# ThingPark Wireless PHY and MAC layer specifications



| Version | By | Date | |
|---|---|---|---|
| 0.1 | OH | 28/02/2013 | Initial slide deck proposing LoRa MAC based on compressed 802.15.4 to Semtech |
| 0.8 | OH | 20/11/2013 | Distribution for review by NS / Semtech |
| 0.9 | OH | 03/12/2013 | Updated header (remarks from NS that LoRA transmits full bytes) |
| 1.0 | OH | 27/12/2013 | Aligned PHY and MAC with LoRa MAC R1 + Review for publication. |

## Overview

The Medium Access Control (MAC) Layer is a critical part of a radio network design. It ensures bootstrapping (e.g. allocation of network addresses, creation of session keys), authentication and protection of data and also ensures proper scaling of the network (collision avoidance, transmit power management). The MAC layer must take into account the specific properties of the PHY layer in order to make sure network links always operate in the optimal PHY settings for given link budget and network load conditions.

As requirements evolve and network functionality is enhanced, the MAC layer design must also ensure that enhanced communication can be introduced without requiring a complete re-commissioning of the network.

This document explains how these design objectives are achieved in the ThingPark Wireless infrastructure.

# Table of contents

# Abbreviations

| | |
|---|---|
| ACK | Acknowledge |
| ADR | Adaptive Data Rate |
| AFA | Adaptive Frequency Agility |
| AppSKey | Application Session Key |
| CAD | Channel Activity Detection |
| CEPT | European Conference of Postal and Telecom administrations (www.cept.org) |
| CR | Coding Rate |
| CRC | Cyclic Redundancy Check code |
| CSL | Coordinated Sampled Listening |
| DC | (in this context) Duty Cycle |
| DevAddr | Device Address |
| FHDR | Frame Header |
| FSK | Frequency Shift Keying modulation |
| G-MSK | Gaussian Minimal Shift Keying modulation |
| IEEE | Institute of Electrical and Electronics Engineers (www.ieee.org) |
| LBT | Listen Before Talk |
| M2M | Machine to machine communications |
| MAC | Medium Access Control layer |
| MHDR | MAC header |
| MIC | Message Integrity Code |
| NwkSKey | Network Session Key |
| OTA | Over The Air |
| PDU | Packet Data Unit |
| PHY | Physical layer |
| RSSI | Received Signal Strength Indication |
| RX | Reception |
| SF | Spreading Factor |
| TX | Transmission |

# PHY layer

The physical layer is based on Semtech LoRa™ spread spectrum chipsets. The main settings are outlined in the following sections.

## PHY radio channels

### Available channels and PHY layer performance

Contrarily to other spread spectrum technologies, the LoRa™ receiver is compatible with mobile nodes, as its correlation technology supports a strong Doppler effect.

The channels usable in Europe in the 868MHz frequency for LoRa spread spectrum modulation are presented in Figure 1:

| Modulation | Bandwidth [kHz] | Channel Frequency [MHz] | FSK Bitrate LoRa™ Spreading Factor | Nb channels | Sub-band | Comments |
|---|---|---|---|---|---|---|
| **LoRa™** | 250 | 868.30 | SF7: 10 kbps | 1 | g1 | [LC7]Bidirectional |
| **FSK** | 250 | 868.30 | 100 kbps | 1 | g1 | [FC1]Bidirectional |
| **LoRa™** | 125 | 868.10<br>868.30<br>868.50 | SF7-SF12<br>0.3 kbps-5 kbps | 3 | g1 | [LC1]Bidirectional<br>[LC2]Bidirectional<br>[LC3]Bidirectional |
| **LoRa™** | 125 | 868.85<br>869.05 | SF7-SF12<br>0.3 kbps-5 kbps | 2 | g2 | [LC4]Bidirectional<br>[LC5]Bidirectional |
| **LoRa™** | 125 | 869.525 | SF7-SF12<br>0.3 kbps-5 kbps | 1 | g3 | [LC6]Bidirectional |

Figure 1: RF channels used by ThingPark Wireless in Europe

The spectrum uses the unlicensed bands reserved by CEPT Electronic Communications Committee, recommendation ERC 70-03 (1997, updated oct. 9th, 2013) in annex 1 for “Non-specific Short range devices”, and allowing spread spectrum modulation. This recommendation provides the technical constraints for channels g1, g2, g3 as outlined in Figure 2 and Figure 3.

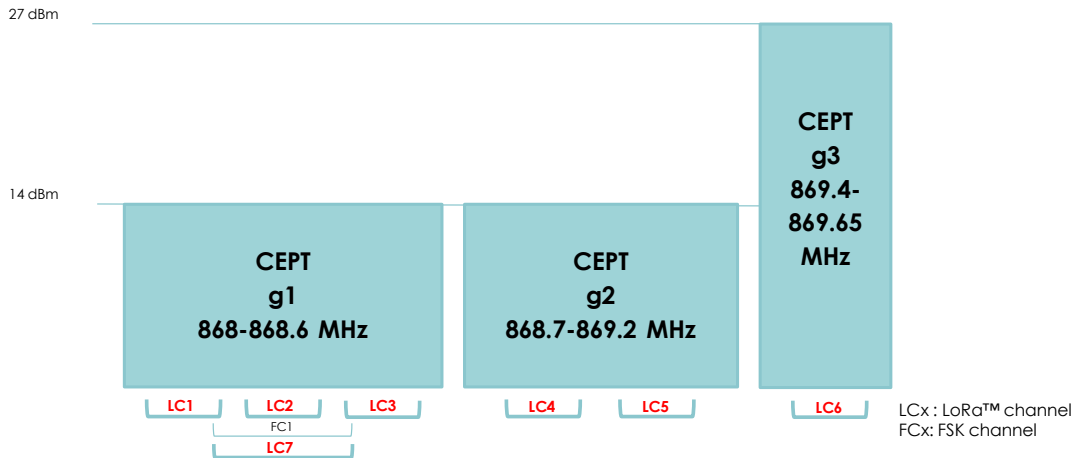



Figure 2: Use of CEPT ERC 70-03 annex 1 g1-g3 channels by ThingPark Wireless

| Band | Edge Frequencies | | Field / Power | Spectrum Access | Band Width |
|---|---|---|---|---|---|
| | Fe- | Fe+ | | | |
| **g** | 865 MHz | 868 MHz | +6.2 dBm / 100 kHz | 1 % or LBT AFA | 3 MHz |
| **g** | 865 MHz | 870 MHz | -0.8 dBm / 100 kHz | 0.1% or LBT AFA | 5 MHz |
| **g1** | 868 MHz | 868.6 MHz | 14 dBm | 1 % or LBT AFA | 600 kHz |
| **g2** | 868.7 MHz | 869.2 MHz | 14 dBm | 0.1% or LBT AFA | 500 kHz |
| **g3** | 869.4 MHz | 869.65 MHz | 27 dBm | 10 % or LBT AFA | 250 kHz |
| **g4** | 869.7 MHz | 870 MHz | 7 dBm | No requirement | 300 kHz |
| **g4** | 869.7 MHz | 870 MHz | 14 dBm | 1 % or LBT AFA | 300 kHz |

Figure 3: CEPT ERC 70-03 technical constraints for channels g, g1, g2, g3 supporting wideband modulation

The channels impose a transmission duty cycle limit or use of a Listen Before Talk Adaptive Frequency Agility (see section Channel selection and access).

ThingPark Wireless uses the instant synchronization property of Semtech LoRa™ technology to dynamically adapt the spreading factor and coding rate to the actual link budget between any node and its bases station(s), among the available options listed in Figure 4.

**Base station key performance indicators**

(Co-channel GMSK rejection : 25dB)

| Spreading factor | Chips /symbol | 125kHz (coding 4/5) | 250kHz (coding 4/5) | 500kHz (coding 4/5) |
|---|---|---|---|---|
| 6 | 64 | 9380bps (-127.5dBm) | 18750bps (-124.5dBm) | 37500bps (-111.5dBm) |
| 7 | 128 | 5469bps(-130.0dBm) | 10938bbs (-127.0dBm) | 21875bps (-124.0dBm) |
| 8 | 256 | 3125bps(-132.5dBm) | 6250bps (-129.5dBm) | 12500bps (-126.5dBm) |
| 9 | 512 | 1758bps(-135.0dBm) | 3516bps (-132dBm) | 7031bps (-129.0dBm) |
| 10 | 1024 | 977bps(-137.5dBm) | 1953bps (-134.5dBm) | 3906bps (-131.5dBm) |
| 11 | 2048 | 537bps(-140.0dBm) | 1074bps (-137.0dBm) | 2148bps (-134.0dBm) |
| 12 | 4096 | 293bps (-142.5dBm) | 586bps(-139.5dBm) | 1172bps (-136.5dBm) |

SNR for 32 octets @ 10% PER

Available cyclic codes for forward error correction:

| Coding rate | Overhead |
|---|---|
| 4/5 | 1.25 |
| 4/6 | 1.5 |
| 4/7 | 1.75 |
| 4/8 | 2 |

**Figure 4: ThingPark Wireless automatically selects most efficient LoRa bitrate and sensitivity options**

The automatic selection of fastest possible transmit bitrates (lower spreading factors which reduce airtime and collisions) is a key contributor to the performance of ThingPark Wireless networks, which achieve a cell spectral efficiency about 5 to 10 times that of equivalent ultra-narrowband M2M networks, despite use of frequency spreading. See section Link Adaptative Data Rate (ADR) for more details.

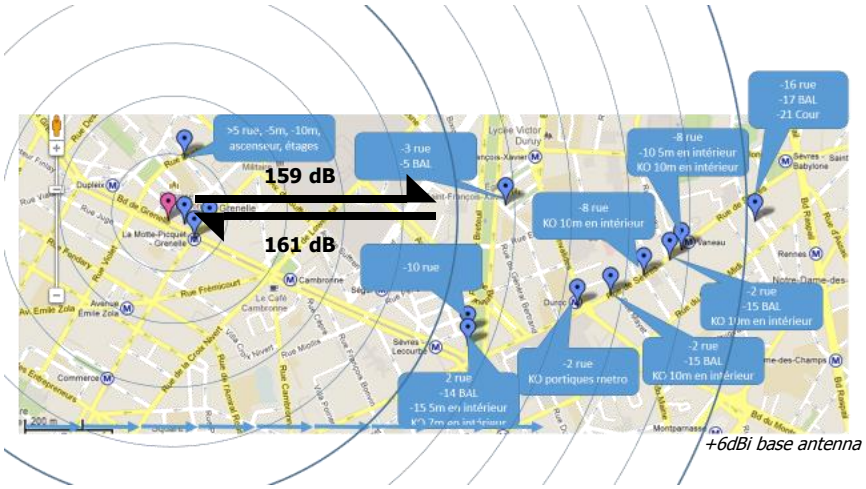
The typical maximum bidirectional link budget in a cell with 25mW (14dBm) node and base transmitters is about 159dB decomposed as follows:

| **Uplink** | **Downlink** |
|---|---|
| Node TX power : 14dBm | Base station TX power (g3): 21dBm[1] |
| Max. path loss : 162dB | Max path loss: 159dB |
| Rx Antenna Gain : 6dB | Rx Antenna Gain : 0dB |
| Base station sensitivity : -142dBm | Node sensitivity :-138 dBm |

The link margin of first generation 2G GSM networks was approximately 150dB: due to the higher performance of LoRa, and despite of the much lower transmit power, equivalent ThingPark Wireless coverage may be achieved by equipping only 1/4th of first generation 2G base stations.

The network cells are dimensioned so that 99% of nodes may connect with a 4dBm margin (Figure 5).

[1] ETSI power density limitations limit the maximum power density, therefore the 6dB antenna gain must be taken into account : 27dBm=21dBm+6dB




| % of nodes | Link budget |
|---|---|
| 10% | <120dB |
| 50% | <138dB |
| 90% | <148dB |
| 99% | <155dB |

**Figure 5: Typical cell link budget, and link budget distribution among cell nodes**

## Channel selection and access

Nodes perform a LoRa™ channel activity detection (CAD) on the selected channel before accessing it. The CAD sequence lasts for about 3 symbols and has a reliability of 99.9%. In addition a listen before talk (LBT) procedure makes sure that the measured RSSI is lower than 90dBm before any transmission.

Any transmitting node must comply with the duty cycle limit of the corresponding channel (the duty cycle is limited individually for each channel). In addition a given channel must never be accessed more than once every 2 seconds. Nodes must select channels pseudo-randomly for each transmission so this is not a strong constraint for responsiveness of the communication to fast events (each transmission may use a different channel), as long as such event bursts remain exceptional.

Single channel private networks:

If the level of signal detected during LBT is above -90dBm, then the node delays transmission by a random delay value, and restarts LBT.

Multichannel public networks:

If the level of signal detected during LBT is above -90dBm, then the node selects a new channel, and restarts LBT.

After accessing a channel and transmitting a PHY PDU, the node must select another channel for the next transmission.

## End device commissioning

Each end device must be commissioned with the following information:

- DevAddr: A device ID of 32 bits that uniquely identifies the end device. This identifier is provided by Actility, or during the network join procedure if supported by the node.

- NwkSKey: A device-specific network session key, which is used to compute the data integrity MIC code, and encryption/decryption of payload field between ThingPark wireless network and the end device.
- AppSKey: One or more device-specific application session keys used for end to end encryption of the payload field for certain application ports (see section Application port). If this AppSKey is not provided to ThingPark wireless, then network cannot access the payload information.

Alternatively, an end device may support Over The Air activation (see section Annex A: optional Over-The-Air device activation using Join request/response), in which case the device needs to be commissioned with non-network specific data only, and the session keys are derived during the OTA activation process.

## PHY layer PDU format

At the lowest datarate of 300bps, the PHY layer frames should not exceed 64 bytes in order to maintain the transmission time below 2 seconds.

In the following sections, the octet order for all multi-octet fields is *little endian*.

### Uplink PDU from nodes

All node transmissions uses the LoRa™ radio PDU explicit mode: the LoRa™ PHY header (PHDR) with PHY header CRC is included on the PHY layer PDU. The optional PHY PDU CRC is enabled and added by the transceiver.

| Preamble | PHDR | PHY_Header_CRC | PHY_Payload | PHY_Payload_CRC |
|---|---|---|---|---|

**Figure 6 – Node PHY PDU transmission format**

### Downlink PDUs from base stations

#### *Beacon PDUs*

Beacon PDU transmissions uses the LoRa™ radio implicit mode: there is no LoRa™ header, and no PHY PDU level CRC.

| Preamble | Payload |
|---|---|

**Figure 7 – Gateway beacon PDU transmission format**

#### *Normal packets (Downlink)*

Normal packets transmission uses the LoRa™ radio packet explicit mode: the LoRa™ header with header CRC is included on the PHY layer PDU. The optional PHY PDU CRC is disabled in order to keep the PDU as short as possible.

| Preamble | PHDR | Phy_Header_CRC | PHY_Payload |
|---|---|---|---|

**Figure 8 – Gateway normal packet transmission format**

#### *Spreading sequence*

The spreading sequence of the Downlink depends on the deployment context of the ThingPark wireless system:

- For private, medium capacity systems, the spreading sequences used for downlink are the same as those used for the uplink.

- For public, high capacity networks, the spreading sequences used for the downlink and the spreading sequences used for the uplink are orthogonal (they do not see each other, and interfere as white noise would do). This makes it possible to transmit ACKs from base stations with a precise timing, as there is no risk of collision and therefore no need to perform a channel activity detection (CAD) sequence for downlink communication.

## Bidirectional communication

### Class A1 nodes

Class A1 nodes provide only mono-directional communication from a user point of view, and the node application may only transmit unconfirmed data frames (therefore Class A1 nodes not consume the critical TX duty cycle of base stations). However, from a PHY and MAC layer perspective, these sensors are bidirectional, and do implement spreading factor and power management primitives, as the latter are essential to ensure proper management and scaling of ThingPark Wireless networks.

Class A1 nodes must wake up and listen for an incoming MAC command about one second after each transmission, and optionally once more (see Class 2 nodes).

### Class A2 nodes

Class A2 nodes use a Receiver Initiated Transmission (RIT) strategy for bidirectional communication. After any transmission, the node:

- waits in sleep mode for RX_DELAY_1 seconds minus low power clock drift during RX_DELAY_1 (DRIFT_1),
- then uses Channel Activity Detection (CAD) on the same channel to detect incoming packets for 2*DRIFT_1 + 5 symbol intervals (depends on data rate used).

The node may iterate the periodic listening process more than once, according to configuration of the node in the ThingPark Wireless network database (e.g. returns to sleep for RX_DELAY_2 seconds minus clock drift DRIFT_2 during RX_DELAY_1 + RX_DELAY_2, then uses CAD for 5 symbol intervals plus 2*DRIFT_2, etc.).

A node is not allowed to begin a new transmission before the second receive window has elapsed.

Class A2 nodes may support multicast transmission from the network, but multicast packets towards groups subscribed by the node are actually delayed by the network until the next RIT cycle of the device. Multicast is not more efficient than multi-unicast for class A2 nodes.

### Class B nodes

Class B nodes use Coordinated Sampled Listening (CSL) strategy for bidirectional communication with low downlink latency, and adding the possibility of network initiated transactions. The node periodically listens to the channel at a known instant and may be “pinged” by the network infrastructure during those slots. The node synchronizes its listening slots to a beacon transmitted periodically by the gateways.

Class B devices are battery powered or mains powered and target all kinds of applications where real time control may be necessary. Class B nodes may support multicast: unconfirmed multicast frames are transmitted only once by the network towards all class 3 nodes.

*Class B node operation is still under development and will be added in later releases of this specification document.*

# MAC layer

## Relationship to IEEE 802.15.4 MAC and 6LoWPAN

IEEE 802.15.4 is a reference MAC layer specification, which incorporates the know-how and feedback of many experts and years of deployment experience. As such, it is the foundation of many application layer specifications (e.g. ZigBee application layer, ISA 100.11a, 6LoWPAN…), and well tested security frameworks are also available on top of the IEEE 802.15.4 MAC layer.

For all these reasons, one of our key design objectives of the ThingPark MAC layer was to keep a full compatibility with the 802.15.4 MAC frame format, in order to facilitate the adaptation of existing application and security layers. Standard 802.15.4 frames may be compressed into ThingPark MAC frames, and vice-versa ThingPark MAC frames may uncompressed into 802.15.4 frames.

The adaptation of 802.15.4 MAC frame format to low throughput networks is outlined in Figure 9.

| | Bytes | |
|---|---|---|
| Frame Control Field | 2 | 000---------- : Beacon frame Uses dedicated channel<br>001---------- : Data Frame<br>010---------- : Ack Frame Becomes a flag because of possibility to encapsulate payload in ack<br>011---------- : Command frame Carried as option or over port 0<br>---1---------- : security enabled at MAC layer Removed : always enabled<br>----1---------- : frame pending<br>-----1---------- : ack request<br>------1---------- :PAN ID compression Removed, PAN ID always elided compressed<br>-------XXX------ : reserved<br>----------XX---- :Destination address mode Removed, star topology requires single address<br>----------XX-- :Frame version (00 : 2003, 01 : 2006)<br>----------XX :Source address mode Removed, star topology requires single address |
| Sequence number | 1 | Merged with frame counter as security is always enabled |
| Destination PAN ID | 0 or 2 | Removed |
| Destination address | 0 or 2 or 8 | Only from base station, 4 byte address |
| Source PAN ID | 0 or 2 | Removed |
| Source address | 0 or 2 or 8 | Only from node to base station, 4 byte address |
| Auxiliary security | variable | Contains security control, Frame counter, Key identifier fields |
| Payload | variable | |
| FCS | 2 | CRC 16 replaced by security MIC as security is always enabled |

**Figure 9: Adaptations of IEEE 802.15.4 to Low Throughput Networks (LTNs)**

Among the various application layers, 6LoWPAN is particularly important because it supports all low bitrate IP applications. The 6LoWPAN compression strategy relies heavily on the underlying MAC layer. Actility has adapted the 6LoWPAN compression libraries in parallel of the development of the ThingPark Wireless MAC layer, so that IP based applications are able to run transparently on ThingPark Wireless networks. Obviously, developers need to take into account the low throughput and duty cycle constraints of the underlying radio network.

## General MAC frame format

The general MAC frame format is outlined in Figure 10. The Beacon frame is carried over a dedicated channel, and uses a specific format.

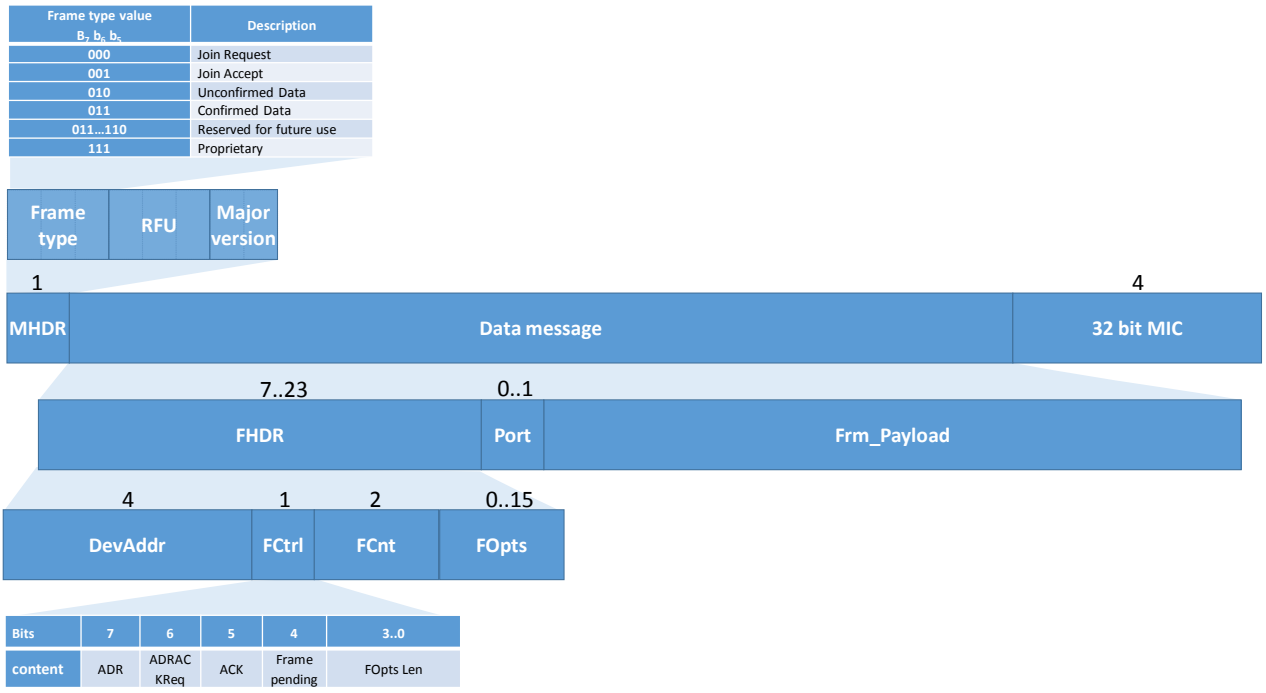



Figure 10: ThingPark wireless MAC frame format

The single byte MAC header (MHDR) encodes the frame type and the MAC major version (currently 00 for major version 1). The frame types currently defined are:

- Unconfirmed Data: a data frame that is not expecting an ACK
- Confirmed Data: a data frame that is expecting an ACK and will retransmit if no ACK is received during the reception timeslots following the frame transmission.
- Join Request: Optional network join message for a device that has not been pre-commissioned.
- Joint Accept : Network reply for a Join Request

A given major version determines the exact location of the address field in the frame. For version 1, the next four bytes must contain an address field. The minor MAC version is then indexed in the ThingPark Wireless network database by the node address field. The minor MAC versions 1.x may change the frame format of any octet after the first four, or even use node specific frame formats (as long as they are supported by the network through a special agreement).

Message integrity is ensured in different ways for different message types and described per message type below.

### Optional Join procedure frame format (MAC layer v1)

The format of the MAC payload for frames with MHDR.FrameType=000 or 001 is outlined in Annex A: optional Over-The-Air device activation using Join request/response.

### Data message format (MAC layer v1)

Frame types “Unconfirmed Data” and “Confirmed Data” are formatted as indicated in Figure 10.

The Data message frame begins with a Frame header FHDR, followed by an optional application port, and the data frame payload.

#### *Application port*

If present an application port value of 0 indicates that the data frame payload (Frm_payload) carries one or more MAC commands, see section MAC commands.

Application port values of 1 to 255 may be used by the application. Application keys (AppKey), when used, may be associated to specific application ports in a way that is opaque to the network infrastructure. Network routing of messages may take into account application ports in order to reach specific cloud servers for specific applications from the same device.

#### *Data frame header FHDR*

##### Address field

This 4 octet field contains the node address for node transmissions (uplink), and contains the node address or a group address for downlink transmissions.

Depending on the deployment context, neighboring network providers may agree to use the first 7 bits of the address field to encode a specific network. The sensor address may be allocated during the join procedure, or flashed at sensor production together with the network session key NwkSkey and the application session key AppSKey.

The network may register a multicast address to the node. A MAC command will communicate the multicast address, a 128 bit AES key, and the current multicast sequence number.

##### Frame control field (FCtrl)

The frame control field contains the following items:

- **ADR flag** signals that the node currently supports Adaptative Rate MAC commands. Nodes must support ADR whenever possible, however, when a node detects that it is moving and that ADR would cause potential communication issues, a node may elect to clear the ADR flag. ADR must be set again as soon as the node detects that is becomes fixed again.
- **ADRAckReq** flag. When end device is currently using a data rate higher than its default applicative data rate, it periodically needs to validate the network still receives the uplink frames. After each uplink, the device increments an ADR_ACK_CNT counter. After ADR_ACK_LIMIT uplinks (ADR_ACK_CNT == ADR_ACK_LIMIT) without any downlink response validating the uplink transmission, it sets the ADR acknowledgment request bit (ADRACKReq) in the next uplink data frame. The network is required to respond with a downlink frame within the next ADR_ACK_DELAY, whereby any received downlink following an uplink resets the ADR_ACK_CNT counter. The downlink ACK bit does not need to be set, since any response during the receive slot of the end device indicates that the gateway still receives the uplinks from this device. If no reply is received within the next ADR_ACK_DELAY, the end-device may try to regain connectivity by switching back to its default spreading factor providing longer reach. The ADRACKReq flag shall not be set if the device uses its default spreading factor because in that case no action can be taken to improve the link range.

  Note: Not requesting an immediate response to an ADR acknowledgement request provides flexibility to the network to optimally schedule its downlinks.
- **ACK flag** means that this frame acknowledges the last transmitted Confirmed data frame of the node. A node *must not* transmit another Confirmed data frame until it has received an

ACK for it or until the retransmission procedure has timed out. It may of course retransmit the same frame after a timeout of 5 seconds if it has not received an ACK within 5 seconds. An ACK will be correlated by the node to the corresponding Confirmed data frame, which will be (1) the last transmitted Confirmed data frame and (2) the difference between the ACK Rx time stamp and the acknowledged Confirmed data frame Tx time stamp must be an exact multiple of one second (considering clock drift tolerances).

- **Frame pending flag** is used only by the network towards a node and means that the network has more frames waiting for the node. After the end of the current frame, the node should wake up after a sleep time of RX_DELAY_1 –DRIFT_1 and start listening for the new frame. Figure 11 illustrates the use of the frame pending (FPending) bit on a downlink.
  If a frame with the FPending bit set requires an acknowledgement from the node, the node shall transmit a frame with ACK flag set as described before. If no acknowledgment is required, the end device may send an empty data message to open additional receive windows at its own discretion, or wait until it has some data to transmit itself and open receive windows as usual.

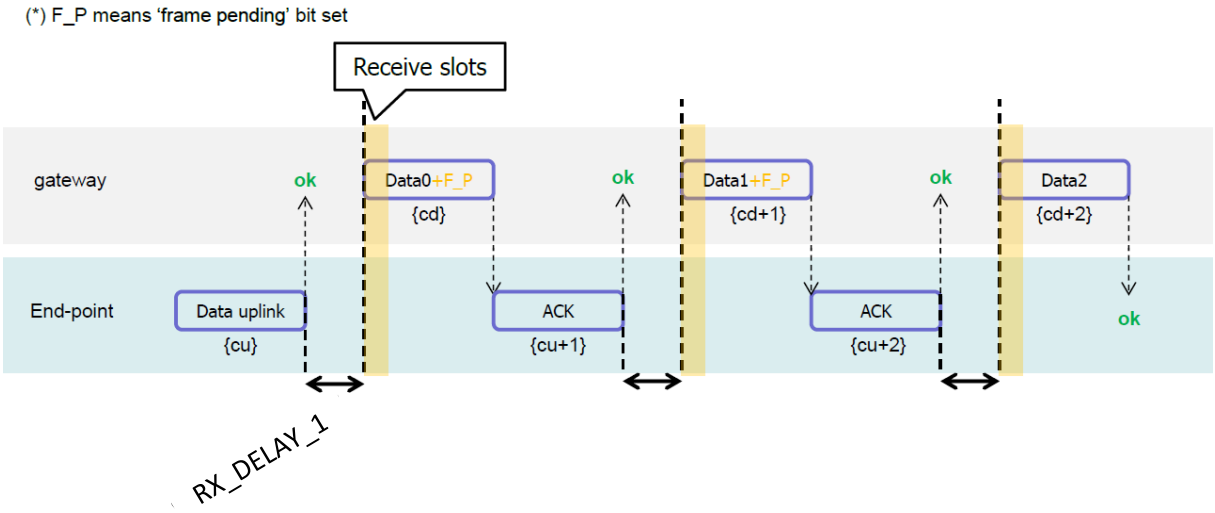



Figure 11: Sequence diagram illustrating use of Frame pending flag

In the example of Figure 11 the network has three data frames to transmit to the end device. The frame exchange is initiated by the end-device via a normal "unconfirmed" uplink message on channel A. The network uses the first receive window to transmit frame Data0 with bit FPending set as a confirmed data message. The device acknowledges the reception of the frame by transmitting back an empty frame with the ACK bit set on a new channel B. RX_DELAY_1 seconds later, the network transmits the second frame Data1 on channel B, again using a confirmed data message. The end device acknowledges on channel C. The last frame Data2 is transmitted as an unconfirmed data message with the FPending bit cleared. The end-device accordingly does not acknowledge reception of this last frame.

- **Options length flag (FOptsLen)** signals the octet size of the Options field in the header. A value of 0 indicated that there are no frame options. The frame options field maximum size is 15 octets.
- **Frame options (FOpt)** contains one or more MAC commands when present for a total of 15 octets maximum.

### Frame counter field

Each device maintains exactly two frame counters to keep track of the number of data frames sent uplink to the network (**FCntUp**) and received by the device downlink from the network (**FCntDown**). An additional downlink counter is needed for each multicast group.
The network in turn maintains the same frame counters for each device. At initialization, the frame counters on the device and the frame counters on the network for that device are reset to 0. Subsequently FCntUp and FCntDown are incremented at the sender side by 1 for each data frame sent in the respective direction. At the receiver side, the corresponding counter is kept in sync with the value received provided the value received is larger than the current counter value but in close proximity (considering rollovers), that is, at most MAX_FCNT_GAP data frames got lost. Data frames with a frame counter out of this range are silently discarded.

The MAC layer allows the use of either 16-bits or 32-bits frame counters. The network needs to be informed out-of-band about the width of the frame counter implemented internally by a given end device. If a 16-bits frame counter is used, the Fcnt field can be used directly as the counter value, possibly extended by leading zero octets if required. If a 32-bits frame counter is used, the FCnt field corresponds to the least-significant 16 bits of the 32-bits frame counter (i.e., FCntUp for data frames sent uplink and FCntDown for data frames sent downlink). Since the FCnt field carries only the least-significant 16 bits of the 32-bits frame counter, the server must infer the 16 most-significant bits of the frame counter from the observation of the traffic.

When the network infrastructure acknowledges a frame transmitted by an endpoint, the sequence counter value is the **FCntDn** (or multicast group counter) value, not the **FCntUp** value used by the end-point uplink that is being acknowledged. This is because the ACK flag may be part of a Confirmed data frame that will later need to be acknowledged by the endpoint.
Similarly, when an endpoint acknowledges a frame transmitted by an endpoint, the sequence counter value is **FCntUp** .

**Send only conditions**
Class 1, class 2 and class 3 devices may operate in send-only conditions, due to the slightly higher uplink budget versus downlink (the 6dB antenna at base station which increases Rx sensitivity but not Tx power due to ETSI power limitations). In this situation, nodes must transmit using SF12 and Frame counter is processed by the network as follows:
When the received 16 bit *Frame Counter* value is smaller than the last successfully received value, one of the 2 following conditions may be happening.

1. The device was reset. In that case the 16 MSBs of the counter were reset to 0.
2. The 32bits frame counter just went over a multiple of $2^{16}$. In that case the 16 MSBs of the frame counter should be incremented by 1.

Both computations must be performed on the server side to solve the uncertainty.
This uncertainty does not exist on the end point side, as the network infrastructure will never reset unexpectedly the downstream frame counter. The server will reset to 0 the 16 MSBs of the downstream frame counter for a class 1 device if a reset of the device is detected.

### Port field

The *Port* field specifies the type and format of the payload field. The payload and port field can be both absent. In this case frame might only carry MAC commands in the options field. It is also legal for payload to be absent and port to be present. A port value of zero is reserved for the MAC. All other ports (1..255) are application specific.

## Payload field

The *Frame Payload* field contains information specific to individual frame types. It may be cryptographically protected if the ThingPark Wireless Network database has encryption services enabled for this node. The encryption key may be managed by the network when the network provides additional value added services (ETSI M2M services), or may be opaque to the network (end to end packet tunneling to an external application server).

## Message Integrity Code (MIC)

A 32bit Message Integrity Code (MIC) contains a cryptographic hash which protects the frame against any modification and also detects transmission errors.

The MIC protects the frame fields as indicated in figure

| Octets | 1 | 7..23 | Variable | | 4 |
|---|---|---|---|---|---|
| | | | 0/1 | Variable | |
| Contents | MHDR | FHDR | Port | FRMPayload | MIC |

**Figure 12: Cryptographic protection of data fields : integrity and authentication**

The MIC of Data frames is calculated over the fields

msg = MHDR | FHDR | FPort | Encrypted_FRMPayload

where len(msg) denotes the length of the msg string in octets. The MIC is now calculated as per RFC 4493:

cmac = aes128_cmac(NwkSKey, $B_0$ | msg)

MIC = cmac[0..3]

where block $B_0$ is defined as follows:

| Octets | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $B_0$ (uplink) | 0x49 | 4* 0x00 | Dir = 0 (uplink) | DevAddr | FCntUp (uplink) | 0x00 | Len(msg) |
| $B_0$ (downlink) | 0x49 | 4* 0x00 | Dir = 1 (downlink) | DevAddr | FCntDn(downlink) | 0x00 | Len(msg) |

## Payload encryption

If a data frame carries a payload, that payload has to be encrypted before message integrity code is computed. The encryption scheme used is based on the generic algorithm described in IEEE 802.15.4/2006 Annex B using AES with a key length of 128 bits. The key used depends on the FPort of the data message as outlined in Figure 13.

| Fport | Encryption key |
|---|---|
| 0 | NwkSkey |
| 1..255 | AppSkey |

**Figure 13: selection of encryption key**

The fields encrypted are highlighted in Figure 14.

| Octets | 1 | 7..23 | 0..15 | Variable | | 4 |
|---|---|---|---|---|---|---|
| | | | | 0/1 | Variable | |
| Contents | MHDR | FHDR | Frame | Port | Payload | MIC |

| | | | options | | | |
|---|---|---|---|---|---|---|

Figure 14: Cryptographic protection of data fields: encryption

For each data message, the algorithm defines a sequence of Blocks Ai for i = 1..k with k = ceil(len(pld) / 16):

| Octets | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| $A_i$ (uplink) | 0x01 | 4* 0x00 | Dir = 0 (uplink) | DevAddr | FCntUp (uplink) | 0x00 | i |
| $A_i$ (downlink) | 0x01 | 4* 0x00 | Dir = 1 (downlink) | DevAddr | FCntDn (downlink) | 0x00 | i |

Each block Ai is then encrypted to get a sequence S of blocks Si:

$S_i$ = aes128_encrypt(K, $A_i$) for i = 1..k with k = ceil(len(FRMPayload) / 16)

$S = S_1 \mid S_2 \mid .. \mid S_k$

Encryption and decryption of the payload finally is done by truncating as follows:

(FRMPayload | $pad_{16}$) xor S

Then truncate to the first len(FRMPayload) octets.

## Beacon frame format

Besides relaying messages between end devices and network servers, the network base stations participate in providing a time-synchronization mechanism by sending beacons at regular fixed intervals (BEACON_INTERVAL). All beacons are transmitted using LoRa SF9/CR1 in radio packet implicit mode, that is, without a LoRa physical header and with no header CRC being appended by the radio.

The beacon frame shall be formatted as illustrated in Figure 15.

| Octets | 4 | 3 | 2 | 4 | 2 | 7 | 2 |
|---|---|---|---|---|---|---|---|
| Contents | Reserved | Network ID | Authorized Channels Mask | Current Time | CRC | Gateway ID | CRC |
| | Common to all base stations | | | | | Base station specific | |

Figure 15 – Beacon frame format

The beacon frame is designed so that the first 15 bytes fill exactly one block of the interleaver + error correction code. These first 15 bytes are identical for all gateways of the network and transmitted synchronously, there are no visible on-air collision for the end points listening to the beacons even if they simultaneously receive beacons from several base stations.
The optional second part of the beacon (bytes 16 and after) may be base station specific, and as such collision may occur. However a node within proximity of more than one base station will still be able to decode the strongest beacon with high probability. This way, the node can gather some basic understanding of its geographical location.
This optional field is used in the current version of the MAC layer to broadcast the gateway geographic coordinates and antenna number. The *Gateway ID* field composition is:

| Bits | 4 | 4 | 24 | 24 |
|---|---|---|---|---|
| Contents | Information Descriptor | Information | Latitude | Longitude |

Table 1 – GatewayID field description

The *Information Descriptor* field describes how the following Information field should be interpreted. The possible values for the *Information Descriptor* field are:

| Information Descriptor | Information |
|---|---|
| 0 | Index of the antenna broadcasting the beacon.<br>For a single antenna gateway, this field is 0. |
| 1-15 | Reserved |

Table 2 – GatewayID information field description

The gateway position is encoded on 48 bits as follows:

- the gateway North-South latitude is encoded using a signed 24 bit word where $-2^{23}$ corresponds to 90° South (the South Pole) and $2^{23}-1$ corresponds to 90° North (the North Pole). The equator corresponds therefore to 0,
- the gateway East-West longitude is encoded using a signed 24 bit word where $-2^{23}$ corresponds to 180° West and $2^{23}-1$ corresponds to 180° East. The Greenwich meridian corresponds therefore to 0.

Example 1: A gateway with 10°N latitude and 15°W longitude with a single antenna (so broadcasting the beacon on antenna 0) would be described by the following fields:

| Bits | 4 | 4 | 24 | 24 |
|---|---|---|---|---|
| Contents | Information Descriptor | Information | Latitude | Longitude |
| Value | 0 | 0 | 932070 | -699050 |

Table 3 – GatewayID example 1

Example 2: A gateway with 10°S latitude and 15°E longitude with two antennas (and broadcasting the beacon on antenna 1) would be described by the following fields:

| Bits | 4 | 4 | 24 | 24 |
|---|---|---|---|---|
| Contents | Information Descriptor | Information | Latitude | Longitude |
| Value | 0 | 1 | -932070 | 699050 |

Table 4 – GatewayID example 2

## MAC commands

For network administration, a set of MAC commands may be exchanged between the network and the MAC layer of any node.
A single data frame can contain any sequence of MAC commands, either piggybacked in the FOpts field or, when sent as a separate data frame, in the FRMPayload field with the FPort field being set to 0. Piggybacked MAC command sequences are always sent in the clear (not encrypted) and may not exceed 15 octets. MAC commands sent as FRMPayload are always encrypted and may not exceed the maximum FRMPayload length.

A MAC command consists of a command identifier (CID) of 1 octet followed by a possibly empty command-specific sequence of octets. For commands that have a request/answer format, the same command identifier is used for the command and the response. The length of a MAC command is not explicitly given and must be implicitly known by the MAC implementation. That is, unknown MAC commands cannot be skipped and the first unknown MAC command terminates the processing of a MAC command sequence.

This paragraph describes the syntax for the following list of MAC commands (Table 5).

| Command frame identifier | Command name | Transmitted by | | Comments |
|---|---|---|---|---|
| | | Node | Base Station | |
| 0x02 | LinkCheckReq | X | | Short frame used by the node to check it is connected to a network, expects an ACK from network. Used at communication setup. Should always use long source address. Trials should be transmitted first at SF8 then SF10 then SF12. |
| 0x02 | LinkCheckAns | | x | Contains the received signal power estimation. Indicates to the node the quality of reception (link margin). |
| 0x03 | LinkADRReq | | x | Requests the node to change data rate/power/channel. The node should comply with the request and needs to acknowledge reception. |

| | | | | |
|---|---|---|---|---|
| 0x03 | LinkADRAns | X | | Used to acknowledge reception of LinkADRReq |
| 0x05 | MacRxSlotsReq | | x | (future use)<br>Forces node to retransmit MacRxSlotsDeclareReq |
| 0x05 | MacRxSlotsDeclareReq | X | | (Future use)<br>Used by node to declare the periodicity / channel / data rate of its reception windows for class B devices. The network must acknowledge this frame. A reception slot of 1s means always ON. |
| 0x06 | DevStatusReq | | x | Request ndevice status |
| 0x06 | DevStatusAns | X | | Returns end device status: Battery level, RF duty cycle, RF power, number of packets transmitted. |

Table 5 – MAC command frames

### Link check

The LinkCheckReq sent by a node has no payload. If a LinkCheckReq is received by the network via one or multiple base stations, it responds with a LinkCheckAns command. The payload of this command is formatted as follows:

| Octets | 1 | 1 |
|---|---|---|
| LinkCheckAns payload | Margin | GwCnt |

The demodulation margin (Margin) is an 8-bit unsigned integer in the range of 0..254 indicating the link margin in dB of the last successfully received LinkCheckReq command, from the best base station. A value of 0 means that the packet was demodulated with no margin while a value of 20, for example, means that packet reached the nearest gateway 20dB above the demodulation floor.

The gateway count (GwCnt) is the number of base stations that successfully received the last LinkCheckReq command.

The network counts Link check requests as acknowledged data traffic for the device data plan. Therefore nodes are encouraged to check their connectivity primarily by using normal application traffic and relying on network ADR management. Nodes should use this MAC command only when needing to know the link margin specifically.

### Link Adaptative Data Rate (ADR)

The LinkADRReq command is sent by the network towards a node to change the node data rate, RF transmission power or channel used.

| Octet | 4 | 4 | 16 |
|---|---|---|---|
| LinkADRReq payload | DataRate | TxPower | ChMask |

The requested date rate (DataRate) and TX output power (TXPower) are encoded as follows:

| Data rate (4 MSBs) | Requested mode |
|---|---|
| 0 | LoRa™: SF12/125kHz |
| 1 | LoRa™: SF11/125kHz |
| 2 | LoRa™: SF10/125kHz |
| 3 | LoRa™: SF9/125kHz |
| 4 | LoRa™: SF8/125kHz |
| 5 | LoRa™: SF7/125kHz |
| 6 | LoRa™: SF7/250kHz |
| 7 | FSK: 100kbps |
| 8 ..15 | Reserved |

Table 6 – LinkADRReq data rate correspondence

| TX power | |
|---|---|
| 0 | 20dBm (if supported) |
| 1 | 14dBm |
| 2 | 11dBm |
| 3 | 8dBm |
| 4 | 5dBm |
| 5 | 2dBm |
| 6..15 | Reserved |

Table 7 – LinkADRReq Tx power correspondence

The channel mask (ChMask) encodes the available center frequencies as follows with bit 0 corresponding to the LSB:

| Bit index | Center frequency (in MHz) |
|---|---|
| 0 | 868.100 |
| 1 | 868.300 |
| 2 | 868.500 |
| 3 | 868.850 |
| 4 | 869.050 |
| 5 | 869.525 |
| 6 | 868.950 |
| 7..15 | Reserved for future use |

Table 8 – LinkADRReq allowed channel bit mask

The node should acknowledge this request by transmitting a LinkADRAns command.

| Octet | 1 |
|---|---|
| LinkADRAns payload | Margin |

The margin (Margin) is the demodulation signal-to-noise ratio in dB for the last successfully received LinkADRReq command. This is a signed integer of 8 bits with a minimum value of -127 and a maximum value of 127. The value -128 is reserved and means the demodulation margin could not be computed. This field can be fetched directly from the RegPktSnrValue register of the SX1272 chip.

### End device status

With the DevStatusReq command a network server may request status information from an end device. The command has no payload. If a DevStatusReq is received by an end device, it responds with a DevStatusAns command.

| Octets | 1 | 1 |
|---|---|---|
| DevStatusAns payload | Battery | Margin |
| | Indicates the power source status:<br>• 0 means the node is connected to an external power source,<br>• 254 means battery is full,<br>• 1 battery level is minimum,<br>255 is reserved and means the node is not able to measure the battery voltage. | Demodulation Signal To Noise ratio in dB for the last successfully received packet.<br>Margin = round($SNR_{dB}$ x 4)<br>This is an 8 bit signed integer.<br>Minimum value is -127, maximum is 127: -128 is reserved and means the demodulation margin could not be computed.<br>This field can be fetched directly from the RegPktSnrValue register of the SX1272 chip. |

# Annex A: optional Over-The-Air device activation using Join request/response

## Prerequisites

The device must be personalized with the following information:

- DevEUI: a globally unique device ID in EUI64 format
- AppEUI: a globally unique application ID in EUI64 format as described before
- AppKey: one or more device-specific AES-128 application keys, associated to application ports. These keys may be obtained by deriving the device-specific key from an application-specific root key exclusively known to and under the control of the application provider

## Join Request/Response

Whenever an end device joins a network via over-the-air activation, the AppKey(s) are used to derive the device-specific sessions key NwkSKey and AppSKey(s) to encrypt and verify network communication and application data.

A device must join the network through a formal MAC layer join procedure and obtain the NwkSkey and AppSKey(s) before it can participate in any other communication with application servers.

The cryptographic algorithms protecting the security of data frames rely on the assumption of nonces. A nonce is a unique number for each data transfer as long as a given key remains in use. Using the same nonce for two different messages compromises the security.

## Join Request

Before a device sends the Join Request it may try to track a beacon. A beacon might provide channel information and allow for a more sophisticated algorithm to guess an initial data rate at the cost of increased latency. Simple devices might join using the lowest data rate (SF12). The join message is designed to be very short to minimize air time. It does not contain any padding nor information beyond what is required to create session keys.

The MAC frame format of the join request is outlined in Figure 16:

| Octets | 1 | 8 | 8 | 2 | 4 |
|---|---|---|---|---|---|
| Contents | MHDR Frame type = 000 | AppEui | DevEui | DevNonce | MIC |

**Figure 16: Join Request MAC frame format**

*AppEui* field is the application provider unique id. The network maps this AppEui into an address (domain name, URI, IP address + port, etc.) and forwards the join request to the application router for authorization.

*DevEui* field is a unique device id embedded during the device personalization process. The application router maintains a list of devices and can decide if it wants to accept this particular device.

*DevNonce* field is a random value contributed by the device for the session key derivation. The device generates this random value e.g. by issuing radio RSSI measurements. For each end device, the network server keeps track of the DevNonce values used by the end device in the past and ignores join requests with a DevNonce values seen before from that end device. This mechanism prevents

replay attacks by sending previously recorded join-request messages with the intention of disconnecting the respective end device from the network.

*MIC* : The message integrity code of a *Join Request* is calculated as per RFC 4493:

cmac = aes128_cmac(AppKey, MHDR | AppEUI | DevEUI | DevNonce)

MIC = cmac[0..3]

The join-request message is not encrypted.

### Join Response

The device expects a *Join Response* message a fixed time after the end of transmission of the *Join Request* (JOIN_RESPONSE_DELAY). If the application router decides to accept the device it will create a *Join Response* message and forward it to the network server which in turn will forward it to the device.
The join response MAC frame is outlined in Figure 17.

| Octets | 1 | 6 | 4 | 2 | 4 |
|---|---|---|---|---|---|
| Contents | MHDR Frame type = 001 | AppNonce | DevAddr | Reserved | MIC |

**Figure 17: Join Response MAC frame format**

The join-accept message contains an application nonce (AppNonce) of 6 octets and a device address (DevAddr) as described above. The AppNonce is a random value or some form of unique ID provided by the network server and used by the end device to derive the two session keys NwkSKey and AppSKey as follows:

NwkSKey = aes128_encrypt(AppKey, AppNonce | DevNonce | $pad_{16}$)
AppSKey = aes128_encrypt(AppKey, DevNonce | AppNonce | $pad_{16}$)

The MIC value for a join-accept message is calculated as follows [RFC4493]:

cmac = aes128_cmac(AppKey, MHDR | AppNonce | DevAddr | RFU)
MIC = cmac[0..3]

The join-accept message itself is encrypted with the AppKey as follows[2]:
aes128_decrypt(AppKey, AppNonce | DevAddr | RFU | MIC)

When the device is configured to have several application keys, by convention the device and the application server must agree on the preferred AppKey to use for the above derivations.

The application server conveys the NwkSkey to the ThingPark wireless network via a cloud API.

[2] The network server uses an AES decrypt operation to encrypt the message so that the end device can use an AES encrypt operation to decrypt the message. This way an end device only has to implement AES encrypt but not AES decrypt.