
WIREPAS RANGE
3.X.X - USER GUIDE
Wirepas

Table of contents

1 INTRODUCTION

Welcome to the Wirepas v300 Client User Guide, introducing the latest release of the Blue Puck Mesh series, which is based on Wirepas Mesh technology. This new release incorporates the latest Wirepas technology (Wirepas 5), offering new features, enhanced performance, and bug fixes. Just like the previous series, the Wirepas Mesh technology enables the creation of decentralized mesh networks, which can be utilized for device localization, sensor data transmission, such as temperature monitoring, and even a combination of both functionalities.

One of the major improvements is the introduction of a new version of Over-The-Air Programming (OTAP), enabling application updates to be performed remotely. This feature will be extremely useful for future firmware upgrades/releases. The improved version is much more reliable.

It is, however, not possible to perform OTAP on devices with previous versions (v2xx and lower) to upgrade to this new release.

In this comprehensive User Guide, we will walk you through the various functionalities and features of the Blue Puck Mesh product series. We hope that it will provide you with all the necessary information to seamlessly configure and operate you devices and your Wirepas network. In case you have any further questions, do not hesitate to contact the ELA customer support team. We are happy to help and provide supplementary information.

2 PRODUCT LIST

Please find below a comprehensive list of all the products currently available. Their different functionalities and how to best deploy and manage them in a Wirepas Mesh network are described in the subsequent sections.

Beacons and Anchors

DESCRIPTION	***DESCRIPTION***
Blue ANCHOR	Tag Wirepas Mesh with localization option, data routing; can serve as positioning reference point
Blue PUCK **ID MESH**	Tag Wirepas Mesh with localization option
Blue PUCK **ID+ MESH**	Tag Wirepas Mesh with localization and mode + option*
Blue PUCK **BUZZ MESH**	Tag Wirepas Mesh with localization option and Buzzer
Blue PUCK **BUZZ+ MESH**	Tag Wirepas Mesh with localization option, Buzzer and mode + option*
Blue COIN **ID MESH**	Tag Wirepas Mesh with localization option
Blue COIN **ID+ MESH**	Tag Wirepas Mesh with localization and mode + option*
Blue SLIM **ID MESH**	Tag Wirepas Mesh with localization option
AERO **ID+ MESH**	Tag Wirepas Mesh with localization and mode + option*
Blue SLIM **ID+ MESH**	Tag Wirepas Mesh with localization and mode + option*
Blue LITE **ID MESH**	Tag Wirepas Mesh with localization option
Blue LITE **ID+ MESH**	Tag Wirepas Mesh with localization and mode + option*

Mode + option*: This means that the tag is equipped with a motion sensor and has the ability to change its transmission period when a movement is detected.

Sensors

DESCRIPTION	DESCRIPTION
Blue PUCK **T MESH**	Temperature Sensor
Blue PUCK **RHT MESH**	Humidity and Temperature Sensor
Blue PUCK **MAG MESH**	Magnetic Sensor
Blue PUCK **MOV MESH**	Motion Sensor
Blue PUCK **DI MESH**	Digital Input Sensor
Blue PUCK **PIR MESH**	Presence Detection

Gateways

- ***Raspberry Pi Gateway and Wirepas Mesh Wireless Dongle (2.4 GHz)***

Raspberry Pi3 B+ or Pi4

Wirepas Mesh 2.4 GHz wireless dongle

- ***SolidRun Gateway***

SolidSense N6

3 GENERAL INFORMATION ON MESH NETWORKS

Mesh networks

A mesh network is a network topology (wired or wireless) in which all hosts are connected "peer-to-peer" without a centralized hierarchy, thus creating a net-type structure. With this architecture, every node can send, receive, and relay data. This eliminates the presence of "backbone" points that can isolate parts of the network in case of malfunction. If a host stops working, data simply takes another route to its destination. A mesh network can relay data via "flooding" (broadcasting data so that it is received by all nodes within direct wireless range). It can also use predefined routes, in which case the network must plan for uninterrupted connections or alternative routes.

Wirepas Mesh

The Wirepas Mesh protocol is a wireless network protocol that uses a multi-jump, self-organizing, and decentralized design. Decentralized network topology enables extremely dense network deployment.

Wirepas focuses on providing a connectivity solution that is highly **reliable**, **optimized**, **scalable**, and **easy to deploy**.

This solution was specifically designed to meet two major challenges facing wireless mesh networks: network reliability regardless of its size and density; and low energy consumption by router devices in the network.

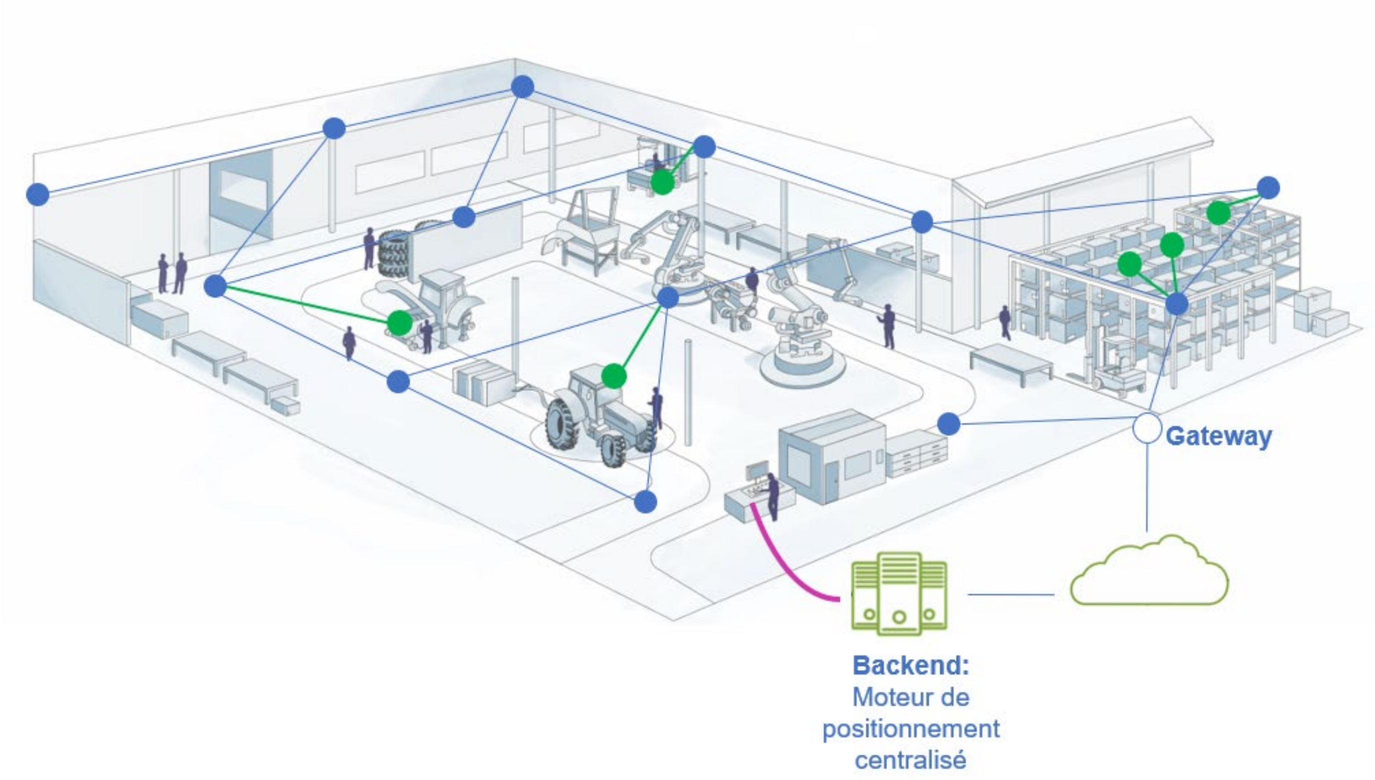
Information about Wirepas Mesh technology is available here:

www.wirepas.com

4 GENERAL NETWORK SETUP

In general, there is a lot of freedom in setting up a Wirepas Mesh network. The essential components are Gateway(s) connected to the a Wirepas backend to transmit the data from devices. Networks can be used for localization, transmission of sensor data, or even a combination of both. A typical network consists of Anchors that transmit data to the Gateway and serve as localization reference points, as well as mobile devices which, depending on their functionality, can also route data from other devices. Mobile tags can be localized and/or send sensor data, depending on their configuration. The different functionalities of the devices will be explained in detail in the following chapter. In this chapter, we will illustrate the general setup of a Wirepas network.

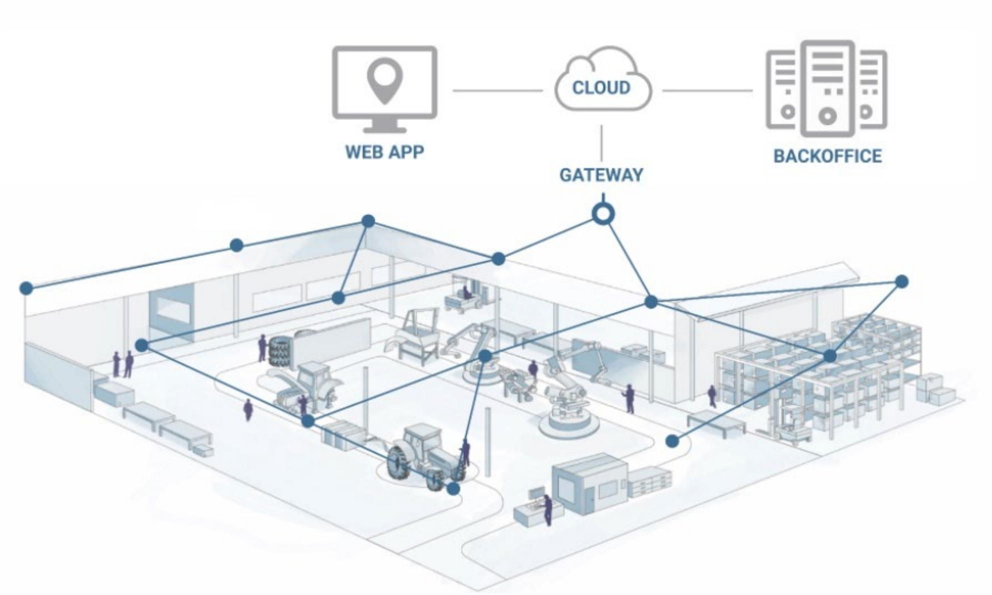
Location network diagram (exemplary)



Network components	Products
ANCHOR	BLUE PUCK **ID MESH** / Blue ANCHOR
MOBILE Beacon	BLUE PUCK/COIN/SLIM/LITE **ID+ MESH**
GATEWAY	ELA Innovation **MESH** Gateway

The graphic show the typical setup of localization network. Typically, Anchors (depicted in blue) have a fixed position, therefore they can serve as reference points for localization. Mobile Beacons (depicted in green) can, for example, be attached to vehicles, equipment and even people, and they are localizable due to the Anchors. The Gateways (depicted in white) transmit the data from the devices to a centralized backend, where they then can be processed and interpreted. A notable advantage lies in the dynamic nature of the network setup, which can be readily modified after the initial deployment. Adding devices to an existing network is easily achievable and can help to extend the covered surface area, increase location accuracy, or reinforce coverage in difficult zones if necessary.

Sensor network diagram (exemplary)



Network components	Products
MESH sensor	BLUE PUCK T **MESH** - RHT **MESH** - MAG **MESH** - MOV **MESH** - PIR **MESH** - Digi IN **MESH**
GATEWAY	ELA Innovation **MESH** Gateway

Different from the localization network presented above, no Anchors are required for a Sensor network, as the Mobile devices themselves facilitate the transmission of data from other devices to a Gateway. They can, nevertheless, be added if desired. Such a network typically consists of devices equipped with sensors, which can be, for example, used for temperature monitoring or presence detection. As for the localization network, devices can be easily added and removed from the network.

The two network setups presented above are, of course, merely illustrative examples. As previously mentioned, there is a lot of freedom in the setup of the network and the selection of devices, according to the specific needs and requirement of the project. In the following, we will provide you with some additional general information on network setup. Complementary information can be found directly on the Wirepas developer portal, please refer to Overview and Concepts section[1].

[1] https://developer.wirepas.com/support/solutions/folders/77000315348
e.g. Wirepas Massive Overview (as of Thu, 8 Sep, 2022);
Wirepas Mesh Concepts (as of Thu, 30 Jun, 2022);
Wirepas Software and APIs Overview (as of Mon, 14 Nov, 2022)

General Information on network setup

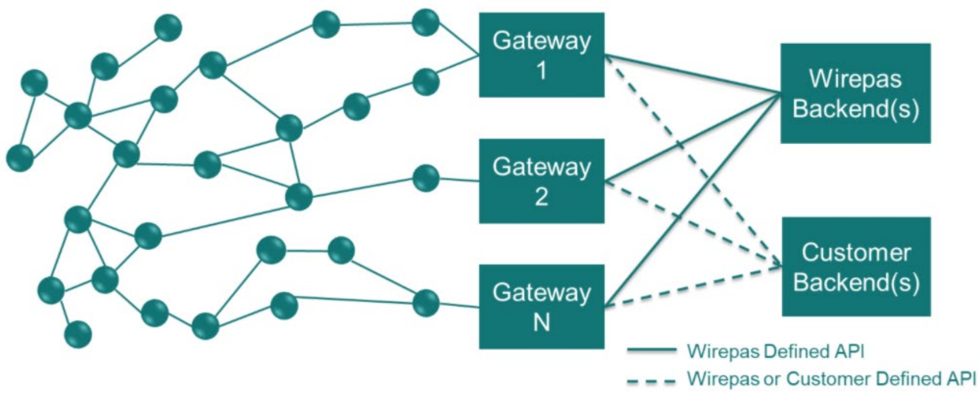
When setting up the network, it is essential to consider several factors. Firstly, it is important to ensure that an adequate number of devices are present to route the data effectively, with approximately 10 devices connected to an Anchor or Router device. Additionally, it is worth noting that non-router devices typically consume less energy compared to router devices and should therefore be deployed, whenever possible. When configuring the transmission period of the device, it should be noted that it not only directly affects the power consumption of the device itself but also that of the routing devices. This is because a higher transmission period of the devices leads to a greater amount of data routing required.

Taking all of these aspects into account is essential for establishing an efficient and sustainable network infrastructure.

Gateways

The gateway receives network data and transmits it to one or more back-end servers.

The example below shows the connection between tags, gateway(s), and back-end(s).



There are two ways to retrieve data:

- Connect to the Wirepas API and retrieve the data stream respecting the format required by Wirepas.
- Develop your own API and handle message collection yourself.

Messages are received in a generic format described in Wirepas documentation and encoded in protocol buffer format: https://developers.google.com/protocol-buffers.

Complete information related to message reception and encoding is provided in the following section of the Wirepas GitHub:

https://github.com/wirepas/backend-apis/tree/master/gateway_to_backend

Data and MQTT topics

Data is sent by the Gateway to a specific MQTT broker server. The list of topics corresponding to the various sensors formats and positioning information in the BLUE MESH product line are provided in the Section Data Received.

5 OPERATING MODES

In general the devices can be grouped into three categories:

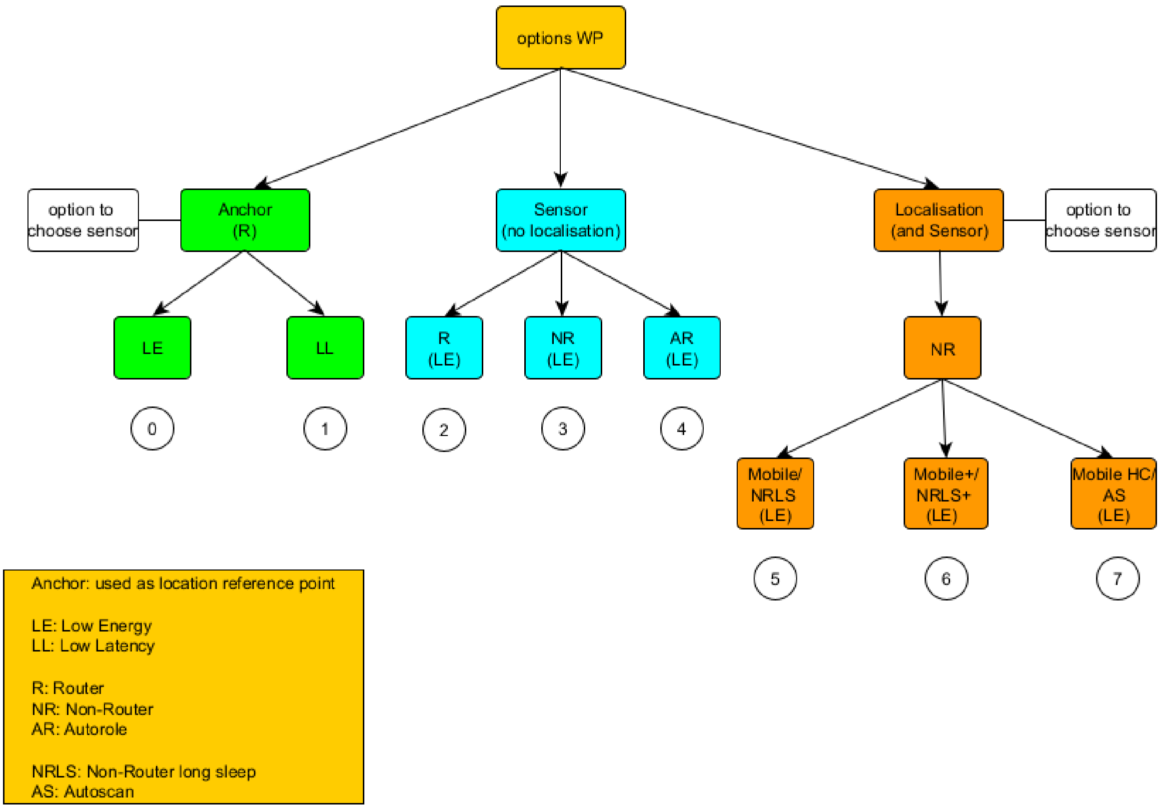
- **Anchors** (Low Energy or Low Latency)
- **Sensors** (Router, Non-Router or Auto Role)
- **Mobile Tags** (NRLS, NRLS+ or Auto Scan)

Anchors are mainly responsible for routing data of other devices and they can serve as reference points for localization. Optionally, they can be equipped with a sensor.

Sensors are devices equipped with a sensor, their principal function is to measure and send its sensor data. They can or cannot route data of other devices, depending on their functionality. It is important to note that these devices cannot be localized.

Mobile Tags are devices that can be localized. They never route data of other devices.

The different options are summarized in the graph below and will be explained in detail in the following:



Operating Modes

Parameter	Possible Values	Function	Availability
Tag WP Function	0 1 2 3 4 5 6 7	Anchor Low Energy (LE) Anchor Low Latency (LL) Sensor Router (R) Sensor Non-Router (NR) Sensor Autorole (AR) Tag mobile NRLS Tag mobile NRLS+ Tag mobile Autoscan (AS)	Hardware dependent

Summary of the functionalities for the different Options:

Function	Data routing	Positioning Information	Localization reference point	Option Sensor
Anchor LE	yes	yes	yes	yes
Anchor LL	yes	yes	yes	yes
Sensor R	yes	-	-	yes
Sensor NR	-	-	-	yes
Sensor AR	yes[2]	-	-	yes
NRLS	-	yes	-	yes
NRLS+	-	yes	-	limited
AS	-	yes	-	yes

A – Anchors

The principal functionality of anchors is the transmission/routing of data of other devices to a Gateway. They can also serve as location reference points and allow the localization of Mobile Tags. In addition, they can have a sensor integrated and send sensor data themselves. They can also send frames in Bluetooth Low Energy (BLE), if this feature is activated.

Anchors are connected to the Wirepas network all the time. They are routers, which means that they transmit data between several tags (router or non-router) or Gateways. An anchor/router can establish up to 14 connections at the same time (4 to other anchors and 10 to other mobile devices), which should be taken into account, when setting up the network.

The anchors can operate in Low Energy (LE) mode or Low Latency (LL) mode. In the LL mode the data transmission is faster, however the energy consumption is much higher. This mode can therefore only be used for anchors powered by an external power source. All battery-powered anchors (and mobile devices) must therefore use the LE mode.

All devices configured as Anchors, periodically send their battery information, and if they have a sensor integrated, their sensor data. They also send their positioning information at irregular intervals[3], depending on the stability of the network. The format of the data received and how to (re)configure the devices, will be explained in the subsequent chapters.

[2] If the option Auto Role (AR) is selected, the device functions either as a Router or as a Non-Router, depending on the network setup and the battery level of the device.

[3] For a stable network, positioning packages are sent approximately one time per day.

B – Sensors

As the name suggest, the principal functionality of the devices in the Sensor mode, is the transmission of sensor data. The devices cannot be localized, they therefore only send sensor and battery information.

There are three different configuration options for the Sensor devices:

- Router (R) -> the device transmits data between multiple Tags, Anchors or Gateways
- Non-Router (NR) -> the device sends its sensor data, but does not route the data of other devices
- Auto-Role (AR) -> the device automatically selects between the Router or Non-Router option

The choice of the configuration strongly depends on the network setup. It has to be ensured that enough devices that can route data, Sensor R or Anchors, are present in the network. The advantage of the Non-Router option is that the devices consume less energy and therefore have a higher expected lifetime than router devices.

The available sensors can be grouped into two categories:

- Binary sensors: sensors with only two possible states (0 – not detected; 1 – detected), e.g. MAG, MOV, PIR, TOUCH

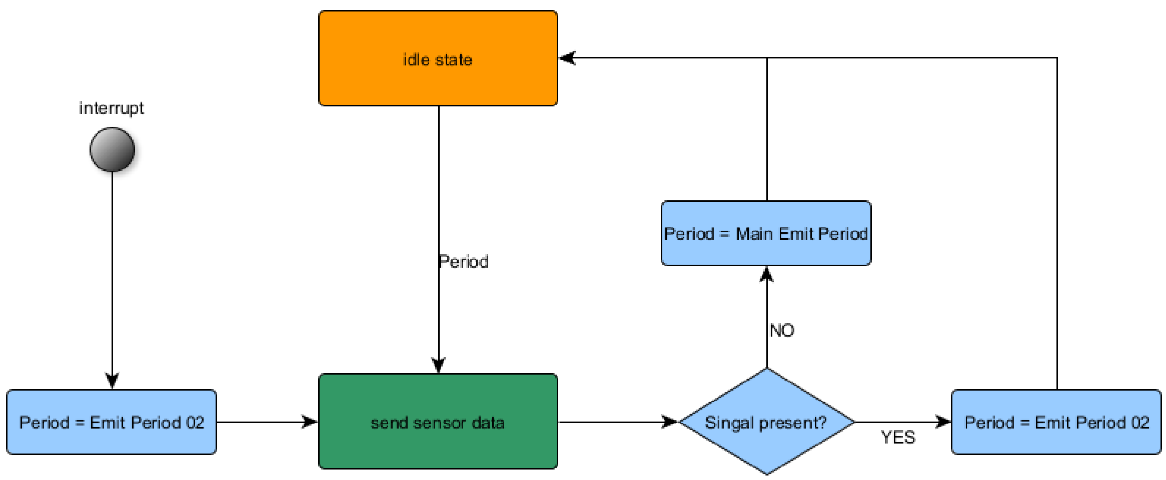
- Analog sensors: that measure a value in a continuous spectrum, e.g. T, RHT, ANG, ProxIR

In general, the sensor data is sent periodically, according to the period configured. In addition, the binary sensors are interrupt-operated, which means that they immediately register and event upon detection of a signal and they directly send their data with the updated value.

Binary Sensors configured with two transmission periods

It is also possible to configure the binary sensors with two different transmission periods (Main Emit Period and Emit Period 02). As sons as an interruption/signal is detected, they switch to the Emit Period 02 period and keep sending at this period as long as the signal is present. It is therefore possible to modify the data transmission period, as illustrated in the graph below. This allows, to for example, configure the device at a relatively long Main Emit Period and switch to a faster period upon signal detection.

Schema illustrating the functionality of the Sensor format with two Emission Periods



Sensors – Parameters

Parameter	Possible Values	Function
Main Emit Period in ms	[8000 – 86400000] ms	Principal Emission Period
Emit Period 02 in ms	[8000 – 86400000] ms	Secondary Emission Period
Sensor Format	0x01 0x02 0x03 0x04 0x05 0x06 0x09 0x0C	ID (no Sensor) T RHT MAG MOV ANG DI PIR
Sensor Threshold MOV	[32 - 8000] in mg	Sensor threshold – for the detection of a movement
PIR sensitivity level	[0, 1, 2, 3]	Sensitivity of the Detector 0 (low sensitivity) -> 3 (high sensitivity)

<u>C – Tag Mobile</u>

The primary functionality of Mobile Tags is localization. They can be used, for example, for asset tracking and other situations where localization is required. These tags periodically send their positioning information, which enables for them to be localized. They do not perform data routing and are always non-routers. Additionally, they can be equipped with sensors to periodically transmit sensor data.

Three different options for Mobile Tags, which will be presented below:

- Auto Scan (AS) – (formerly also referred to as Tag HC, for High Consumption)
- Non-Router Long Sleep (NRLS) - (formerly also referred to as Tag Mobile)
- NRLS+ - (formerly also referred to as Tag Mobile +)

<u>Auto Scan</u>

A mobile Tag in mode Auto Scan is constantly connected to the Wirepas network. Due to this, it consumes more energy than the NRLS/NRLS+ option; in the past this option was therefore sometimes also referred to as Tag High Consumption (HC). However, being constantly connected to the network enables the device to directly receive and process information or commands send from the Gateway.

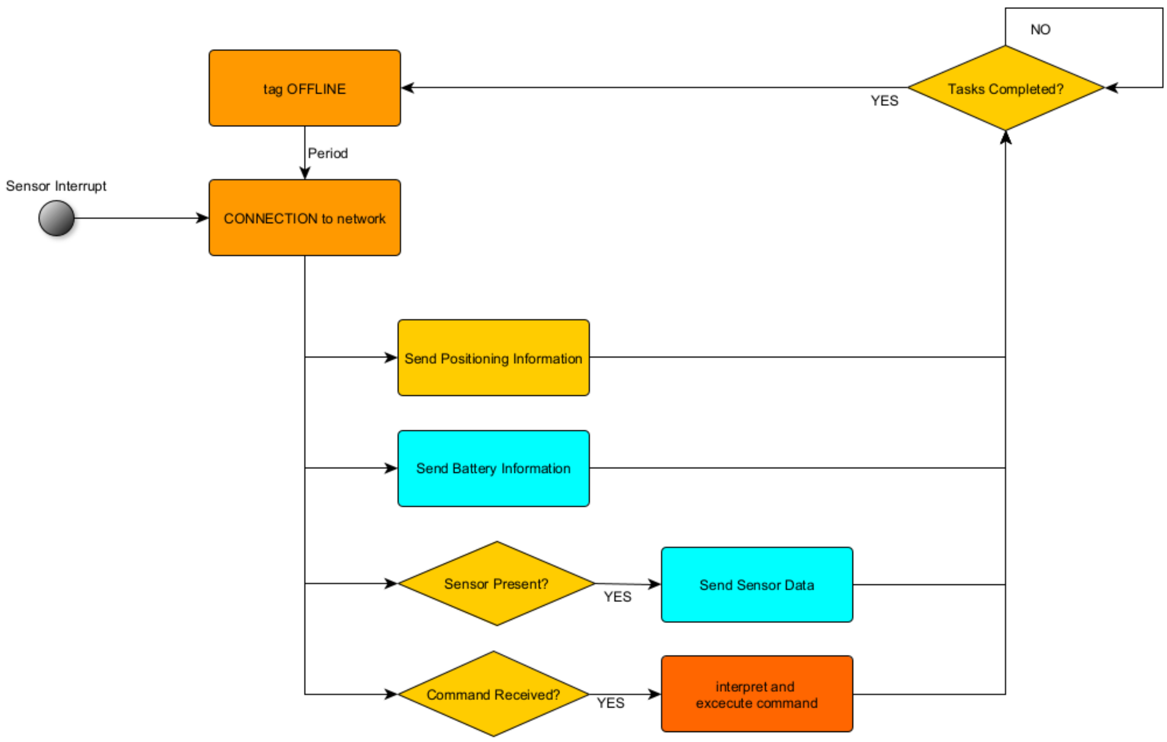
<u>NRLS</u>

Unlike the Auto Scan devices, a tag in mode NRLS is not continuously connected to the network. NRLS stands for "Non Router Long Sleep", indicating that the device only periodically connects to the network to send data and receive potential commands. The rest of the time, the devices are in a phase called "sleep", where they are disconnected from the network and unresponsive.

One significant advantage of this option is its lower power consumption compared to the other options. However, a notable disadvantage is that the tag is less responsive since it can only receive commands when it wakes up and connects to the network. This means that if a device is configured with a relatively long transmission period, it may take some time before it receives a message. Additionally, NRLS tags can only receive messages received via the option "Application Data" option, as these messages are persistent in the network. They cannot receive messages sent via MQTT, which are not persistent, thereby running the risk of missing them if the tag is not connected at the time the message is sent.

The connection/ disconnection process of an NRLS tag to an Anchor takes approximately one minute. This has to be taken into account when configuring the transmission period and also when estimating the number of Anchors needed for a certain number of tags.

Schema illustrating the functionality NRLS devices



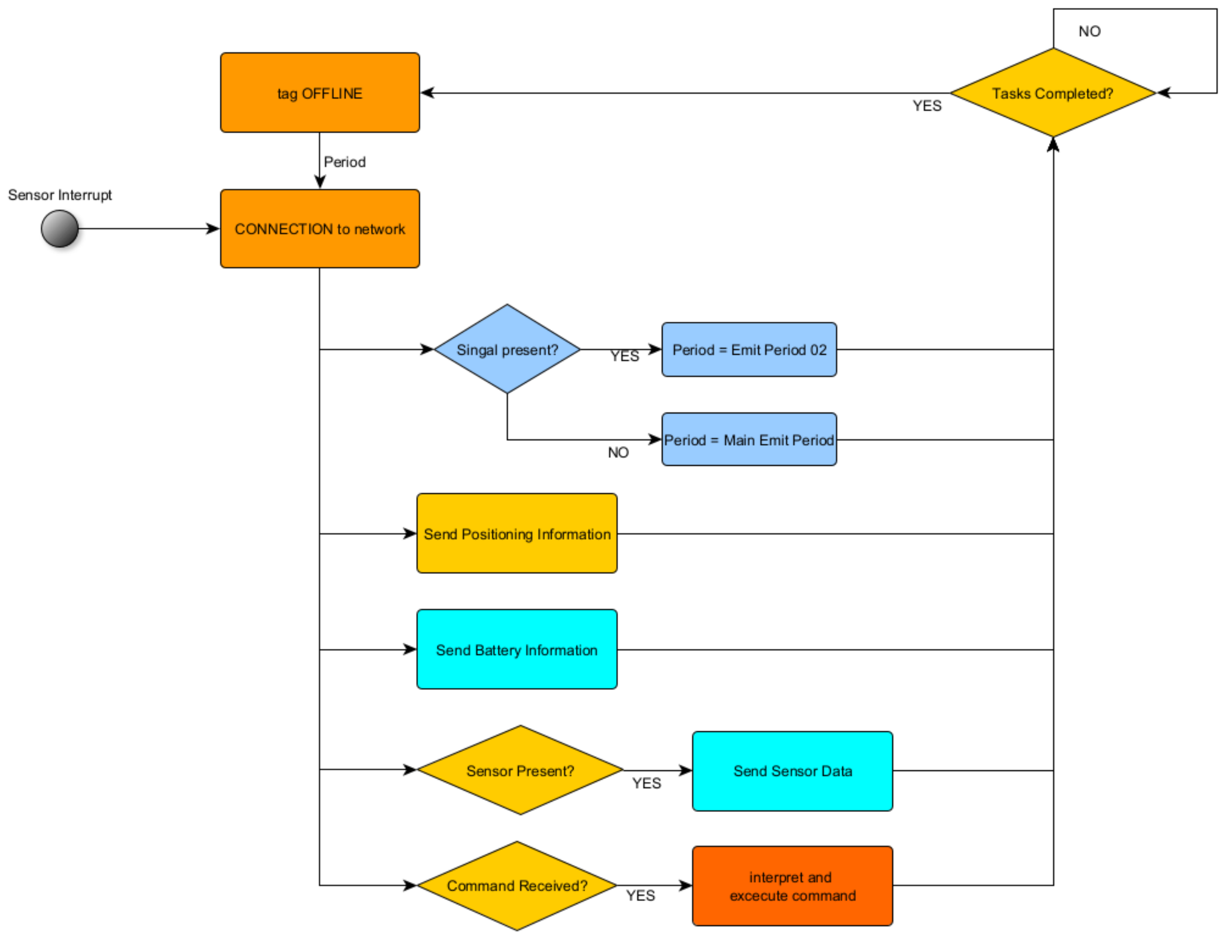
NRLS+

A tag in NRLS+ mode offers the same functionalities as in NRLS mode. Additionally, it has the capability to change its transmission period upon signal detection. When a movement is detected, the device immediately connects to the network and sends its information. As a result, two periods are used in this mode, similar to the Sensor option with two periods: the Main Emit Period (previously referred to as the slow period) and the Emit Period 02 (previously referred to as the fast period).

If the tag does not detect any movement, its behavior is identical to that of an NRLS device, waking up periodically with the Main Emit Period. However, once an interruption is detected, the wakeup period changes to the Emit Period 02, and the device immediately connects to the network to transmit its information. If the movement persists, the tag continues to wake up periodically with the Emit Period 02. Once the interruption ends, the transmission period reverts back to the Main Emit Period.

The purpose of having two emission periods is to choose an Emit Period 02 that is faster than the Main Emit Period. This ensures that when movement is detected, the device transmits its information at a higher frequency. The entire process is illustrated in the graph below:

Schema illustrating the functionality NRLS devices



6 DATA RECEIVED

The devices periodically transmit information over the network based on their functionality and configuration. The messages are transmitted through other tags or anchors that function as routers, ultimately reaching a gateway. Subsequently, the data is forwarded from the gateway to an MQTT broker. Anchors, tags NRLS/NRLS+ (Mobile/Mobile+) and AS transmit positioning and battery information. If equipped with a sensor, they also send sensor data. Sensors solely transmit their sensor data and battery information, but no positioning information, as they are not localizable. Moreover, diagnostic information is sent, which can be valuable in identifying network malfunctions.

The list below contains all the essential information required for interpreting the received data: the endpoints (EP) associated with various sensor formats, the data format and type, and the applicable sensor range. All received data frames are encoded using the TLV (type-length-value) scheme and transmitted in hexadecimal, utilizing the little-endian format. Additionally, the sensor data is consistently accompanied by the "Battery Voltage" information. A detailed example for the humidity and temperature sensor can be found below.

RHT Sensor data and interpretation:

Message received: 03 04 51 00 BD 0B 01 02 7A 0B (received via EP 110/110)

-> RHT data :

- Type (T): 03 -> RHT
- Length (L): 04 -> 4 bytes
- Value (V): 51 00 BD 0B
  - ➔ Relative Humidity: 0x0051 = 81 ≙ 81 %
  - ➔ Temperature: 0x0BBD = 3005 ≙ 30.5°C

-> Battery Voltage:

- T : 01 -> Battery
- L : 02 -> 2 bytes
- V : 7A 0B
  - ➔ Voltage: 0x0B7A = 2938 ≙ 2939 mV

The format of the data received is exactly the same as for the previous versions. For users already familiar with the different data formats and their interpretation, nothing has changed. Only the format of the positioning information (received via EP 238/238) has changed slightly[4]. Detailed examples for all the different sensor formats can be found in the Appendix.

In general, the data is encoded and stored in 16 bits (2 bytes), however there are a view exceptions where the data is stored in 32 bits (4 bytes). For most sensors, the data comprises multiple blocks. For example, in the case of the RHT data, the Relative Humidity (RH) data (2 bytes) is followed by the temperature data (also 2 bytes). Binary sensors, i.e. sensors with two states like detected/not detected for MOV sensor, have the state (2 bytes) followed by a counter (4 bytes) indicating the number of times this state has been detected since the last device reboot. However, the ANG sensor provides acceleration data on three axes, with each axis's acceleration encoded in 2 bytes. The ProxIR sensor, on the other hand, sends the measurement state (2 bytes), followed by the measured distance (2 bytes).

Format and Endpoints of the data received for the different sensor formats:

---

[4] More information: Wirepas Positioning Application Reference Manual v1.5 (section Measurement message) (as of Wed, 21 Sep, 2022)

Functionality/ Sensor Format	Endpoint (EP)	Data				
	Source/ destination	Type	Length/ Length of subunit		Description/ Data Type [unit]	Range
Battery Voltage	11/11	01	02		Battery Voltage U [mV]	2500 mV – 3600 mV
Temperature **(T)**	100/100	02	02		Temperature T [c°C]	-4000 c°C - +8500 c°C
Humidity and Temperature **(RHT)**	110/110	03	04	02	Relative Humidity Φ [%]	*0% – 100%*
				02	Temperature T [c°C]	-4000 c°C - +8500 c°C
Digital Input **(DI)**	120/120	04	06	02	Input State	0 – activated 1 – deactivated
				04	Counter	0 – 32767 (0x7FFF)
Digital Output **(DO)**	130/130	05	06	02	Output State	0 – activated 1 – deactivated
				04	Counter	0 – 32767 (0x7FFF)
Magnetic Field Detection **(MAG)**	150/150	07	06	02	Magnetic Field Detection	0 – not detected 1 – detected
				04	Counter	0 – 32767 (0x7FFF)
Movement Detection **(MOV)**	160/160	08	06	02	Movement Detection	0 – not detected 1 – detected
				04	Counter	0 – 32767 (0x7FFF)
Acceleration **(ANG)**	170/170	09	06	02	Acceleration – 3 axis a_x *[mg]*	*-16 000 mg – +16 000 mg*
				02	a_y *[mg]*	

				02	*a_z [mg]*	
Presence Detection **(PIR)**	200/200	0C	06	02	Presence Detection	0 – absence 1 – presence detected
				04	Counter	0 – 32767 (0x7FFF)
Positioning Information	238/238	up to 102 bytes			contains RSSI information and possibly other data; cf. WP documentation[6]	
Diagnostics	247/255	variable			used to analyze the nodes functionality in the network cf. WP documentation[7]	

[6] Wirepas Positioning Application Reference Manual v1.5 (Measurement message – version modified on: 21 Sep, 2022 at 11:40 AM)
[7] https://developer.wirepas.com/support/solutions/articles/77000406799-wirepas-mesh-diagnostics-v2-reference-manual (version modified on: 18 Feb, 2022 at 11:03 AM– confidential document)

7 COMMANDS

There are two principal ways of sending a command to a device or class of devices:

- a command sent via the “Application Data” (AppConfigData)
- a command sent via an MQTT topic

It is also possible to modify certain parameters using a Remote Api provided by Wirepas. For more information please refer to the documentation provided by Wirepas[8]. In this document the two methods mentioned above will be presented.

The commands that can be send are either a parameter change (full list of modifiable parameters can be found in the Appendix), or a basic command like the activation of an LED or Buzzer.

It is important to note that devices in mode Anchor, Sensor and Mobile AS can use both options. However, devices in NRLS/NRLS+ (Mobile/Mobile+) mode can only receive commands send via the Application Data. Messages sent via an MQTT topic are, unlike Application Data Messages, not persistent in the network. As a result, there is a possibility of the message getting lost if a device in NRLS/NRLS+ mode is in its "sleep" phase. When sending a command via an MQTT topic, it is therefore absolutely necessary for the device is to be in Anchor, Sensor or Mobile AS mode.

Commands “Application Data”

The Application Data is a functionality provided by Wirepas that allows sending a command to an individual device, a network class, or even all nodes in the network. The message persists in the network until it is actively replaced by a new message. This feature enables a new device entering the network to receive the message, and devices in NRLS/NRLS+ mode can also receive it when they wake up and connect to the network.

The maximum length of a message send via the Application Data is 80 bytes, allowing for the transmission of multiple commands simultaneously. The list of modifiable parameters and accessible commands can be found in the appendix.

In order to send messages, they must adhere to the following format:

> Sending a message to an *entire class*:
>
> **F6 7E 01 C1 (*LL*+2) *CT LL CMD***
>
> (*LL*: length of the command (in hexadecimal) ; *CT*: class of the tag/device; *CMD*: command)
>
> Sending a message to an *individual device*:
>
> **F6 7E 01 C1 (*LL*+7) F8 (*LL*+5) 86 XX XX XX XX *CMD***
>
> (*LL*: length of the command; F8: do not modify!; *XX*: tag address (hexadecimal in little-endian); *CMD*: command)

The command (CMD) will be generated by the Device Manager and does not need any further modification. However, it is important to know the length of the command, as well as the device class or the address of the tag to which the command is sent.

[8] Wirepas Massive Remote API Reference Manual (version modified on: 16 Jun, 2021 at 8:52 AM– confidential document)

In the following table you can find some basic examples:

Examples of messages sent via « App Config Data »

Command	Device Address/ Class	Complete Message
Parameter Modification		
Change the Sensor Format to format Sensor Format T	Individual device (ID : 754077429 = 0x2CF24EF5)	F6 7E 01 C1 *23* F8 *21* 86 F5 4E F2 2C 1A *1A* **00 10 16 C3 10 EF 09 01 07 45 4C 41 31 32 33 34 10 50 07 04 02 01 02 01 01 02** (encoding preamble; *length*; type; tag ID; **CMD**)
	class: 0xFB	F6 7E 01 C1 *1E* FB *1C* 1A *1A* **00 10 16 C3 10 EF 09 01 07 45 4C 41 31 32 33 34 10 50 07 04 02 01 02 01 01 02** (class)
Change the device name to "TOTO"	Individual device (ID : 754077429 = 0x2CF24EF5)	F6 7E 01 C1 *22* F8 *20* 86 F5 4E F2 2C 1A *19* **00 10 15 19 10 EF 09 01 07 45 4C 41 31 32 33 34 10 10 06 01 04 54 4F 54 4F**
	class: 0xFA	F6 7E 01 C1 *1D* FA *1B* 1A *19* **00 10 15 19 10 EF 09 01 07 45 4C 41 31 32 33 34 10 10 06 01 04 54 4F 54 4F**
Commands		
Buzzer OFF	class: 0xFB	F6 7E 01 C1 *0A* FB *08* 1B *06* **00 20 02 9F 20 04**
LED ON pendant 20s	Individual device (ID : 754077429 = 0x2CF24EF5)	F6 7E 01 C1 *12* F8 *10* 86 F5 4E F2 2C 1B *09* **00 20 05 EB 20 01 02 14 00**

Commands MQTT

Commands are directly sent to an individual device by selecting its address (identifier) through an MQTT endpoint (EP). However, unlike the Application Data, these messages are not persistent in the network.

Important note: These commands can only be used for devices that remain connected to the network, such as Anchors, Mobile AS/HC, and Sensors. Mobile beacons in NRLS mode (Mobile or Mobile+) are unable to receive these commands due to network disconnection during their sleep phase between two measurement updates.

Similar to the Application Data, it is possible to modify parameters and send basic commands to the devices. A comprehensive list of available options and their corresponding endpoints (EP) will be provided below:

Parameter modification:

Functionality	Source Endpoint	Destination Endpoint	Command
Parameter Modification	70	70	CMD
Command	80	80	CMD

The command (CMD) can be generated by the Device Manager and does not need any further modification.

Basic commands (ASCII):

Functionality	Source Endpoint	Destination Endpoint	Information	
			Command	ACK
LED ON	20	20	LED_ON	OK : 00 NOK : 01
LED OFF	20	20	LED_OFF	OK : 00 NOK : 01
LED ON Time *time in seconds	20	20	LED_ON 10 *10 seconds	OK : 00 NOK : 01
BUZZ ON	20	20	BUZZ_ON	OK : 00 NOK : 01
BUZZ OFF	20	20	BUZZ_OFF	OK : 00 NOK : 01
BUZZ ON Time *time in seconds	20	20	BUZZ_ON 10 *10 seconds	OK : 00 NOK : 01
LEDBUZZ ON	20	20	LEDBUZZ_ON	OK : 00 NOK : 01
LEDBUZZ OFF	20	20	LEDBUZZ_OFF	OK : 00 NOK : 01
LEDBUZZ ON Time *time in seconds	20	20	LEDBUZZ_ON 10 *10 seconds	OK : 00 NOK : 01
DIGITAL Output ON	130	130	DIGI_ON	Response send via EP 160/160 see section Data Received
DIGITAL Output OFF	130	130	DIGI_OFF	Response send via EP 160/160 see section Data Received
DIGITAL Output ON Time *time in seconds	130	130	DIGI_ON 10 *10 seconds	Response send via EP 160/160 see section Data Received

REBOOT	40	40	REBOOT	OK : 00 NOK : 01

Note: When sending several commands LED/BUZZER ON one after another to the same device, a command LED/BUZZER OFF must be send in-between, in order to ensure that the commands are received and executed correctly.

Request Sensor Data

Sensor	Source Endpoint	Destination Endpoint	Command
T	30	30	T_DATA
RHT	30	30	RHT_DATA
MOV	30	30	MOV_DATA
ANG	30	30	ANG_DATA
MAG	30	30	MAG_DATA
AI	30	30	AI_DATA
DI	30	30	DI_DATA
PIR	30	30	PIR_DATA

It is possible to send a command to a device to immediately request the sensor data. The response is send in the format described in section Data Received.

Diagnostic commands

Functionality	Endpoint source	Endpoint destination	Description	
			Command	Information
Battery level	50	50	GET_BATT_VOLTAGE	Returns the battery level
Firmware version	50	50	FW_VERS	Returns firmware version
Sequence Number	50	50	SCRATCHPAD_INFO	Returns the value of the scratchpad sequence number and the processed

				scratchpad sequence number

8 CONFIGURATION

The devices can be configured with an NFC reader using the ELA "Device Manager" application. For minor configurational changes it is also possible to configure the devices via the network. Both concepts will be explained in the following.

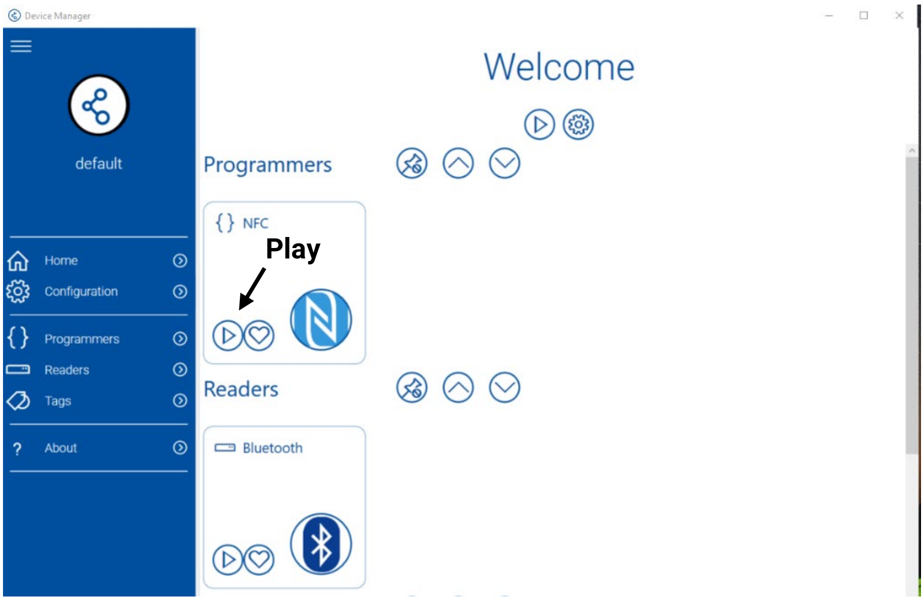
A. VIA NFC (Device Manager)

For configuring the devices, please follow the subsequent steps:

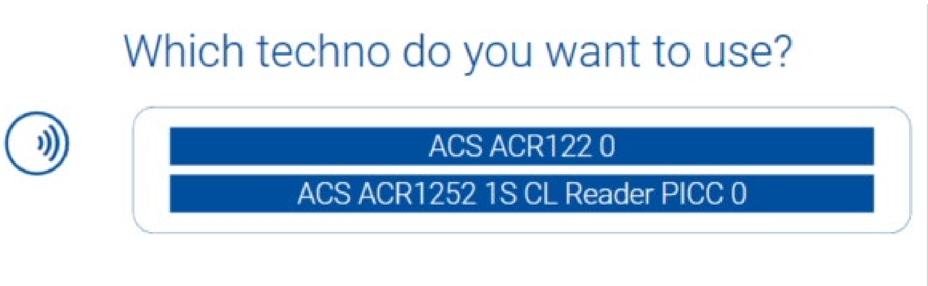
1. Connect an NFC reader to your PC (for example: NFC R/W 01 - ref. ACIOM177)

2. Start the Device Manager application[9]

3. On the welcome page click the "Play" button on the "NFC" widget



4. Select your NFC reader

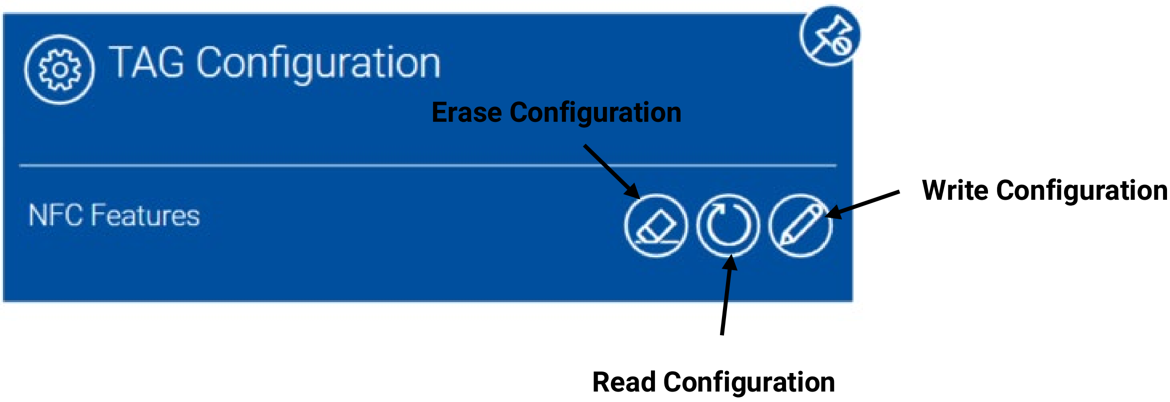


Upon selection, the following widget will appear:

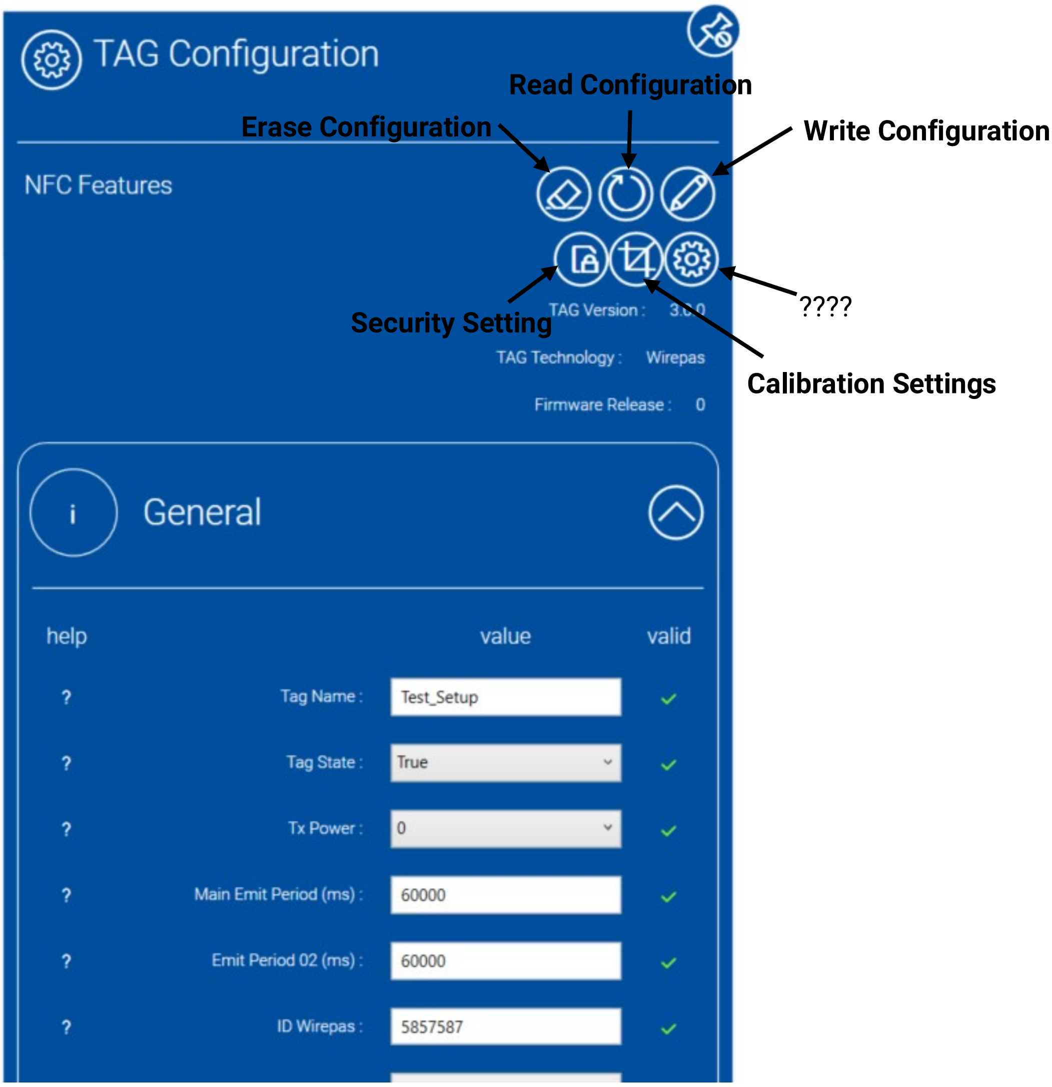
[9] The software can be downloaded directly from the ELA Innovation homepage. Tutorials and installation guidelines are available online.

5. Place the device on the NFC reader as indicated below:

6. Click on the Configuration  icon. This opens up the TAG Configuration window:



7. Click on “Read Configuration” icon  to read and display the current configuration of the device.



8. To modify a parameter, change the value(s) in the respective field(s) and click on the write icon . Afterward, remove the device from the NFC reader. The tag will reboot and operate with the new configuration. To confirm that the new configuration has been successfully applied, place the device on the NFC reader once again and read the configuration.

A list of all modifiable parameters, their function and their ranges can be found in the Appendix.

B. OVER THE NETWOK

It is possible to configure a reduced set of parameters throught app data service through the network: main Emit Period, Emit Period 02, Acceleration Threshold, Activation of emulated BLE advertising, Ble Tx period and Tx power. Refer to appendix for the code information.

9 VIEWING TOOLS

Wirepas Network Tool (WNT)

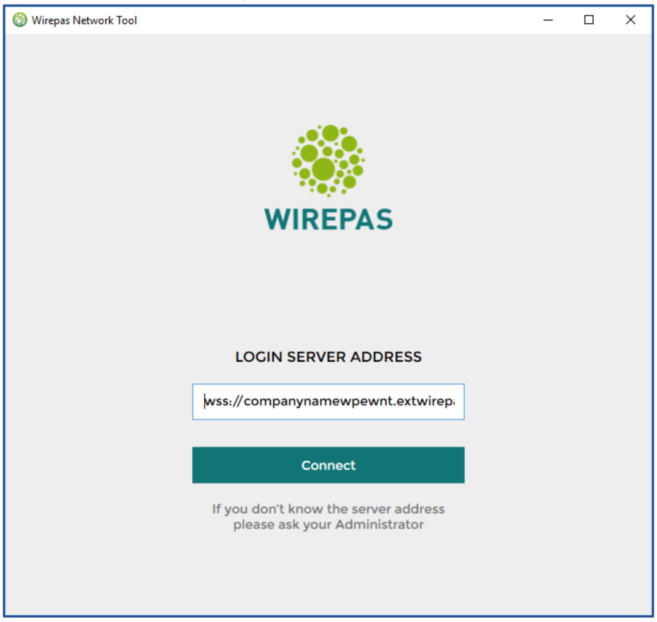
The Wirepas Network Tool (WNT) is a tool for managing and monitoring a Wirepas network. It is provided by Wirepas and can be downloaded directly from their web page. It is strongly recommended to use the latest version.

The application enables network monitoring, e.g. viewing the network status, managing node configurations, updating nodes using remote updates and configuring the Wirepas Positioning Engine. It also allow building and managing floorplans to visualize the network topology in a given environment.

All these features and how to correctly setup a WNT instance are explained in full detail in the Client User guide provided by Wirepas[10].

Upon launching WNT, you will need to enter the "Login Server Address", as illustrated in the screenshot below, along with your login name and password. This information must correspond to your WIREPAS licence details.

Example - Credentials input dialog WNT



[10] Wirepas Network Tool v4 Client User Guide (version: Thu, 13 Apr, 2023)

10 BLE ADVERTISING

It is possible to send non-connectable BLE advertising packets with BLUE MESH Beacons and Anchors at regular intervals. Currently, there are three different formats available: ID ELA, Eddystone UID and iBeacon. The frame specifications can be found in the table below[11], where "NwAdr" corresponds to the network address of the tag and "NodeAdr" to the address of the tag (sometimes also referred to as WirepasID).

Frame Specifications for BLE trams and their default values:

Format		ID ELA	Eddystone UID	iBeacon
Frames Bytes	1	Length : 0x02	Length : 0x02	Length : 0x02
	2	Type : 0x01	Type : 0x01	Type : 0x01
	3	Data : 0x06	Data : 0x06	Data : 0x06
	4	Length : <=0x16	Length : 0x03	Length : 0x1A
	5	Type : 0x09	Type : 0x03	Type : 0xFF
	6	Name[0]	Eddystone_UUID_LSB : 0xAA	Apple CIN_LSB : 0x4C
	7	Name[1]	Eddystone_UUID_MSB : 0xFE	Apple CIN_MSB : 0x00
	8	Name[2]	Length : 0x17	Beacon type : 0x02
	9	Name[3]	Type : 0x16	Data size : 0x15
	10	Name[4]	Eddystone_UUID_LSB : 0xAA	UUID[0] = 'w'
	11	Name[5]	Eddystone_UUID_MSB : 0xFE	UUID[1] = 'i'
	12	Name[6]	Frame type UUID : 0x00	UUID[2] = 'r'
	13	Name[7]	Power TX at 0m	UUID[3] = 'e'
	14	Name[8]	NID[0] = 'w'	UUID[4] = 'p'
	15	Name[9]	NID[1] = 'i'	UUID[5] = 'a'
	16	Name[10]	NID[2] = 'r'	UUID[6] = 's'
	17	Name[11]	NID[3] = 'e'	UUID[7] = ' '
	18	Name[12]	NID[4] = 'p'	UUID[8] = 'm'
	19	Name[13]	NID[5] = 'a'	UUID[9] = 'e'
	20	Name[14]	NID[6] = 's'	UUID[10] = 's'
	21		NID[7] = NwAdr[2]	UUID[11] = 'h'
	22		NID[8] = NwAdr[1]	UUID[12] = 0x00
	23		NID[9] = NwAdr[0]	UUID[13] = NwAdr[2]
	24		BID[0] = 0x00	UUID[14] = NwAdr[1]
	25		BID[1] = 0x00	UUID[15] = NwAdr[0]
	26		BID[2] = NodeAdr[3]	Major[0]
	27		BID[3] = NodeAdr[2]	Major[1]
	28		BID[4] = NodeAdr[1]	Minor[0]
	29		BID[5] = NodeAdr[0]	Minor[1]
	30		0x00 (RFU)	Power TX at 1m
	31		0x00 (RFU)	

[11] The devices are not sending a "Scan Response frame", as it is the case for other the ELA Bluetooth devices.

For the advertisement all three BLE advertising channels (37,38 and 39) are used in order to ensure reception. The transmission period can be varied between 1 and 10s and the transmit power can be varied and is independent of the Wirepas stack transmit power. There are two principal options for BLE advertising, either the tag is advertising all the time, or only if the tag is outside the Wirepas network coverage. Further information, also regarding task scheduling, can be found in the official Wirepas documentation[12].

For all formats, the physical address of the device is fixed and has the following format, based on the 32 bit device ID (NodeAdr):
{0xC0, 0x00, NodeAdr[3], NodeAdr[2], NodeAdr[1], NodeAdr[0]}

For configuring a device and to enable BLE advertising the following parameters are used:

Advertising BLE – Parameters 1

Parameter	Possible Values	Function	Availability
BLE Emulated Frame Type	0 1 2	ID ELA format Eddystone UID format iBeacon format	Anchor, Tag Mobile/ Mobile+/ AS
BLE Emulated	0 1 2	Disabled Activated all the time Activated only if the Device is outside of network coverage	Anchor, Tag Mobile/ Mobile+/ AS
BLE Emulated Period	[1000 – 10000] in ms	Period of the Emulated BLE Advertising in ms	Anchor, Tag Mobile/ Mobile+/ AS
Tx Power	[-8, -4, 0, +4] in dB	Transmit Power for BLE advertising	Anchors, Tag Mobile/ Mobile+/AS

In the basic "ID ELA" format the name of the tag is advertised, which can be modified. The "Eddystone UID" and the "iBeacon" format follow the Eddystone and iBeacon Protocol Specifications respectively, allowing for the modification of the NID and the BID (Eddystone) and the UUID, Major and Minor (iBeacon). The default values can be found in the frame specifications above and the corresponding values are summarized in the following table:

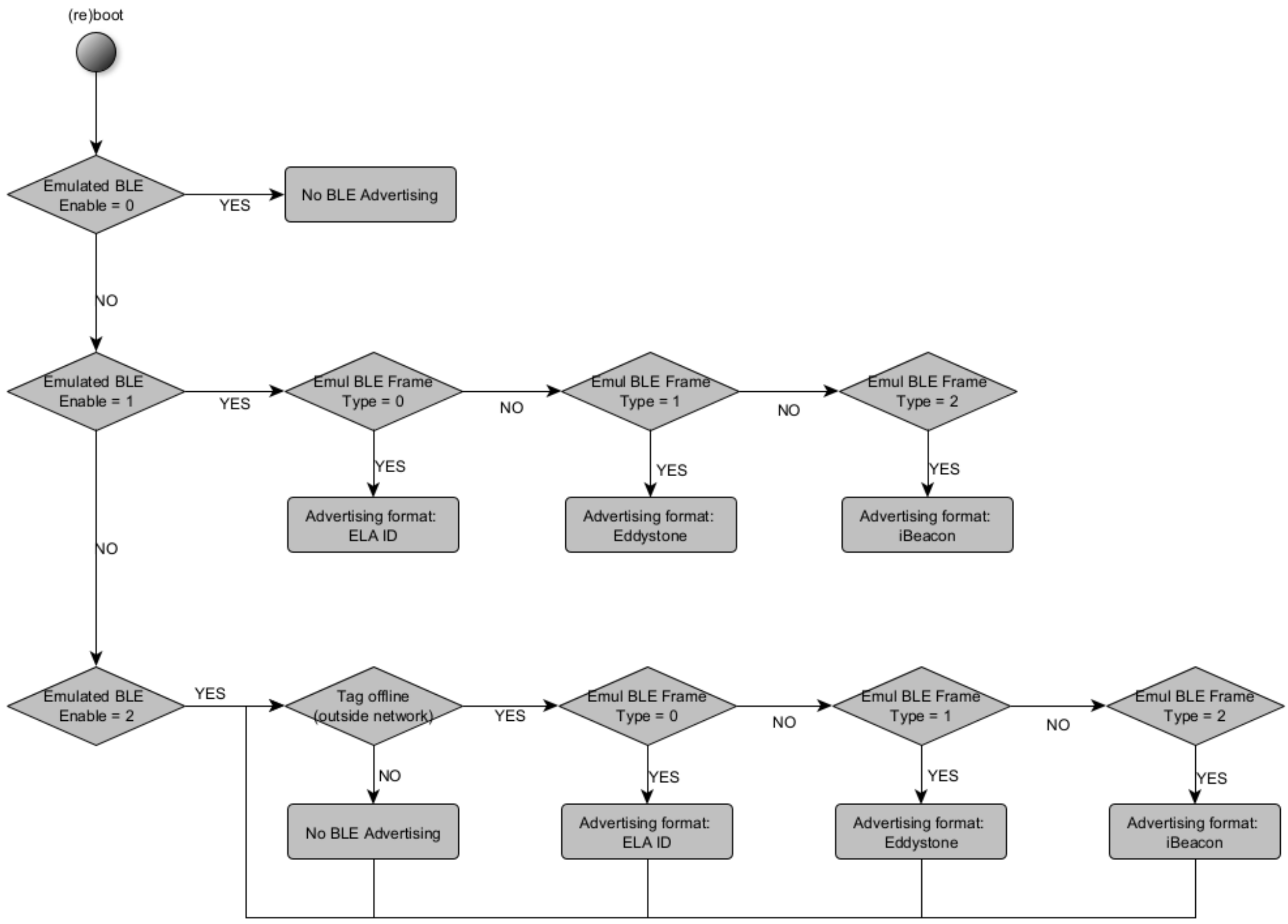
Advertising BLE – Parameters 2

Parameter	Possible Values	Function	Availability
Tag Name	Maximum 15 characters [0-9 ; A-Z ; a-z ; SPACE, _, -]	Name used for BLE advertising – format "ID ELA"	Anchors, Tag Mobile/ Mobile+/AS
UUID (iBeacon)	32 characters [0-9 ; A-F]	Definition of iBeacon UUID, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
Major (iBeacon)	4 characters [0-9 ; A-F]	Definition of iBeacon Major, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS

[12] Wirepas Mesh BLE advertising and scanning application note (version: Tue, 14 Feb, 2023)
Wirepas Positioning Application Reference Manual v1.5 (section: BLE beacons) (version: Wed, 21 Sep, 2022)

Minor (iBeacon)	4 characters [0-9 ; A-F]	Definition of iBeacon Minor, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
NID (Eddystone)	20 characters [0-9 ; A-F]	Definition of Eddystone NID, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
BID (Eddystone)	12 characters [0-9 ; A-F]	Definition of Eddystone BID, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS

All available options summarized in diagram below:



Important information: BLE advertising is only available for the formats Anchor (LE/LL), NRLS/NRLS+ and AS. It is not available for the Sensor formats.

11 OTAP

It is possible to perform a firmware update over-the-air (OTAP[13]) for all devices connected to a Wirepas Network. This method offers a fast and convenient way to deploy a new release. For further information, please contact the ELA customer support team, who will guide you through the procedure and provide all the necessary information. The following section explains the principal steps of the procedure.

Important note: Devices with a firmware version prior to v3.0.0 cannot be updated to the current v3.x.x series.

OTAP v2

The latest stack WP introduces a new and improved OTAP procedure called OTAP v2, which offers enhanced reliability and is easier to control. While the previous mechanism, Legacy OTAP, is still supported, it is strongly recommended to use the new procedure. In an OTAP procedure, it is possible to update all nodes and sinks in a Wirepas network. A significant improvement is that the sinks now serve as the control point for the update, eliminating the possibility of node-initiated OTAP, which has caused minor issues in the past.

An OTAP can be performed using the Wirepas Network Tool (WNT), presented in a previous section, or the Wirepas MQTT library. For more information on the latter, please contact Wirepas customer support directly. In this document, we will present the OTAP procedure using the WNT. It is important to note that OTAP is a sensitive operation. Before carrying out an OTAP, please read the following information and guidelines carefully:

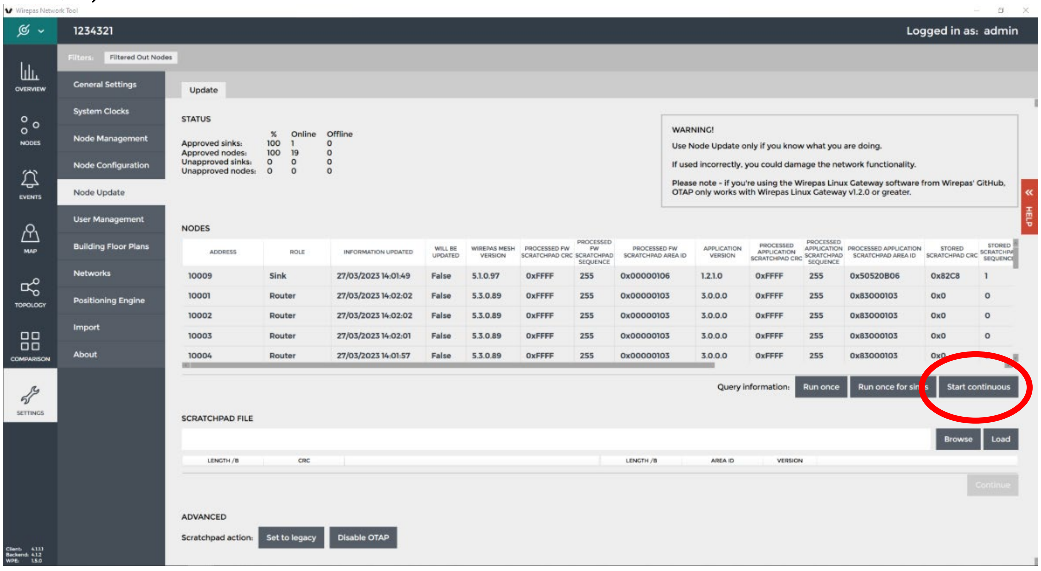
- Sinks and nodes have to be updated separately.
- The Wirepas Stack and the Application (provided by ELA) can be updated together or individually.
- The file containing the update is typically referred to as scratchpad file. It has the file ending *.otap. The ELA customer support will provide you with these files.
- For experienced users: It is no longer necessary to manually define a sequence number. A newer image is now identified explicitly by a given sequence number along with a specified CRC that is computed automatically.
- Once the OTAP has finished successfully, it is important to disable the OTAP procedure.

Node Update

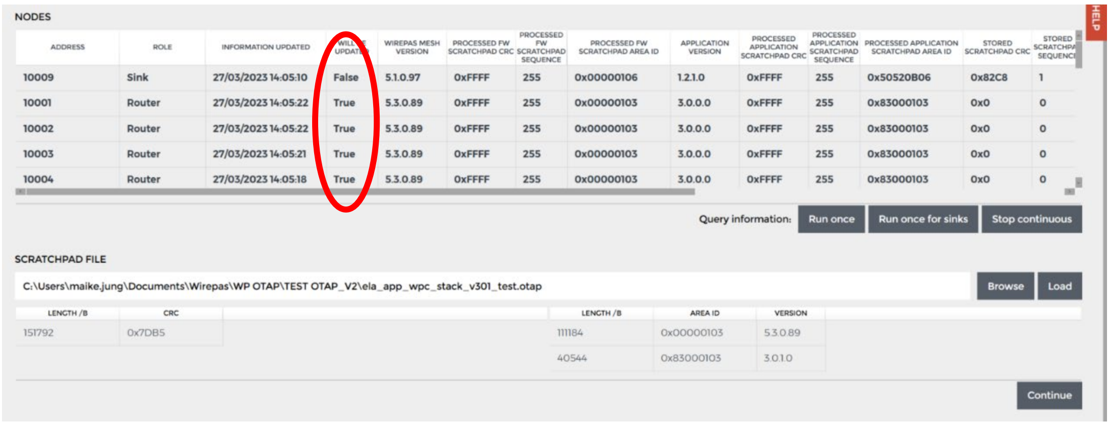
1. Open WNT and select the network to be updated
2. Select: Settings -> Node Update

[13] Short for Over-the-Air Programming

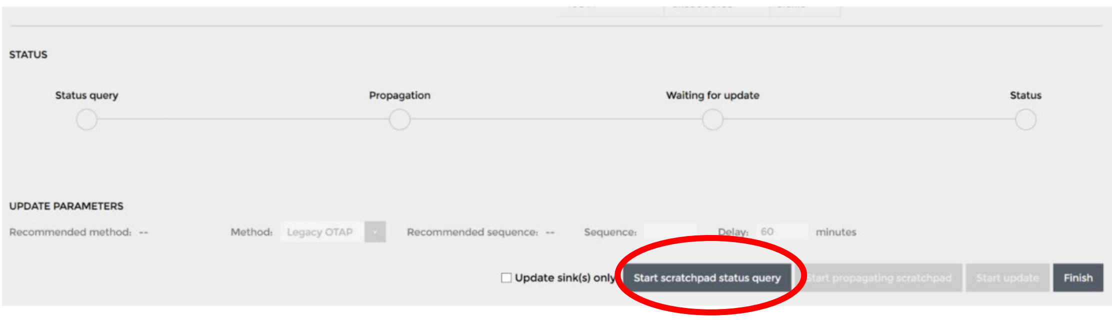
3. Click on “Start continuous” to request node information (firmware version, firmware and application area ID, …)



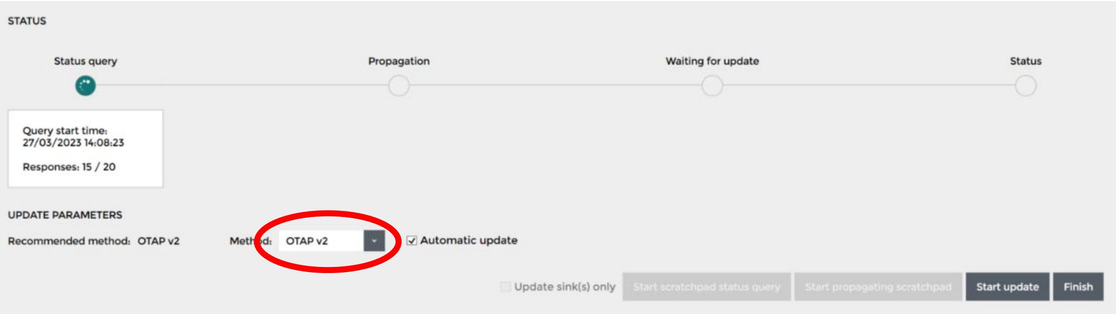
4. Select and Load the scratchpad file containing the update
5. Firmware and Application area ID need to match to be able to perform an OTAP
   -> “will be updated” will change to True if the node can be updated
6. Click “continue”



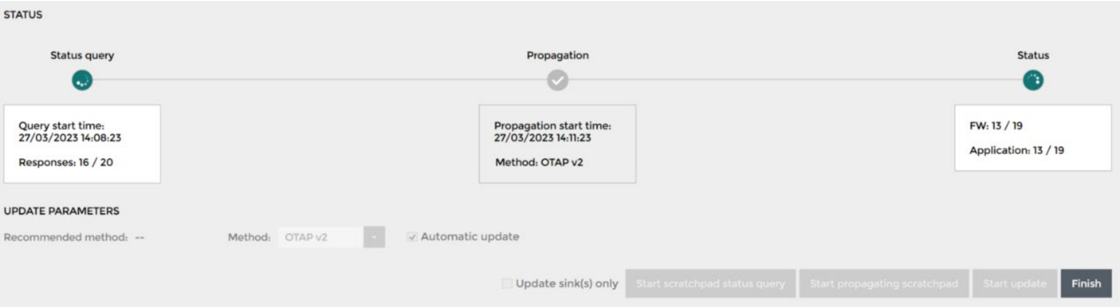
7. Select “Start scratchpad status query”
   (This operation can take some time, especially when the network contains tags with role NRLS/NRLS+ with a relatively long advertising period.)
8. It is not necessary to wait for all the nodes to respond, before continuing. However, it is encouraged to wait until the majority of the nodes have responded.



9. Select “Method”: OTAP v2 and select “Automatic update”
(The scratchpad will automatically be processed when it reaches a node.)



10. Click “Start update” to start the OTAP process
11. WAIT -> this step can again take some time, especially when the network contains tags with role NRLS/NRLS+ with a relatively long advertising period
12. The scratchpad will now propagate and be processed by the tags
13. The progress of the OTAP process is show on the right



14. Once all the nodes have responded, click “Finish”
15. The OTAP of the nodes is now completed.

Sink Update

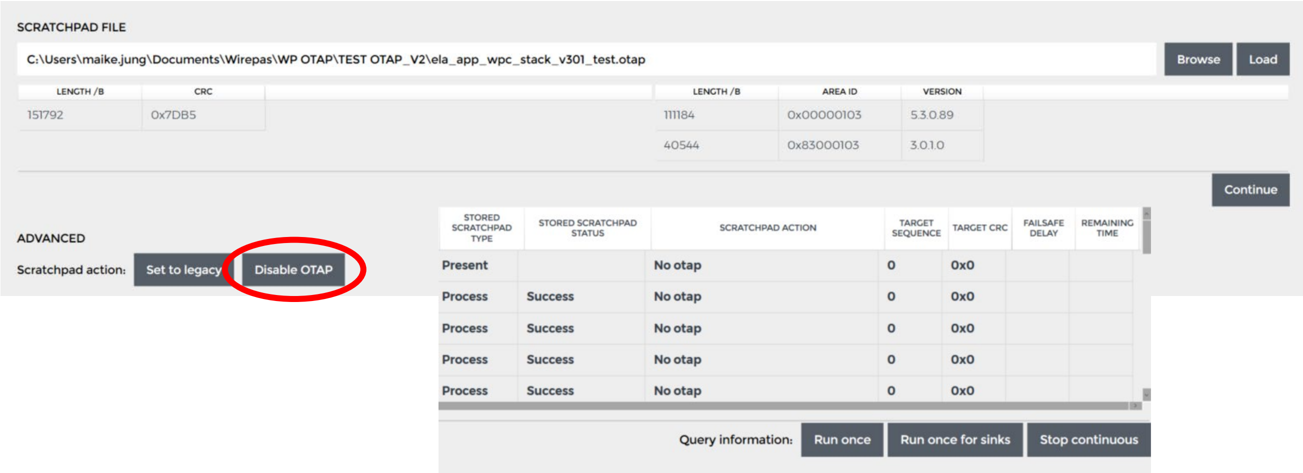
The sinks are updated separately, because they usually have a different firmware. An update of the application usually does not require an update of the sinks. In most cases this step is therefore not necessary.

To update the sinks:
- Upload the new scratchpad to the sinks (ensure that all sinks are running)
- Send a “process scratchpad locally” command to the sinks -> the update is started
-

Disable OTAP

Important: Do not leave out this step!

To avoid further propagation of the scratchpad file and uncontrolled OTAP of nodes
-> Click “Disable OTAP” (Once the OTAP has finished)
The scratchpad action should now change from “Propagate and process” to “No otap”, as illustrated below (this also takes some time).

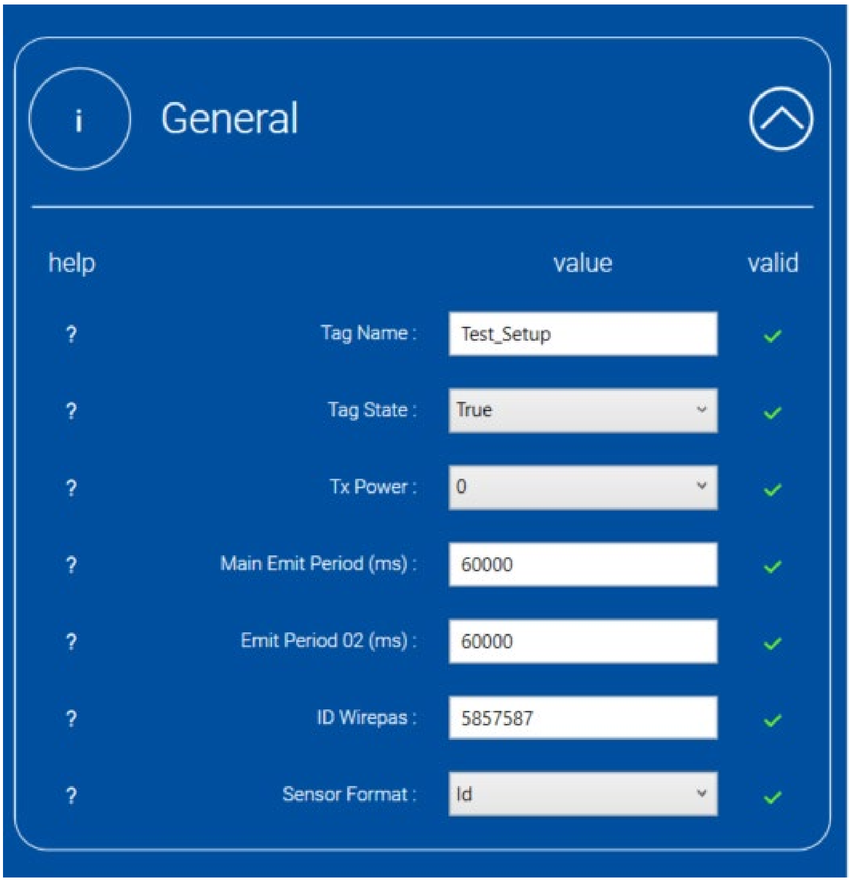

SCRATCHPAD FILE
C:\Users\maike.jung\Documents\Wirepas\WP OTAP\TEST OTAP_V2\ela_app_wpc_stack_v301_test.otap
Browse
Load
LENGTH /B
CRC
151792
0x7DB5
LENGTH /B
AREA ID
VERSION
111184
0x00000103
5.3.0.89
40544
0x83000103
3.0.1.0
Continue
ADVANCED
Scratchpad action:
Set to legacy
Disable OTAP
STORED SCRATCHPAD TYPE
STORED SCRATCHPAD STATUS
SCRATCHPAD ACTION
TARGET SEQUENCE
TARGET CRC
FAILSAFE DELAY
REMAINING TIME
Present
No otap
0
0x0
Process
Success
No otap
0
0x0
Process
Success
No otap
0
0x0
Process
Success
No otap
0
0x0
Process
Success
No otap
0
0x0
Query information:
Run once
Run once for sinks
Stop continuous

12 APPENDIX

<u>Configuration of Tags using Device Manager (Supplementary information)</u>

In the following, you can find a comprehensive list of all the parameters that can be modified using the Device Manager application, along with their respective ranges.

General Settings – Device Manager

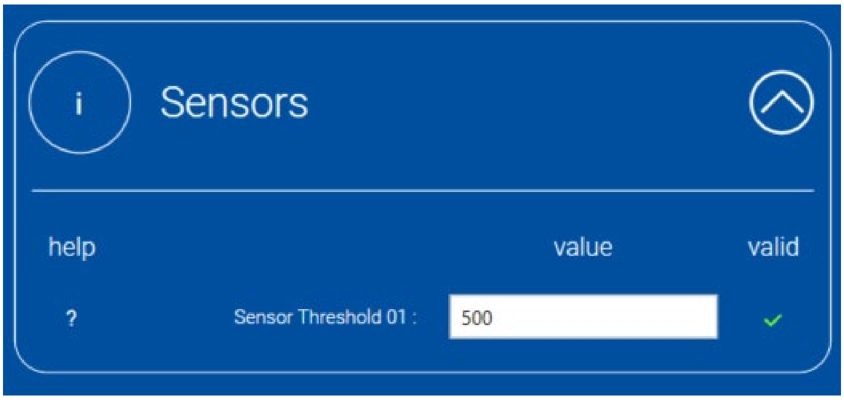


General Settings – Parameters

Parameter	Possible Values	Function	Availability
Tag Name	Maximum 15 characters [0-9 ; A-Z ; a-z ; SPACE, _, -]	Name used for BLE advertising – format "ID ELA"	Anchors, Tag Mobile/ Mobile+/AS
Tag State	True/ False	True: Device is enabled and operational False: Turn OFF device	all
Tx Power	[-8, -4, 0, +4] in dB	Transmit Power for BLE advertising	Anchors, Tag Mobile/ Mobile+/AS
Main Emit Period	[8000 - 86400000] in ms	Standard duration between two consecutive advertising events	all
Emit Period 2	[8000 - 86400000] in ms	Duration between two advertising events upon signal detection (Sensor and NRLS+ formats only!)	Sensor/ NRLS+

Wirepas ID (aka Node Address)	32 bits [0x00000000 – 0xFFFFFFFD]	Unique Address of the Device	all
Sensor Format	[ID, T, RHT, MAG, MOV, ANG, DI, DO ,AI, PIR, T Probe, TOUCH, ProxIR]	Select the sensor format of the device (ID: no sensor)	all (if the respective sensor is present on the device)

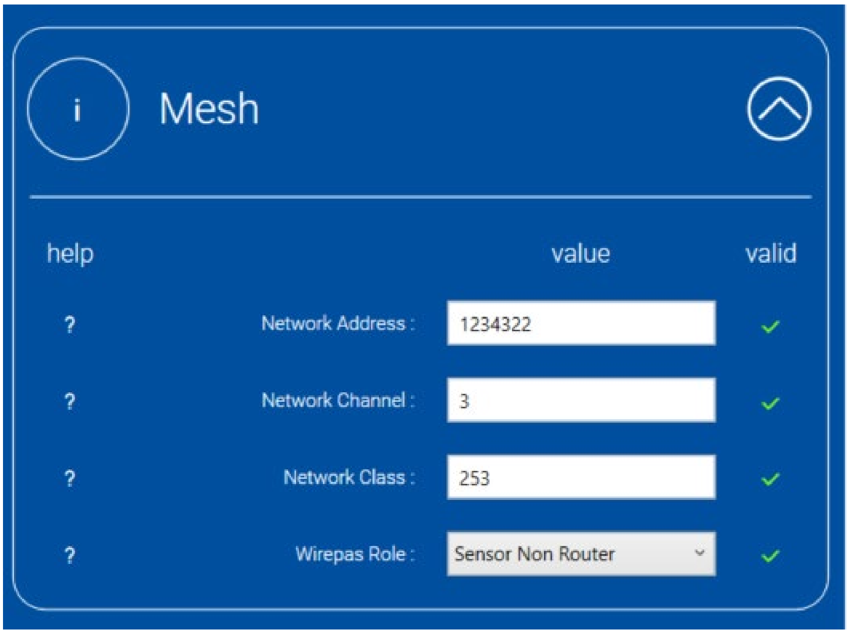
Sensor Settings – Device Manager



Sensor Settings – Parameters

Parameter	Possible Values	Function	Availability
Sensor Threshold	[32 – 8000] in mg	Sensor threshold for the motion detection Sensor (MOV) (mg)	Devices with motion Sensor/ Mobile+

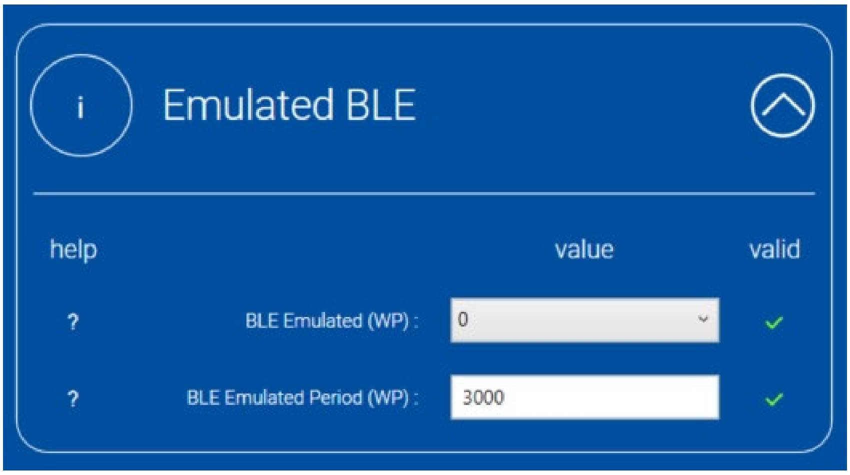
Network Setting – Device Manager



Network Settings – Parameters

Parameter	Possible Values	Function	Availability
Network Address	u32 [0x00000000 – 0xFFFFFFFD]	Address of the Wirepas Network	all
Network Channel	[1 – 40]	Network Channel of the Wirepas Network	all
Network Class	[0xF9; 0xFA; … ; 0xFF]	Class of the network or a subset of the network	all
Wirepas Role	Anchor LE (Low Energy) Anchor LL (low Latency) Sensor Router Sensor Non Router Sensor Autorole NRLS (former Mobile) NRLS+ (former Mobile+) AS (former HC)	Role of the device	all

Emulated BLE Setting – Device Manager



Emulated BLE Settings – Parameters

Parameter	Possible Values	Function	Availability
BLE Emulated	0 1 2	Disabled Activated all the time Activated only if the Device is outside of network coverage	Anchor, Tag Mobile/ Mobile+/ AS
BLE Emulated Period	[1000 – 10000] in ms	Period of the Emulated BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS

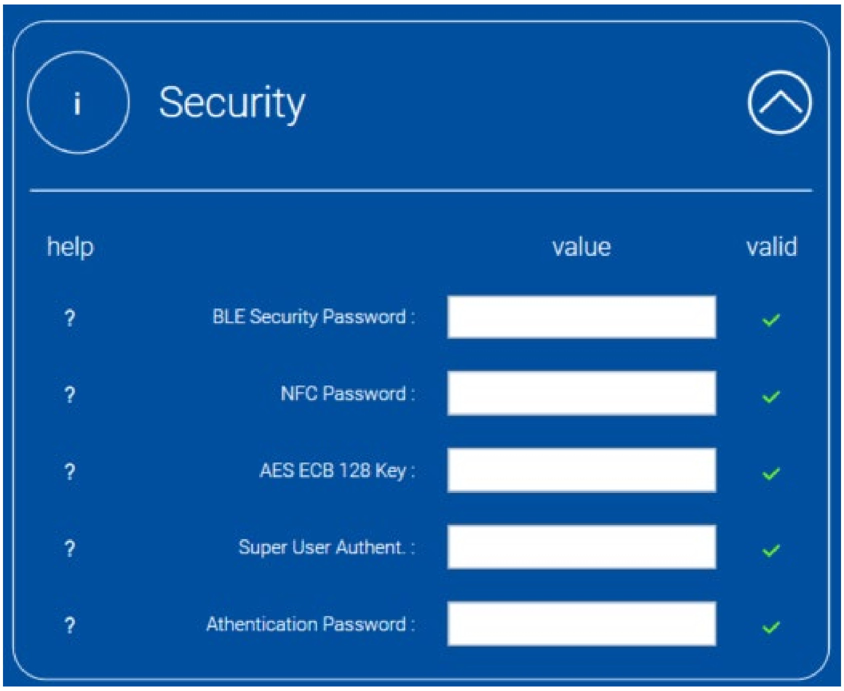
Emulated BLE Setting (Part 2) – Device Manager

Emulated BLE Settings (Part 2) – Parameters

Parameter	Possible Values	Function	Availability
UUID (iBeacon)	32 characters [0-9 ; A-F]	Definition of iBeacon UUID, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
Major (iBeacon)	4 characters [0-9 ; A-F]	Definition of iBeacon Major, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
Minor (iBeacon)	4 characters [0-9 ; A-F]	Definition of iBeacon Minor, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
NID (Eddystone)	20 characters [0-9 ; A-F]	Definition of Eddystone NID, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS
BID (Eddystone)	12 characters [0-9 ; A-F]	Definition of Eddystone BID, transmitted by BLE Advertising	Anchor, Tag Mobile/ Mobile+/ AS

Security Setting – Device Manager

➔ can be used to modify the password used for device configuration over the network



Security Settings – Parameters

Parameter	Possible Values	Function	Availability
BLE Security Password	0x00000000000000000000 – 0xFFFFFFFFFFFFFFFFFFFF Default = 0x454C4131323334	Field to change the BLE Security password. Enter here your new password	all
Authentication Password	0xE62DD700	Authentication password to change the BLE Security password. This must be entered in the same time as changing the BLE Security password	all

<u>Appdata CMD frame making process</u>

To create appdata CMD frames, the following frame structure (value in hexadecimal) must be respected if:

- You want to change a parameter: 0x1A(LL+4)0010LLCC10EF(PL+2)01(PL)PPVV
- You want to send a command: 0x1B(LL+4)0020LLCCVV

With:

- LL+4: The length LL added of 4
- LL: The number of bytes of the data value (VV) (min: 0x02, max: 0xFF)

- CC: The CRC8-Maxim calculated on the entire frame[14]
- PL+2: The BLE Security password length added of 2
- PL: The BLE Security password length
- PP: The BLE Security password
- VV: The data value

The data value follows a TLV frame format. The type will give the parameter/command we want to interact with, the length will be the length of the value and the value will give the argument for the parameter/command we want to use.

/!\: In the value section, the data must be filled following the little-endian format.

In the following table, you will find the data value characteristics for the different parameter/command available in the ELA tags:

Characteristic	Type	Length	**Data** (VAL_X = value to set)	Example
Main emit period (ms)	**0x1037**	**0x06**	**0x0104VAL_0VAL_1VAL_2VAL_3**	**3600s = 3600000ms: 0x103706010480EE3600**
Emit period 02 (ms)	**0x1038**	**0x06**	**0x0104VAL_0VAL_1VAL_2VAL_3**	**7200s = 7200000ms:** **0x103806010400DD6D00**
Acceleration threshold (mg)	**0x1054**	**0x04**	**0x0102VAL_0VAL_1**	**1000mg:** **0x1054040102E803**
Activation BLE (0/1)	**0x103D**	**0x03**	**0x0101VAL_0**	**Activation:** **0x103D03010101** **Deactivation:** **0x103D03010100**
BLE Tx period (ms)	**0x103E**	**0x06**	**0x0104VAL_0VAL_1VAL_2VAL_3**	**5s = 5000ms:** **0x103E06010488130000**
Tx Power (dBm)	**0x1031**	**0x03**	**0x0101VAL_0**	**+4dBm:** **0x103103010104**
LED ON (time s)	**0x2001**	**0x02**	**VAL_0VAL_1**	**10s:** **0x2001020A00** **Note: A time of 0s will blink the LED indefinitely**
LED OFF	**0x2002**	**0x00**		**0x200200**

[14] See following website to calculate CRC8-Maxim: https://tomeko.net/online_tools/crc8.php?lang=en

BUZZ ON (time s)	**0x2003**	**0x02**	**VAL_0VAL_1**	**5s:** **0x2003020500** **Note: A time of 0s will turn on the buzzer indefinitely**
BUZZ OFF	**0x2004**	**0x00**		**0x200400**
LED BUZZ ON (time s)	**0x2005**	**0x02**	**VAL_0VAL_1**	**20s:** **0x2005021400** **Note: A time of 0s will turn on the led and the buzzer indefinitely**
LED BUZZ OFF	**0x2006**	**0x00**		**0x200600**

The process to construct a CMD frame, with the default BLE Security password, is the following (example: set main emit period at 3600s):

- 1) Create or take a data value from the above table (0x103706010480EE3600)
- 2) Replace the VV field of the frame structure by the data value:
  0x0010LLCC10EF090107454C41313233334**103706010480EE3600**
- 3) Calcul the number of bytes of the frame (i.e. number of bytes after the CRC (CC) field) and replace the LL field in the frame structure:
  0x0010**15**CC10EF090107454C41313233334103706010480EE3600
- 4) Calcul CRC8-Maxim on the total frame (without the CRC field) and replace the CRC field in the frame structure.
  - 4-1) Remove the CC field in the frame:
    0x00101510EF090107454C41313233334103706010480EE3600
  - 4-2) Calcul the CRC8-Maxim on this frame: 0xFF
  - 4-3) Replace the calculated CRC8-Maxim in the CRC field of the frame:
    0x001015**FF**10EF090107454C41313233334103706010480EE3600
- 5) Replace the (LL+4) field in the 0x1A(LL+4) preamble by the number of bytes of the VV field (equaled in this example as 0x15 calculated in point 3): 0x1A19
- 6) Place the preamble at the beginning of the constructed frame:
  0x**1A19**001015FF10EF090107454C41313233334103706010480EE3600
- 5) Your frame is now ready to be encapsulated as described in AppConfig frames in the **Erreur ! Source du renvoi introuvable.** section by replacing the **CMD** field by the CMD data frame created

Here are some common examples of data frames:

- Set main emit period at 3600s:
  0x1A19001015FF10EF090107454C41313233334103706010480EE3600
- Set emit pediod 02 at 7200s:
  0x1A19001015B210EF090107454C41313233334103806010400DD6D00
- Set acceleration threshold to 1000mg:
  0x1A170010134B10EF090107454C4131323333410540401 02E803

- Set emulated BLE function: 0x1A160010126B10EF090107454C4131323334103D03010101
- Set BLE Tx emit period at 5s: 0x1A190010159010EF090107454C4131323334103E06010488130000
- Set Tx Power at +4dBm: 0x1A160010127510EF090107454C4131323334103103010104
- Turn the LED ON for 6 seconds: 0x1B09002005962001020600
- Turn the LED OFF: 0x1B0700200375200200
- Turn the buzzer ON for 5 seconds: 0x1B09002005C42003020500
- Turn the buzzer OFF: 0x1B07002003DF200400
- Turn the LED and the buzzer ON for 20 seconds: 0x1B09002005E52005021400
- Turn the LED and the buzzer OFF: 0x1B070020034E200600

Sensor Data Interpretation - Examples

In the following some examples on how to interpret the data received for the different sensor formats can be found. General information on the different data formats and how they are encoded can be found in section Data Received.

Examples of data received for the different sensor formats:

Functionality/ Sensor Type	Endpoint (EP)	Data			Result	
	Source/ destination	Type	Length	Data Received (exemplary)	Interpreted Data	Value
Tension Batterie	11/11	01	02	BD 0B	0x0BBD	3005 mV = 3.005 V
Temperature (T)	100/100	02	02	92 0B	0x0B92	2962 c°C = 29.62°C
Relative Humidity and Temperature (RHT)	110/110	03	04	27 00 BA 0B	0x0027	39 %
					0x0BBA	3002 c°C = 30.02°C
Digital Input (DI)	120/120	04	06	01 00 2A 00 00 00	00 01	activated: 01
					00 00 00 2A	Counter: 0x2A = 42
				00 00 2A 00 00 00	00 00	deactivated: 00
					00 00 00 2A	Counter: 0x2A = 42
Digital Output (DO)	130/130	05	06	01 00 01 36 00 00	00 01	activated: 01
					00 00 36 01	Counter: 0x3601 = 13825
				00 00 01 36 00 00	00 00	deactivated: 00

					00 00 36 01	Counter: 0x3601 = 13825
Magnetic Field Detection (MAG)	150/150	07	06	01 00 B5 00 00 00	00 01	detected: 01
					00 00 00 B5	Counter: 0xB5 = 181
				00 00 B5 00 00 00	00 00	not detected: 00
					00 00 00 B5	Counter: 0xB5 = 181
Movement Detection(MOV)	160/160	08	06	01 00 2A 00 00 00	00 01	Movement detected: 01
					00 00 00 2A	Counter: 0x2A = 42
				00 00 2A 00 00 00	00 00	No movement: 00
					00 00 00 2A	Counter: 0x2A = 42
Acceleration (ANG)	170/170	09	06	B8 00 58 FF 8E 04	00 B8	a_x: 0x00B8 → 184 mg
					FF 58	a_y: 0xFF58 → -168 mg (signed 2's complement)
					04 8E	a_z: 0x048E → 1166 mg
Analogue Input (AI)	180/180	0A	02	C7 09	09 C7	Tension: 0x09C7 → 2503 mV
Presence Detection (PIR)	200/200	0C	06	01 00 05 00 00 00	00 01	Presence detected: 01
					00 00 00 05	Counter: 5
				00 00 0A 00 00 00	00 00	No presence detected: 00
					00 00 00 0A	Counter: 0x0A = 10