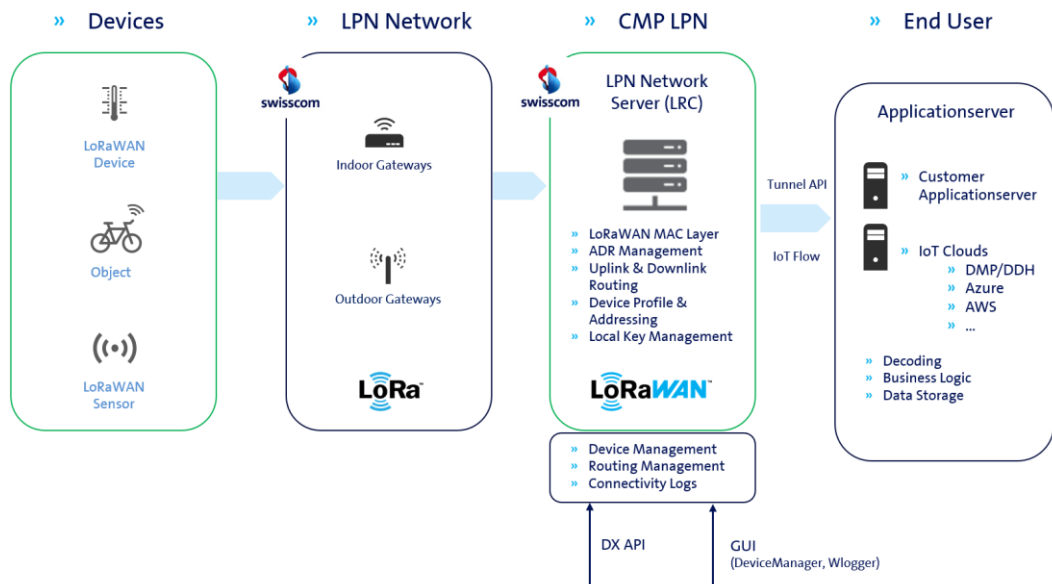
RIT	Receiver Initiated Transmit
RSSI	Received Signal Strength Indicator
Rx	Receiver
RX1	First receive window
RX2	Second receive window
SaaS	Software As A Service
SF	Spreading Factor
SNR	Signal to Noise Ratio
SSO	Single Sign On
TWA	ThingPark Wireless Application
Tx	Transmitter
UNB	Ultra Narrow Band
URL	Uniform Resource Locator
XML	Extensible Markup Language

1. Available Documents and Scope

Please get the newest version of this document via [this link](#). There are also LoRaWAN specifications, gateway guides, csv templates and tutorial videos available to get you started. The Scope of this Developer Guide is to provide the guidelines to a developer during the Swisscom LPN Portal connectivity integration.

2. Swisscom LPN System

This chapter will help you establishing the communication between a device and the network. Let's have a look at the LPN system:



Everything related to the LoRaWAN MAC layer happens in the Network Server. You will need to provision some information for each device, and you can also define the routing strategy. You have three options for the provisioning, that are described in the following chapters:

- Provision your devices one by one using the GUI (only recommended for tests)
[See section 3.1](#)
- Generate a csv in the right format and use the csv upload function
[See section 5](#)
- Automate your device provisioning using the DX API
[See section 13](#)

Concerning the message routing (receiving your device's messages on your end application), you currently have two options:

- Use the standard REST API connector (HTTP / HTTPS) to push each device message to your application server when it arrives
[See section 3.3 \(setup\) and 3.4 \(specification\)](#)
- Use the IoT Flow connector for direct integrations into Microsoft Azure, AWS, your MQTT broker and more.
[See section 12](#)

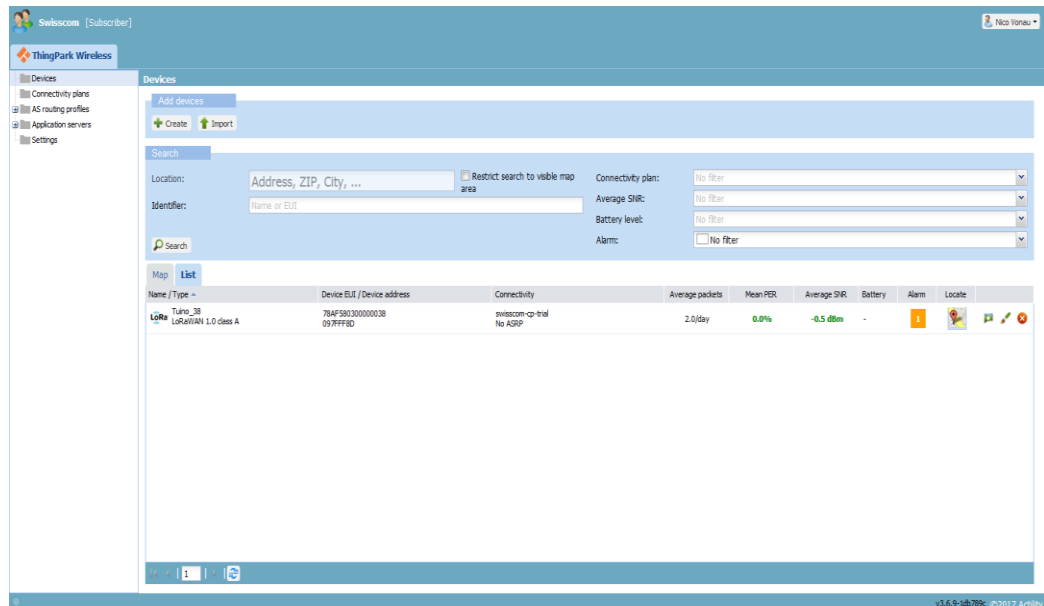
3. Device Manager

The device manager is one of the options to provision and manage your devices.

- > Launch your Device Manager via the following link:
<https://portal.lpn.swisscom.ch/thingpark/wireless/gui/deviceManager/>

When you login for the first time, you need to setup a 2 factor authentication. If you struggle with the setup, please check the instructions in chapter 3.5.

Once you login to the Device Management portal, you will get an overview of all the devices of your account. You can easily shift between the **Map** and **List** view of devices, by clicking the corresponding tabs.



The interface is based on 2 frames, a left sidebar menu and a main application frame showing device data.

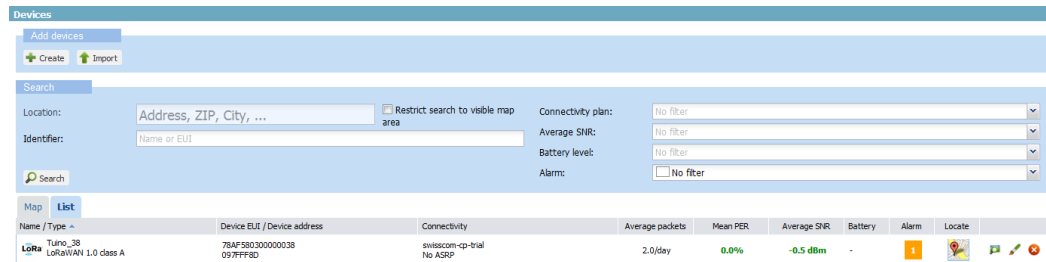
The left sidebar menu gives access to devices, connectivity plans, AS routing profiles, application servers and Settings.

The first main frame contains a Search bar, allowing users to search devices by Location, device Identifier or other filtering criterias.

3.1. Devices Creation and Management

3.1.1 Device list

Device List displays all the filtered devices in a list.



The screenshot shows the 'Devices' management interface. It includes a search bar with fields for 'Location' (Address, ZIP, City, ...), 'Identifier' (Name or EUI), and 'Connectivity plan'. There are also dropdown menus for 'Average SNR', 'Battery level', and 'Alarm'. Below the search filters, there are 'Map' and 'List' tabs. The 'List' tab is active, displaying a table of device data.

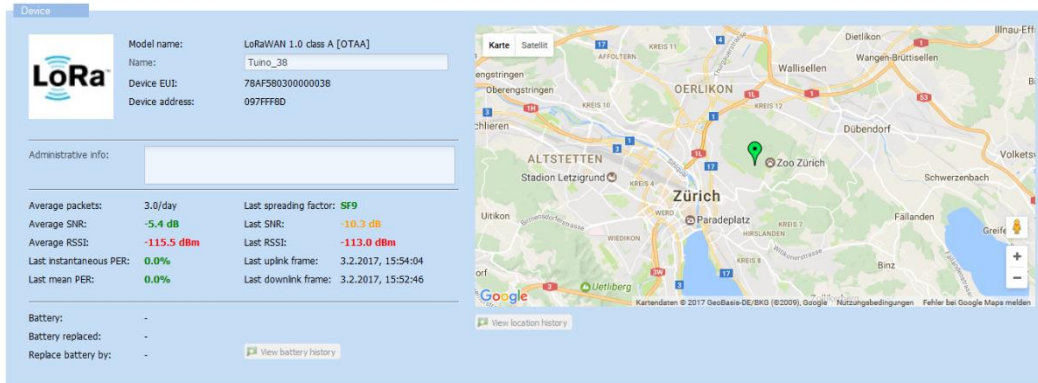
Name / Type	Device EUI / Device address	Connectivity	Average packets	Mean PER	Average SNR	Battery	Alarm	Locate
L0Ra Turbo_38 LoRaWAN L1.0 class A	78AF580300000038 092FFFD0	swisscom-cp-trial No ASRP	2.0/day	0.0%	-0.5 dBm	-	1	

The displayed field are:

- > Name / Type: name and device profile
- > MAC IEEE address / Device Address: DevEUI and DevAddr of the device
- > Connectivity plan / application routing profile
- > Mean packet error rate
- > Average amount of packets per day
- > Average SNR: based on the last 5 packets received
- > Battery status
- > Alarm: number of alarms not acknowledged
- > Locate: open a pop-up and display the device on a map
- > Button () to view more info of the device
- > Button () to edit the settings of the device
- > Button () to delete the device

The filtering/sorting can be done on most of the above mentioned properties. Users can therefore easily display all devices for which e.g. the battery level is low, that have raised critical alarms, etc.

3.1.2 Device details



Device frame

This frame displays the basic information of the device such as the model (device profile), name, DevEUI, DevAddr.

The interface also provides information on the LoRaWAN traffic such as the average number of packets, average RSSI and SNR, last instantaneous/mean PERs, last RSSI/SNR/SF, date/time of the last message received/sent.

The battery status information is not requested from the devices in the productive connectivity plans and can therefore not be displayed here.

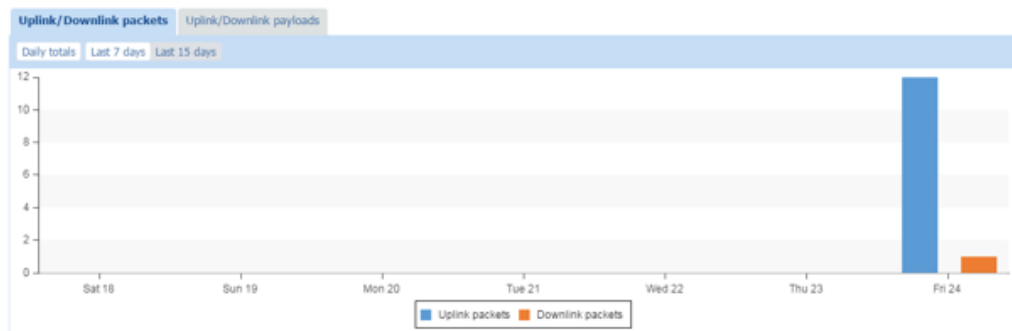
The location of the device on the map could be provided in 2 different ways:

- > Manual location: The position of the device is "hardcoded" in the platform
- > Network location: Using the location solver of the platform (see section 4.4 of this guide)

If a location is set, it will also be sent to the Application Server inside the *CustomerData* field. An approximation of the device's position can also be taken from the position of the last seen gateway, transmitted to the Application Server in the fields *LrrLAT* and *LrrLON*.

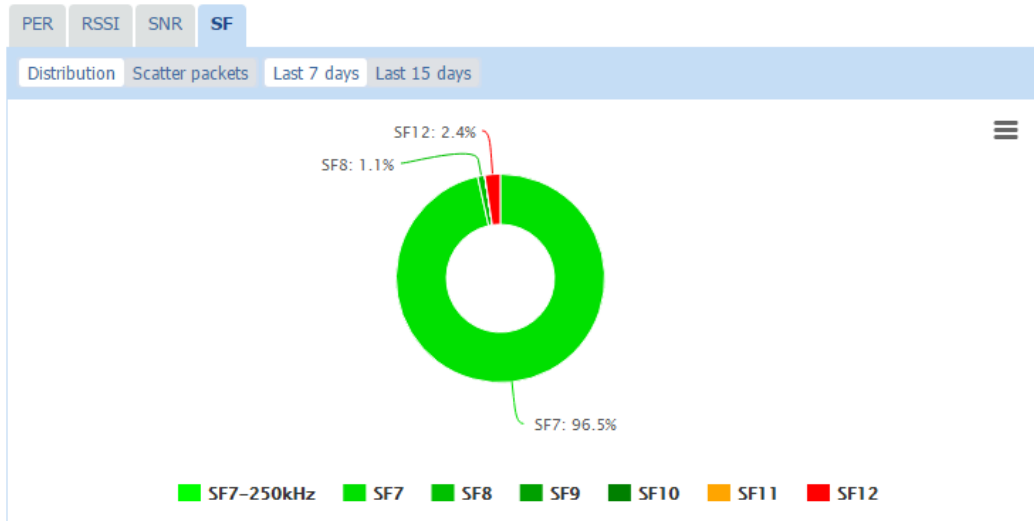
The **View location history** button displays a map with markers showing where the devices are located.

Uplink/Downlink frame



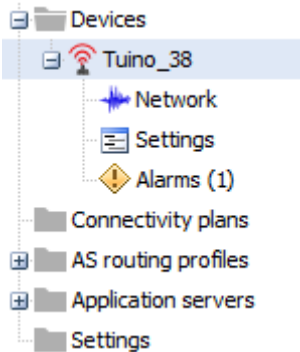
The graphic above displays the number of uplink/downlink packets and payloads (bytes) over the selected period (Daily totals, last 7 days, last 15 days).

PER/RSSI/SNR/SF graph



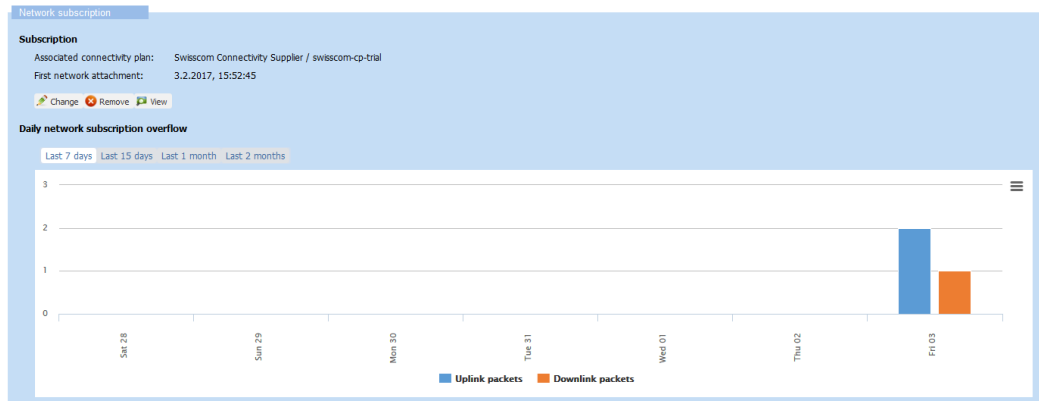
The graphic above displays the PER, RSSI, SNR or SF distribution over the selected period.

Device Network



The Devices/Network section provides information on the Network subscription, associated connectivity plan, routing plans associated to the device and the Network Coverage.

Network subscription



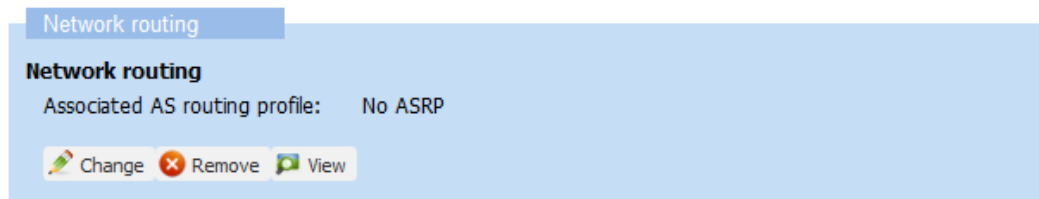
The Subscription section displays the current associated connectivity plan and the first date on which the device has communicated.

The graphic displays the usage of the device in the connectivity plan, showing the number of packets on the selected period.

HOW-TO change the Connectivity plan

- > Go to the device Edit view
- > Go to the **Network** section (blue icon in the left-hand menu)
- > Click on **Change** in the subscription section
- > Select the desired Connectivity Plan from the drop-down menu
- > Click **Save**

Network/cloud routing



The Network routing section displays the current associated AS routing profile. It is possible to view the details of this routing profile, or also change and remove it.

HOW-TO change or assign an AS Routing Profile

- > Go to the device Edit view
- > Go to the **Network** section (blue icon in the left-hand menu)
- > Click on **Change** in the Network routing section
- > Select the preferred AS routing profile from the drop-down menu in the pop-up
- > Click **Save**

3.2. Device provisioning

The device provisioning allows users to create devices and register them on the network through **Activation By Personalization (ABP)** or **Over the Air Activation (OTAA)**.

There are three ways to create a device

- > Manual creation: create devices one by one
- > Batch creation: mass import from a csv file to create several devices in one go
- > API device creation (see section 12 of this guide)

ABP Address range



The preferred way to add a device is OTAA. In this case, the device address is handled by the Network Server. If you still want to use ABP, you need to request a Swisscom address range from Support.LPN@swisscom.com and please mind that ABP shall not be used in production.

Information required to create a new device:

ABP:

- > DevEUI
- > DevAddr
- > NwkSKey

Optional information may be required:

- > AppSKey (if none is provided, data is sent encrypted to the Application Server)

OTAA:

- > DevEUI
- > AppEUI
- > AppKey



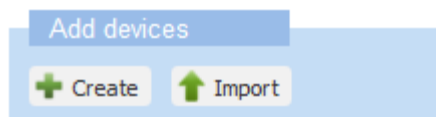
Note: Please do not enter any fixed DevAddr for OTAA devices! The address will be dynamically allocated withing the Swisscom address range.

ABP Manual Device creation

The manual creation is mainly for support or development purpose. Once the device is in production, refer to the Batch provisioning mode.

To create a new device manually:

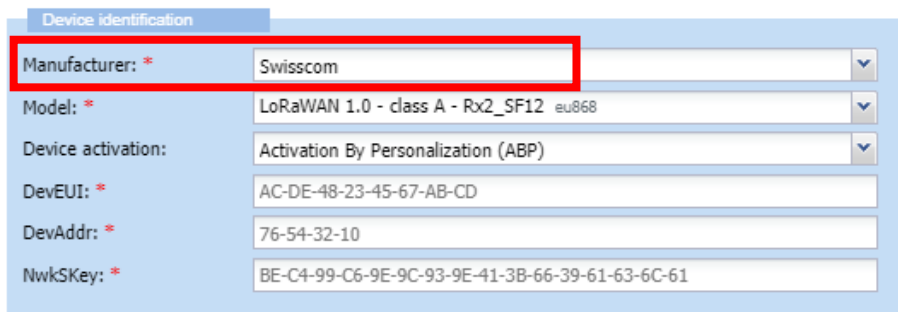
- > Click on "Create" in the **Devices** view



- > A new pop-up appears with 4 sections: Administrative data, Device identification, Network parameters, Application layer handling.
- > Enter a name for the device
- > The Administrative Info and Location are optional.
- > Change the marker if you wish to customize the device marker
- > Enter any relevant administrative information (Such information will be displayed in the e-mail generated by an alarm)
- > Set a location

- Network location
- Manual location

Fill out the Device identification section. Please note that all fields are mandatory



- > Device EUI (hex): **DevEUI**, device EUI, globally unique IEEE EUI-64 address
- > Network address (hex): **DevAddr**, device address
- > Network key (hex): **NwkSKey**, Network Session Key, 128-bit key
- > Manufacturer: **Swisscom**. All the supported profiles are listed here.
- > Model: Choose the correct model for your device (Class A/B/C, LoRaWAN specification etc...)

Particularities in the Device Profile choice	
Name	Description
LoRaWAN 1.0 -	Designation for specification 1.0.1 and older
LoRaWAN 1.0 – class A – RX2_SF9	Profile for old devices still needing RX2 on SF9 as OTAA boot parameters
LoRaWAN 1.0 – class A – RX2_SF12	Most commonly used boot parameters
LoRaWAN 1.0.2 revB	Latest revision of 1.0.2, most devices of 1.0.2 stack use this.
LoRaWAN 1.0.3	Has the same functions as 1.0.2 revB
LoRaWAN 1.0.4	Has the same functions as 1.0.3

Fill the Network section



- > Choose a **Connectivity plan** in the drop-down menu, the displayed count indicates the remaining connectivity plans available

- > Choose an **AS routing profile** in the drop-down menu

Application layer handling

Application server routing profile: noas

AppSKeys:

Application Session Key	Port
000102030405060708090A0B0C0D0E0F	3

Update Cancel

+ Add x Delete

Fill out the Application layer section

- > To add a new Application Key
 - Click on **Add**
 - Fill the **Key**, 128-bit key
 - Enter a **port** number, " * " sends the data to all ports
 - Click on **Update**

The port 0 is encrypted by the NwkSKey

The optional 128-bit AppSkey is used to encrypt the payload of the messages, and has to be shared with the application server. You may decide to use a unique AppSkey for all LoRaWAN ports used by your device (* keyword), or to allocate one AppSkey for each port.

- > If you do not provision the AppSkey, The Swisscom LPN Portal will forward the payload in encrypted form to the application servers and has no access to the payload clear-form content.
- > If you provision the AppSkey(s), the LRC will decode the payload before forwarding it to the application server(s).

OTAA Manual Device creation

The Join device activation procedure is similar as the ABP provisioning, except the keys to inquire.

Device identification	
Manufacturer: *	Swisscom
Model: *	LoRaWAN 1.0.3 - class A - Rx2_SF12 eu868
Device activation:	Over The Air Activation (OTAA)
Join server: *	Local Join server with software encryption
DevEUI: *	AC-DE-48-23-45-67-AB-CD
JoinEUI (AppEUI):	AC-DE-48-23-45-67-AB-CD
Key format:	Clear text
AppKey: *	BE-C4-99-C6-9E-9C-93-9E-41-3B-66-39-61-63-6C-61

19

Fill out the Device identification section. Please note that all fields are mandatory

- > Device EUI (hex): **devEUI**, globally unique IEEE EUI-64 address
- > Device profile: Bidirectional communication class
- > Application EUI (hex): AppEUI is a global application ID in IEEE EUI64 address space that uniquely identifies application provider of the end-device; The Swisscom LPN AppEUI is: **F0:3D:29:AC:71:00:00:01**
- > Application key (hex): is an AES-128 application key specific for the end-device that is assigned by the application owner to the end-device and is responsible to encrypt JOIN communication.
- > Manufacturer: **Swisscom**. All the supported profiles are listed here.
- > Model: Choose the correct model for your device (Class A/B/C, LoRaWAN version). If you are unsure about this, ask your device manufacturer for the LoRaWAN version of your device.
- > Key format: Choose clear text to enter your key unencrypted, or RSA encryption (described in section 14.2)

Modify a device

Modifying a device allows you to update device-related data such as the name, the manually entered location, define another connectivity plan or define a routing plan.

Start by opening a device in Edit view:

- > Click on **Edit** to enter the Edit view



Modification allowed:

- > Device name
- > Administrative info
- > Device location
- > Device marker
- > Change/remove a connectivity plan
- > Change/remove a AS routing plan

Finally, to confirm the changes, go back to the device details by selecting the device in the column in the left sidebar, then click **Save** in the top-right corner of the screen.

Delete a device

Deleting a device is an action which cannot be undone and should be handled with care. All device details and device status information will be lost.

Below are the steps to delete a device:

- > Click on **Delete** to delete the device



- > Confirm in the pop-up to delete the device
- > Go back to the device list, click on **Refresh** to refresh the list

3.2.1 Secure key provisioning

Instead of clear text, you can also upload an RSA encrypted AppKey by choosing "RSA encrypted" in the Key format field.



The key can be provided as base64 string or as a file. Here are the commands to generate such a file with OpenSSL (Linux):

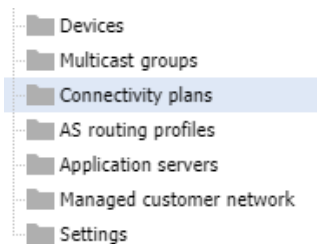
```
echo "0018B244415246320018B2000000CD7" > appKey
xxd -r -p appKey > appKey.bin
openssl rsautl -encrypt -in appKey.bin -inkey TWK1.pem -pubin -pkcs -out
encryptedAppKey.bin
base64 encryptedAppKey.bin >> encryptedAppKey.text
```

3.2.2 Connectivity plan (CP)

The Connectivity plan defines the network connectivity features (e.g. confirmed messages, downlink traffic), and traffic policing parameters (token bucket regulators for uplink and downlink traffic) and is associated to a given activation and recurring fee.

To access Connectivity plans, click on Connectivity plans in the left sidebar menu:

- > Connectivity plan section



First Connectivity plan section displays the available plans in your account:

- > Connectivity plans: name of the connectivity plan
- > ID of the connectivity plan (required for batch provisioning)
- > Purchased credit for end devices: number of maximum devices allowed in the plan
- > Used up credit by end devices: number of devices registered on the selected plan

The screenshot shows a table titled 'Purchased network subscriptions' with the following data:

Connectivity plans	ID	Purchased credit for end devices	Used up credit by end devices
Swisscom Connectivity Supplier / swisscom-cp-trial (15)	swisscom-cs/swisscom-cp-trial	16	1

So the remaining devices that could be provisioned are “Purchased credit” – “Used up credit”. Additional connectivity plans can be purchased from your preferred Swisscom sales person.

Naming examples

Price plan (contract)	Connectivity plan name in the platform
Basic Low	swisscom-cs/swisscom-cp-basic-low
Advanced High	swisscom-cs/swisscom-cp-advanced-high
Trial	swisscom-cs/swisscom-cp-trial

3.2.3 Removal of AS Routing Profile or CP

Devices can be disabled by removing the Connectivity Profile. The device is then unprovisioned and will not be billed any more from the following month on.



Attention: Removing the Connectivity Plan or AS Routing Profile deletes the provisioning context of the device. It needs to **send a new Join Request** in order to reconnect to the network. Please keep this in mind when dealing with devices that are already deployed in the field!

3.2.4 Billing of different message types

- > Customers in the PAYG model are billed for the number of uplink and downlink messages. Here is an overview of the messages that will be billed (✓) and not be billed (✗).

Billed as uplink message		Billed as downlink message	
Uplink payload messages other than MAC (FPort > 0)	✓	Downlink payload messages other than MAC (FPort > 0)	✓
Join-requests	✓	Join-accepts	✓
MAC request: LinkCheckReq	✓	MAC answer: LinkCheckAns	✓
MAC answer: DevStatusAns	✓	MAC request: DevStatusReq	✓
MAC request: DeviceTimeReq	✓	MAC answer: DeviceTimeAns	✓
ACK after confirmed downlink	✓	ACK after confirmed uplink	✓
All other MAC traffic (ACK bit not set and FPort=0)	✗	All other MAC traffic (ACK bit not set and FPort=0)	✗

3.2.5 Connectivity Plan details

In the **Connectivity Plan details** tab, the name, ID and description of the plan are displayed. Furthermore, this tab provides a view on the following characteristics of the plan:

> Uplink frame parameters

- Acknowledged uplink frame: When taking delivery of an uplink packet from a device, the network server sends an acknowledgment of receipt (ACK) to the device. By default, this parameter is enabled.
- Rate regulator: nb of frames allowed per hour, nb of frames allowed in burst
- Uplink Regulator policy: Describes how the network behaves in case of uplink traffic overload. By default, this parameter is set to Mark.
 - Mark: The system will keep track of the device packets exceeding the limits' set. Exceeding packets are sent and marked.
 - Drop: The system will drop the device packets exceeding the limits' set. Exceeding packets are deleted.
- Base station buffering policy: RFU
- Force Adaptive Data Rate: The Adaptive Data Rate is always used even if not requested by the device. By default, this parameter is disabled.
- Asynchronous UL processing: Disabled by default.

> Downlink frame parameters

- Downlink transmission: Enable/ Disable downlink packets transmission to the device. By default, this parameter is enabled.
- Acked downlink frame: Allow to send downlink confirmed
- Rate regulator: number of frames allowed per hour, number of frames allowed in burst
- Regulator policy: uplink and downlink regulator policy (see Uplink frame Regulatory policy above)
- Device status request frequency: Number of DevStatusReq sent by the server in 24h

- Report device battery level: Report Device battery usage level to the Device Manager application and to Third Party Application Servers
- Report device margin: Report Device signal margin to the Device Manager application and to Third Party Application Servers
- Minimal RX1 delay (ms): Specifies at the device level the delay in milliseconds between the uplink and the first downlink receive window (RX1). If provided, this value takes precedence over the RX1 value defined in the RF Region. By default, this parameter is not set.

> **Network parameters**

- Mobility: RFU
- Network geolocation: Enables the device network geolocation services (GPSfree). The Adaptive Data Rate policy seeks to keep the device within reach of three base stations.
- Add Base station metadata information: Provides LRR meta information (RSSI, SR, SNR, LRR, ...) into routing messages to third party applications servers.
- Class B support: If supporting the feature, allows an OTAA or ABP device to switch to LoRaWAN™ Class B mode after a start-up procedure.
- Managed Customer Network: Not available.

> **Routing parameters**

- ThingPark X routing: Allows routing messages to and from the ThingPark X service (Activity proprietary Application Server, and more).
- Third party application routing: Feature flag to allow routing to third party servers
- Max. destinations per route: maximum number of destinations per route
- Add PER information: Forward PER to the Application Server
- Downlink sent indication: Forward sent downlink indication to the Application Server
- ThingPark Kafka routing: Allows message routing to Third Party Application Servers using a Kafka topic.

The connectivity plan associated with the device must enable the routing parameter corresponding to the Application Server used by the device

> **Roaming parameters**

- Roaming activation allowed: RFU
- Passive roaming allowed: RFU
- Handover Roaming allowed: RFU

> **Security**

Hardware Security Module (HSM) protection: AppKey protection for OTAA devices / encrypted key storage on the NS.

> **Grade-of-Service based ADR parameters**

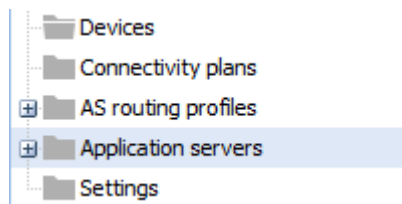
- Minimal/Maximal SF: Lowest/highest spreading factor allowed for a device
- Force channel mask: Force a specific channel mask
- Minimum antenna (macro) diversity: Minimum number of base stations simultaneously receiving the device packets, typically used for network geolocation.
- Force RX2 data rate: Specifies a data rate for the RX2 receive window to overcome the default data rate defined in the RF Region settings.
- Margin offset: The offset to apply on top of the global SNR margin set at RFregion level to tune the global margin differently for different class of services to control uplink PER.

- Number of transmissions offset: Defines the offset added to the number of transmissions configured in the RF Region.
- Adaptive Data Rate algorithm: ADR algorithm Version, ADR v2: Signal to Noise Ratio-based ADR optimization or ADR v3: Packet Error Rate-based optimization. The ADR algorithm is used to manage the data rate, number of transmissions and RF output power for each device individually, in order to maximize both battery life of devices and overall network capacity.
- Target packet error rate: "Packet Error Rate value targeted by the ADR algorithm
- Macro diversity reliability: Minimum probability target of having N Base Stations receiving Device uplink packets
- Minimum/Maximum number of transmissions: Minimum/Maximum number of Device uplink transmissions to ensure quality of service will not be degraded

Connectivity plan details																							
Connectivity Plan Name:	Swisscom Connectivity Supplier / swisscom-cp-nb-trial (59)																						
Connectivity Plan ID:	swisscom-cs/swisscom-cp-nb-trial																						
Connectivity:	LoRaWAN																						
Communication Type:	Unicast																						
Connectivity Plan Description:	swisscom-cp-nb-trial																						
<table border="1"> <thead> <tr> <th colspan="2">Uplink frame parameters</th> </tr> </thead> <tbody> <tr> <td>Acknowledged uplink frame:</td> <td>Enabled</td> </tr> <tr> <td>Rate regulator:</td> <td>6 frame(s)/hour, 144 frame(s) burst</td> </tr> <tr> <td>Uplink regulator policy:</td> <td>Mark</td> </tr> <tr> <td>Base Station buffering policy:</td> <td>Not set.</td> </tr> <tr> <td>Force Adaptive Data Rate:</td> <td>Disabled</td> </tr> <tr> <td>Asynchronous UL processing:</td> <td>Not set.</td> </tr> </tbody> </table>		Uplink frame parameters		Acknowledged uplink frame:	Enabled	Rate regulator:	6 frame(s)/hour, 144 frame(s) burst	Uplink regulator policy:	Mark	Base Station buffering policy:	Not set.	Force Adaptive Data Rate:	Disabled	Asynchronous UL processing:	Not set.								
Uplink frame parameters																							
Acknowledged uplink frame:	Enabled																						
Rate regulator:	6 frame(s)/hour, 144 frame(s) burst																						
Uplink regulator policy:	Mark																						
Base Station buffering policy:	Not set.																						
Force Adaptive Data Rate:	Disabled																						
Asynchronous UL processing:	Not set.																						
<table border="1"> <thead> <tr> <th colspan="2">Downlink frame parameters</th> </tr> </thead> <tbody> <tr> <td>Downlink transmission:</td> <td>Enabled</td> </tr> <tr> <td>Acknowledged downlink frame:</td> <td>Enabled</td> </tr> <tr> <td>Rate regulator:</td> <td>0.59 frame(s)/hour, 14 frame(s) burst</td> </tr> <tr> <td>Downlink regulator policy:</td> <td>Drop</td> </tr> <tr> <td>Device status request rate (request/day):</td> <td>0.0/day</td> </tr> <tr> <td>Report Device battery level:</td> <td>Disabled</td> </tr> <tr> <td>Report Device signal margin:</td> <td>Disabled</td> </tr> <tr> <td>Minimal RX1 delay:</td> <td>Not set.</td> </tr> </tbody> </table>		Downlink frame parameters		Downlink transmission:	Enabled	Acknowledged downlink frame:	Enabled	Rate regulator:	0.59 frame(s)/hour, 14 frame(s) burst	Downlink regulator policy:	Drop	Device status request rate (request/day):	0.0/day	Report Device battery level:	Disabled	Report Device signal margin:	Disabled	Minimal RX1 delay:	Not set.				
Downlink frame parameters																							
Downlink transmission:	Enabled																						
Acknowledged downlink frame:	Enabled																						
Rate regulator:	0.59 frame(s)/hour, 14 frame(s) burst																						
Downlink regulator policy:	Drop																						
Device status request rate (request/day):	0.0/day																						
Report Device battery level:	Disabled																						
Report Device signal margin:	Disabled																						
Minimal RX1 delay:	Not set.																						
<table border="1"> <thead> <tr> <th colspan="2">Network parameters</th> </tr> </thead> <tbody> <tr> <td>Mobility:</td> <td>Disabled</td> </tr> <tr> <td>Network geolocation:</td> <td>Disabled</td> </tr> <tr> <td>Add Base Station metadata information:</td> <td>Enabled</td> </tr> <tr> <td>Class B support:</td> <td>Disabled</td> </tr> <tr> <td>Managed customer network:</td> <td>Disabled</td> </tr> </tbody> </table>		Network parameters		Mobility:	Disabled	Network geolocation:	Disabled	Add Base Station metadata information:	Enabled	Class B support:	Disabled	Managed customer network:	Disabled										
Network parameters																							
Mobility:	Disabled																						
Network geolocation:	Disabled																						
Add Base Station metadata information:	Enabled																						
Class B support:	Disabled																						
Managed customer network:	Disabled																						
<table border="1"> <thead> <tr> <th colspan="2">Routing parameters</th> </tr> </thead> <tbody> <tr> <td>ThingPark X routing:</td> <td>Disabled</td> </tr> <tr> <td>Third Party Application Servers routing:</td> <td>Enabled</td> </tr> <tr> <td>Maximum allowed destinations per Application Server routes:</td> <td>3</td> </tr> <tr> <td>Third Party Application Server PER information:</td> <td>Enabled</td> </tr> <tr> <td>Third Party Application Server downlink sent indication:</td> <td>Disabled</td> </tr> <tr> <td>ThingPark Kafka routing:</td> <td>Disabled</td> </tr> </tbody> </table>		Routing parameters		ThingPark X routing:	Disabled	Third Party Application Servers routing:	Enabled	Maximum allowed destinations per Application Server routes:	3	Third Party Application Server PER information:	Enabled	Third Party Application Server downlink sent indication:	Disabled	ThingPark Kafka routing:	Disabled								
Routing parameters																							
ThingPark X routing:	Disabled																						
Third Party Application Servers routing:	Enabled																						
Maximum allowed destinations per Application Server routes:	3																						
Third Party Application Server PER information:	Enabled																						
Third Party Application Server downlink sent indication:	Disabled																						
ThingPark Kafka routing:	Disabled																						
<table border="1"> <thead> <tr> <th colspan="2">Roaming parameters</th> </tr> </thead> <tbody> <tr> <td>Roaming Activation Allowed:</td> <td>Disabled</td> </tr> <tr> <td>Passive Roaming Allowed:</td> <td>Disabled</td> </tr> <tr> <td>Handover Roaming Allowed:</td> <td>Disabled</td> </tr> </tbody> </table>		Roaming parameters		Roaming Activation Allowed:	Disabled	Passive Roaming Allowed:	Disabled	Handover Roaming Allowed:	Disabled														
Roaming parameters																							
Roaming Activation Allowed:	Disabled																						
Passive Roaming Allowed:	Disabled																						
Handover Roaming Allowed:	Disabled																						
<table border="1"> <thead> <tr> <th colspan="2">Security</th> </tr> </thead> <tbody> <tr> <td>Hardware Security Module (HSM) protection:</td> <td>Disabled</td> </tr> </tbody> </table>		Security		Hardware Security Module (HSM) protection:	Disabled																		
Security																							
Hardware Security Module (HSM) protection:	Disabled																						
<table border="1"> <thead> <tr> <th colspan="2">Grade-of-Service based ADR parameters</th> </tr> </thead> <tbody> <tr> <td>Minimum Spreading Factor:</td> <td>Not set.</td> </tr> <tr> <td>Maximum Spreading Factor:</td> <td>Not set.</td> </tr> <tr> <td>Force channel mask:</td> <td>Not set.</td> </tr> <tr> <td>Minimum antenna (macro) diversity:</td> <td>1</td> </tr> <tr> <td>Force RX2 Data Rate:</td> <td>Not set.</td> </tr> <tr> <td>Adaptive Data Rate algorithm:</td> <td>ADR v3: Packet Error Rate (PER) based ADR optimization</td> </tr> <tr> <td>Device PER target:</td> <td>10%</td> </tr> <tr> <td>Macro diversity reliability target:</td> <td>80%</td> </tr> <tr> <td>Minimum number of Device uplink transmissions:</td> <td>1</td> </tr> <tr> <td>Maximum number of Device uplink transmissions:</td> <td>3</td> </tr> </tbody> </table>		Grade-of-Service based ADR parameters		Minimum Spreading Factor:	Not set.	Maximum Spreading Factor:	Not set.	Force channel mask:	Not set.	Minimum antenna (macro) diversity:	1	Force RX2 Data Rate:	Not set.	Adaptive Data Rate algorithm:	ADR v3: Packet Error Rate (PER) based ADR optimization	Device PER target:	10%	Macro diversity reliability target:	80%	Minimum number of Device uplink transmissions:	1	Maximum number of Device uplink transmissions:	3
Grade-of-Service based ADR parameters																							
Minimum Spreading Factor:	Not set.																						
Maximum Spreading Factor:	Not set.																						
Force channel mask:	Not set.																						
Minimum antenna (macro) diversity:	1																						
Force RX2 Data Rate:	Not set.																						
Adaptive Data Rate algorithm:	ADR v3: Packet Error Rate (PER) based ADR optimization																						
Device PER target:	10%																						
Macro diversity reliability target:	80%																						
Minimum number of Device uplink transmissions:	1																						
Maximum number of Device uplink transmissions:	3																						

3.3. Application servers

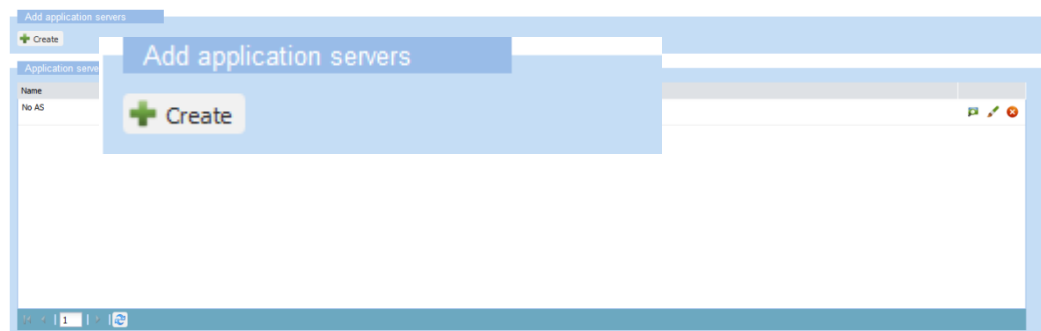
The Application Server (AS) defines where the data is routed to. **Data is never stored** on the LPN portal. Redundancy may be added to handle a failover to a second application server with the "sequential strategy explained later. The AS needs to be defined prior to adding a routing profile to it.



To access the Application Server settings, click on Application servers on the left sidebar menu:

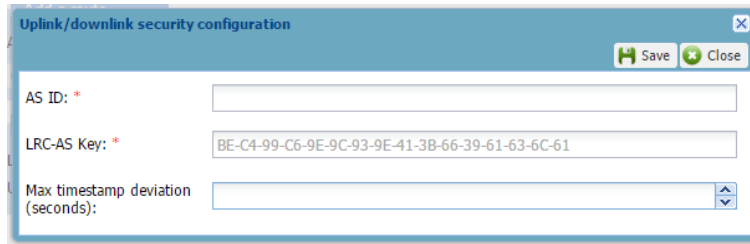
The right-hand side will then be populated with the following dialog:

3.3.1 Create a new application server



- > In the 'New application server' section, click on
- > In the name section of the new dialog, enter the name of your new Application server and click "Create":

- > A new dialog appears.
- > Select the type of content your Application Server can handle (xml or json) for the metadata, including a field for the payload
- > For production it is strongly recommended to activate up- and downlink security: Click on 'Activate' in the 'Uplink/downlink security' section
 - Enter the Application Server unique ID, signature key and allowed timestamp deviation between the LRC and your Application Server. This can seriously affect communication if both sides are not properly synchronized.
 - Click on 'Save' to validate the information

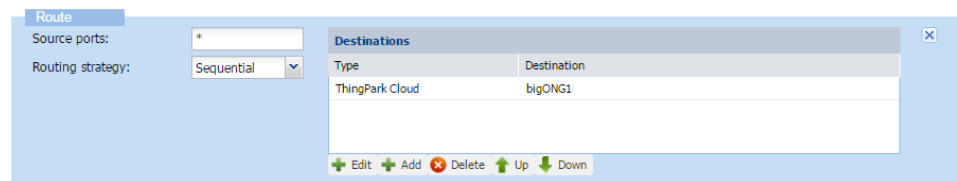


- > Add a new route – URL or IP adress
- > Click on 'Save' to save the new Application server, then close.

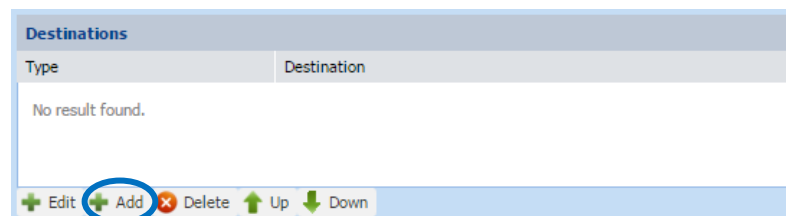
Application servers	
Name	ID
My AppServer	TWA_100000807.471.AS

3.3.2 Edit an Application server

- > In the list of application server, select the server you want to edit and click on "✎", you might have to confirm that you want to edit if not already done.
- > In the 'Add a route' section, click on 'Add'
 - A new layer "Route" appears

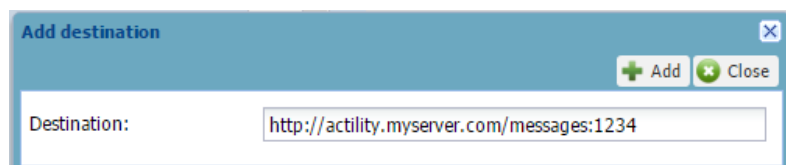


- Enter the **Source ports** to route
The Source ports are the LoRaWAN ports, it could be only one port (1), a range of ports (1-4), or all ports (*).
- Choose the **Routing strategy**
In case multiple destinations are given, the routing strategy defines how the data will be sent to these destinations. "Sequential" is preselected and should always be selected. "Sequential" means the data will be sent to the first destination and only be sent to the subsequent destination if the previous one is not available, i.e., if there is no answer in 5 seconds. If none of the destinations are available, the message will be discarded. In order to send one message to multiple endpoints, multiple application have to be defined (as



explained in this section) and these application servers have to be added to an AS routing profile as explained in section 3.4.1. The routing strategy "blast" is deprecated and should not be used anymore.

- Click on **Add** to create a new one
- Select the Destination of the new route



- Click on **Add** to add this new destination
- > Sort between different destinations, for order selection if using Sequential strategy
 - You can change the order in which messages will be sent using the arrow buttons

Destinations	
Type	Destination
Third party AS (HTTP)	http://activity.myserver.com/messages:1234
Third party AS (HTTP)	http://www.otherserver.com/messages:4567

- > To edit a destination, select it and click on **Edit**
- > To delete a destination, select it and click on **Delete**

Note: Basic authentication is supported. Please use the following format in the destination URL:

`https://user:password@www.example.com/`

Or use the field HTTP custom headers above. Note that you need to do the base64 encoding yourself if you use the HTTP custom header field.

HTTP custom headers	
Name	Value
Authorization	Basic dXNlcm5hbWU6cHc=

3.3.3 Activate UL/DL Security

Uplink and Downlink security is inactive by default. However it is **highly recommended** to set a key in this part, in order to use the authentication functions described in 10.3 and 11.3.

Uplink/downlink security

Status: **Inactive**

Max timestamp deviation: -

Uplink/downlink security configuration

AS ID: *

LRC-AS Key: * BE-C4-99-C6-9E-9C-93-9E-41-3B-66-39-61-63-6C-61

Max timestamp deviation (seconds):

First click on "Activate" in the Uplink/Downlink security part, then enter an ID and a random key for this specific Application Server. You can also set the maximum timestamp deviation that will be accepted by the LRC.



If you have multiple Application Servers set in your AS Routing Profile, all of them need to have UL/DL security activated, otherwise you can still send downlinks to the device without using a token.

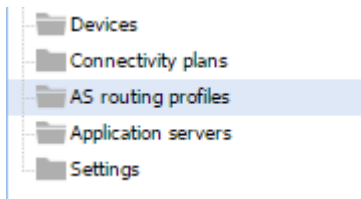
3.3.4 Delete an Application server

- > In the list of application server, select the server you want to delete and click on ; then confirm that you want to delete the application server.

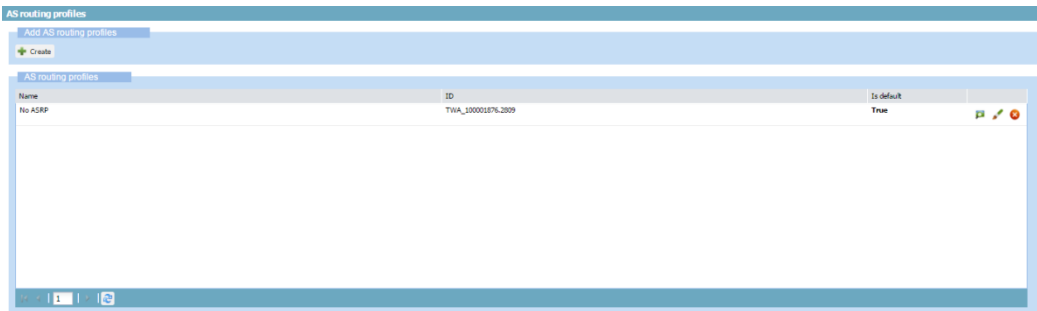
3.4. AS routing profiles

An AS routing profile is attributed to one or multiple devices, and defines to which Application Servers the device's messages will be sent. You can add one or multiple Application Servers, but please mind that the maximum number of end destinations per device is limited in your connectivity plan (normally 3 destinations).

To access an AS routing profile, click on AS routing profiles on the left sidebar menu:



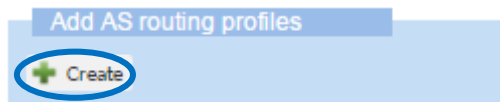
You will see the existing AS routing profiles in the list where you can View details or Edit an AS routing profile.



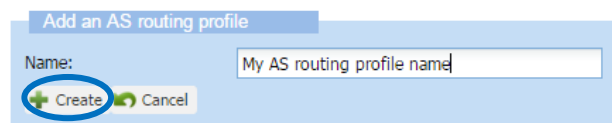
3.4.1 Create an AS Routing Profile

In order to create a new AS Routing Profile, go through the following steps:

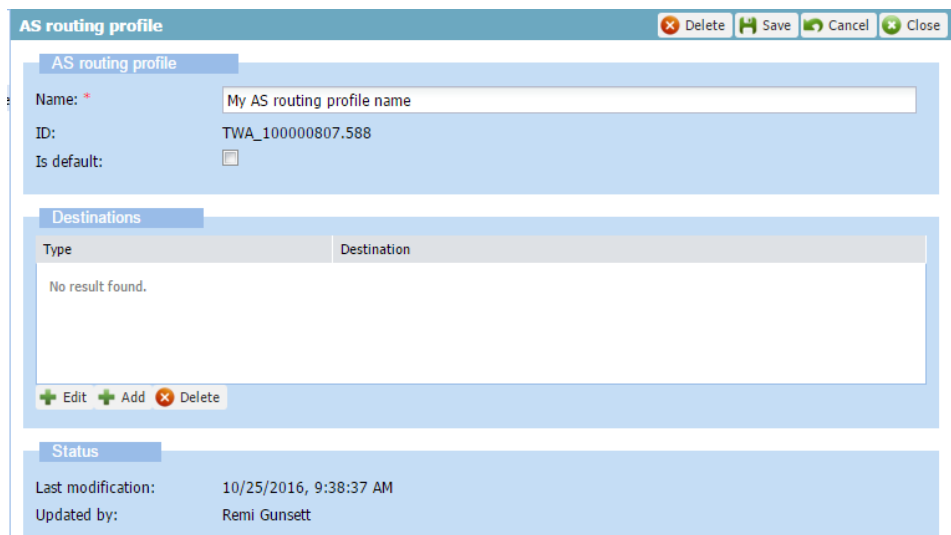
- > Click on **Create** in the **Add AS routing profiles** section



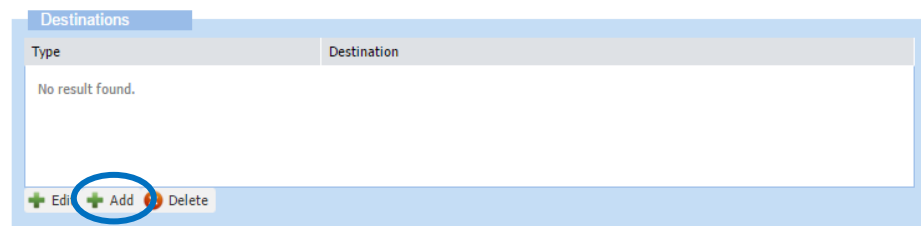
- > Enter a name to the desired new AS routing profile



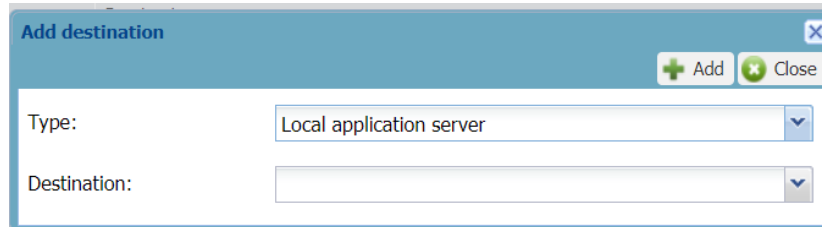
- > Click on **Create**
- > This new AS routing profile is opened and you can now edit it



- > Set or unset the profile as default: check **Is default** checkbox
- > Add a route
 - Click on **Add** in the "Destinations" section



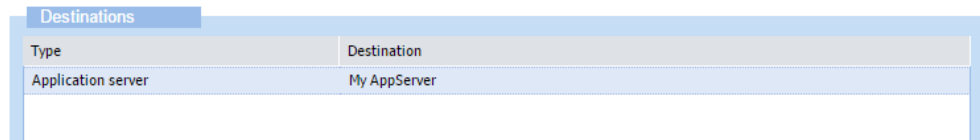
- A new pop up 'Add destination' appears



The screenshot shows a dialog box titled "Add destination" with a close button (X) in the top right corner. Below the title bar are two buttons: a green plus sign followed by "Add" and a green minus sign followed by "Close". The main area contains two dropdown menus. The first is labeled "Type:" and is currently set to "Local application server". The second is labeled "Destination:" and is currently empty.

Settings for your HTTP/HTTPS application servers

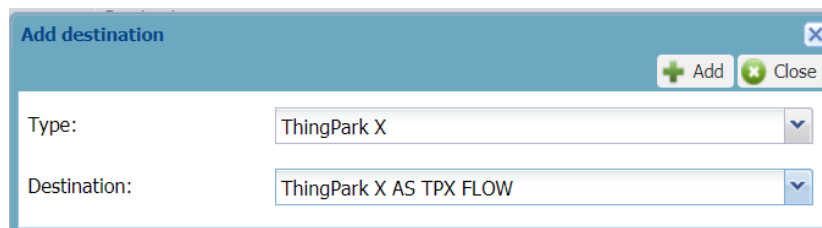
- Select the type of destination (Application server) and the destination
 - Click on 'Add'
- > The Application server is now in the list



Type	Destination
Application server	My AppServer

- > If you want to route your messages to multiple Application servers at the same time, you can add multiple Destinations in the routing profile. Then all listed Destinations receive the same message.

Note: Choose "Local application server" to find your list of Application Servers you have created previously. If you want to use the IoT Flow connector (see chapter 12), choose "ThingPark X" as type and "ThingPark X AS TPX FLOW" as destination.



The screenshot shows the same "Add destination" dialog box. The "Type:" dropdown is now set to "ThingPark X" and the "Destination:" dropdown is set to "ThingPark X AS TPX FLOW".

Settings to add the IoT Flow connector

3.4.2 Modify or Delete an AS Routing Profile

In order to modify an AS Routing Profile

- > Select the applicable Profile in the list
- > Click on 

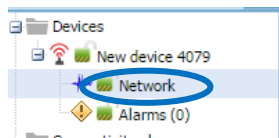


Remember to click on Save after any modification made in case of edition.

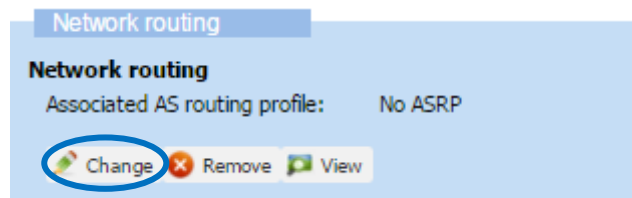
3.4.3 Assign or Remove an AS Routing Profile

First, go to the Device Edit view in order to modify a device:

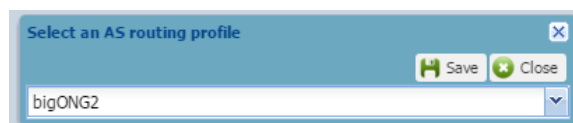
- > Select a device in the list
- > Click on  to enter the Edit view
- > Go to the **Network** section



- > In the **Network routing** section
- > Click on **Change**



- > Select your new AS routing profile in the drop-down menu



- > To remove the current associated AS routing profile, click on **Remove**

3.5. Multi factor authentication (MFA) setup

Upon your first login to Device Manager, Wireless Logger, Network Manager or IoT Flow, you will need to scan the displayed QR Code with an authenticator app on your smartphone to activate multi factor authentication. You can use any authenticator app such as FreeOTP, Google Authenticator, Microsoft Authenticator, Authy, etc. In any case you need to add an account, scan the QR code, enter the code displayed in the app to the portal and click submit. With every login you will need to put the one-time code generated in your authenticator app. If you need to relink the account to a new app, click on "forgot password" in the login interface to start the process again.



You need to set up Mobile Authenticator to activate your account.

1. Install one of the following applications on your mobile
 - FreeOTP
 - Google Authenticator
2. Open the application and scan the barcode



[Unable to scan?](#)

3. Enter the one-time code provided by the application and click Submit to finish the setup

SUBMIT



Username or email

Password

[Forgot Password?](#)

LOG IN

4. Other LoRaWAN features

Features like pricing plans and pricing models are configured by choosing the according **Connectivity Plan**. Also other features like roaming, network geolocation and third-party routing (Microsoft Azure, MQTT, Amazon etc...) can be activated on a device by attributing the right connectivity plan to the device. If you are missing a specific connectivity plan in your profile, please contact your sales person or IoT.SPOC@swisscom.com to get it added to your contract.

Features and default connectivity plans

Here is a list of our default connectivity plans and the features connected to them. Depending on your contract, you might also have other combinations.

	Basic		Plus		Advanced		Trial
	Low	High	Low	High	Low	High	
Class A, B, C	X	X	X	X	X	X	X
Roaming			X	X	X	X	
IoT-Flow			X	X	X	X	X
Multicast					X	X	
Geolocation					X	X	

4.1. Roaming

The use of roaming features will involve some more technical requirements on the device side:

- The use of the Swisscom JoinEUI (see section 14.2.2) is mandatory
- Roaming involves some more MAC-layer features that might be unused when running the device only in Switzerland. Make sure that your device is Swisscom IoT Qualified before using it in roaming, see section 15.1.1.

4.2. IoT Flow network connectors

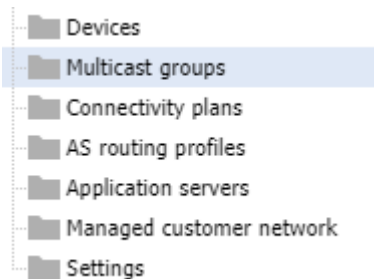
It is possible to route directly to third party services like Microsoft Azure, Amazon Web Services or to your MQTT broker. You will need a connectivity plan that includes the IoT Flow feature. For more details on how to configure the connection, please [visit chapter 12](#) of this guide.

Currently supported

- HTTPS (REST API): Available as standard connector, without IoT Flow. [Visit chapter 3.3](#)
- Microsoft Azure IoT Hub / Event Hubs
- Amazon Web Services (AWS) IoT Core
- MQTT over SSL, WSS or TCP (you need your own MQTT broker)

4.3. Multicasting

Multicasting allows you to create groups of class C devices. You can then send a downlink to all of them at the same time. This feature makes sense for some specific use cases, please contact Support.LPN@swisscom.com to get onboarded and use the feature. You will need the **Advanced** connectivity plan.



4.4. Network Geolocation

The Swisscom LPN network allows you to localize your LoRaWAN device, based on the signal strength (RSSI) received at each gateway, and the time difference on arrival (TDOA) of the message at each gateway. The solver is already integrated in our network server infrastructure. What will be provided to the application server, are the approximate coordinates of your device, together with some precision estimations.

Network geolocation is included in the **Advanced** connectivity plan. The approximate precision of the localization can vary between 200m (urban areas) and a few kilometers (rural areas). There is no requirement on the hardware side, any LoRaWAN device sending uplinks can be localized. Please follow the separate network localization user guide for more details.

5. Upload Devices with CSV file

The creation of multiple devices can either be done in an automated way using the DX API, or by batch upload with a csv file. Please make sure that you are using the up to date version of the example-input and csv generator file available [here](#).

5.1. Guide on using the csv generator

Use the corresponding tabs for OTAA or ABP devices and fill all the mandatory information.



Task (Mandatory)	'CREATE_OTAA': OTAA device creation 'CREATE_ABP': ABP device creation
DevEUI (Mandatory)	Globally unique IEEE EUI-64 address in hexadecimal format.
DevADDR	OTAA: Very important to leave this empty. Other wise your device address might not be supported by the Swisscom network server. ABP: Your address range is to be requested from Support.LPN@swisscom.com beforehand. Please notice that OTAA is the preferred way of activation for massive deployments.
Device profile ID (Mandatory)	Choose one of the following IDs corresponding to the LoRaWAN verison and class of your device. These are also valid for DX API: LORA/SwisscomA.1.0.2a_ETSI_Rx2-SF12 LORA/SwisscomB.1.0.2a_ETSI_Rx2-SF12 LORA/SwisscomC.1.0.2a_ETSI_Rx2-SF12 LORA/SwisscomA.1.0.2b_ETSI_Rx2-SF12 LORA/SwisscomB.1.0.2b_ETSI_Rx2-SF12 LORA/SwisscomC.1.0.2b_ETSI_Rx2-SF12 LORA/SwisscomA.1_ETSI_Rx2-SF12 LORA/SwisscomA.1_FCC_Rx2-SF9 LORA/SwisscomC.1_ETSI_Rx2-SF12 LORA/SwisscomC.1_ETSI_Rx2-SF9 LORA/SwisscomA.1.0.3_ETSI_Rx2-SF12 LORA/SwisscomB.1.0.3_ETSI_Rx2-SF12 LORA/SwisscomC.1.0.3_ETSI_Rx2-SF12 LORA/SwisscomA.1.0.4_ETSI_Rx2-SF12 LORA/SwisscomB.1.0.4_ETSI_Rx2-SF12 LORA/SwisscomC.1.0.4_ETSI_Rx2-SF12
AppEUI (Mandatory)	F03D29AC71000001 for Swisscom
AppKey (OTAA only, mandatory)	AES-128 application key specific for the device that is assigned by the application owner to the device. Can be generated randomly and is recommended to be unique for each device.
NWkSKey (ABP only, mandatory)	Network Session Key, 128-bit key

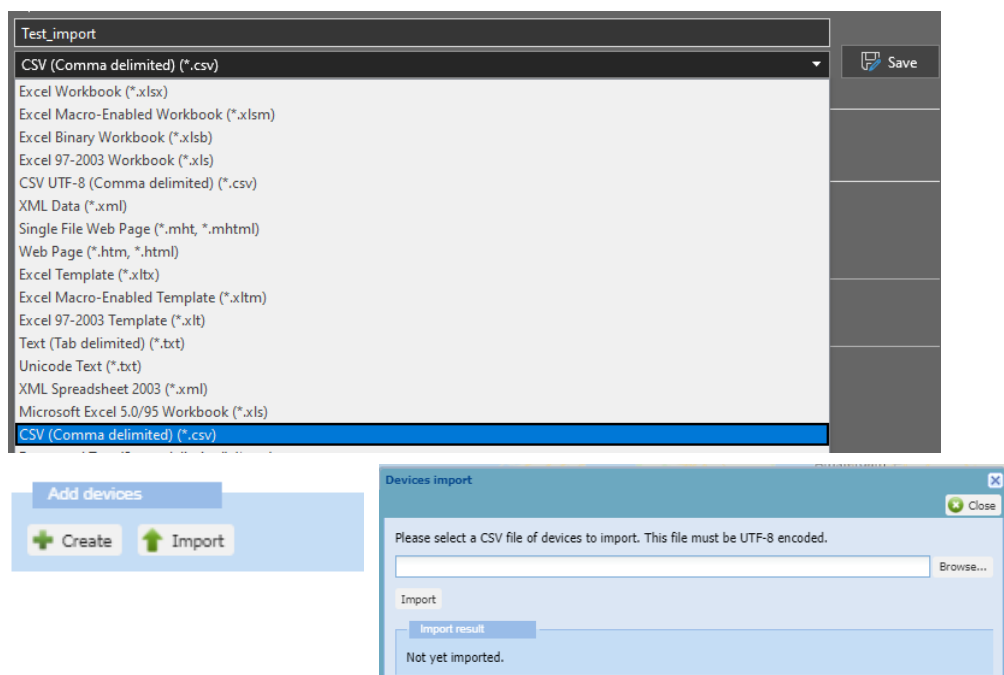
AppSKeys (ABP only, optional) XML encoded application keys, 128-bit key per port:

```
<AppSKeys>
  <AppSKey Port="1">2B7E151628AED2A6ABF7158809CF4F3C
</AppSKey>
  <AppSKey Port="*">2B7E151628AED2A6ABF7158809CF4F3C
</AppSKey>
</AppSKeys>
```

Connectivity plan ID	Name of the connectivity plan, e.g. swisscom-cs/swisscom-cp-nb-trial
AS routing profile ID (optional)	To be found in your device manager, under AS routing profiles. e.g. TWA_100000304.1177 Default will be taken if none specified
Device name (optional)	Free text

	X
CSV for Import	
CREATE_OTAA_7888580310301230_LORAIGenericA.1F03D29AC71000001.3A7E15362DAED2562AF7A5C439CA2F3F_.,swisscom-cs/swisscom-cp-nb-trial,TWA_100000304.1177,.,Device_7888	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	
CREATE_OTAA_.....	

The csv-format is generated in the column "X". Copy the necessary lines into an empty file and save it with a .csv ending. Then go to device manager by hitting the "import" button in "add devices", then upload for your file. Make sure to use the correct encoding as shown below.



6. Wireless Logger

The Wireless Logger is a convenient debugging tool to check the uplink and downlink message sequence of each device. It will show the complete list of LoRaWAN transactions from your account with all its metadata (Timestamp, Signal strength ESP, Gateway IDs, MAC commands etc...) However, it will show neither decoded, nor raw payload. This is sent to your application server only, and not stored on the Swisscom LPN portal.

- > Launch the Wireless Logger from here: <https://portal.lpn.swisscom.ch/wlogger/>
- > The credentials are the same as for the Device Manager.

When you login to the Wireless Logger interface, the interface automatically displays the 50 last messages received from the devices provisioned into your Device Manager.

The screenshot shows the 'WIRELESS-LOGGER' interface with a 'Dashboard [100000807]' header. Below the header are several filter fields: 'DevAddr Filtering', 'DevEUI Filtering', 'LRR Id Filtering', and 'LRC Id Filtering', each with a 'Clear' button. There are also fields for 'From:' and 'To:', a 'Decoder:' dropdown set to 'raw', and a 'Last:' dropdown set to '50'. At the bottom of the filter section are 'Auto Reload:' and 'Expand All:' options, along with 'Refresh', 'Export', and 'Map' buttons.

The main content area is titled '50 last packets' and contains a table with the following columns: UTC Timestamp, Local Timestamp, DevAddr, DevEUI, FPort, FCnt, FCnt #, RSSI, SNR, ESP, SF, and SubBand. The table displays 50 rows of data, each representing a packet. The 'FCnt' and 'FCnt #' columns are highlighted in green, and the 'RSSI' and 'SNR' columns are highlighted in red. The 'SF' column is highlighted in red, and the 'SubBand' column is highlighted in green. The 'DevAddr' and 'DevEUI' columns are highlighted in blue.

	UTC Timestamp	Local Timestamp	DevAddr	DevEUI	FPort	FCnt	FCnt #	RSSI	SNR	ESP	SF	SubBand
↓	2016-10-24 14:16:30.904	2016-10-24 16:16:30.904	04CEB974	0000000036035332	1	12					7	G20
↑ data	2016-10-24 14:16:29.904	2016-10-24 16:16:29.904	04CEB974	0000000036035332	1	13	-27	8.75	-27.5436	7	G20	
↓	2016-10-24 14:16:19.476	2016-10-24 16:16:19.476	04CEB974	0000000036035332	1	11					7	G20
↑ mac	2016-10-24 14:16:18.476	2016-10-24 16:16:18.476	04CEB974	0000000036035332	1	12	-24	9.75	-24.4373	7	G20	
↓ data	2016-10-24 14:16:07.636	2016-10-24 16:16:07.636	04CEB974	0000000036035332	0	10					12	G20
↑ data	2016-10-24 14:16:06.636	2016-10-24 16:16:06.636	04CEB974	0000000036035332	1	11	-36	7.75	-36.674	12	G20	
↓	2016-10-24 14:15:54.690	2016-10-24 16:15:54.690	04CEB974	0000000036035332	1	9					12	G20
↑ data	2016-10-24 14:15:53.690	2016-10-24 16:15:53.690	04CEB974	0000000036035332	1	10	-24	8.25	-24.6056	12	G20	
↓	2016-10-24 14:15:40.512	2016-10-24 16:15:40.512	04CEB974	0000000036035332	1	8					12	G20
↑ data	2016-10-24 14:15:39.512	2016-10-24 16:15:39.512	04CEB974	0000000036035332	1	9	-24	9.5	-24.4618	12	G20	
↓	2016-10-24 14:15:26.925	2016-10-24 16:15:26.925	04CEB974	0000000036035332	1	7					12	G20
↑ data	2016-10-24 14:15:25.925	2016-10-24 16:15:25.925	04CEB974	0000000036035332	1	8	-25	9.25	-25.4877	12	G20	
↓	2016-10-24 14:15:13.376	2016-10-24 16:15:13.376	04CEB974	0000000036035332	1	6					12	G20
↑ mac	2016-10-24 14:15:12.376	2016-10-24 16:15:12.376	04CEB974	0000000036035332	1	7	-23	10	-23.4139	12	G20	
↓ data	2016-10-24 14:14:59.289	2016-10-24 16:14:59.289	04CEB974	0000000036035332	0	5					12	G20
↑ data	2016-10-24 14:14:58.289	2016-10-24 16:14:58.289	04CEB974	0000000036035332	1	6	-30	9.5	-30.4618	12	G20	
↓	2016-10-24 14:14:45.719	2016-10-24 16:14:45.719	04CEB974	0000000036035332	1	4					12	G20
↑ data	2016-10-24 14:14:44.719	2016-10-24 16:14:44.719	04CEB974	0000000036035332	1	5	-27	9.25	-27.4877	12	G20	
↓	2016-10-24 14:14:32.154	2016-10-24 16:14:32.154	04CEB974	0000000036035332	1	3					12	G20
↑ data	2016-10-24 14:14:31.154	2016-10-24 16:14:31.154	04CEB974	0000000036035332	1	4	-24	9.5	-24.4618	12	G20	
↓	2016-10-24 14:14:18.569	2016-10-24 16:14:18.569	04CEB974	0000000036035332	1	2					12	G20
↑ data	2016-10-24 14:14:17.569	2016-10-24 16:14:17.569	04CEB974	0000000036035332	1	3	-24	8	-24.6389	12	G20	
↓	2016-10-24 14:14:04.150	2016-10-24 16:14:04.150	04CEB974	0000000036035332	1	1					12	G20
↑ mac	2016-10-24 14:14:03.150	2016-10-24 16:14:03.150	04CEB974	0000000036035332	1	2	-23	10	-23.4139	12	G20	
↓ data	2016-10-24 14:13:49.755	2016-10-24 16:13:49.755	04CEB974	0000000036035332	0	0					12	G20

The interface contains a search bar and a result window displaying the messages.

6.1. Metadata

The metadata available for each message are:

- > Direction of the data: up or down represented by a directional green or red arrow
- > Type of transmission: data, mac or simply an acknowledge
- > UTC Timestamp
- > Local Timestamp: UTC Timestamp translated to browser timezone
- > Device Address



- > Device EUI
- > Port: Application port of the message
- > Counter UP and Counter DOWN
- > LRR RSSI: RSSI of the received message on LRR side
- > LRR SNR: SNR of the received message on LRR side
- > LRR ESP: ESP of the received message on LRR side
- > SF: Spreading Factor
- > Sub Band: LoRa sub band used for the message
- > Channel: LoRa logical channel used for the message
- > LRC Id: Id of the LRC server
- > LRR Id: LRR with better SNR
- > LRR Lat: LRR latitude
- > LRR Long: LRR longitude
- > LRR Count: number of LRR receiving this message. The system performs a 250ms buffering upon receiving a message to check if the same message arrives through other LRR, in which case LRR Count is incremented. If latency is uneven in the network, a message (with the same Counter Up and payload) may appear more than once in the Wlogger.
- > Device Lat: Device latitude
- > Device Lon: Device longitude
- > LoS Distance (m): distance between the device and the LRR
- > Map: displays the device and LRR on a map
- > Trip: displays the location path of the device (if device location available)
- > MIC: Checksum

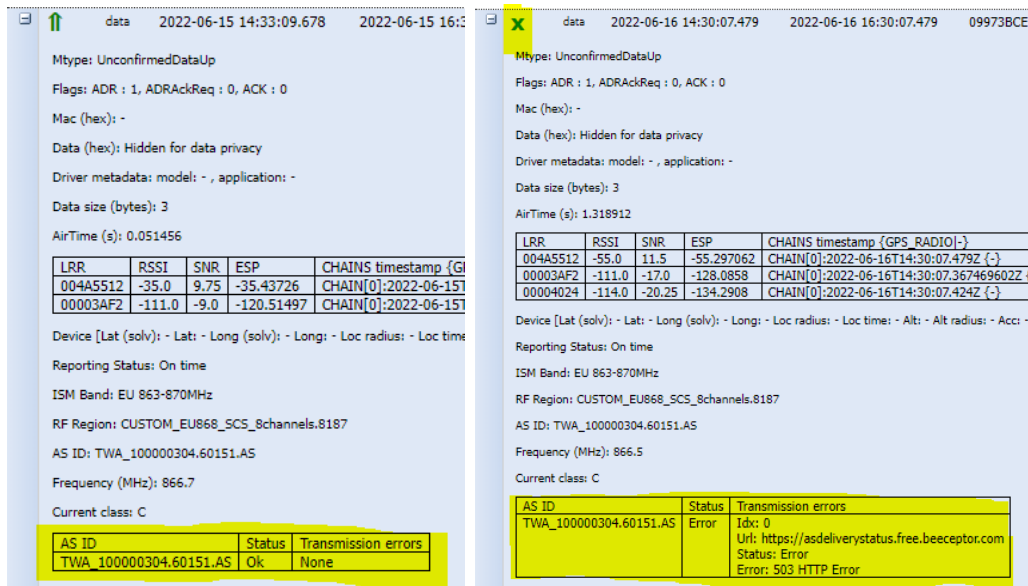


The GPS data (Device lat/lon, LoS, map and trip) are filled only if the device gives its location:

- > Manual location
- > Decoder selected if the location is in the payload

6.2. Expanding a message

Click on the  icon on the left of a message in order to expand and display the message details. You will see the flags, packet size, the list of gateways which have received your signal and Application server delivery information. It is displayed if a message arrived at the Application server in the table at the bottom of every uplink frame. In case an error occurred, it will be visible as a green  icon and more information will be displayed in this field.



Message 1 (Successful):

- data 2022-06-15 14:33:09.678 2022-06-15 16:12
- Mtype: UnconfirmedDataUp
- Flags: ADR : 1, ADRAckReq : 0, ACK : 0
- Mac (hex): -
- Data (hex): Hidden for data privacy
- Driver metadata: model: -, application: -
- Data size (bytes): 3
- AirTime (s): 0.051456

LRR	RSSI	SNR	ESP	CHAINS timestamp {GPS_RADIO[-]}
004A5512	-35.0	9.75	-35.43726	CHAIN[0]:2022-06-15T14:33:09.678Z (-)
00003AF2	-111.0	-9.0	-120.51497	CHAIN[0]:2022-06-15T16:12:00.000Z (-)

Device [Lat (solv): - Lat: - Long (solv): - Long: - Loc radius: - Loc time: - Alt: - Alt radius: - Acc: -]

Reporting Status: On time

ISM Band: EU 863-870MHz

RF Region: CUSTOM_EU868_SCS_8channels.8187

AS ID: TWA_100000304.60151.AS

Frequency (MHz): 866.7

Current class: C

AS ID	Status	Transmission errors
TWA_100000304.60151.AS	Ok	None

Message 2 (Failed):

- data 2022-06-16 14:30:07.479 2022-06-16 16:30:07.479 09973BCE
- Mtype: UnconfirmedDataUp
- Flags: ADR : 1, ADRAckReq : 0, ACK : 0
- Mac (hex): -
- Data (hex): Hidden for data privacy
- Driver metadata: model: -, application: -
- Data size (bytes): 3
- AirTime (s): 1.318912

LRR	RSSI	SNR	ESP	CHAINS timestamp {GPS_RADIO[-]}
004A5512	-55.0	11.5	-55.297062	CHAIN[0]:2022-06-16T14:30:07.479Z (-)
00003AF2	-111.0	-17.0	-128.0858	CHAIN[0]:2022-06-16T14:30:07.367469602Z (-)
00004024	-114.0	-20.25	-134.2908	CHAIN[0]:2022-06-16T14:30:07.424Z (-)

Device [Lat (solv): - Lat: - Long (solv): - Long: - Loc radius: - Loc time: - Alt: - Alt radius: - Acc: -]

Reporting Status: On time

ISM Band: EU 863-870MHz

RF Region: CUSTOM_EU868_SCS_8channels.8187

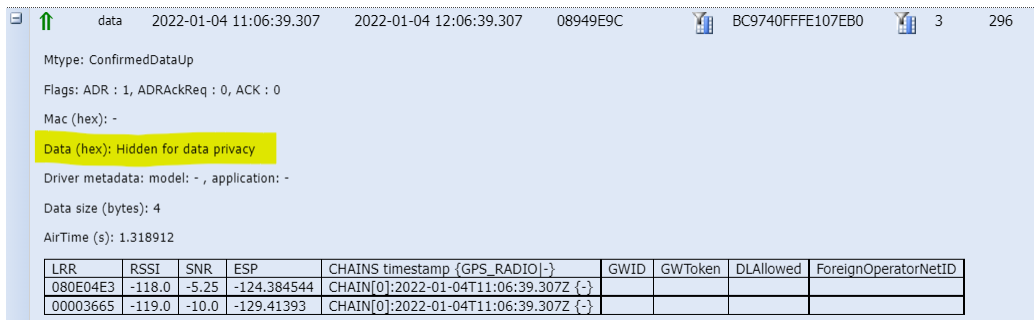
AS ID: TWA_100000304.60151.AS

Frequency (MHz): 866.5

Current class: C

AS ID	Status	Transmission errors
TWA_100000304.60151.AS	Error	Idx: 0 Url: https://asdeliverystatus.free.beeceptor.com Status: Error Error: 503 HTTP Error

Note: Your payload data is not stored on the LPN Network server. It can therefore not be displayed here, please send it to your application server instead.



data 2022-01-04 11:06:39.307 2022-01-04 12:06:39.307 08949E9C BC9740FFFE107E80 3 296

Mtype: ConfirmedDataUp

Flags: ADR : 1, ADRAckReq : 0, ACK : 0

Mac (hex): -

Data (hex): Hidden for data privacy

Driver metadata: model: -, application: -

Data size (bytes): 4

AirTime (s): 1.318912

LRR	RSSI	SNR	ESP	CHAINS timestamp {GPS_RADIO[-]}	GWID	GWToken	DLAllowed	ForeignOperatorNetID
080E04E3	-118.0	-5.25	-124.384544	CHAIN[0]:2022-01-04T11:06:39.307Z (-)				
00003665	-119.0	-10.0	-129.41393	CHAIN[0]:2022-01-04T11:06:39.307Z (-)				

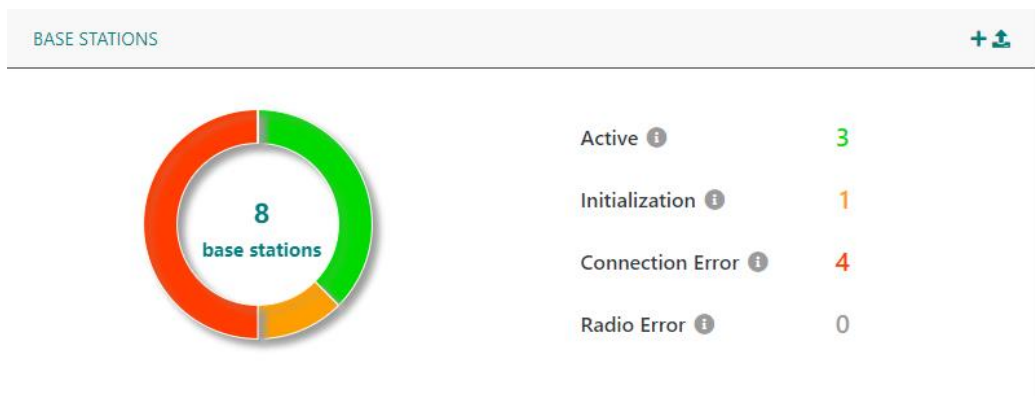
7. Network Manager

The Network Manager is a tool to manage gateways. It shows a complete list of Swisscom LoRaWAN indoor and outdoor gateways of your account, if you choose to manage the gateways on your own. Additionally, Network Manager gives access to metadata such as name, LRR-ID, connection status, traffic history, served devices, etc. of all your gateways.

- > Launch the Network Manager from here:
<https://portal.lpn.swisscom.ch/thingpark/wireless/gui/networkManager/>
- > The credentials are the same as for the Device Manager.

7.1. Dashboard

The dashboard gives a compact status overview of the gateways. It displays how many gateways are connected or initializing and how many gateways are experiencing connection or radio errors and it shows the ten most recent alarms ordered by descending severity. The dashboard allows you to get a direct access to full lists of gateways and all the active gateway alarms of your account.



7.2. Base Station List

The Base Station List provides a list of all gateways of your account. It can be sorted by name or last uplink and filtered across various aspects such as name, LRR ID, health state, software version, tags or alarm state. Clicking on a gateway in this list opens the gateway view where a lot of information regarding the selected gateway is visible. The page shows name, manufacturer, model, LRR ID of the gateway and its status information. The "%" sign can be used in the search field to filter more easily.

Filter base stations

Health state

- Active
- Initialization
- Connection Error
- Radio Error
- Suspended

Software version

Tags

Minimum alarm severity

RF Region

Certificate

7.3. Base Station View

Clicking on a gateway in the Base Station List leads to the gateway view, where the status of a gateway is easily visible. The status overview shows if the gateway is connected and online, if its connected via mobile or ethernet, if the lorawan software is running and the last uplink/downlink timestamp and the amount of uplinks/downlinks in the last 24h. Furthermore, the served devices of a given gateway can be inspected on this page.

STATUS	
Connection	● ACTIVE
Network Connection	📶 NOT ACTIVE 📶 CONNECTED
LoRaWAN™ Radio Status	📶 STARTED
Clock Synchronization	🕒 NTP SYNCHRONIZED
Base Station Restart Time	2023-01-24 14:04:42
Last Uplink ⓘ	Today 14:55:59
Last Downlink ⓘ	Today 14:25:55
Uplink Packets for the last hour	5
Downlink Packets for the last hour	1

7.4. Managing Alarms

Use this information to learn about the gateway alarms that are triggered by the Network Manager application. The gateway alarms are accessible by clicking Alarms in the navigation panel on the left.

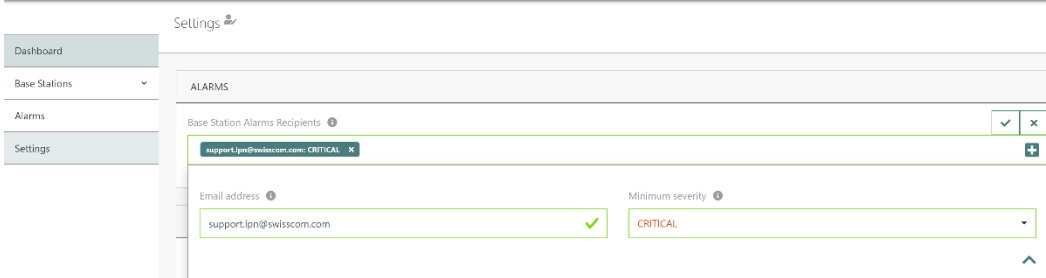
To help you identify and prioritize corrective actions when required, the Network Manager application triggers alarms regarding a gateway when a gateway is not working properly or a gateway is exposed to a replay attack threat. When a gateway is not working properly, the following alarms are triggered automatically: beacon transmission failure, abnormal log activity, base station connection status, downlink frame rate exceeds the RF cell capacity.

An alarm is a visible signal used to indicate that a gateway malfunction, a process deviation or an abnormal condition requires a response. When an error occurs on a gateway, an alarm is triggered in the Network Manager application. The alarm management system qualifies the alarm and assigns a specific state to it. The alarm states are associated with colors and relate to the following severity levels:

Alarm state	Color	Definition
Critical	Red	The service is affected and an immediate corrective action is required
Major	Orange	The service is partly affected and an urgent action is required
Minor	Yellow	A fault that does not affect the service should be corrected to prevent a more serious problem
Warning	Blue	A potential or impending fault affecting the service should be diagnosed and corrected if necessary
Indeterminate	Purple	The severity cannot be determined

Cleared	Green	The alarm has satisfied the clearing condition and has been cleared by the system. Note that the current status of a cleared alarm is available in the active alarms panel and the last status before clearance of the alarm is available in the alarms history panel during 15 days after clearance
---------	-------	--

Through the setting panel on the left, email notifications can be setup to multiple email addresses. We recommend setting up a notification for critical alarms in order to get notified when a gateway goes offline:



For bigger gateway numbers we recommend using DX-API (chapter 13) to monitor gateway status.

8. Swisscom LPN API overview

In the Swisscom LPN environment, the different APIs serve for different purposes. Here is a quick overview of the APIs and additional systems available, and where to find information.

API	Purpose	Rate limit	Documentation
Tunnel API (Uplink)	Receiving data uplinks from devices (REST API)	8 parallel HTTP sessions per Application Server Destination*	10. Tunnel API (Uplink)
Tunnel API (Downlink)	Send data downlink messages to the device	51 requests / s per source IP	11. Tunnel API (Downlink)
DX API	Device lifecycle (create, delete, change price plan...)	10 requests / s per source IP	13. DX API
IoT Flow	Data connectors to third party systems (Azure, AWS...)	3000 msg/minute 50 msg/second	12. IoT Flow network connectors

* The maximum number of requests per second will depend on your application server's response time. This number can be increased if necessary, please contact us.

9. Swisscom LPN Portal Tunnel API

The Swisscom LPN Portal provides an interface for developers of applications combined with wireless sensors or actuators compatible with the LoRaWAN specification:

- > **Tunnel mode interface:** A simple message passing interface between the Swisscom LPN PORTAL servers which implement the network MAC layer (LRC servers), and application servers. This interface forwards the uplink radio packets raw payload data and associated metadata (RSSI, SNR...) to one or more application servers associated to the network node MAC address. As the Swisscom LPN Portal supports bidirectional communications, application servers may also send requests to one of the LRC nodes to send downlink frames to a network node identified by its full format (64bits) MAC address

This document provides information on:

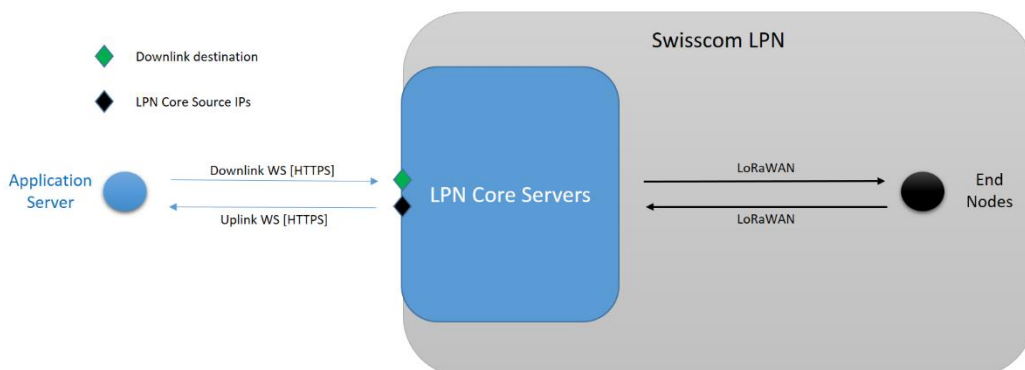
- > Low level LPN core configuration parameters required to associate one or more application servers with a given network node MAC address.
- > Format of LPN core to application server messages that encapsulate uplink payload data and associated metadata
- > The format of application server to LPN core messages that encapsulate downlink payload.

9.1. Connectivity

The tunneling interface is based on HTTPS for uplink & downlink packets flows.

The confidentiality of the uplink/downlink are managed by an HTTPS connection. The uplink HTTPS session is mounted between the LRC cluster and the Application Server. The downlink HTTPS session is mounted between the Application Server and the Reverse HTTP proxy in front of the LRC cluster.

The uplink and downlink packets may be secured by an authentication layer. The LRC will share an



AS key with the Application Server.

The AS key will be used by the Application Server to generate a signature added to downlink packets. This signature will be used by the LPN core to verify the identity and the authorization of the Application Server. If the identity or the authorization cannot be successfully verified, the packet will be dropped.

The AS key will be also used by the LPN core to generate a signature added to uplink packets. The signature will be used by the AS to verify the identity of the LPN core. If the identity cannot be successfully verified, the packet should be dropped by the AS.

- ◆ The downlink destination is as follows:

<https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/downlink/?DevEUI=<DEVEUI>&FPort=<FPORT>&Payload=<PAYLOAD>>

- ◆ The LPN Portal source IPs for uplink messages which should be approved by your ASs firewall are the following:

- > 195.65.47.212
- > 195.65.47.242
- > 194.209.209.249

9.2. Parameters Format

9.2.1 ISO 8601 timestamps

All ISO 8601 timestamps described in this section use the following convention:

YYYY-MM-DDThh:mm:ss.s+|-hh:mm (e.g. 2016-08-01T09:06:06.0+02:00)

Where:

- > YYYY = four-digit year
- > MM = two-digit month
- > DD = two-digit day
- > hh = two digits of hour (00 through 23)
- > mm = two digits of minute (00 through 59)
- > ss = two digits of second (00 through 59)
- > s = one to three digits of millisecond (0 through 999)
- > +|-hh:mm = timezone designator (+hh:mm or -hh:mm)

All timestamp parameters described above are mandatory.

9.2.2 Timestamp Encoding When UL/DL Security Is Activated

When the uplink/downlink security of the Application Server is activated, the timestamp parameter has to be encoded differently in the SHA256 and in the URL of the HTTP request:

Special Characters	In SHA256, use...	In URL, use...
+	+	%2B
-	-	-
:	:	:
.	.	.
Examples	2016-08-01T09:06:06.0+02:00	2016-08-01T09:06:06.0%2B02:00
	2016-11-28T09:06:06.0-04:00	2016-11-28T09:06:06.0-04:00

10. Tunnel API (Uplink)

This section describes the two reports that are generated from the LRC to an Application Server.

10.1. Uplink Frame Report

Uplink destination URLs are defined for each device in the AS profile. The main criteria is the LoRa port numbers expressed as intervals, lists or single values. A default LoRa port destination can be declared. Sequential mode allows packets to be delivered to a list of destinations until one of them confirms receipt (200OK).

Duplicate packets (same counter up and same payload) are not sent to application servers, as long as the multiple copies are received by the network infrastructure within a maximum delay of 250ms (configurable by the LPWA operator). The XML payload sent to the application servers still include the RF metadata corresponding to all receiving base stations.

The uplink frame is transmitted in a HTTP/POST request with query parameters and an XML payload.

Following query parameters are defined for uplink frames (lexicographic order):

AS_ID	Application Server ID (only reported when the authentication is activated) NOTE: AS_ID is not reported when the Uplink/downlink authentication is not activated in the AS profile.
LrnDevEui	Device DevEUI.
LrnFPort	LoRaWAN port number.
LrnInfos	Service profile name used to route the packet.
Time	ISO 8601 timestamp associated to generation of the HTTP request by the LRC (only reported when the authentication is activated) NOTE: Time is not reported when the Uplink/downlink authentication is not activated in the AS profile.
Token	Security token generated by the LRC (only reported when the authentication is activated) NOTE: Token is not reported when the Uplink/downlink authentication is not activated in the AS profile.

Following elements are defined for uplink frames (lexicographic order):

ACKbit	ACKBit set by the device. NOTE: ACKbit is not filled in the XML document if not set in the uplink frame.
ADRbit	ADRBit set by the device. NOTE: ADRbit is not filled in the XML document if not set in the

	uplink frame.
AppSKey	Encrypted AppSkey with the ASkey.
Channel	LC used by the device.
CustomerData	ASCII customer data set by provisioning.
CustomerID	Customer ID associated to the Device Manager account.
DevAddr	Device DevAddr.
DevEUI	Device DevEUI.
DevLrrCnt	Number of LRRs which received this packet.
DriverCfg	Reports the device profile configuration. Should be a swisscom profile if you have registered your device correctly.
DynamicClass	The current LoRaWAN class in which the device is currently set, as devices can change class dynamically.
FCntDn	The counter value to be used for the next downlink frame.
FCntUp	The uplink counter for this packet.
FPort	LoRaWAN FPort used by the device for this packet.
Frequency	The frequency of this LoRaWAN uplink
InstantPER	Instant PER (Packet Error Rate). NOTE: The instance PER is computed on the last 10 packets.
Late	Indicate if the packet was queued by the LRR. NOTE: Late is always filled. 0 means that the packet was not queued by the LRR, 1 means that the packet was queued (the LRR queues packets when the connection between the LRR and the LRC is temporarily broken).
Lrcid	ID of the LRC that processed the packet.
Lrrid	The ID of the LRR that received the packet with the best ESP. This LRR is flagged as "best LRR".
LrrLAT	LAT and LON of the best LRR.
LrrLON	
Lrrs	Can contain up to 10 gateways seen by the device.
LrrRSSI	RSSI measured by the best LRR.
LrrSNR	SNR measured by the best LRR.
Lrrs/Lrr/Lrrid	LRR ID associated to this <Lrr> XML element.
Lrrs/Lrr/LrrESP	ESP measured by the LRR associated to this <Lrr> XML element.
Lrrs/Lrr/LrrRSSI	RSSI measured by the LRR associated to this <Lrr> XML element.
Lrrs/Lrr/LrrSNR	SNR measured by the LRR associated to this <Lrr> XML element.
MeanPER	Mean PER (Packet Error Rate). NOTE: The Mean PER is the average of the instantaneous PER of the last 20 packets.
mic_hex	MIC in hexadecimal ASCII format.
ModelCfg	ASCII ThingPark Cloud data set by provisioning.
MType	LoRaWAN MType of the packet.
NbTrans	Number of transmissions for each uplink frame.
payload_hex	LoRaWAN payload in hexadecimal ASCII format.
SpFact	SF used by the device.

SubBand	SUB-BAND used by the device.
Time	LRR Timestamp for the packet.
TxPower	End-device transmission power in dBm. Reported in ERP for LoRaWAN 1.0/1.0.1 and in EIRP for LoRaWAN 1.0.2 revB and above.

NOTE: The **Lrrs** element is reported for max 10 LRRs which received the packet. If the packet was received by more than 10 LRRs, only the 10 LRRs with the best ESP are reported.

NOTE: When implementing your parser, please be aware that additional fields may be added in future updates and that most of the fields are optional.

10.2. Sample of Uplink Frame HTTP Request

The payload may be provided to the application server either encrypted or decrypted. The following rules apply:

ABP device	Payload is provided decrypted to the AS, if the AppSKey has been provisioned for the relevant FPort. Otherwise the Payload is provided encrypted to the AS.
OTAA device Embedded security server.	Payload is provided decrypted to the AS
OTAA device HSM	The payload and AppSKey are provided encrypted to the AS.

In this sample, `<as-url>` is the destination URL configured in the AS profile:

Note: In an URL, the "+" character must be escaped.

```
>> POST <as-url?LrnDevEui=00000000F1D8693&LrnFPort=2&LrnInfos=UPHTTP_LAB_LORA&AS_ID=appl.s
ample.com&Time=2016-01-
11T14:11:11.333%2B02:00&Token=fd0b0b00464aa798a59282d64eaa70813e33bff87682880db4
9638569d096aad

<?xml version="1.0" encoding="UTF-8"?>
<DevEUI_uplink xmlns="http://uri.actility.com/lora">
  <Time>2020-06-16T07:27:01.618+02:00</Time>
  <DevEUI>0018B2000000CD7</DevEUI>
  <FPort>1</FPort>
  <FCntUp>4</FCntUp>
  <ADRbit>1</ADRbit>
  <MType>4</MType>
  <FCntDn>4</FCntDn>
  <payload_hex>af1c03030e681f07</payload_hex>
  <mic_hex>9fa820cd</mic_hex>
  <Lrcid>00000404</Lrcid>
  <LrrRSSI>-28.000000</LrrRSSI>
  <LrrSNR>10.750000</LrrSNR>
  <SpFact>12</SpFact>
  <SubBand>G0</SubBand>
  <Channel>LC1</Channel>
  <DevLrrCnt>1</DevLrrCnt>
  <Lrrid>004A10D7</Lrrid>
  <Late>0</Late>
  <LrrLAT>0.000000</LrrLAT>
  <LrrLON>0.000000</LrrLON>
  <Lrrs>
    <Lrr>
      <Lrrid>004A10D7</Lrrid>
      <Chain>0</Chain>
      <LrrRSSI>-28.000000</LrrRSSI>
      <LrrSNR>10.750000</LrrSNR>
      <LrrESP>-28.350851</LrrESP>
    </Lrr>
  </Lrrs>
  <CustomerID>100001792</CustomerID>
  <CustomerData>{"alr":{"pro":"ADRF/DEMO","ver":"2"}}</CustomerData>
  <ModelCfg>0</ModelCfg>

<DriverCfg>{"mod":{"pId":"swisscom","mId":"swisscom","ver":"1"},"app":{"pId":"sw
isscom","mId":"swisscom","ver":"1"}}</DriverCfg>
  <InstantPER>0.000000</InstantPER>
  <MeanPER>0.000000</MeanPER>
  <DevAddr>099802E2</DevAddr>
  <TxPower>14.000000</TxPower>
  <NbTrans>1</NbTrans>
  <Frequency>868.1</Frequency>
  <DynamicClass>A</DynamicClass>
</DevEUI_uplink>
```

10.3. LRC Authentication for UL Frame and DL Frame

Securing LRC to AS frame is implemented with the following principles:

- > The LRC adds the AS ID and the generation time stamp in the message.
- > Then, the LRC adds a security token to sign the message based on a pre-shared AS key.
- > When the AS receives a message, the AS will re-compute the security token.
- > If the re-computed security token matches the security token provided by the LRC, and if the time deviation (between the generation by the LRC and the reception by the AS) is acceptable (e.g. less than 10 seconds), the AS can trust the message and process it accordingly.

The AS ID / AS Key are part of the AS profile configuration associated to the device. The generation of the security token by the LRC can be deactivated by not setting an AS_ID and AS key in the AS Profile.

Token must be verified as following by the Application Server:

- > The application server retrieves the `<query-parameters>` WITHOUT the Token QP (Query parameters include the AS_ID and the Time):

For an uplink frame (based on the example provided section 10.1):

e.g. `<query-parameters> := LrnDevEui=00000000F1D8693&LrnFPort=2&LrnInfos=UPHTTP_LAB_LORA&AS_ID=appl.sample.com&Time=2016-01-11T14:11:11.333+02:00`

- > The application server builds the `<body-elements>` as the concatenation, without separator, of the following values:

For an uplink frame (extract from the `<DevEUI_uplink>` body): CustomerID, DevEUI, FPort, FCntUp, payload_hex.

e.g. `<body-elements> := 100000507000000000F1D8693270110027bd00`

- > The application server re-computes the `<token>` as:
SHA-256(`<body-elements><query-parameters><AsKey>`):

For an uplink frame:

e.g. `<token> := SHA-256(100000507000000000F1D8693270110027bd00LrnDevEui=000000000F1D8693&LrnFPort=2&LrnInfos=UPHTTP_LAB_LORA&AS_ID=appl.sample.com&Time=2016-01-11T14:11:11.333+02:0046ab678cd45df4a4e4b375Eacd096acc)`

Where `46ab678cd45df4a4e4b375Eacd096acc` is the 128 bits pre-shared key (lower case hex string representation) between the Application Server and the LRC as defined in the AS profile.

- > The `<token>` is encoded as a hex string AND can be compared to the `<token>` provided by the LRC in the `<query parameters>` line.

For an uplink frame:

e.g. `<encrypted-token> := fd0b0b00464aa798a59282d64eaa70813e33bfff87682880db49638569d096aad`

- > Finally, if the token is valid, the application server can verify the deviation between the emission (as provided in the Time query parameter) and the reception by the application server.

10.4. XML or JSON Encoding

The Uplink frame HTTP request body may be encoded by the LRC as an XML payload or as a JSON payload, which can be configured in Device Manager > Application Server. The default is set to XML. Information elements in the XML document (as defined in 4.2.1 *Uplink frame*) can be mapped one-to-one with information elements in the JSON document.

A 1-to-1 mapping must be assumed between information elements present in the XML document (as defined section 10.1 for uplink frame) and information elements present in the JSON document.

10.4.1 Sample of Uplink JSON Payload

```
{
  "DevEUI_uplink": {
    "Time": "2020-06-08T11:08:41.018+02:00",
    "DevEUI": "0018B2000000CD7",
    "FPort": 1,
    "FCntUp": 4,
    "ACKbit": 1,
    "ADRbit": 1,
    "MType": 4,
    "FCntDn": 4,
    "payload_hex": "af1a03030ea82107",
    "mic_hex": "78137e75",
    "Lrcid": "00000404",
    "LrrRSSI": -28.0,
    "LrrSNR": 10.5,
    "SpFact": 12,
    "SubBand": "G1",
    "Channel": "LC2",
    "DevLrrCnt": 1,
    "Lrrid": "004A10D7",
    "Late": 0,
    "LrrLAT": 0.0,
    "LrrLON": 0.0,
    "Lrrs": {
      "Lrr": [
        {
          "Lrrid": "004A10D7",
          "Chain": 0,
          "LrrRSSI": -28.0,
          "LrrSNR": 10.5,
          "LrrESP": -28.370777
        }
      ]
    },
    "CustomerID": "100001792",
    "CustomerData": {
      "alr": {
        "pro": "ADRF/DEMO",
        "ver": "2"
      }
    },
    "ModelCfg": "0",
    "DriverCfg": {
      "mod": {
        "pId": "swisscom",
        "mId": "swisscom",
        "ver": "1"
      },
      "app": {
        "pId": "swisscom",
        "mId": "swisscom",
        "ver": "1"
      }
    },
    "InstantPER": 0.0,
    "MeanPER": 0.0,
    "DevAddr": "099802E2",
    "TxPower": 14.0,
    "NbTrans": 1,
    "Frequency": 868.3,
    "DynamicClass": "A"
  }
}
```

11. Tunnel API (Downlink)

This section describes how the downlinks are sent from the application server to a device.

URL base path:

V1: <https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/downlink>

V2: <https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/v2/downlink>

Using the v2 interface will provide more detailed HTTP responses, see section 11.1.4

11.1. Downlink Frame

Depending on the device provisioning encryption/decryption can be performed by the LRC. The following rules apply:

Note: When the payload is encrypted, Wireless Logger cannot decrypt it as this application does not embed the associated decryption key.

ABP device	Payload can be provided not encrypted by the AS and will be encrypted by the LRC, if AppSKey has been provisioned for the relevant FPort and if the FCntDn parameter is absent. Otherwise the Payload must be provided encrypted by the AS and the FCntDn parameter must be present.
OTAA device Embedded security server	Payload must be provided not encrypted by the AS and will be encrypted by the LRC. The FCntDn parameter must be absent.
OTAA device HSM	The payload must be provided encrypted by the AS. The FCntDn parameter must be present.

The following HTTP/POST message format is used to tunnel the radio frame payload and associated metadata from the target application server to the LRC. The application server acts as a HTTP client and the reverse HTTP proxy (PROXY_HTTP server) acts as a HTTP server. Rerouting of the HTTP request to the primary LRC or the backup LRC is handled by the reverse HTTP proxy.

The LoRaWAN™ MAC message integrity code (MIC) is always computed by the LRC, as part of the MAC frame formatting. The MAC payload may be encrypted either by the application or by the LRC (see table above).

Such POST command may be generated easily by tools such as curl or POSTman.

```
curl -H "Content-type:application/x-www-form-urlencoded" -X POST
"https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/downlink?DevEUI=00000000F1D8693&FPort=1&Payload=0102030405060708090A&FCntDn=1234"
```

Following query parameters are defined for downlink frame (lexicographic order):

DevEUI (Mandatory)	Target device IEEE EUI64 in hexadecimal form (representing 8 octets)
FPort (Mandatory)	Target port (in decimal format)
Payload (Mandatory)	Hexadecimal payload. The hexadecimal payload will be encrypted by the LRC cluster if FCntDn parameter is absent, and if the LRC has been configured with an AppSKey for the specified LoRaWAN port, otherwise the Payload must be encrypted by the Application Server

according to the LoRaWAN specification and the FCntDn parameter must be present. The Application Server encryption uses the downlink counter, which is why the FCntDn query parameter is required in this case.

FCntDn (Optional)	LoRaWAN Downlink Counter value used to encrypt the payload. This query parameter is needed only if the Application server (not the LRC) encrypts the payload. If present, FCntDn will be copied in the LoRaWAN header field FCnt, and the encrypted payload will be copied as-is to the LoRaWAN downlink frame by the LRC.
Confirmed (Optional)	A value of Confirmed=0 requests transmission of an UNCONFIRMED downlink frame. A value of Confirmed=1 requests transmission of a CONFIRMED downlink frame. Default value Confirmed=0 (UNCONFIRMED).
FlushDownlinkQueue (Optional)	Empties the device AS downlink queue of the device (Boolean). When this parameter is set to FlushDownlinkQueue=1, the AS requests the LRC to purge the AS downlink queue of the device prior to add the downlink payload transported by this HTTP POST.
ValidityTime (Optional)	Associates the AS downlink payload with an expiration date (ISO 8601 timestamp or Duration in seconds) in the device AS downlink queue. If the AS downlink payload has not yet been sent to the device, the AS downlink payload will be discarded by the LRC when the expiration date is reached.
AS_ID (Optional)	Application Server ID, as provisioned in the AS Profile. The Application server ID is mandatory if the Application server authentication has been activated in the AS Profile. In this case the LRC will check that the Application Server is authorized to send downlink command to the device.
Time (Optional)	ISO 8601 time of the request. The Time is mandatory when the Application server authentication has been activated in the AS Profile. In this case the LRC will verify the time deviation between the generation and the reception of the request. The deviation must be lower than "Max Time Deviation" as defined in the AS Profile. Note: In the URL of the HTTP request, use "%2B" ASCII code for the "+" character.
Token (Optional)	Security token to sign the downlink frame. The Token is mandatory when the Application server authentication has been activated in the AS Profile.

11.1.1 Sample of Downlink Frame HTTP Request

Sample with v1:

```
>> POST
https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/downlink?DevEUI=00000000F1D86
93&FPort=1&Payload=00&AS_ID=appl.sample.com&Time=2016-01-
11T14:28:00.333%2B02:00&Token=ea8f31d2299cbece8e180a3012766c4df15fe3cf2e142d9fdf
4035b5894ec886
```

Sample with v2:

```
>> POST
https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/v2/downlink?DevEUI=00000000F1
D8693&FPort=1&Payload=00&AS_ID=appl.sample.com&Time=2016-01-
11T14:28:00.333%2B02:00&Token=ea8f31d2299cbece8e180a3012766c4df15fe3cf2e142d9fdf
4035b5894ec886
```

Confirmed frames are declared as a message in the LPN Portal, therefore a confirmation is subject to deduction of your up-/downlink budget.

11.1.2 Confirmed Downlink

Unconfirmed Downlink messages are not acknowledged at LoRaWAN level and therefore the network, and the tunnel mode Application server does not know whether they have been received or not.

Confirmed Downlink messages are acknowledged by the target device, but LoRaWAN section 4.3.1.2 “Message acknowledge bit and acknowledgement procedure (ACK in FCtrl)” lets the device free of sending delayed ACKs. Therefore, it is not possible to let the network manage retransmissions.

When the LRC receives a possibly empty (no payload) uplink message with ACK set in the FCtrl field, the LRC will add an “ACKbit” flag in the XML/JSON metadata of the uplink frame sent to the Application server. The retransmit policy is up to the application server.

11.1.3 LRC HTTP Response Codes (v1):

- > **200 “Request queued by LRC”**: request accepted and queued until the class A device opens Rx slots by sending an uplink. In the case of a class C device, the downlink command will be sent as soon as the LRR base station radio is available and the maximum regulatory Tx duty cycle allows transmission.
- > **350 “Invalid DevEUI”**: An Invalid DevEUI was entered.
- > **350 “Downlink counter value already used. Expected=1238”**: the downlink counter value was already used, for instance due to a race condition with another Application server.
- > **350 “Downlink counter value increment too large. Expected=1001”**: the AS supplied downlink counter value is much larger than the expected downlink counter value and was rejected by the LRC.
- > **350 “Confirmed downlink is not authorized for this device”**: the request for transmission of a confirmed downlink packet was rejected by the LRC due to absence of “ackedDownlinkFrame” feature flag in the Connectivity plan associated to the device.
- > **350 “Invalid LoRa port 0”**: sending on port 0 (port reserved for LoRaWan MAC commands) is unauthorized from the tunneling interface.
- > **350 “Security Check. AS_ID is mandatory”**: speaking to this device needs AS_ID. The Application Server authorization has been activated for this device and the application must be identified.
- > **350 “Security Check. missing timestamp/token”**: Time and Token query parameter are mandatory when application server authentication is activated.
- > **350 “Security Check. bad AS_ID”**: AS_ID is not declared in the database or is not authorized for the targeted device.
- > **350 “Security Check. Server Decrypt Error”**: Missing or badly formatted security token.
- > **350 “Security Check. malformed ISO8601 time”**: An ISO 8601 date/time must be used (YYYY-MM-DDThh:mm:ss.s+|-hh:mm) representing a local time with a time zone offset in hours and minutes.

- > **350 "Security Check. Invalid downlink frame timestamp"**: the time deviation between the frame generation by the application server and the reception by the LRC exceeds the MAX deviation configured in AS profile.
- > **350 "Security Check. bad token"**: Token was not accepted by the LRC
- > **350 "ValidityTime expired or invalid"**: The date or duration in the ValidityTime attribute is wrongly formatted or is invalid (for instance, date in the past).
- > **350 "Payload too big or invalid size"**: The payload size is greater than 5000 characters or is not a multiple of 2.
- > **350 "Downlink transmission disabled"**: The request for transmission of the downlink packet was rejected by the LRC due to the absence of the downlinkTransmission feature flag in the Connectivity plan associated to the device.
- > **404 "No Base Station Available"**: The request for transmission of the downlink packet was rejected by the LRC because no base station near to the device is connected.
- > **350 "Invalid payload size according to current DR"**: The payload is too big to be transmitted according to the data rate currently used by the device.
- > **503 "service temp unavailable"**: The service is temporarily unavailable due to throttling. Try again at a later point.

11.1.4 LRC HTTP Response Codes (v2):

HTTP response when the downlink frame is accepted:

- > HTTP Response status code: 202
- > HTTP Response body: empty.

HTTP response when the downlink frame is rejected:

- > HTTP Response status code: 409
- > HTTP Response body: JSON document

```
{
  "code": <code>,
  "message": "<message>"[,
  "xxx": <additional info 1>][,
  "yyy": <additional info 2>]
  ...
}
```

- > The code is mandatory (see table below) and is an immutable unsigned integer.
- > The message is mandatory (see table below) and is an ASCII string.
- > On an error basis, additional information elements (xxx, yyy...) may be added to the JSON document (see table below).
- > The table below gives the description of the error codes

Error code	Error message	Additional info 1	Additional info 2
104	Invalid LoRa port	-	-
105	Security Check: AS_ID is mandatory	-	-
106	Security Check: Missing timestamp	-	-
107	Security Check: Bad AS_ID	-	-
108	Security Check: Server	-	-

	Decrypt Error		
109	Security Check: Malformed ISO 8601 time	-	-
110	Security Check: Invalid downlink frame timestamp	-	-
111	Security Check: Bad token	-	-
112	Validity time expired or invalid	-	-
113	Payload too big or invalid size	Attribute: maxSize Type: Integer Description: When set this means that the max payload size was reached. When not set this means that the payload size is invalid (must be a multiple of 2).	-
114	Downlink transmission disabled	-	-
115	No Base Station Available	-	-
116	Invalid payload size according to currents DR	Attribute: rx2DR or pingSlotDR Type: Integer Description: MAX Data Rate associated to RX2 (class A/C) or PingSlot (class B). The current RX1 DR is not checked.	Attribute: maxSize Type: Integer Description: Max payload size according to RX2 (class A/C) or PingSlot (class B). The current RX1 DR is not checked.
117	Invalid Correlation ID. Must be a 64 bits hexadecimal value encoded as string	-	-
118	Invalid payload. Must not be empty	-	-
119	Payload must be provided encrypted with the downlink counter value	-	-

Note: regarding queuing of several messages: The Swisscom LPN Portal Wireless network may queue up to 5 messages per device. The network uses the FPending flag defined in the LoRaWAN protocol to signal to the device that additional messages are queued. Messages will be sent, one at a time, in the receive window following the next uplink message received from the device. A downlink has no timeout in the queue unless it is forced by the "FlushDownlinkQueue" query parameter.

11.2. Downlink Multicast

At Phy level Multicast support in LoRaWAN class C amounts to having multiple End devices listen to the same network address. This is already supported as part of class C support in the Swisscom LPN Portal, an Application server just needs to send an unconfirmed downlink message to the target group address (configured as dummy device).

However, the LoRaWAN™ roadmap includes future work on multicast, including group membership and key management procedures, as well as large payload fragmentation transmission e.g. for firmware updates.

11.3. Application Server Authentication for Downlink Frame

Securing downlink frame is implemented with the following principles.

The AS must not be able to send downlink POST if:

- > The AS is not in possession of AS key.
- > The AS has not been authorized to send downlink packet to the device
- > The time between the generation of the request by the AS and the reception of the request by the LRC is too high.

The AS ID / AS Key and max time deviation are part of the AS profile configuration associated to the device. The Application Server authentication can be deactivated by not setting an AS_ID and AS key in the AS Profile.

Token must be computed as follows by the Application Server:

- > The downlink message <query-parameters> (Query parameters must include the AS_ID and the Time query parameters) are constructed WITHOUT the Token:
e.g. <query-parameters> :=
DevEUI=000000000F1D8693&FPort=1&Payload=00&AS_ID=appl.sample.com
&Time=2016-01-11T14:28:00.333+02:00
- > The <token> is computed as SHA-256(<query-parameters><Askey>)
e.g. <token> :=
SHA-256 (DevEUI=000000000F1D8693&FPort=1&Payload=00&AS_ID=appl.sample.com&Time=2016-01-11T14:28:00.333+02:0046ab678cd45df4a4e4b375Eacd096acc)
where 46ab678cd45df4a4e4b375Eacd096acc is the 128 bits pre-shared key (lower case hex string representation) between the Application Server and the LRC as defined in the AS profile.
- > The <token> is encoded as an hex string (e.g.
ea8f31d2299cbece8e180a3012766c4df15fe3cf2e142d9fdf4035b5894ec886)
AND added at the end of the query parameters line
e.g.
<https://proxy1.lpn.swisscom.ch/thingpark/lrc/rest/downlink?De>

vEUI=000000000F1D8693&FPort=1&Payload=00&AS_ID=appl.sample.com&Time=2016-01-11T14:28:00.333%2B02:00&Token=ea8f31d2299cbece8e180a3012766c4df15fe3cf2e142d9fdf4035b5894ec886

where: 2016-01-11T14:28:00.333%2B02:00 contains the "%2B" ASCII code for the "+" character where as the ":" character has to be sent unescaped.

12. IoT Flow network connectors

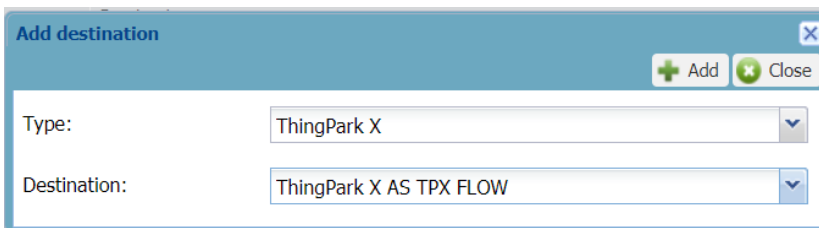
It is possible to route directly to third party services like Microsoft Azure, Amazon Web Services or to your MQTT broker. You will need a connectivity plan that includes the IoT Flow feature. To learn more about connectivity plans and features, please [visit chapter 4](#) of this guide.

Currently supported

- HTTPS (REST API): Available as standard connector, without IoT Flow. [Visit chapter 3.3](#)
- Microsoft Azure IoT Hub / Event Hubs
- Amazon Web Services (AWS) IoT Core
- MQTT over SSL, WSS or TCP (you need your own MQTT broker)

12.1. Set up the routing

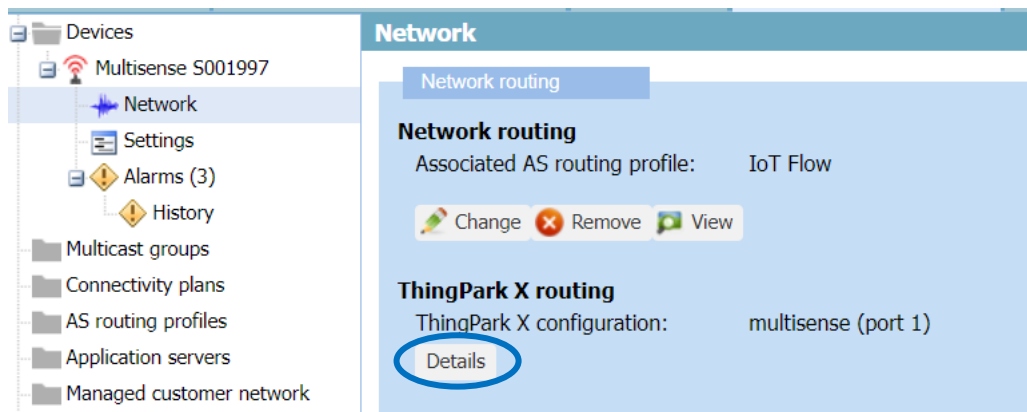
In order to route your traffic to the IoT Flow connector, you only need to set up an Application Server Routing Profile, described in chapter 3.4 of this guide. There is no need to create an Application Server to use IoT Flow, you can directly route your traffic from the AS Routing Profile: Choose **"ThingPark X"** as type and **"ThingPark X AS TPX FLOW"** as destination.



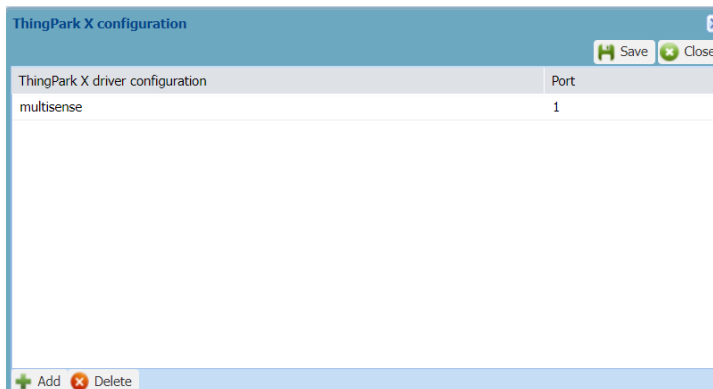
Settings to add the IoT Flow connector

12.2. Add tags to your devices

If you want to route all of your traffic to the same place, you don't need to follow this step. However, if you have different destinations for your devices or if you want to decode payloads of different manufacturers, you can use tags on your devices.



Go to your device in edit mode -> Network -> Network routing -> ThingPark X routing.

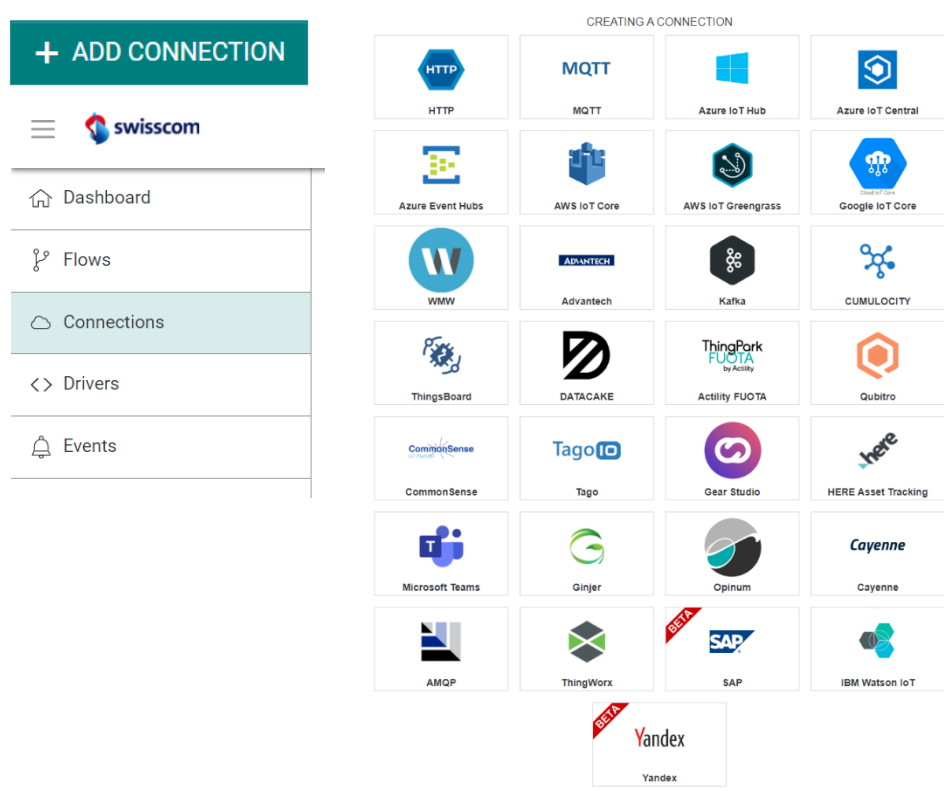


Now, add your tag by clicking on "Add". The port functionality is deprecated and does not matter, just use a positive integer value.

12.3. Set up your connection

After you have routed your traffic to the IoT Flow service, please log in to the IoT Flow GUI:

<https://portal.lpn.swisscom.ch/tpx/login>



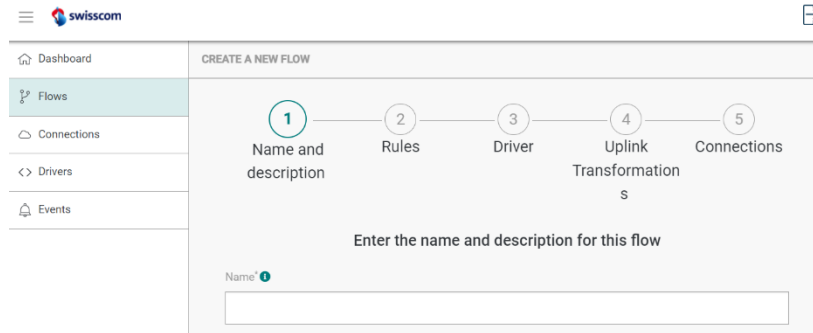
Click on "Add Connection", choose your service and enter your credentials depending on your cloud service to connect to.



Warning: Do not use connectors marked "BETA" in production! These are only a preview to be used for tests. Changes to the configuration and connector restarts are reserved without notification! Please also note, that the Google Clouds IoT Core will be shut down in 2023.

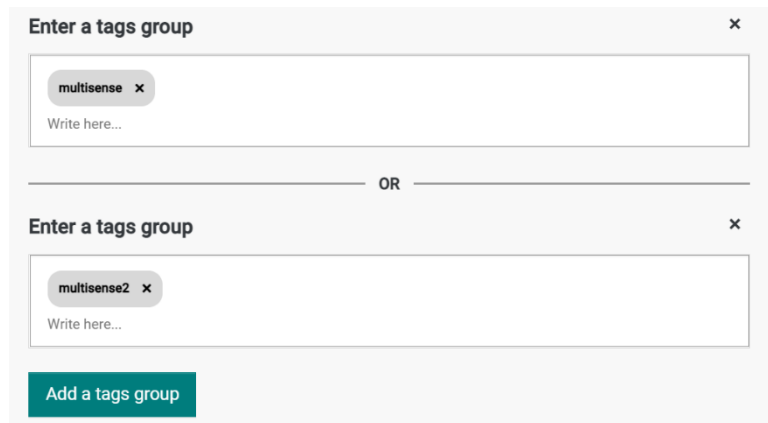
12.4. Set up your Flow

Click on "Flow" on the left hand menu and "Create new Flow".



You will be guided through the following steps:

- > **Name and description:** Give your flow a name
- > **Rules:** Choose "Keys" to match this flow on specific DevEUIs. Choose "Tags" in order to use the tags which you have set up previously in section 12.2. Please mind that only devices **that match all of the tags in the group** will be routed to this flow. You can create "OR" rules by adding another tags group.
Note: Tags are optional, you can set none and all the traffic will be routed to this flow.



- > **Driver:** Choose "Force driver" and choose your device's driver from the list. You can create your own driver following section 12.5. Please note that automatic driver functionality is only supported for Swisscom Multisense devices at this moment.
Note: Message decoding on IoT Flow is optional. If you prefer not to activate any driver, choose "Automatic". The "Swisscom" profiles used on our platform do not have any drivers associated to them. Only Swisscom Multisense will be decoded with configuration on "automatic".
- > **Uplink Transformations:** This step is not mandatory
- > **Connections:** Choose at least one connection where you would like your data to be sent.

After having set both a connection and a flow, IoT Flow is ready to route your data. You can monitor the state of your connections in the "Connections" menu. You will see a list of open and closed connections, as well as the number of routed messages.

Last Restart	Active Devices (last 1h/24h)	Uplinks (last 1h/24h)	Downlinks (last 1h/24h)	State
5 days ago	1 / 1	1 / 24	0 / 0	OPENED ▼
-	0 / 0	0 / 0	0 / 0	CLOSED ▼

12.5. Create custom drivers

In the "Drivers" menu point, you can create and test your own Javascript driver. A custom driver will be created, that is only visible inside your organization. Please follow the Github links and further documentation available in the portal. Please let us know if you want to make your driver public and we will assist you. This enables other users to use the same driver.

12.6. Queueing behavior

The default queueing time "uplinkTimeValidity" is set to 72h. This means that, in case of a connection failure or server downtime, the messages will be stored in a kafka topic for 72h and IoT Flow will attempt to re-deliver the packets. Messages older than "uplinkTimeValidity" will be discarded. The queuing behavior can be adapted if necessary.

12.7. Sample of Uplink JSON with decoded payload

Below listed is an uplink JSON with a driver selected. Therefore, it additionally includes the decoded payload according to the selected driver. If the flow does not decode the payload, there is no entry for payload, only for the plain payload_hex.

```
{
  "DevEUI_uplink" : {
    "Time": "2020-01-06T12:46:54.285+01:00",
    "DevEUI": "20635F0108000E09",
    "FPort": 17,
    "FCntUp": 265,

```



```

"ADRbit": 1,
"MType": 2,
"FCntDn": 9,
"payload_hex": "0520be8800400108c9000000",
"mic_hex": "3bcf2927",
"Lrcid": "00000127",
"LrrRSSI": -65.0,
"LrrSNR": 9.5,
"SpFact": 7,
"SubBand": "G1",
"Channel": "LC3",
"DevLrrCnt": 2,
"Lrrid": "08050376",
"Late": 0,
"LrrLAT": 43.615501,
"LrrLON": 7.066182,
"Lrrs": {
  "Lrr": [
    {
      "Lrrid": "08050376",
      "Chain": 0,
      "LrrRSSI": -65.0,
      "LrrSNR": 9.5,
      "LrrESP": -65.461838
    }
  ]
},
"CustomerID": "1000xxxx",
"CustomerData": {
  "alr": {
    "pro": "ABEE/APY",
    "ver": "1"
  }
},
"DriverCfg": {
  "id": "abeeway:asset-tracker:1"
},
"payload": {
  "messageType": "HEARTBEAT",
  "mode": "MOTION_TRACKING",
  "batteryVoltage": 3.85,
  "ackToken": 0,
  "firmwareVersion": "1.8.201",
  "bleFwVersion": "0.0.0",
  "resetCause": 40,
  "periodicPosition": false,
  "temperature": 24.8,
  "userAction": 0,
  "appState": 0,
  "moving": false,
  "onDemand": false,
  "payload": "0520be8800400108c9000000",
  "InstantPER": 0.0,
  "MeanPER": 0.0,
  "DevAddr": "055E1C4E",
  "AckRequested": 0,
  "rawMacCommands": "",
  "TxPower": 2.0,
  "NbTrans": 1,
  "Frequency": 867.9,
  "DynamicClass": "A"
}
}
}

```

12.8. Uplink Transformation

If you prefer to use a custom JSON format rather than the standard format, you can make use of the uplink transformation. This allows to easily transform the uplink such that it can be directly integrated to for example chirpstack. The uplink transformation is done on the connection. You can also define a custom uplink via JSLT. On "edit transformation" the changes to the JSON are defined for each tag in the JSON.

The image shows two screenshots of the 'CREATE NEW OPERATION' dialog, illustrating the steps to select an uplink transformation type.

Left Screenshot:

- Step 1:** Select a transformation type. Three options are shown: Filter Messages, Standard Transformation Processor, and Transform Custom Uplink.
- Step 2:** Select a standard transformation type. Four options are shown: ChirpStack, Message Flattener, Objenious, and Abeeway LocationSolver.
- Buttons:** CANCEL and ADD.

Right Screenshot:

- Step 1:** Select a transformation type. Three options are shown: Filter Messages, Standard Transformation Processor, and Transform Custom Uplink.
- Step 2:** Select a custom transformation type. Three options are shown: JMESPath, JSLT (Recommended), and JSONATA.
- Step 3:** Modify transformation output. A button labeled EDIT TRANSFORMATION is visible.

12.9. Sample of Downlink JSON payload

For any connector, the following scheme must be used by IoT cloud platforms for downlink messages to IoT Flow.

```
{
  "DevEUI_downlink": {
    "Time": "2019-07-10T15:38:46.882+02:00",
    "DevEUI": "0018B2000000B20",
    "FPort": 1,
    "payload_hex": "9e1c4852512000220020e3831071"
  }
}
```

67

This downlink message also contains optional fields.

```
{
  "DevEUI_downlink": {
    "Time": "2019-07-10T15:38:46.882+02:00",
    "DevEUI": "0018B2000000B20",
    "FPort": 1,
    "AS_ID": "TWA 199983788.1972.AS",
    "AS_KEY": "9311e22d7d44fc52215b0dc154aald22",
    "payload": {
      "DownMessageType": "SET_PARAMETER",
      "ParameterName": "TRANSMIT_STRAT",
      "TransmitStrat": "DOUBLE_FIXED",
      "AckToken": 1
    },
    "Confirmed": "1",
    "ValidityTime": "2019-07-10T16:38:46.882+02:00",
    "FlushDownlinkQueue": "1",
    "DriverCfg": {
      "app": {
        "pId": "abeeway",
        "mId": "asset-tracker",
        "ver": "1"
      }
    }
  }
}
```

13. DX API

The purpose of the DX API is to provide the best developer experience for all developers who intend to interact with the LPN Portal for device lifecycle management.

Every call to the DX API requires three standard HTTP headers: Content-Type, Accept and Authorization. The API supports content types application/json and application/xml for requests and responses. Usage of the Authorization header is detailed in chapter 13.1.

You will find detailed documentation including examples from our platform supplier Activity on the following link.

Documentation: <https://portal.lpn.swisscom.ch/thingpark/dx/core/latest/doc/index.html>

Please note that for the Swisscom DX API you will only have the rights to use the Device Operations, all other operations can be ignored.

You can also download the Swagger contract (<https://portal.lpn.swisscom.ch/thingpark/dx/core/latest/tpdx-core-api-contract.yaml>) of the DX API, or start using the API right away with the Swagger UI (<https://portal.lpn.swisscom.ch/thingpark/dx/core/latest/swagger-ui/index.html?shortUrl=tpdx-core-api-contract.json>). For other tools such as client SDK generators you can also check the Swagger homepage (<http://swagger.io/>).

13.1. Authentication

The DX API relies on an OAuth2 authorization workflow: in order to use the API, one must first get an API Key (also called access token), providing access to specific parts of the API, until it expires.

A new API Key can be obtained using the DX Admin API (<https://portal.lpn.swisscom.ch/thingpark/dx/admin/latest/swagger-ui/index.html?shortUrl=tpdx-admin-api-contract.json>), by providing the valid existing LPN Portal credentials. For more convenience, the Swagger UI can also be provided the login credentials on Core API as described in chapter 13.1.2.

While it is valid, an API Key is associated with a scope. A scope is a group of permissions (creation, update, etc.) over a set of resources, granted by the LPN SUBSCRIBER role. As an LPN customer you will only be able to use the Device operations described under "Device operations" in the online DX API documentation.

The generated API Key should be set in the Authorization HTTP header of every request as a Bearer token, e.g.: Authorization: Bearer <access_token>.



Note: To avoid token issues, please use a separate user for DX-API rather than an existing GUI-user.

13.1.1 DX Admin API

The Admin API is used to get the Bearer Token to authenticate for API calls. Follow the instructions of the page and use a valid `client_id` and `client_secret` from Device Manager.

POST /`oauth/token` Token generation

Generates and retrieves a token for a client.

Parameters Cancel Reset

Name	Description
<code>renewToken</code> boolean (query)	Forces the token to be renewed. If false, and a token already exists for the client, it will be reused. If true, a new token is always returned. Default is false.
<code>validityPeriod</code> string (query)	Validity of the new token. Possible values are '5minutes', '12hours', '7days', '90days' or 'infinite' (never expires, until revocation). Default is '7days'. Note that in order to properly use an 'infinite' token, the user password MUST NOT be updated at any time (thus requiring an API-Only user account or a non-expiring-password security policy).

Request body application/x-www-form-urlencoded

grant_type * required
string
Type of the OAuth2 grant workflow. Its value should always be 'client_credentials', which is the only workflow currently supported.

`client_credentials`

client_id * required
string
Email used for the ThingPark login.

`lpndev1+test@gmail.com`

client_secret * required
string(\$password)
Secret of the client. It's value should be the password for the ThingPark login specified in the 'client_id' parameter.

.....
















Execute Clear



Please note that a user's password will expire after 4 months, unless you create an user with the flag **permanent password** (user creation action see 13.1.2 Core API)+

13.1.2 DX Core API

Is used for everything else e.g. creation of end users, device (add, delete, change connectivity plan, etc.), base station operations (get, rename) or to retrieve alarms. It is also possible to send downlinks for testing purposes. To authorize, the Authorize button on top can be used to log in with a Thingpark account (via Admin API) or to enter the bearer token, incase you have it already.

Device		Device operations provide the ability to manage ThingPark devices, device profiles, connectivity plans.		Authorize 
GET	/devices	Devices retrieval	✓	
POST	/devices	Device creation	✓	
GET	/devices/{deviceRef}	Device retrieval	✓	
PUT	/devices/{deviceRef}	Device update	✓	
DELETE	/devices/{deviceRef}	Device deletion	✓	
GET	/deviceProfiles	Device profiles retrieval	✓	
GET	/connectivityPlans	Connectivity plans retrieval	✓	
GET	/routingProfiles	Routing profiles retrieval	✓	
POST	/routingProfiles	Routing profiles creation	✓	
GET	/routingProfiles/{routingProfileRef}	Routing profile retrieval	✓	
PUT	/routingProfiles/{routingProfileRef}	Routing profile update	✓	
DELETE	/routingProfiles/{routingProfileRef}	Routing profile deletion	✓	
GET	/deviceFrameStatistics	Frame statistics retrieval	✓	
GET	/deviceHealthStatistics	Health statistics retrieval	✓	



Please note that sending Downlinks via DX API is only for testing purposes. The preferred way for sending downlinks is using the Tunnel API described in chapter 9. Tunnel API is the direct option and data won't leave Switzerland.

13.1.3 DX API versioning

Please make sure to use the **/latest** endpoint. With every update on the API, all API users will be notified at least 2 weeks in advance about the change, and the new features shall be backward-compatible. The new API will be available for testing on the **/new** endpoint. For the case you have troubles with your integration after the update, you can use the **/old** endpoint while troubleshooting your integration for at least 6 months after the removal from **/new**.

Example:

```
https://portal.lpn.swisscom.ch/thingpark/dx/core/latest/api/devices
```

13.1.4 DX API error codes

In case of a problem, you might get an answer containing an OSS-xxx error code. The http error is always Error 400, and the specific error code can be seen in the body.

Example:

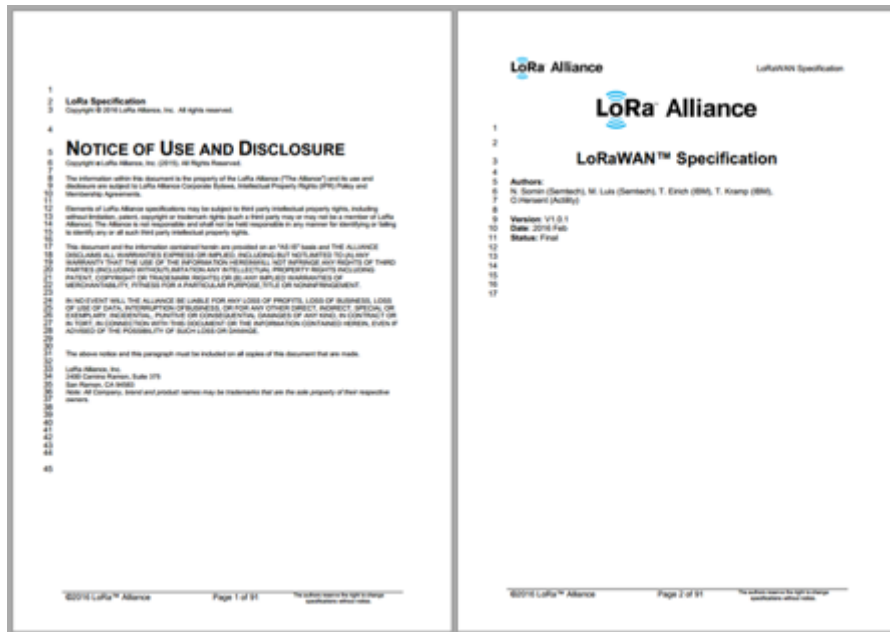
```
{"code":400,"message":"Bad request: The operation is still in progress and has been switched to asynchronous mode. It should end in a few moments.","errorId":"OSS-522"}
```

The complete list of error codes is available as an attachment to this document, please request it from Support.LPN@swisscom.com.

14. LoRaWAN specification

The LoRaWAN specification is publicly available from the LoRa™ alliance web site: lora-alliance.org

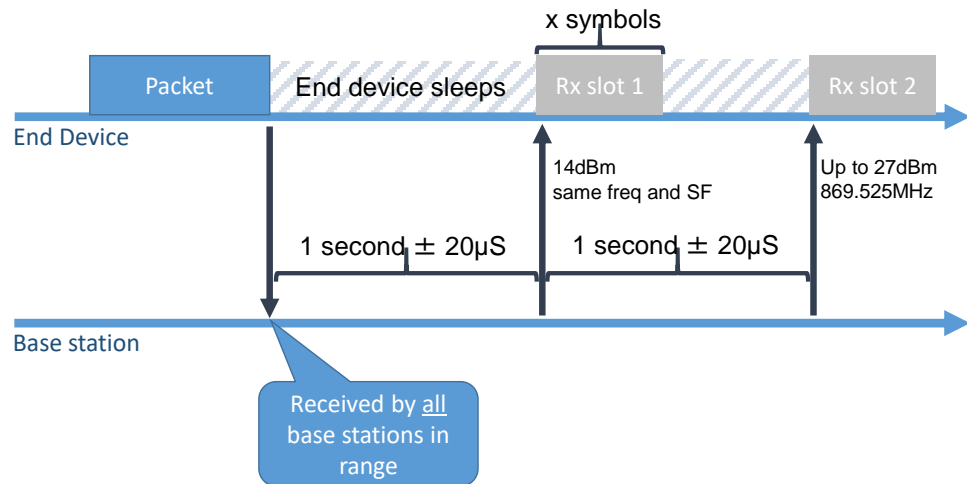
The LoRaWAN specification enables bidirectional communications, and currently offers three



variants of the MAC layer: class A, optimized for battery usage and sensors; class B, optimized for battery powered actuators with low command latency requirements, and class C for mains powered devices including actuators.

Class name	Intended usage
A (« all »)	Battery powered sensors, or actuators with no latency constraint Most energy efficient communication class. Must be supported by all devices
B (« beacon »)	Battery powered actuators Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon.
C (« continuous »)	Mains powered actuators Devices which can afford to listen continuously. No latency for downlink communication.

LoRaWAN class A employs the well-known receiver initiated transmit strategy to enable communication bidirectionality.



The End-device, e.g. a temperature sensor, wakes-up to transmit its measurement. The LoRaWAN radio frame will be received by all nearby base stations of the RF network. The device then immediately goes to sleep for a specified amount of time, by default 1 second, in order to preserve the battery.

After the exact sleep time, the End-device must wake up to receive potential downlink communication from the network. The downlink communication may be an ACK from the network if the End-device had sent a confirmed LoRaWAN frame, or it may be a command from the MAC layer network controller or from an application server.

The Core network will use, at its choice, this first receive window (RX1), or the second receive window (RX2) to send downlink frames. So if no frame has been received during the RX1 slot, the device must go to sleep again and wake up another time for the RX2 slot.

14.1. Globally unique EUI-64: DevEUI

Each LoRaWAN end device has a **globally unique** IEEE EUI-64 address, the DevEUI. These addresses are allocated by manufacturers within address blocks that must be purchased from IEEE, three blocks are available:

- > Organizational Unique Identifier (OUI) / MAC Address Block Large (MA-L)
- > MAC Address Block Medium (MA-M)
- > MAC Address Block Small (MA-S)

These 3 different blocks of addresses can be purchased from IEEE here:

- > <http://standards.ieee.org/develop/regauth/oui/>
- > <http://standards.ieee.org/develop/regauth/oui28/index.html>
- > <http://standards.ieee.org/develop/regauth/oui36/index.html>

14.2. Over-The-Air Activation (OTAA)

OTAA device derives its NwkSKey and AppSKey using the Join key negotiation procedure as they first attach to a network. This procedure uses a master AppKey secret that must be personalized at production in the device.

A single application server, identified by its AppEUI (JoinEUI), is supported per device.

OTAA	Who?	What is it?
DevEUI	IEEE/Device manufacturer	The DevEUI identifies the device on the LoRaWAN network during the JOIN request
JoinEUI (former AppEUI)	Operator	The AppEUI identifies the join server during the JOIN request.
AppKey	Device manufacturer	The AppKey encrypts the data during the JOIN request

14.2.1 128 bit application key: AppKey

The 128 bit AppKey **must be personalized in each device** during production. It may be distinct per device or unique per application depending on the use-case.

LoRaWAN uses symmetric encryption, which means that the same key needs to be provisioned both on the device and on the network server. The 128 bit AppKey is never transmitted on the air.

Typically, the device manufacturer provisions a unique AppKey into each device and transmits it to you in a secure way, so that you can provision the correct key for each device in the network server, using either the GUI, the API or CSV upload.

14.2.2 64 bit application server identifier: JoinEUI (former AppEUI)

The JoinEUI, previously also called AppEUI, is a globally unique identifier of the target application server that will process all exchanges with the device. It will have an increasing significance when it comes to roaming between different operators.

The network forwards the join message to the application server identified by the AppEUI. This application server is supposed to have been provisioned with the Device AppKey. Based on the AppKey and the content of the Join message sent by the device, the Application server:

1. Generates a NwkSKey and AppSKey and sends the NwkSKey information to the Core Network
2. Forms a cryptographic Join response payload that will allow the device to compute a NwkSKey and AppSKey

As part of the Join procedure, the network also allocates a DevAddr address to the LoRaWAN device. The Swisscom JoinEUI is **F0:3D:29:AC:71:00:00:01**



Some device manufactureres still use other or random JoinEUIs. But with the increasing number of roaming partners it is important to choose the Swisscom AppEUI, or roaming will never be possible with this device.

14.3. Activation By Personalization (ABP)

ABP	Who?	What is it?
DevEUI	IEEE/Device manufacturer	DevEUI is not used in LoRa communication in ABP but is used to identify the device at the Network Server side
DevAddr	Operator	The DevAddr is the Device Address on the LPN Network
NwkSKey	Device manufacturer	The NwkSKey encrypts the data during the transmission. Gateways from other networks cannot see the content of messages. The NwkSKey authenticates the device on a LoRa network
AppSKeys	Device manufacturer	The AppSKey encrypts the payload data

14.3.1 Device address: DevAddr

The DevAddr identifies the device on the network, together with the Network secret for the sensor.

The group (DevAddr, NwkSKey) must be globally unique.

If the end device is not using the JOIN LoRaWAN procedure, it must also be personalized with the DevAddr.



Note: If you really need to use ABP for some reason or testing, Swisscom can provide you with a DevAddr range. Other wise please avoid ABP as much as possible and use OTAA instead. Please mind that ABP shall not be used in production. **Please do not choose a random DevAddr yourself, this can lead to problems with your connection.**

14.3.2 128 bit network secret: NwkSKey

The 128 bit NwkSKey is used by the Core network to verify the authenticity and integrity of each message. Use a **random NwkSKey for each device.**

Allocating a **random NwkSKey per device** is very important for security, but also to ensure that the short address collision resolution algorithm will work appropriately. The pair (DevAddr, NwkSKey) must be globally unique.

128 bit application secret: AppSKey

The 128 bit AppSKey is used to encrypt the payload of messages. You may decide to use a unique AppSKey for all LoRaWAN ports used by your device, or to allocate one AppSKey for each port.

AppSKeys must be known to the Application Server. Commonly AppSKeys are part of a production Excel file providing the associations between DevEUI, DevAddr, NwkSKey, AppSKey(s) of the devices part of the production batch.

When adding the device to your account using the Device Manager application:

- > It is not mandatory to provision the AppSKey. The Swisscom LPN Portal will then forward the payload in encrypted form to the application servers and has no access to the payload clear-form content.
- > If you provision the AppSKey(s), then the Core Network will decode the payload before forwarding it to the application server(s).

14.4. Channel plans

14.4.1 ETSI EU 868

The implementation in Europe is the following default and mandatory channel plan:

- > LC1: 868.1 MHz
- > LC2: 868.3 MHz
- > LC3: 868.5 MHz
- > RX2: 869.525 MHz / SF12

Then the network will configure the device with the operator settings (add new channels, change RX2 configuration).

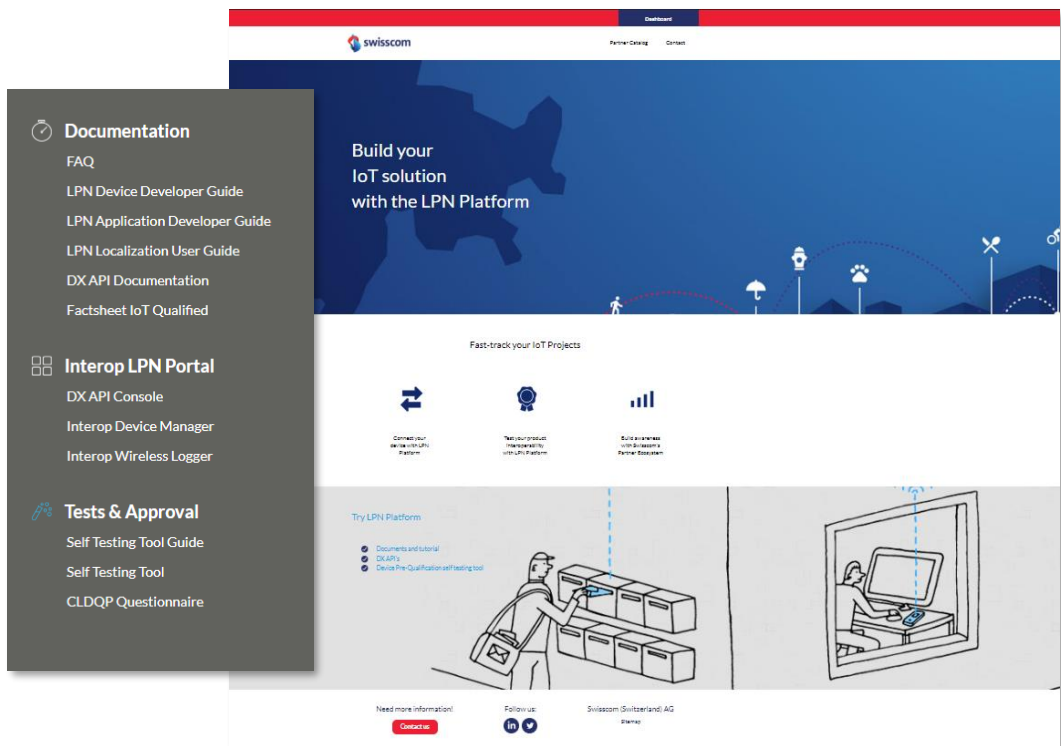
The channels currently used by the Swisscom LPN:

- > LC1: 868.1 MHz
- > LC2: 868.3 MHz
- > LC3: 868.5 MHz
- > LC4: 867.1 MHz
- > LC5: 867.3 MHz
- > LC6: 867.5 MHz
- > LC7: 867.7 MHz
- > LC8: 867.9 MHz
- > RX2: 869.525 MHz / SF12

15. Developing on LoRaWAN

15.1. The LPN Developer Portal

The Swisscom LPN Developer Portal consolidates all the necessary information for developers in one place: Documentation, FAQ, news, device qualification and new features to test. Get your account on <https://developer.lpn.swisscom.ch>



15.1.1 Swisscom IoT Qualified for LoRaWAN devices

Before starting your IoT project, you want to make sure that your device properly interoperates with the Swisscom LPN network. Therefore, you should only use devices that have passed the "**Swisscom IoT Qualified**" tests. Customers are also contractually obligated to use only qualified devices when deploying more than 50 of the same type.

If you are a device manufacturer, you can use the self-testing function on the developer portal to qualify your device. This function can also be used for purchasers to evaluate devices from different manufacturers.

On this platform you will also find the CLDQP (Collective LoRaWAN Device Qualification Program) form, which is used by the main public LoRaWAN operators and defines all questions and test procedures accepted by the participating operators.

For further information regarding the qualification, please refer to the quick guide available on the developer portal.

