

# LoRa Specification

Copyright © 2015 LoRa Alliance Inc. All rights reserved.

## NOTICE OF USE AND DISCLOSURE

Copyright © LoRa Alliance Inc. All rights reserved.

The information within this document is the property of the LoRa Alliance. The "LoRa Alliance" and its use and disclosure are subject to LoRa Alliance Corporate Bylaws, Intellectual Property Rights, IPR Policy and Membership Agreement.

Elements of LoRa Alliance Specifications may be subject to third party intellectual property rights. Inclusion without limitation patent, copyright or trademark rights. Such a third party may or may not be a member of LoRa Alliance. The Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

This document and the information contained herein are provided on an "AS IS" basis and THE ALLIANCE DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF ANY PARTY INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT, COPYRIGHT OR TRADEMARK RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT.

IN NO EVENT WILL THE ALLIANCE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND IN CONTRACT OR IN TORT IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The above notice and this paragraph must be included on all copies of this document that are made.

LoRa Alliance Inc.

24th Canal Road Suite 303

San Ramon, CA 94583

*Note: All company, brand and product names may be trademarks that are the sole property of their respective owners.*

1  
2  
3  
4  
5



# LoRaWAN™ Specification

**Authors:**

N. Sornin (Semtech), M. Luis (Semtech), T. Eirich (IBM), T. Kramp (IBM),  
O.Hersent (Actility)

**Version:** V1.1 Draft 186 . versions 16 and above integrate CRs from TC7 / Paris

**Date:** 2015 Dec

**Status:** Draft

## Contents

1	1	Introduction.....	9
2	1.1	LoRaWAN Classes.....	9
3	1.2	Conventions.....	10
4	2	Introduction on LoRaWAN options.....	11
5	2.1	LoRaWAN Classes.....	11
6	2.2	Specification scope.....	12
7		Class A – All end-devices.....	13
8	3	Physical Message Formats.....	14
9	3.1	Uplink Messages.....	14
10	3.2	Downlink Messages.....	14
11	3.3	Receive Windows.....	14
12	3.3.1	First receive window channel, data rate, and start.....	15
13	3.3.2	Second receive window channel, data rate, and start.....	15
14	3.3.3	Receive window duration.....	15
15	3.3.4	Receiver activity during the receive windows.....	15
16	3.3.5	Network sending a message to an end-device.....	15
17	3.3.6	Important notice on receive windows.....	15
18	3.3.7	Receiving or transmitting other protocols.....	15
19	4	MAC Message Formats.....	16
20	4.1	MAC Layer (PHYPayload).....	16
21	4.2	MAC Header (MHDR field).....	16
22	4.2.1	Message type (MType bit field).....	16
23	4.2.2	Major version of data message (Major bit field).....	17
24	4.3	MAC Payload of Data Messages (MACPayload).....	17
25	4.3.1	Frame header (FHDR).....	18
26	4.3.2	Port field (FPort).....	21
27	4.3.3	MAC Frame Payload Encryption (FRMPayload).....	22
28	4.4	Message Integrity Code (MIC).....	23
29	5	MAC Commands.....	24
30	5.1	Link Check commands ( <i>LinkCheckReq</i> , <i>LinkCheckAns</i> ).....	25
31	5.2	Link ADR commands ( <i>LinkADRReq</i> , <i>LinkADRAns</i> ).....	25
32	5.3	End-Device Transmit Duty Cycle ( <i>DutyCycleReq</i> , <i>DutyCycleAns</i> ).....	27
33	5.4	Receive Windows Parameters ( <i>RXParamSetupReq</i> , <i>RXParamSetupAns</i> ).....	28
34	5.5	End-Device Status ( <i>DevStatusReq</i> , <i>DevStatusAns</i> ).....	29
35	5.6	Creation / Modification of a Channel ( <i>NewChannelReq</i> , <i>NewChannelAns</i> ).....	29
36	5.7	Setting delay between TX and RX ( <i>RXTimingSetupReq</i> , <i>RXTimingSetupAns</i> ).....	30
37	5.8	Reset indication commands ( <i>ResetInd</i> , <i>ResetConf</i> ).....	31
38	5.9	ADR parameters ( <i>ADRParamSetupReq</i> , <i>ADRParamSetupAns</i> ).....	32
39	6	End-Device Activation.....	34
40	6.1	Data Stored in the End-device after Activation.....	34
41	6.1.1	End-device address (DevAddr).....	34
42	6.1.2	Application identifier (AppEUI).....	34
43	6.1.3	Uplink Network session key (NwkSKeyUp).....	34
44	6.1.4	Downlink Network session key (NwkSKeyDwn).....	34
45	6.1.5	Application session key (AppSKey).....	35
46	6.2	Over-the-Air Activation.....	35
47	6.2.1	End-device identifier (DevEUI).....	35
48	6.2.2	Application root keys (AppKey NwkKey).....	35
49	6.2.3	Join procedure.....	36
50			

1	6.2.4	Join-request message.....	36
2	6.2.5	Join-accept message.....	37
3	6.2.6	ReJoin-request message.....	39
4	6.3	Activation by Personalization.....	39
5	7	Protocol option negotiation.....	41
6	7.1	Two Downlink FCNTs scheme.....	41
7	7.2	32-bits frame counter.....	41
8	7.3	ADRParmSetupReq/Ans support.....	41
9	8	Physical Layer.....	42
10	8.1	EU 863-870MHz ISM Band.....	42
11	8.1.1	EU863-870 Preamble Format.....	42
12	8.1.2	EU863-870 ISM Band channel frequencies.....	42
13	8.1.3	EU863-870 Data Rate and End-point Output Power encoding.....	43
14	8.1.4	EU863-870 JoinAccept CFList.....	43
15	8.1.5	EU863-870 LinkAdrReq command.....	44
16	8.1.6	EU863-870 Maximum payload size.....	44
17	8.1.7	EU863-870 Receive windows.....	45
18	8.1.8	EU863-870 Default Settings.....	46
19	8.2	US 902-928MHz ISM Band.....	47
20	8.2.1	US902-928 Preamble Format.....	47
21	8.2.2	US902-928 Channel Frequencies.....	47
22	8.2.3	US902-928 Data Rate and End-point Output Power encoding.....	47
23	8.2.4	US902-928 JoinAccept CFList.....	48
24	8.2.5	US902-928 LinkAdrReq command.....	48
25	8.2.6	US902-928 Maximum payload size.....	49
26	8.2.7	US902-928 Receive windows.....	50
27	8.2.8	US902-928 Default Settings.....	50
28	8.3	China 779-787MHz ISM Band.....	51
29	8.3.1	CN779-787 Preamble Format.....	51
30	8.3.2	CN779-787 ISM Band channel frequencies.....	51
31	8.3.3	CN779-787 Data Rate and End-point Output Power encoding.....	52
32	8.3.4	CN779-787 JoinAccept CFList.....	52
33	8.3.5	CN779-787 LinkAdrReq command.....	53
34	8.3.6	CN779-787 Maximum payload size.....	53
35	8.3.7	CN779-787 Receive windows.....	54
36	8.3.8	CN779-787 Default Settings.....	54
37	8.4	EU 433MHz ISM Band.....	55
38	8.4.1	EU433 Preamble Format.....	55
39	8.4.2	EU433 ISM Band channel frequencies.....	55
40	8.4.3	EU433 Data Rate and End-point Output Power encoding.....	55
41	8.4.4	EU433 JoinAccept CFList.....	56
42	8.4.5	EU433 LinkAdrReq command.....	57
43	8.4.6	EU433 Maximum payload size.....	57
44	8.4.7	EU433 Receive windows.....	58
45	8.4.8	EU433 Default Settings.....	58
46	8.5	Australia 915-928MHz ISM Band.....	错误！未定义书签。
47	8.5.1	AU915-928 Preamble Format.....	错误！未定义书签。
48	8.5.2	AU915-928 Channel Frequencies.....	错误！未定义书签。
49	8.5.3	AU915-928 Data Rate and End-point Output Power encoding.....	错误！未定义书签。
50	8.5.4	AU915-928 JoinAccept CFList.....	错误！未定义书签。
51	8.5.5	AU915-928 LinkAdrReq command.....	错误！未定义书签。
52	8.5.6	AU915-928 Maximum payload size.....	错误！未定义书签。

1	8.5.7 AU915-928 Receive windows.....	错误！未定义书签。
2	8.5.8 AU915-928 Default Settings.....	错误！未定义书签。
3	8.6 CN 470-510MHz Band.....	64
4	8.6.1 CN470-510 Preamble Format.....	64
5	8.6.2 CN470-510 Channel Frequencies.....	64
6	8.6.3 CN470-510 Data Rate and End-point Output Power encoding.....	65
7	8.6.4 CN470-510 JoinResp CFlist.....	65
8	8.6.5 CN470-510 LinkAdrReq command.....	65
9	8.6.6 CN470-510 Maximum payload size.....	66
10	8.6.7 CN470-510 Receive windows.....	66
11	CN470-510 Default Settings.....	66
12	Class B – Beacon.....	69
13	9 Introduction to Class B.....	70
14	10 Principle of synchronous network initiated downlink (Class-B option).....	71
15	11 Uplink frame in Class B mode.....	73
16	12 Downlink Ping frame format (Class B option).....	74
17	12.1 Physical frame format.....	74
18	12.2 Unicast & Multicast MAC messages.....	74
19	12.2.1 Unicast MAC message format.....	74
20	12.2.2 Multicast MAC message format.....	74
21	13 Beacon acquisition and tracking.....	75
22	13.1 Minimal beacon-less operation time.....	75
23	13.2 Extension of beacon-less operation upon reception.....	75
24	13.3 Minimizing timing drift.....	75
25	14 Class B Downlink slot timing.....	76
26	14.1 Definitions.....	76
27	14.2 Slot randomization.....	77
28	15 Class B MAC commands.....	78
29	15.1 PingSlotInfoReq.....	78
30	15.2 BeaconFreqReq.....	79
31	15.3 PingSlotChannelReq.....	80
32	15.4 BeaconTimingReq.....	81
33	15.5 BeaconTimingAns.....	82
34	16 Beacons (Class B option).....	83
35	16.1 Beacon physical layer.....	83
36	16.1.1 EU 863-870MHz ISM Band.....	83
37	16.1.2 US 902-928MHz ISM Band.....	83
38	16.2 Beacon frame content.....	84
39	16.3 Beacon GwSpecific field format.....	85
40	16.3.1 Gateway GPS coordinate:InfoDesc = 0, 1 or 2.....	85
41	16.4 Beacons precise timing.....	86
42	16.5 Network downlink route update requirements.....	86
43	17 Class B unicast & multicast downlink channel frequencies.....	88
44	17.1 EU 863-870MHz ISM Band.....	88
45	17.2 US 902-928MHz ISM Band.....	88
46	Class C – Continuously listening.....	89
47	18 Class C: Continuously listening end-device.....	90
48	18.1 Second receive window duration for Class C.....	90
49	18.2 Class C Multicast downlinks.....	90
50	19 Class C MAC commands.....	91
51	19.1 Device Mode ( <i>DeviceModeInd</i> , <i>DeviceModeConf</i> ).....	91
52	Support information.....	92

20	Examples and Application Information.....	93
20.1	Uplink Timing Diagram for Confirmed Data Messages.....	93
20.2	The third ACK frame in this example also carries an application payload. A downlink frame can carry any combination of ACK, MAC control commands and payload. Downlink Diagram for Confirmed Data Messages.....	93
20.3	Downlink Timing for Frame-Pending Messages.....	94
20.4	Data-Rate Adaptation during Message Retransmissions.....	95
21	Recommendation on contract to be provided to the network server by the end-device provider at the time of provisioning.....	97
22	Recommendation on finding the locally used channels.....	98
23	Revisions.....	99
23.1	Revision 1.0.....	99
23.2	Revision 1.1.....	99
24	Glossary.....	100
25	Bibliography.....	101
25.1	References.....	101
26	NOTICE OF USE AND DISCLOSURE.....	102

## Tables

Table 1:	MAC message types.....	17
Table 2:	Major list.....	17
Table 3:	FPort list.....	22
Table 4:	MAC commands.....	25
Table 5:	Channel state table.....	26
Table 6:	LinkADRAAns status bits signification.....	27
Table 7:	RX2SetupAns status bits signification.....	28
Table 8:	Battery level decoding.....	29
Table 9:	NewChannelAns status bits signification.....	30
Table 10:	Del mapping table.....	31
Table 11:	EU863-870 synch words.....	42
Table 12:	EU863-870 default channels.....	42
Table 13:	EU863-870 JoinReq Channel List.....	43
Table 14:	Data rate and TX power table.....	43
Table 15:	Data rate and TX power table.....	43
Table 16:	ChMaskCntl value table.....	44
Table 17:	EU863-870 maximum payload size.....	45
Table 18 :	EU863-870 maximum payload size (not repeater compatible).....	45
Table 19:	Data rate and TX power table.....	48
Table 20:	ChMaskCntl value table.....	48
Table 21:	US902-928 maximum payload size (repeater compatible).....	49
Table 22 :	US902-928 maximum payload size (not repeater compatible).....	49
Table 23:	US902-928 data rate RX offset.....	50
Table 24:	CN779-787 synch words.....	51
Table 25:	CN780 JoinReq Channel List.....	51
Table 26:	Data rate and TX power table.....	52
Table 27:	ChMaskCntl value table.....	53
Table 28:	CN780 maximum payload size.....	53
Table 29 :	CN780 maximum payload size (not repeater compatible).....	54
Table 30:	EU433 synch words.....	55
Table 31:	EU433 JoinReq Channel List.....	55

1	Table 32: Data rate and TX power table.....	56
2	Table 33: ChMaskCntl value table.....	57
3	Table 34: EU433 maximum payload size.....	57
4	Table 35 : EU433 maximum payload size (not repeater compatible).....	58
5	Table 36: Data rate and TX power table.....	错误！未定义书签。
6	Table 37: ChMaskCntl value table.....	错误！未定义书签。
7	Table 38: AU915-928 maximum payload size.....	错误！未定义书签。
8	Table 39: AU915-928 maximum payload size (not repeater compatible).....	错误！未定义书签。
9	Table 40: Data rate mapping.....	错误！未定义书签。
10	Table 41: Data rate and TX power table.....	65
11	Table 42: ChMaskCntl value table.....	65
12	Table 43: CN470-510 maximum payload size.....	66
13	Table 44: Data rate offset.....	66
14	Table 45: Beacon timing.....	76

15

## 16 **Figures**

17	Figure 1: LoRaWAN Classes.....	11
18	Figure 2: Uplink PHY structure.....	14
19	Figure 3: Downlink PHY structure.....	14
20	Figure 4: End-device receive slot timing.....	14
21	Figure 5: Radio PHY structure (CRC* is only available on uplink messages).....	16
22	Figure 6: PHY payload structure.....	16
23	Figure 7: MAC payload structure.....	16
24	Figure 8: Frame header structure.....	16
25	Figure 9: LoRa message format elements.....	16
26	Figure 10: US902-928 channel frequencies.....	47
27	Figure 11: AU915-928 channel frequencies.....	错误！未定义书签。
28	Figure 12: CN470-510 channel frequencies.....	64
29	Figure 13: Beacon reception slot and ping slots.....	72
30	Figure 14 : beacon-less temporary operation.....	75
31	Figure 15: Beacon timing.....	76
32	Figure 16: Class C end-device receive slot timing.....	90
33	Figure 17: Uplink timing diagram for confirmed data messages.....	93
34	Figure 18: Downlink timing diagram for confirmed data messages.....	94
35	Figure 19: Downlink timing diagram for frame-pending messages, example 1.....	94
36	Figure 20: Downlink timing diagram for frame-pending messages, example 2.....	95
37	Figure 21: Downlink timing diagram for frame-pending messages, example 3.....	95

38



## 1 Introduction

This document describes the LoRaWAN™ network protocol which is optimized for battery-powered end-devices that may be either mobile or mounted at a fixed location.

LoRaWAN networks typically are laid out in a star-of-stars topology in which **gateways**<sup>1</sup> relay messages between **end-devices**<sup>2</sup> and a central **network server** at the backend. Gateways are connected to the network server via standard IP connections while end-devices use single-hop LoRa™ or FSK communication to one or many gateways.<sup>3</sup> All communication is generally bi-directional, although uplink communication from an end-device to the network server is expected to be the predominant traffic.

Communication between end-devices and gateways is spread out on different **frequency channels** and **data rates**. The selection of the data rate is a trade-off between communication range and message duration, communications with different data rates do not interfere with each other. LoRa data rates range from 0.3 kbps to 50 kbps. To maximize both battery life of the end-devices and overall network capacity, the LoRa network infrastructure can manage the data rate and RF output for each end-device individually by means of an **adaptive data rate** (ADR) scheme.

End-devices may transmit on any channel available at any time, using any available data rate, as long as the following rules are respected:

- The end-device changes channel in a pseudo-random fashion for every transmission. The resulting frequency diversity makes the system more robust to interferences.
- The end-device respects the maximum transmit duty cycle relative to the sub-band used and local regulations.
- The end-device respects the maximum transmit duration (or dwell time) relative to the sub-band used and local regulations.

**Note:** Maximum transmit duty-cycle and dwell time per sub-band are region specific and are defined in the Chapter 8.

### 1.1 LoRaWAN Classes

All LoRaWAN devices implement at least the Class A functionality as described in this document. In addition they may implement options named Class B, Class C as also described in this document or others to be defined. In all cases, they must remain compatible with Class A.

<sup>1</sup> Gateways are also known as **concentrators** or **base stations**.

<sup>2</sup> End-devices are also known as **nodes**.

<sup>3</sup> Support for intermediate elements – repeaters – is not described in the document, however payload restrictions for encapsulation overhead are included in this specification. A repeater is defined as using LoRaWAN as its backhaul mechanism.

## 1.2 Conventions

MAC commands are written ***LinkCheckReq***, bits and bit fields are written **FRMPayload**, constants are written RECEIVE\_DELAY1, variables are written *N*.

In this document,

- The octet order for all multi-octet fields is little endian and
- EUI are 64 bits integers and are transmitted as little endian.

## 2 Introduction on LoRaWAN options

LoRa™ is a wireless modulation for long-range low-power low-data-rate applications developed by Semtech. Devices implementing more than Class A are generally named “higher Class end-devices” in this document.

### 2.1 LoRaWAN Classes

A LoRa network distinguishes between a basic LoRaWAN (named Class A) and optional features (Class B, Class C ...):

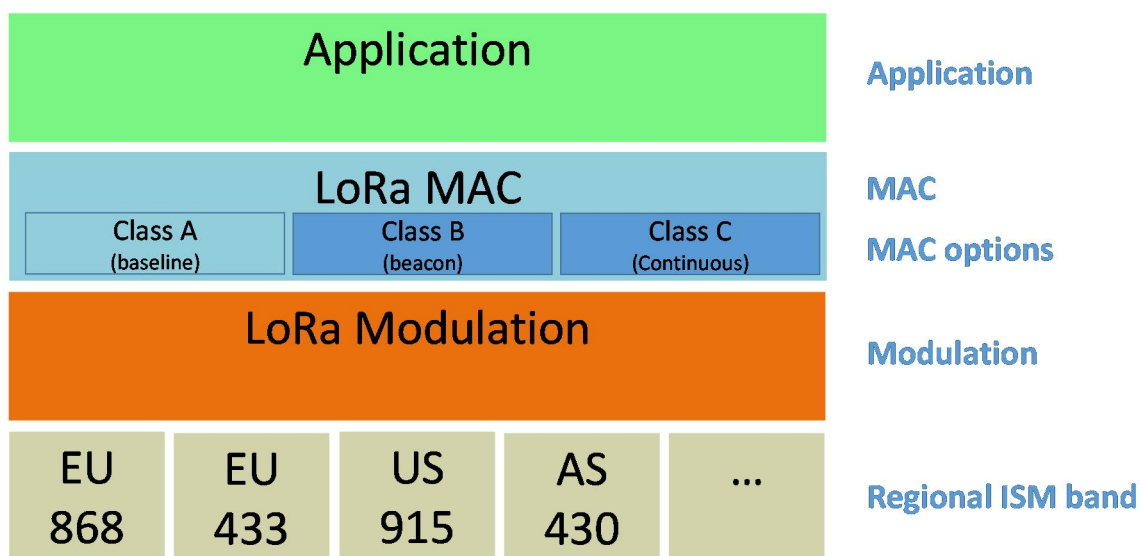


Figure 1: LoRaWAN Classes

- **Bi-directional end-devices (Class A):** End-devices of Class A allow for bi-directional communications whereby each end-device’s uplink transmission is followed by two short downlink receive windows. The transmission slot scheduled by the end-device is based on its own communication needs with a small variation based on a random time basis (ALOHA-type of protocol). This Class A operation is the lowest power end-device system for applications that only require downlink communication from the server shortly after the end-device has sent an uplink transmission. Downlink communications from the server at any other time will have to wait until the next scheduled uplink.
- **Bi-directional end-devices with scheduled receive slots (Class B):** End-devices of Class B allow for more receive slots. In addition to the Class A random receive windows, Class B devices open extra receive windows at scheduled times. In order for the End-device to open its receive window at the scheduled time it receives a time synchronized Beacon from the gateway. This allows the server to know when the end-device is listening.
- **Bi-directional end-devices with maximal receive slots (Class C):** End-devices of Class C have nearly continuously open receive windows, only closed when transmitting. Class C end-device will use more power to operate than Class A or Class B but they offer the lowest latency for server to end-device communication.

## 2.2 Specification scope

This LoRaWAN specification describes the additional functions differentiating an end-device higher Class from one of Class A. A higher Class end-device shall also implement all the functionality described in the LoRaWAN Class A specification.

**NOTE:** Physical message format, MAC message format, and other parts of this specification that are common to both end-devices of Class A and higher Classes are described only in the LoRaWAN Class A specification to avoid redundancy.

# **CLASS A – ALL END-DEVICES**

---

1

2

All LoRaWAN end-devices must implement Class A features.

### 3 Physical Message Formats

The LoRa terminology distinguishes between uplink and downlink messages.

#### 3.1 Uplink Messages

**Uplink messages** are sent by end-devices to the network server relayed by one or many gateways.

Uplink messages use the LoRa radio packet explicit mode in which the LoRa physical header (**PHDR**) plus a header CRC (**PHDR\_CRC**) are included.<sup>1</sup> The integrity of the payload is protected by a CRC.

The **PHDR**, **PHDR\_CRC** and payload **CRC** fields are inserted by the radio transceiver.

*Uplink PHY:*



Figure 2: Uplink PHY structure

#### 3.2 Downlink Messages

Each **downlink message** is sent by the network server to only one end-device and is relayed by a single gateway.<sup>2</sup>

Downlink messages use the radio packet explicit mode in which the LoRa physical header (**PHDR**) and a header CRC (**PHDR\_CRC**) are included.<sup>3</sup>

*Downlink PHY:*

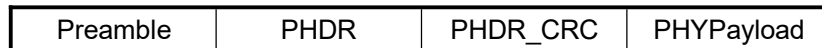


Figure 3: Downlink PHY structure

#### 3.3 Receive Windows

Following each uplink transmission the end-device opens two short receive windows. The receive window start times are defined using the end of the transmission as a reference.

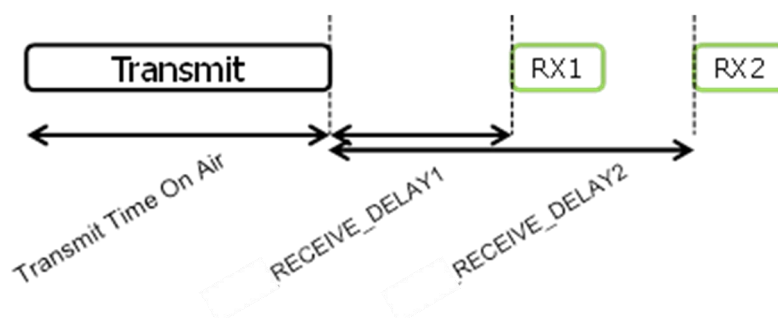


Figure 4: End-device receive slot timing.

<sup>1</sup> See the LoRa radio transceiver datasheet for a description of LoRa radio packet implicit/explicit modes.

<sup>2</sup> This specification does not describe the transmission of multicast messages from a network server to many end-devices.

<sup>3</sup> No payload integrity check is done at this level to keep messages as short as possible with minimum impact on any duty-cycle limitations of the ISM bands used.

### 3.3.1 First receive window channel, data rate, and start

The first receive window RX1 uses the same frequency channel as the uplink and a data rate that is a function of the data rate used for the uplink. RX1 opens RECEIVE\_DELAY1<sup>1</sup> seconds (+/- 20 microseconds) after the end of the uplink modulation. The relationship between uplink and RX1 slot downlink data rate is region specific and detailed in the Section 8. By default the first receive window datarate is identical to the datarate of the last uplink.

### 3.3.2 Second receive window channel, data rate, and start

The second receive window RX2 uses a fixed configurable frequency and data rate and opens RECEIVE\_DELAY2<sup>1</sup> seconds (+/- 20 microseconds) after the end of the uplink modulation. The frequency and data rate used can be modified through MAC commands (see Section 5). The default frequency and data rate to use are region specific and detailed in the Section 8 .

### 3.3.3 Receive window duration

The length of a receive window must be at least the time required by the end-device's radio transceiver to effectively detect a downlink preamble.

### 3.3.4 Receiver activity during the receive windows

If a preamble is detected during one of the receive windows, the radio receiver stays active until the downlink frame is demodulated. If a frame was detected and subsequently demodulated during the first receive window and the frame was intended for this end-device after address and MIC (message integrity code) checks, the end-device does not open the second receive window.

### 3.3.5 Network sending a message to an end-device

If the network intends to transmit a downlink to an end-device, it will always initiate the transmission precisely at the beginning of one of those two receive windows.

### 3.3.6 Important notice on receive windows

An end-device shall not transmit another uplink message before it either has received a downlink message in the first or second receive window of the previous transmission, or the second receive window of the previous transmission is expired.

### 3.3.7 Receiving or transmitting other protocols

The node may listen or transmit other protocols or do any transactions between the LoRaWAN transmission and reception windows, as long as the end-device remains compatible with the local regulation and compliant with the LoRaWAN specification.

---

<sup>1</sup> RECEIVE\_DELAY1 and RECEIVE\_DELAY2 are described in Chapter 6.

## 4 MAC Message Formats

All LoRa uplink and downlink messages carry a PHY payload (**Payload**) starting with a single-octet MAC header (**MHDR**), followed by a MAC payload (**MACPayload**)<sup>1</sup>, and ending with a 4-octet message integrity code (**MIC**).

Radio PHY layer:



Figure 5: Radio PHY structure (CRC\* is only available on uplink messages)

PHYPayload:

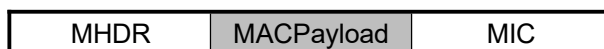


Figure 6: PHY payload structure

MACPayload:

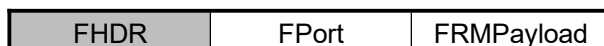


Figure 7: MAC payload structure

FHDR:

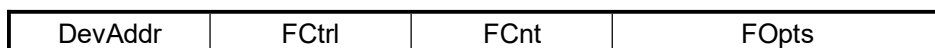


Figure 8: Frame header structure

Figure 9: LoRa message format elements

### 4.1 MAC Layer (PHYPayload)

Size (bytes)	1	1..M	4
PHYPayload	MHDR	MACPayload	MIC

The maximum length ( $M$ ) of the **MACPayload** field is region specific and is specified in Chapter 6.

### 4.2 MAC Header (MHDR field)

Bit#	7..5	4..2	1..0
MHDR bits	MType	RFU	Major

The MAC header specifies the message type (**MType**) and according to which major version (**Major**) of the frame format of the LoRaWAN layer specification the frame has been encoded.

#### 4.2.1 Message type (MType bit field)

The LoRaWAN distinguishes between six different MAC message types: **join request**, **join accept**, **unconfirmed data up/down**, and **confirmed data up/down**.

MType	Description

<sup>1</sup> Maximum payload size is detailed in the Chapter 6.



000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	Rejoin Request
111	Proprietary

**Table 1: MAC message types**

#### 4.2.1.1 Join-request and join-accept messages

The join-request, Rejoin-request and join-accept messages are used by the over-the-air activation procedure described in Chapter 6.2.

#### 4.2.1.2 Data messages

Data messages are used to transfer both MAC commands and application data, which can be combined together in a single message. A **confirmed-data message** has to be acknowledged by the receiver, whereas an **unconfirmed-data message** does not require an acknowledgment.<sup>1</sup> **Proprietary messages** can be used to implement non-standard message formats that are not interoperable with standard messages but must only be used among devices that have a common understanding of the proprietary extensions.

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

#### 4.2.2 Major version of data message (Major bit field)

Major bits	Description
00	LoRaWAN R1
01..11	RFU

**Table 2: Major list**

**Note:** The Major version specifies the format of the messages exchanged in the join procedure (see Chapter 6.2) and the first four bytes of the MAC Payload as described in Chapter 4. For each major version, end-devices may implement different minor versions of the frame format. The minor version used by an end-device must be made known to the network server beforehand using out of band messages (e.g., as part of the device personalization information).

### 4.3 MAC Payload of Data Messages (MACPayload)

The MAC payload of the data messages, a so-called “data frame”, contains a frame header (**FHDR**) followed by an optional port field (**FPort**) and an optional frame payload field (**FRMPayload**).

<sup>1</sup> A detailed timing diagram of the acknowledge mechanism is given in Section 21.

### 4.3.1 Frame header (FHDR)

The **FHDR** contains the short device address of the end-device (**DevAddr**), a frame control octet (**FCtrl**), a 2-octets frame counter (**FCnt**), and up to 15 octets of frame options (**FOpts**) used to transport MAC commands.

Size (bytes)	4	1	2	0..15
FHDR	DevAddr	FCtrl	FCnt	FOpts

For downlink frames the FCtrl content of the frame header is:

Bit#	7	6	5	4	[3..0]
FCtrl bits	ADR	RFU	ACK	FPending	FOptsLen

For uplink frames the FCtrl content of the frame header is:

Bit#	7	6	5	4	[3..0]
FCtrl bits	ADR	ADRACKReq	ACK	RFU	FOptsLen

#### 4.3.1.1 Adaptive data rate control in frame header (ADR, ADRACKReq in FCtrl)

LoRa network allows the end-devices to individually use any of the possible data rates. This feature is used by the LoRaWAN to adapt and optimize the data rate of static end-devices. This is referred to as Adaptive Data Rate (ADR) and when this is enabled the network will be optimized to use the fastest data rate possible.

A mobile device's data rate cannot be efficiently controlled by the network because the radio channel changes faster than the control loop can deal with. In that case the data rate used by the end-device should be set by the end-device's application.

If the uplink **ADR** bit is set, the network will control the data rate of the end-device through the appropriate MAC commands. If the **ADR** bit is not set, the network will not attempt to control the data rate nor the transmit power of the end-device regardless of the received signal quality. The network may still send commands to change the Channel mask or the frame repetition parameters.

When the downlink ADR bit is set, it informs the end-device that the network server is in a position to send ADR commands. The device may set/unset the uplink ADR bit.

When the downlink ADR bit is unset, it signals the end-device that due to rapid changes of the radio channel, the network temporarily cannot estimate the best data rate. In that case the device has the choice to either

- unset the ADR uplink bit, and control its uplink data rate following its own strategy. This should be the typical strategy for a mobile end-device.
- Ignore it (keep the uplink ADR bit set) and apply the normal data rate decay in the absence of ADR downlink commands. This should be the typical strategy for a stationary end-device.

The **ADR** bit may be set and unset by the end-device or the Network on demand. However, whenever possible, the ADR scheme should be enabled to increase the battery life of the end-device and maximize the network capacity.

**Note:** Even mobile end-devices are actually immobile most of the time. So depending on its state of mobility, an end-device can request the network to optimize its data rate using the ADR uplink bit.

If an end-device whose data rate is optimized by the network to use a data rate higher than its default data rate, or a TXPower lower than its default TXPower, it periodically needs to validate that the network still receives the uplink frames. Each time the uplink frame counter is incremented (for each new uplink, repeated transmissions do not increase the counter), the device increments an `ADR_ACK_CNT` counter. After `ADR_ACK_LIMIT` uplinks (`ADR_ACK_CNT >= ADR_ACK_LIMIT`) without any downlink response, it sets the ADR acknowledgment request bit (**ADRACKReq**). The network is required to respond with a downlink frame within the next `ADR_ACK_DELAY` frames, any received downlink frame following an uplink frame resets the `ADR_ACK_CNT` counter. The downlink **ACK** bit does not need to be set as any response during the receive slot of the end-device indicates that the gateway has still received the uplinks from this device. If no reply is received within the next `ADR_ACK_DELAY` uplinks (i.e., after a total of `ADR_ACK_LIMIT + ADR_ACK_DELAY`), the end-device may try to regain connectivity by first stepping up the transmit power if possible then switching to the next lower data rate that provides a longer radio range. The end-device will further lower its data rate step by step every time `ADR_ACK_LIMIT` is reached. The **ADRACKReq** shall not be set if the device uses its default data rate because in that case no action can be taken to improve the link range.

**Note:** Not requesting an immediate response to an ADR acknowledgment request provides flexibility to the network to optimally schedule its downlinks.

**Note:** In uplink transmissions the **ADRACKReq** bit is set if `ADR_ACK_CNT >= ADR_ACK_LIMIT` and the current data-rate is greater than the device defined minimum data rate or its transmit power is lower than the default, it is cleared in other conditions.

#### 4.3.1.2 Message acknowledge bit and acknowledgement procedure (ACK in FCtrl)

When receiving a *confirmed data* message, the receiver shall respond with a data frame that has the acknowledgment bit (**ACK**) set. If the sender is an end-device, the network will send the acknowledgement using one of the receive windows opened by the end-device after the send operation. If the sender is a gateway, the end-device transmits an acknowledgment at its own discretion.

Acknowledgements are only sent in response to the latest message received and are never retransmitted.

**Note:** To allow the end-devices to be as simple as possible and have as few states as possible it may transmit an explicit (possibly empty) acknowledgement data message immediately after the reception of a data message requiring a confirmation. Alternatively the end-device may defer the transmission of an acknowledgement to piggyback it with its next data message.

### 4.3.1.3 Retransmission procedure

The number of retransmissions (and their timing) for the same message where an acknowledgment is requested but not received is at the discretion of the end device and may be different for each end-device.

**Note:** Some example timing diagrams of the acknowledge mechanism are given in Chapter 21.

**Note:** If an end-device has reached its maximum number of retransmissions without receiving an acknowledgment, it can try to re-gain connectivity by moving to a lower data rate with longer reach. It is up to the end-device to retransmit the message again or to forfeit that message and move on.

**Note:** If the network server has reached its maximum number of retransmissions without receiving an acknowledgment, it will generally consider the end-device as unreachable until it receives further messages from the end-device. It is up to the network server to retransmit the message once connectivity to the end-device in question is regained or to forfeit that message and move on.

**Note:** The recommended data rate back-off strategy during re-transmissions is described in Chapter 21.4

### 4.3.1.4 Frame pending bit (FPending in FCtrl, downlink only)

The frame pending bit (**FPending**) is only used in downlink communication, indicating that the gateway has more data pending to be sent and therefore asking the end-device to open another receive window as soon as possible by sending another uplink message.

The exact use of **FPending** bit is described in Chapter 21.3.

### 4.3.1.5 Frame counter (FCnt)

Each end-device has at least two frame counters to keep track of the number of data frames sent uplink to the network server (FCntUp), incremented by the end-device and sent downlink from the network server (FCntDown),.

In the downlink direction two different frame counter scheme exists; a single counter scheme in which all ports share the same downlink frame counter (default on all LoRaWAN 1.0 and earlier), and a two-counter scheme in which a separate NFCntDown is used for MAC communication on port 0, and another AFCntDown is use for all other ports.

In the two counter scheme the NFCntDown is managed by the network server, whereas the AFCntDown is managed by the application server.

The use of the two-counter scheme is signaled through the reset indicator option bitmap. The single counter scheme **shall** be supported by end-device and network server. The two-counter scheme **should** be supported by all LoRaWAN 1.1 capable end-devices and network servers.

**Note:** LoRaWAN v1.0 and earlier support only one FCntDown counter (shared across all ports) and the network server must take care to support this scheme for devices prior to LoRaWAN v1.1.

Whenever a ResetInd command is sent by an end-device to indicate re-initialization, the frame counters on the end-device and the frame counters on the network server for that end-device are reset to 0. Subsequently FCntUp is incremented with each uplink. NFCntDown is incremented with each downlink on FPort 0. AFCntDown is incremented with each downlink on a port different than 0.. At the receiver side, the corresponding counter is kept in sync with the value received provided the value received has incremented compared to the current counter value and is less than the value specified by MAX\_FCNT\_GAP<sup>1</sup> after considering counter rollovers. If this difference is greater than the value of MAX\_FCNT\_GAP then too many data frames have been lost then subsequent will be discarded.

The LoRaWAN allows the use of either 16-bits or 32-bits frame counters. The network side needs to be informed out-of-band about the width of the frame counter implemented in 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).

The end-device shall not reuse the same FCntUp value, except for retransmission, with the same application and network session keys.

**Note:** 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.

#### 4.3.1.6 Frame options (FOptsLen in FCtrl, FOpts)

The frame-options length field (**FOptsLen**) in **FCtrl** byte denotes the actual length of the frame options field (**FOpts**) included in the frame.

**FOpts** transport MAC commands of a maximum length of 15 octets that are piggybacked onto data frames; see Chapter 4.4 for a list of valid MAC commands.

If **FOptsLen** is 0, the **FOpts** field is absent. If **FOptsLen** is different from 0, i.e. if MAC commands are present in the **FOpts** field, the port 0 cannot be used (**FPort** must be either not present or different from 0).

MAC commands cannot be simultaneously present in the payload field and the frame options field. Should this occur, the device should ignore the message.

#### 4.3.2 Port field (FPort)

If the frame payload field is not empty, the port field must be present. If present, an **FPort** value of 0 indicates that the **FRMPayload** contains MAC commands only; see Chapter 4.4 for a list of valid MAC commands. **FPort** values 1..223 (0x01..0xDF) are application-specific. FPort value 224 is dedicated to LoRaWAN Mac layer test protocol.

**Note :** The purpose of Fport value 224 is to provide a dedicated Fport to run Mac compliance test scenarios over-the-air on final versions of

<sup>1</sup> Actual value for MAX\_FCNT\_GAP, RECEIVE\_DELAY1 and RECEIVE\_DELAY2 can be found at 8.1.7 for EU863-870 or 8.2.7 for US902-928.

devices, without having to rely on specific test versions of devices for practical aspects. The test is not supposed to be simultaneous with live operations, but the Mac layer implementation of the device shall be exactly the one used for the normal application. The test protocol is normally encrypted using the AppSKey. This ensures that the network cannot enable the device's test mode without involving the device's owner. If the test runs on a live network connected device, the way the test application on the network side learns the AppSKey is outside of the scope of the LoRaWAN specification. If the test runs using OTAA on a dedicated test bench (not a live network), the way the Appkey is communicated to the test bench, for secured JOIN process, is also outside of the scope of the specification.

The test protocol, running at application layer, is defined outside of the LoRaWAN spec, as it is an application layer protocol.

**FPort** values 224..255 (0xE0..0xFF) are reserved for future standardized application extensions.

Size (bytes)	7..22	0..1	0..N
MACPayload	FHDR	FPort	FRMPayload

$N$  is the number of octets of the application payload. The valid range for  $N$  is region specific and is defined in Section 8

$N$  should be equal or smaller than:

$$N \leq M - 1 - (\text{length of FHDR in octets})$$

where  $M$  is the maximum MAC payload length.

### 4.3.3 MAC Frame Payload Encryption (FRMPayload)

If a data frame carries a payload, **FRMPayload** must be encrypted before the message integrity code (**MIC**) is calculated.

The encryption scheme used is based on the generic algorithm described in IEEE 802.15.4/2006 Annex B [IEEE802154] using AES with a key length of 128 bits.

The key  $K$  used depends on the FPort of the data message:

FPort	Direction	K
0	Uplink	NwkSKeyUp
0	Downlink	NwkSKeyDwn
1..255	Up/downlink	AppSKey

**Table 3: FPort list**

The fields encrypted are:

$$pId = \text{FRMPayload}$$

For each data message, the algorithm defines a sequence of Blocks  $A_i$  for  $i = 1..k$  with  $k = \text{ceil}(\text{len}(pId) / 16)$ :

Size (bytes)	1	4	1	4	4	1	1
$A_i$	0x01	4 x 0x00	Dir	DevAddr	FCntUp or FCntDown	0x00	$i$

The direction field (**Dir**) is 0 for uplink frames and 1 for downlink frames.

The blocks  $A_i$  are encrypted to get a sequence  $S$  of blocks  $S_i$ :

$$S_i = \text{aes128\_encrypt}(K, A_i) \text{ for } i = 1..k$$

$$S = S_1 | S_2 | .. | S_k$$

Encryption and decryption of the payload is done by truncating

$$(p/d | \text{pad}_{16}) \text{ xor } S$$

to the first  $\text{len}(p/d)$  octets.

#### 4.4 Message Integrity Code (MIC)

The message integrity code (**MIC**) is calculated over all the fields in the message.

$$msg = \text{MHDR} | \text{FHDR} | \text{FPort} | \text{FRMPayload}$$

whereby  $\text{len}(msg)$  denotes the length of the message in octets.

The **MIC** is calculated as follows [RFC4493]:

$$cmac = \text{aes128\_cmac}(\text{NwkSKeyUp/Dwn}, B_0 | msg)$$

$$\text{MIC} = cmac[0..3]$$

whereby the block  $B_0$  is defined as follows:

Size (bytes)	1	4	1	4	4	1	1
$B_0$	0x49	4 x 0x00	Dir	DevAddr	FCntUp or FCntDown	0x00	$\text{len}(msg)$

The NwkSKeyUp is used to authenticate uplinks (from end-device to network), NwkSKeyDwn for downlinks.

The direction field (**Dir**) is 0 for uplink frames and 1 for downlink frames.



## 5 MAC Commands

For network administration, a set of MAC commands may be exchanged exclusively between the network server and the MAC layer on an end-device. MAC layer commands are never visible to the application or the application server or the application running on the end-device.

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 commands are always sent without encryption and must not exceed 15 octets. MAC commands sent as **FRMPayload** are always encrypted and must not exceed the maximum **FRMPayload** length.

**Note:** MAC commands whose content shall not be disclosed to an eavesdropper must be sent in the **FRMPayload** of a separate data message.

A MAC command consists of a command identifier (**CID**) of 1 octet followed by a possibly empty command-specific sequence of octets.

CID	Command	Transmitted by		Short Description
		End-de vice	Gateway	
0x02	<b>LinkCheckReq</b>	x		Used by an end-device to validate its connectivity to a network.
0x02	<b>LinkCheckAns</b>		x	Answer to LinkCheckReq command. Contains the received signal power estimation indicating to the end-device the quality of reception (link margin).
0x03	<b>LinkADRReq</b>		x	Requests the end-device to change data rate, transmit power, repetition rate or channel.
0x03	<b>LinkADRAns</b>	x		Acknowledges the LinkRateReq.
0x04	<b>DutyCycleReq</b>		x	Sets the maximum aggregated transmit duty-cycle of a device
0x04	<b>DutyCycleAns</b>	x		Acknowledges a DutyCycleReq command
0x05	<b>RXParamSetupReq</b>		x	Sets the reception slots parameters
0x05	<b>RXParamSetupAns</b>	x		Acknowledges a RXSetupReq command
0x06	<b>DevStatusReq</b>		x	Requests the status of the end-device
0x06	<b>DevStatusAns</b>	x		Returns the status of the end-device, namely its battery level and its demodulation margin
0x07	<b>NewChannelReq</b>		x	Creates or modifies the definition of a radio channel
0x07	<b>NewChannelAns</b>	x		Acknowledges a NewChannelReq command
0x08	<b>RXTimingSetupReq</b>		x	Sets the timing of the of the reception slots
0x08	<b>RXTimingSetupAns</b>	x		Acknowledges RXTimingSetupReq command
0x09	<b>ResetInd</b>	x		Used by an end-device to indicate a reset to the network and negotiate optional protocol features
0x09	<b>ResetConf</b>		x	Acknowledges ResetInd command
0x0A	<b>ADRParamSetupReq</b>		x	Used by the network server to set the ADR_ACK_LIMT and ADR_ACK_DELAY parameters of an end-device
0x0A	<b>ADRParamSetupAns</b>	x		Acknowledges ACKParamSetupReq command



CID	Command	Transmitted by		Short Description
		End-de vice	Gateway	
0x0B	<b>DeviceTimeReq</b>	x		Used by an end-device to request the current date and time
0x0B	<b>DeviceTimeAns</b>		x	Sent by the network, answer to the DeviceTimeReq request
0x80 to 0xFF	Proprietary	x	x	Reserved for proprietary network command extensions

**Table 4: MAC commands**

**Note:** The length of a MAC command is not explicitly given and must be implicitly known by the MAC implementation. Therefore unknown MAC commands cannot be skipped and the first unknown MAC command terminates the processing of the MAC command sequence. It is therefore advisable to order MAC commands according to the version of the LoRaWAN specification which has introduced a MAC command for the first time. This way all MAC commands up to the version of the LoRaWAN specification implemented can be processed even in the presence of MAC commands specified only in a version of the LoRaWAN specification newer than that implemented.

**Note:** Any values adjusted by the network server (e.g., RX2, new or adjusted channels definitions) remain only valid until the next join of the end-device. Therefore after each successful join procedure the end-device uses the default parameters again and it is up to the network server to re-adjust the values again as needed.

## 5.1 Link Check commands (*LinkCheckReq*, *LinkCheckAns*)

With the **LinkCheckReq** command, an end-device may validate its connectivity with the network. The command has no payload.

When a **LinkCheckReq** is received by the network server via one or multiple gateways, it responds with a **LinkCheckAns** command.

Size (bytes)	1	1
<b>LinkCheckAns Payload</b>	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. A value of “0” means that the frame was received at the demodulation floor (0 dB or no margin) while a value of “20”, for example, means that the frame reached the gateway 20 dB above the demodulation floor. Value “255” is reserved.

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

## 5.2 Link ADR commands (*LinkADRReq*, *LinkADRAns*)

With the **LinkADRReq** command, the network server requests an end-device to perform a rate adaptation.

Size (bytes)	1	2	1
<b>LinkADRReq Payload</b>	DataRate_TXPower	ChMask	Redundancy

Bits	[7:4]	[3:0]
<b>DataRate_TXPower</b>	DataRate	TXPower

The requested data rate (**DataRate**) and TX output power (**TXPower**) are region-specific and are encoded as indicated in Chapter 8. A value 0xF (15 in decimal format) of either DataRate or TXPower means that the device should ignore that field, and keep the current parameter value. The channel mask (**ChMask**) encodes the channels usable for uplink access as follows with bit 0 corresponding to the LSB:

Bit#	Usable channels
0	Channel 1
1	Channel 2
..	..
15	Channel 16

Table 5: Channel state table

A bit in the **ChMask** field set to 1 means that the corresponding channel can be used for uplink transmissions if this channel allows the data rate currently used by the end-device. A bit set to 0 means the corresponding channels should be avoided.

Bits	7	[6:4]	[3:0]
<b>Redundancy bits</b>	RFU	ChMaskCntl	NbTrans

In the Redundancy bits the **NbTrans** field is the number of transmissions for each uplink message. This applies only to “unconfirmed” uplink frames. The default value is 1 corresponding to a single transmission of each frame. The valid range is [1:15]. If **NbTrans**==0 is received the end-device should keep the current NbTrans value unchanged. This field can be used by the network manager to control the redundancy of the node uplinks to obtain a given Quality of Service. The end-device performs frequency hopping as usual between repeated transmissions, it does wait after each repetition until the receive windows have expired. . Whenever a downlink message is received during the RX1 slot window, it shall stop any further retransmission of the same uplink message. For class A devices, a reception in the RX2 slot has the same effect.

The channel mask control (**ChMaskCntl**) field controls the interpretation of the previously defined **ChMask** bit mask.. It controls the block of 16 channels to which the **ChMask** applies. It can also be used to globally turn on or off all channels using specific modulation. This field usage is region specific and is defined in Chapter 8.

The channel frequencies are region-specific and they are defined in Chapter 6. An end-device answers to a **LinkADRReq** with a **LinkADRAns** command.

Size (bytes)	1
<b>LinkADRAns Payload</b>	Status

Bits	[7:3]	2	1	0
Status bits	RFU	Power ACK	Data rate ACK	Channel mask ACK

The **LinkADRAAns Status** bits have the following meaning:

	Bit = 0	Bit = 1
<b>Channel mask ACK</b>	The channel mask sent enables a yet undefined channel or the channel mask required all channels to be disabled. The command was discarded and the end-device state was not changed.	The channel mask sent was successfully interpreted. All currently defined channel states were set according to the mask.
<b>Data rate ACK</b>	The data rate requested is unknown to the end-device or is not possible given the channel mask provided (not supported by any of the enabled channels). The command was discarded and the end-device state was not changed.	The data rate was successfully set or the DataRate field of the request was set to 15, meaning it was ignored
<b>Power ACK</b>	The requested power level is not implemented in the device. The command was discarded and the end-device state was not changed.	The power level was successfully set, or the TXPower field of the request was set 15, meaning it was ignored

**Table 6: LinkADRAAns status bits signification**

If any of those three bits equals 0, the command did not succeed and the node has kept the previous state.

### 5.3 End-Device Transmit Duty Cycle (*DutyCycleReq*, *DutyCycleAns*)

The **DutyCycleReq** command is used by the network coordinator to limit the maximum aggregated transmit duty cycle of an end-device. The aggregated transmit duty cycle corresponds to the transmit duty cycle over all sub-bands.

Size (bytes)	1
<b>DutyCycleReq Payload</b>	MaxDCycle

Insert 4 MSB as RFU , MaxDCycle can only be in the range 0 to 15

The maximum end-device transmit duty cycle allowed is:

The valid range for **MaxDutyCycle** is [0 : 15]. A value of 0 corresponds to “no duty cycle limitation” except the one set by the regional regulation.

An end-device answers to a **DutyCycleReq** with a **DutyCycleAns** command. The **DutyCycleAns** MAC reply does not contain any payload.

## 5.4 Receive Windows Parameters (*RXParamSetupReq*, *RXParamSetupAns*)

The ***RXParamSetupReq*** command allows a change to the frequency and the data rate set for the second receive window (RX2) following each uplink. The command also allows to program an offset between the uplink and the RX1 slot downlink data rates.

Size (bytes)	1	3
<b>RXParamSetupReq Payload</b>	DLsettings	Frequency

Bits	7	6:4	3:0
<b>DLsettings</b>	RFU	RX1DROffset	RX2DataRate

The **RX1DROffset** field sets the offset between the uplink data rate and the downlink data rate used to communicate with the end-device on the first reception slot (RX1). As a default this offset is 0. The offset is used to take into account maximum power density constraints for base stations in some regions and to balance the uplink and downlink radio link margins.

The data rate (**RX2DataRate**) field defines the data rate of a downlink using the second receive window following the same convention as the ***LinkADRReq*** command (0 means DR0/125kHz for example). The frequency (**Frequency**) field corresponds to the frequency of the channel used for the second receive window, whereby the frequency is coded following the convention defined in the ***NewChannelReq*** command.

The ***RXParamSetupAns*** command is used by the end-device to acknowledge the reception of ***RXParamSetupReq*** command. The ***RXParamSetupAns*** command should be added in the FOpt field of all uplinks until a class A downlink is received by the end-device. This guarantees that even in presence of uplink packet loss, the network is always aware of the downlink parameters used by the end-device.

The payload contains a single status byte.

Size (bytes)	1
<b>RXParamSetupAns Payload</b>	Status

The status (**Status**) bits have the following meaning.

Bits	7:3	2	1	0
<b>Status bits</b>	RFU	RX1DROffset ACK	RX2 Data rate ACK	Channel ACK

	Bit = 0	Bit = 1
<b>Channel ACK</b>	The frequency requested is not usable by the end-device.	RX2 slot channel was successfully set
<b>RX2 Data rate ACK</b>	The data rate requested is unknown to the end-device.	RX2 slot data rate was successfully set
<b>RX1DROffset ACK</b>	the uplink/downlink data rate offset for RX1 slot is not in the allowed range	RX1DROffset was successfully set

**Table 7: RX2SetupAns status bits signification**

If either of the 3 bits is equal to 0, the command did not succeed and the previous parameters are kept.

## 5.5 End-Device Status (*DevStatusReq*, *DevStatusAns*)

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.

Size (bytes)	1	1
<b>DevStatusAns Payload</b>	Battery	Margin

The battery level (**Battery**) reported is encoded as follows:

Battery	Description
0	The end-device is connected to an external power source.
1..254	The battery level, 1 being at minimum and 254 being at maximum
255	The end-device was not able to measure the battery level.

Table 8: Battery level decoding

The margin (**Margin**) is the demodulation signal-to-noise ratio in dB rounded to the nearest integer value for the last successfully received **DevStatusReq** command. It is a signed integer of 6 bits with a minimum value of -32 and a maximum value of 31.

Bits	7:6	5:0
<b>Status</b>	RFU	Margin

## 5.6 Creation / Modification of a Channel (*NewChannelReq*, *NewChannelAns*)

The **NewChannelReq** command can be used to either modify the parameters of an existing channel or to create a new one. The command sets the center frequency of the new channel and the range of data rates usable on this channel:

Size (bytes)	1	3	1
<b>NewChannelReq Payload</b>	ChIndex	Freq	DrRange

The channel index (**ChIndex**) is the index of the channel being created or modified. Depending on the region and frequency band used, the LoRaWAN specification imposes default channels which must be common to all devices and cannot be modified by the **NewChannelReq** command (cf. Chapter 6). If the number of default channels is  $N$ , the default channels go from 0 to  $N-1$ , and the acceptable range for **ChIndex** is  $N$  to 15. A device must be able to handle at least 16 different channel definitions. In certain region the device may have to store more than 16 channel definitions.

The frequency (**Freq**) field is a 24 bits unsigned integer. The actual channel frequency in Hz is  $100 \times \text{Freq}$  whereby values representing frequencies below 100 MHz are reserved for future use. This allows setting the frequency of a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps. A **Freq** value of 0 disables the channel. The end-device has to check that the frequency is actually allowed by its radio hardware and return an error otherwise.

The data-rate range (**DrRange**) field specifies the data-rate range allowed for this channel. The field is split in two 4-bit indexes:

Bits	7:4	3:0
<b>DrRange</b>	MaxDR	MinDR

Following the convention defined in Section 5.2 the minimum data rate (**MinDR**) subfield designate the lowest data rate allowed on this channel. For example 0 designates DR0 / 125 kHz. Similarly, the maximum data rate (**MaxDR**) designates the highest data rate. For example, DrRange = 0x77 means that only 50 kbps GFSK is allowed on a channel and DrRange = 0x50 means that DR0 / 125 kHz to DR5 / 125 kHz are supported.

The newly defined channel is enabled and can immediately be used for communication.

The end-device acknowledges the reception of a **NewChannelReq** by sending back a **NewChannelAns** command. The payload of this message contains the following information:

<b>Size (bytes)</b>	1
<b>NewChannelAns Payload</b>	Status

The status (**Status**) bits have the following meaning:

<b>Bits</b>	7:2	1	0
<b>Status</b>	RFU	Data rate range ok	Channel frequency ok

	<b>Bit = 0</b>	<b>Bit = 1</b>
<b>Data rate range ok</b>	The designated data rate range exceeds the ones currently defined for this end-device	The data rate range is compatible with the possibilities of the end-device
<b>Channel frequency ok</b>	The device cannot use this frequency	The device is able to use this frequency.

**Table 9: NewChannelAns status bits signification**

If either of those 2 bits equals 0, the command did not succeed and the new channel has not been created.

## 5.7 Setting delay between TX and RX (*RXTimingSetupReq*, *RXTimingSetupAns*)

The **RXTimingSetupReq** command allows configuring the delay between the end of the TX uplink and the opening of the first reception slot. The second reception slot opens one second after the first reception slot.

<b>Size (bytes)</b>	1
<b>RXTimingSetupReq Payload</b>	Settings

The delay (**Delay**) field specifies the delay. The field is split in two 4-bit indexes:

<b>Bits</b>	7:4	3:0
<b>Settings</b>	RFU	Del

The delay is expressed in seconds. **Del** 0 is mapped on 1 s.

<b>Del</b>	<b>Delay [s]</b>
0	1
1	1
2	2
3	3
..	..
15	15

**Table 10: Del mapping table**

An end device answers **RXTimingSetupReq** with **RXTimingSetupAns** with no payload.

The **RXTimingSetupAns** command should be added in the FOpt field of all uplinks until a class A downlink is received by the end-device. This guarantees that even in presence of uplink packet loss, the network is always aware of the downlink parameters used by the end-device.

## 5.8 Reset indication commands (**ResetInd**, **ResetConf**)

With the **ResetInd** command, an end-device indicates to the network that it has been re-initialized and it that he has switched back to its default parameters (id the parameters originally programmed into the device at fabrication). The **ResetInd** command must be added to the FOpt field of all uplinks until a **ResetConf** is received.

The **ResetInd** command includes the major and minor of the LoRaWAN version supported by the end device and a variable length bitmap with supported options.

Size (bytes)	1	1	1	n
<b>ResetInd Payload</b>	LoRaWAN version	Nonce	Options_bitmap	bitmap

Size (bytes)	7:4	3:0
<b>LoRaWAN version</b>	Major	Minor

The Major field indicates the major of the LoRaWAN version supported by the end-device.

Major version	Major
RFU	0
1 (LoRaWAN 1.x)	1
1 (LoRaWAN 2.x)	2

The minor field indicates the minor of the LoRaWAN version supported by the end-device.

Minor version	Minor
0 (LoRaWAN x.0)	0
1 (LoRaWAN x.1)	1
RFU	2:15

**Nonce** is a true (that may be derived from radio RSSI for example) random value. For each end-device, the network server keeps track of a certain number of **{Nonce,FCounter}** couple values used by the end-device in the past, and ignores reset indication with any of these **{Nonce,FCounter}** couple values from that end-device. A new random Nonce value must be generated for each uplink until the ResetConf command is received.

Devices that perform the JOIN procedure may set the Nonce value to zero, and the network shall ignore the Nonce for all over-the-air JOIN procedure based devices.



**Note:** Devices that perform the JOIN procedure shall send the reset indication immediately after receiving the JOIN ACCEPT. Reset Indications sent by such devices at any later time should be discarded by the network as they may be a replay attack.

Size (bytes)	7:4	3:0
Options_bitmap field	Bitmap length	Bitmap

Bitmap length	Bitmap length
Bitmap is made of 4 bits	0
Bitmap is made of 4 bits + 1 Bytes	1
...	...

Each bit of the bitmap indicates whether an optional protocol capability is supported by the end-device or not. The description of the negotiable options is in chapter 7

When a **ResetInd** is received by the network server, it responds with a **ResetConf** command.

The ResetConf command contains the same bitmap as received in the ResetInd command, with each bit indicating which of the options supported by the end-device is also supported by the network. The Network generates this response by performing a bit-wise AND between its own capability mask and that sent by the end-device.

Both network and end-device will then after successful reception of the ResetConf message know the set of common options that will be used in the subsequent communications.

In case the ResetInd and hence option negotiation cannot be performed (OptNeg bit of JOIN ACCEPT message is unset – see section 6.2.5) all options bits shall be considered zero. The device shall behave according to the LoRaWAN1.0 specification.

Size (bytes)	1	0 to 15
ResetConf Payload	Options_bitmap	bitmap

## 5.9 ADR parameters (*ADRParamSetupReq*, *ADRParamSetupAns*)

The **ADRParamSetupReq** command allows to change the ADR\_ACK\_LIMIT and ADR\_ACK\_DELAY parameters defining the ADR back-off algorithm. The ADRParamSetupReq command has a single byte payload.

Size (bytes)	1
ADRParamSetupReq Payload	ADRparam

Bits	7:4	3:0
ADRparam	Limit_exp	Delay_exp

The Limit\_exp field sets the ADR\_ACK\_LIMIT parameter value.

$$\text{ADR\_ACK\_LIMIT} = 2^{\text{Limit\_exp}}$$

The Limit\_exp valid range is 0 to 15, corresponding to a range of 1 to 32768 for ADR\_ACK\_LIMIT



The Delay\_exp field sets the ADR\_ACK\_DELAY parameter value.

$$\text{ADR\_ACK\_DELAY} = 2^{\text{Delay\_exp}}$$

The Delay\_exp valid range is 0 to 15, corresponding to a range of 1 to 32768 for ADR\_ACK\_DELAY

The **ADRParamSetupAns** command is used by the end-device to acknowledge the reception of **ADRParamSetupReq** command. The **ADRParamSetupAns** command has no payload field.

## 5.10 DeviceTime commands (*DeviceTimeReq*, *DeviceTimeAns*)

With the **DeviceTimeReq** command, an end-device may request from the network the current network date and time. The request has no payload.

With the **DeviceTimeAns** command, the network server provides the network date and time in to the end device. The command has a 6 bytes payload defined as follows:

Size (bytes)	4	2
<b>DeviceTimeAns Payload</b>	32-bit integer : Seconds since epoch*	fractional-second in $\frac{1}{2}^{16}$ second steps

~~When a **DeviceTimeAns** command is received by the end device from the network server, it responds with an acknowledgement.~~

=  
 (\*) The GPS epoch (id Sunday January the 6<sup>th</sup> 1980 at midnight) is used as origin. The “seconds” field is the number of seconds elapsed since the origin. This field is monotonically increasing by 1 every second. To convert this field to UTC time, the leap seconds must be taken into account.

Example : Friday 12<sup>th</sup> of february 2016 at 14:24:31 UTC corresponds to 483888 sec since GPS epoch. At this date the GPS time is 17seconds ahead of UTC time.

## 6 End-Device Activation

To participate in a LoRaWAN network, each end-device has to be personalized and activated.

Activation of an end-device can be achieved in two ways, either via **Over-The-Air Activation** (OTAA) when an end-device is deployed or reset, or via **Activation By Personalization** (ABP) in which the two steps of end-device personalization and activation are done as one step.

### 6.1 Data Stored in the End-device after Activation

After activation, the following information is stored in the end-device: a device address (**DevAddr**), an application identifier (**AppEUI**), a pair of network session key (**NwkSKeyUp/Dwn**), and an application session key (**AppSKey**).

#### 6.1.1 End-device address (DevAddr)

The **DevAddr** consists of 32 bits identifies the end-device within the current network. Its format is as follows:

Bit#	[31..25]	[24..0]
DevAddr bits	NwkID	NwkAddr

The most significant 7 bits are used as network identifier (**NwkID**) to separate addresses of territorially overlapping networks of different network operators and to remedy roaming issues. The least significant 25 bits, the network address (**NwkAddr**) of the end-device, can be arbitrarily assigned by the network manager.

#### 6.1.2 Application identifier (AppEUI)

The **AppEUI** is a global application ID in IEEE EUI64 address space that uniquely identifies the entity able to process the JoinReq frame.

The **AppEUI** is stored in the end-device before the activation procedure is executed.

#### 6.1.3 Uplink Network session key (NwkSKeyUp)

The **NwkSKeyUp** is a **network session key** specific for the end-device. It is used by the end-device to calculate the **MIC** (message integrity code) of all uplink data messages to ensure data integrity. It is further used to encrypt the payload field of a uplink MAC-only data messages.

#### 6.1.4 Downlink Network session key (NwkSKeyDwn)

The **NwkSKeyDwn** is a **network session key** specific for the end-device. It is used by the end-device to verify the **MIC** (message integrity code) of all downlink data messages to ensure data integrity. It is further used to decrypt the payload field of a MAC-only data downlink message. When a device connects to a LoRaWAN1.0 network server the same key is used for both uplink & downlink MIC calculation. In that case **NwkSKeyDwn takes the same value than NwkSKeyUp**

### 6.1.5 Application session key (AppSKey)

The **AppSKey** is an **application session key** specific for the end-device. It is used by both the **network-application** server and the end-device to encrypt and decrypt the payload field of application-specific data messages. It is also used to calculate and verify an application-level **MIC** that may be included in the payload of application-specific data messages.

## 6.2 Over-the-Air Activation

For over-the-air activation, end-devices must follow a join procedure prior to participating in data exchanges with the network server. An end-device has to go through a new join procedure every time it has lost the session context information.

The join procedure requires the end-device to be personalized with the following information before it starts the join procedure: a globally unique end-device identifier (**DevEUI**), the application identifier (**AppEUI**), and two AES-128 keys (**NwkKey** and **AppKey**).

The **AppEUI** is described above in 6.1.2.

**Note:** For over-the-air-activation, end-devices are not personalized with the network session key. Instead, whenever an end-device joins a network, a network session key specific for that end-device is derived to encrypt and verify transmissions at the network level. This way, roaming of end-devices between networks of different providers is facilitated. Using both a network session key and an application session key further allows federated network servers in which application data cannot be read or tampered with by the network provider.

### 6.2.1 End-device identifier (DevEUI)

The **DevEUI** is a global end-device ID in IEEE EUI64 address space that uniquely identifies the end-device.

### 6.2.2 Application root keys (AppKey NwkKey)

The **NwkKey** and **AppKey** are AES-128 root keys specific to the end-device that are assigned by the application owner to the end-device during fabrication and most likely derived from an application-specific root key exclusively known to and under the control of the application provider.<sup>1</sup> Whenever an end-device joins a network via over-the-air activation, the **NwkKey** is used to derive the **NwkSKeyUp** and **NwkSKeyDwn** session keys, and **AppKey** is used to derive the **AppSKey** session key.

**Note:** As the application session key is derived only from the **AppKey**, the **NwkKey** can generally be surrendered to the network for the network to manage the JOIN procedure without enabling the operator to eavesdrop on the application payload data.

To ensure backward compatibility with LoraWAN 1.0 and earlier network servers that do not support two root keys, the end-device must default back to the single root key scheme when

1. Since all end-devices are equipped with unique application and network root keys specific for each end-device, extracting the **AppKey**/**NwkKey** from an end-device only compromises this one end-device.

joining such a network. This condition is signaled to the end-device by the “OptNeg” bit (bit 7) of the DLsetting field of the JOIN ACCEPT message being zero.

The end-device in this case must

- Use the NwkKey to derive both the AppSKey and the NwkSKeyUp session keys as in LoRaWAN1.0 specification.
- Set the NwkSKeyDwn equal to NwkSKeyUp, the same network session key is effectively used for both uplink and downlink MIC calculation according to the LoRaWAN1.0 specification.

### 6.2.3 Join procedure

From an end-device’s point of view, the join procedure consists of two MAC messages exchanged with the server, namely a **join request** and a **join accept**.

### 6.2.4 Join-request message

The join procedure is always initiated from the end-device by sending a join-request message.

Size (bytes)	8	8	2
Join Request	AppEUI	DevEUI	DevNonce

The join-request message contains the **AppEUI** and **DevEUI** of the end-device followed by a **nonce** of 2 octets (**DevNonce**).

**DevNonce** is a random value.<sup>1</sup> For each end-device, the network server keeps track of a certain number of **DevNonce** values used by the end-device in the past, and ignores join requests with any of these **DevNonce** values from that end-device.

**Note:** This mechanism prevents replay attacks by sending previously recorded join-request messages with the intention of disconnecting the respective end-device from the network.

The message integrity code (**MIC**) value (see Chapter 4 for MAC message description) for a join-request message is calculated as follows:<sup>2</sup>

$$cmac = aes128\_cmac(NwkKey, MHDR \mid AppEUI \mid DevEUI \mid DevNonce)$$

$$MIC = cmac[0..3]$$

The join-request message is not encrypted. The join-request message can be transmitted using any data rate and following a random frequency hopping sequence across the specified join channels. It is recommended to use a plurality of data rates. The intervals between transmissions of **Join-Requests** shall respect the condition described in chapter 9.

<sup>1</sup> The DevNonce can be extracted by issuing a sequence of RSSI measurements under the assumption that the quality of randomness fulfills the criteria of true randomness

<sup>2</sup> [RFC4493]

### 6.2.5 Join-accept message

The network server will respond to the **join-request** message with a **join-accept** message if the end-device is permitted to join a network. The join-accept message is sent like a normal downlink but uses delays JOIN\_ACCEPT\_DELAY1 or JOIN\_ACCEPT\_DELAY2 (instead of RECEIVE\_DELAY1 and RECEIVE\_DELAY2, respectively). The channel frequency and data rate used for these two receive windows are identical to the one used for the RX1 and RX2 receive windows described in the “receive windows” section of the “Physical Layer” chapter

No response is given to the end-device if the join request is not accepted.

The join-accept message contains an application nonce (**AppNonce**) of 3 octets, a network identifier (**NetID**), an end-device address (**DevAddr**), a delay between TX and RX (**RxDelay**) and an optional list of channel frequencies (**CFList**) for the network the end-device is joining. The CFList option is region specific and is defined in Section 8.

Size (bytes)	3	3	4	1	1	(16) Optional
Join Accept	AppNonce	NetID	DevAddr	DLSettings	RxDelay	CFList

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 three session keys **NwkSKeyUp**, **NwkSKeyDwn** and **AppSKey** as follows:<sup>1</sup>

If the OptNeg is unset, the AppSKey is derived from the AppKey

$\text{AppSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x02 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce} \mid \text{pad}_{16})$

$\text{NwkSKeyUp} = \text{aes128\_encrypt}(\text{AppKey}, 0x01 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce} \mid \text{pad}_{16})$

and  $\text{NwkSKeyDwn} = \text{NwkSKeyUp}$ .

Else if the OptNeg is set, the AppSKey is derived from AppKey:

$\text{AppSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x02 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce} \mid \text{pad}_{16})$

And the NwkSKeyUp&Dwn are derived from the NwkKey:

$\text{NwkSKeyUp} = \text{aes128\_encrypt}(\text{NwkKey}, 0x01 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce} \mid \text{pad}_{16})$

$\text{NwkSKeyDwn} = \text{aes128\_encrypt}(\text{NwkKey}, 0x03 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce} \mid \text{pad}_{16})$

The MIC value for a join-accept message is calculated as follows:<sup>2</sup>

$\text{cmac} = \text{aes128\_cmac}(\text{NwkKey},$

$\text{MHDR} \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevAddr} \mid \text{DLSettings} \mid \text{RxDelay} \mid \text{CFList})$

$\text{MIC} = \text{cmac}[0..3]$

The join-accept message itself is encrypted with the **NwkKey** as follows:

$\text{aes128\_decrypt}(\text{NwkKey}, \text{AppNonce} \mid \text{NetID} \mid \text{DevAddr} \mid \text{DLSettings} \mid \text{RxDelay} \mid \text{CFList} \mid \text{MIC})$

**Note:** The network server uses an AES decrypt operation in ECB mode to encrypt the join-accept message so that the end-device can use an

<sup>1</sup> The  $\text{pad}_{16}$  function appends zero octets so that the length of the data is a multiple of 16.

<sup>2</sup> [RFC4493]

AES encrypt operation to decrypt the message. This way an end-device only has to implement AES encrypt but not AES decrypt.

**Note:** Establishing these three session keys allows for a federated network server infrastructure in which network operators are not able to eavesdrop on application data. In such a setting, the application provider must support the network operator in the process of an end-device actually joining the network and establishing the **NwkSKey** for the end-device, or the application can provide directly the **NwkKey** to the network enabling the network to perform all the JOIN procedure steps. At the same time the application provider commits to the network operator that it will take the charges for any traffic incurred by the end-device and retains full control over the **AppSKey** used for protecting its application data.

**Note:** The device's protocol version (1.0 or 1.1) is registered on the network side out-of-band at the same time than the DevEUI and the device's **AppKey** and possibly **NtwKey**

The **NetID** field uniquely identifies a network operator. This code is managed by the Lora-alliance. The seven LSB of the **NetID** are called **NwkID**

The 7 MSBs of the devAddr of all devices of a network must match **NwkID**. This is to simplify the discovery of a device's home network in case of roaming.

The **DLsettings** field contains the downlink configuration:

Bits	7	6:4	3:0
<b>DLsettings</b>	OptNeg	RX1DRoffset	RX2 Data rate

The **OptNeg** bit indicates whether the network server implements the LoRaWAN1.0 protocol version (unset) or 1.1 and later (set). When the **OptNeg** bit is set

- The optional protocol features are negotiated between the end-device and the network server through the *ResetInd/ResetConf* MAC command exchange.
- The device derives **NwkSKeyUp** & **NwkSKeyDwn** from the **NwkKey**
- The device derives **AppSKey** from the **AppKey**

When the **OptNeg** bit is not set

- The device reverts to LoRaWAN1.0, no options can be negotiated
- The *ResetInd* command is not sent by the device
- The device derives **NwkSKeyUp** & **AppSKey** from the **AppKey**
- The device sets **NwkSKeyDwn** equal to **NwkSKeyUp**

The **RX1DRoffset** field sets the offset between the uplink data rate and the downlink data rate used to communicate with the end-device on the first reception slot (RX1). By default this offset is 0.. The offset is used to take into account maximum power density constraints for base stations in some regions and to balance the uplink and downlink radio link margins.

The actual relationship between the uplink and downlink data rate is region specific and detailed in the "Physical Layer" section

The delay **RxDelay** follows the same convention as the **Delay** field in the **RXTimingSetupReq** command.

The CFlist field is optional. If the Join-accept message is received following the transmission of a Rejoin-request and if the CFlist field is absent, the device should keep its current channels definition unchanged. If the CFlist is present, it overrides all currently defined channels.

## 6.2.6 ReJoin-request message

Once activated a device may periodically transmit a Rejoin-request message on top of its normal applicative traffic. This Rejoin-request message periodically gives the network the opportunity to change the device address and session keys. This may be used to hand-over a device between two networks or to rekey and/or change devAddr of a device on a given network.

The Rejoin procedure is always initiated from the end-device by sending a Rejoin-request message.

Size (bytes)	5	3	8	2
Join Request	RFU	NetID	DevEUI	RJcount

The Rejoin-request message contains the **NetID** and **DevEUI** of the end-device followed by a 16 bits counter (**RJcount**).

**RJcount** is a counter incremented with every Rejoin frame transmitted. For each end-device, the network server keeps track of the last **RJcount** value (called **RJcount\_last**) used by the end-device. It ignores Rejoin requests if  $(\text{RJcount} - \text{RJcount\_last}) \bmod 2^{16} > 2^{15}$

**Note:** This mechanism prevents replay attacks by sending previously recorded Rejoin-request messages

The message integrity code (**MIC**) value (see Chapter 4 for MAC message description) for a Rejoin-request message is calculated as follows:<sup>1</sup>

$$cmac = \text{aes128\_cmac}(\text{NwkKey}, \text{MHDR} \mid 0x000000000000 \mid \text{NetID} \mid \text{DevEUI} \mid \text{RJcount})$$

$$\text{MIC} = cmac[0..3]$$

The Rejoin-request message is not encrypted.

The network server may respond to the **Rejoin-request** message with a **join-accept** (as defined in **Error! Reference source not found.**) message if it wants to modify the device network identity. **RJcount replaces DevNonce in key derivation**

The device's **Rejoin-Req** transmissions duty-cycle shall always be <0.1%

## 6.3 Activation by Personalization

Under certain circumstances, end-devices can be activated by personalization. Activation by personalization directly ties an end-device to a specific network by-passing the **join request - join accept** procedure.

<sup>1</sup> [RFC4493]



1 Activating an end-device by personalization means that the **DevAddr** and the three session  
2 keys **NwkSKeyUp/Dwn** and **AppSKey** are directly stored into the end-device instead of the  
3 **DevEUI**, **AppEUI** and the **AppKey**. The end-device is equipped with the required  
4 information for participating in a specific LoRa network when started.

5 Each device should have a unique set of NwkSKeyUp/Dwn and AppSKey. Compromising the  
6 keys of one device shouldn't compromise the security of the communications of other devices.  
7 The process to build those keys should be such that the keys cannot be derived in any way  
8 from publicly available information (like the node address for example).

9 When a personalized end-device accesses the network for the first time or after a  
10 re-initialization, it shall transmit the ResetInd MAC command in the FOpt field of all uplink  
11 messages until it receives a ResetConf command from the network. After a re-initialization  
12 the end-device must use its default configuration (id the configuration that was used when the  
13 device was first connected to the network).  
14



## 7 Protocol option negotiation

The following table summarizes the list of protocol options that shall be negotiated via the Options\_bitmap field of the *RestInd* / *RestConf* command exchange .

Protocol option	Bit Position
Two Downlink FCNTs scheme	1
32-bits frame counter	2
ADRPParamSetupReq/Ans support	3
RFU	4

A bit set to 1 (resp 0) indicates that the option is supported (resp not supported).

### 7.1 Two Downlink FCNTs scheme

If this option is supported by both the end-device and the network server, two different downlink frame counters shall be used depending on the port as described in 4.3.1.5

### 7.2 32-bits frame counter

If this option is supported by both the end-device and the network server, 32bits frame counters are used for both uplink and downlink frame counters. Only the 16LSBs of the frame counters actually appear in the transmitted frames as described in 4.3.1.5

If this option is not supported, only 16 bits frame counters shall be used.

### 7.3 ADRParamSetupReq/Ans support

If this option is supported by the end-device and the network server, the MAC command ADRParamSetupReq can be send by the network server to change the data rate decay parameters (ADR\_ACK\_LIMIT and ADR\_ACK\_DELAY) of the ADR algorithm.

If this option is not supported, the ADR decay parameters programmed in the end-device cannot be modified.

## 8 Physical Layer

### 8.1 EU 863-870MHz ISM Band

#### 8.1.1 EU863-870 Preamble Format

The following synchronization words should be used:

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols
GFSK	0xC194C1	5 bytes

Table 11: EU863-870 synch words

#### 8.1.2 EU863-870 ISM Band channel frequencies

This section applies to regions where the 868MHz ISM band usage is defined by ETSI [EN300.220].

The network channels can be freely attributed by the network operator. However the three following default channels must be implemented in every EU868MHz end-device. Those channels are the minimum set that all network gateways should always be listening on.

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	3	<1%

Table 12: EU863-870 default channels

In order to access the physical medium the ETSI regulations impose some restrictions such maximum time the transmitter can be on or the maximum time a transmitter can transmit per hour. The ETSI regulations allow the choice of using either a duty-cycle limitation or a so-called **Listen Before Talk Adaptive Frequency Agility** (LBT AFA) transmissions management. The current LoRaWAN specification exclusively uses duty-cycled limited transmissions to comply with the ETSI regulations.

EU868MHz ISM band end-devices should use the following default parameters

- Default radiated transmit output power: 14 dBm

EU868Mhz end-devices should be capable of operating in the 863 to 870 MHz frequency band and should feature a channel data structure to store the parameters of at least 16 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

The first three channels correspond to 868.1, 868.3, and 868.5 MHz / DR0 to DR5 and must be implemented in every end-device. Those default channels cannot be modified through the **NewChannelReq** command and guarantee a minimal common channel set between end-devices and network gateways.

The following table gives the list of frequencies that should be used by end-devices to broadcast the JoinReq message. The JoinReq message transmit duty-cycle shall respect the condition described in chapter 9.

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	868.10 868.30 868.50	DR0 – DR5 / 0.3-5 kbps	6	<0.1%

**Table 13: EU863-870 JoinReq Channel List**

### 8.1.3 EU863-870 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (DR) and End-point Output Power (TXPower) in the EU863-870 band:

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

**Table 14: Data rate and TX power table**

TXPower	Configuration
0	20 dBm (if supported)
1	14 dBm
2	11 dBm
3	8 dBm
4	5 dBm
5	2 dBm
6..15	RFU

**Table 15: Data rate and TX power table**

### 8.1.4 EU863-870 JoinAccept CFList

The EU 863-870 ISM band LoRaWAN implements an optional **channel frequency list** (CFList) of 16 octets in the JoinAccept message.

In this case the **CFList** is a list of five channel frequencies for the channels four to eight whereby each frequency is encoded as a 24 bits unsigned integer (three octets). All these channels are usable for DR0 to DR5 125kHz LoRa modulation.. The list of frequencies is followed by a single **CFListType** octet for a total of 16 octets. The **CFListType** shall be equal to zero (0) to indicate that the **CFList** contains a list of frequencies.

Size (bytes)	3	3	3	3	3	1
<b>CFList</b>	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	<b>CFListType</b>

The actual channel frequency in Hz is 100 x frequency whereby values representing frequencies below 100 Mhz are reserved for future use. This allows setting the frequency of a channel anywhere between 863 and 870MHz in 100 Hz steps. Unused channels have a frequency value of 0. The **CFList** is optional and its presence can be detected by the length of the join-accept message. If present, the **CFList** replaces all the previous channels stored in the end-device apart from the three default channels as defined in Chapter 8. The newly defined channels are immediately enabled and usable by the end-device for communication.

### 8.1.5 EU863-870 LinkAdrReq command

The EU863-870 LoRaWAN only supports a maximum of 16 channels. When **ChMaskCntl** field is 0 the **ChMask** field individually enables/disables each of the 16 channels.

<b>ChMaskCntl</b>	<b>ChMask applies to</b>
0	Channels 1 to 16
1	RFU
..	..
4	RFU
5	RFU
6	All channels ON The device should enable all currently defined channels independently of the <b>ChMask</b> field value.
7	RFU

Table 16: **ChMaskCntl** value table

If the **ChMask** field value is one of values meaning RFU, the end-device should reject the command and unset the “**Channel mask ACK**” bit in its response.

### 8.1.6 EU863-870 Maximum payload size

The maximum **MACPayload** size length (*M*) is given by the following table. It is derived from limitation of the PHY layer depending on the effective modulation rate used taking into account a possible repeater encapsulation layer. The maximum application payload length in the absence of the optional **FOpt** control field (*N*) is also given for information only. The value of *N* might be smaller if the **FOpt** field is not empty:

<b>DataRate</b>	<b><i>M</i></b>	<b><i>N</i></b>
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222

6	230	222
7	230	222
8:15	Not defined	

**Table 17: EU863-870 maximum payload size**

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	M	N
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242
8:15	Not defined	

**Table 18 : EU863-870 maximum payload size (not repeater compatible)**

### 8.1.7 EU863-870 Receive windows

The RX1 receive window uses the same channel than the preceding uplink. The data rate is a function of the uplink data rate and the RX1DROffset as given by the following table. The allowed values for RX1DROffset are in the [0:5] range. Values in the [6:7] range are reserved for future use.

RX1DROffset Upstream data rate	0	1	2	3	4	5
	Downstream data rate in RX1 slot					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0

RX1DROffset Upstream data rate	0	1	2	3	4	5
	Downstream data rate in RX1 slot					
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2

The RX2 receive window uses a fixed frequency and data rate. The default parameters are 869.525 MHz / DR0 (SF12, 125 kHz)

### 8.1.8 EU863-870 Default Settings

The following parameters are recommended values for the EU863-870Mhz band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE\_DELAY1 and RECEIVE\_DELAY2 latency), those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

## 8.2 US 902-928MHz ISM Band

### 8.2.1 US902-928 Preamble Format

The following synchronization words should be used:

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols

LoRaWAN does not make use of GFSK modulation in the US902-928 ISM band.

### 8.2.2 US902-928 Channel Frequencies

The 915 MHz ISM Band shall be divided into the following channel plans.

- Upstream – 64 channels numbered 0 to 63 utilizing LoRa 125 kHz BW varying from DR0 to DR3, using coding rate 4/5, starting at 902.3 MHz and incrementing linearly by 200 kHz to 914.9 MHz
- Upstream – 8 channels numbered 64 to 71 utilizing LoRa 500 kHz BW at DR4 starting at 903.0 MHz and incrementing linearly by 1.6 MHz to 914.2 MHz
- Downstream – 8 channels numbered 0 to 7 utilizing LoRa 500 kHz BW at DR8 to DR13) starting at 923.3 MHz and incrementing linearly by 600 kHz to 927.5 MHz

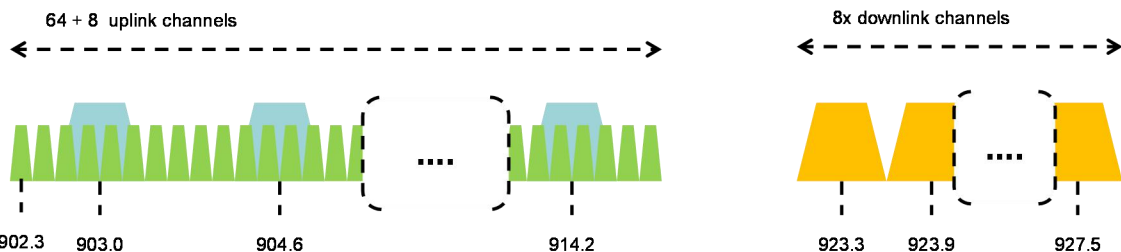


Figure 10: US902-928 channel frequencies

915 MHz ISM band end-devices should use the following default parameters:

- Default radiated transmit output power: 20 dBm
  - Devices, when transmitting with 125 kHz BW may use a maximum of +30 dBm. The transmission should never last more than 400 ms.
  - Devices, when transmitting with 500 kHz BW may use a maximum of +26 dBm

US902-928 end-devices should be capable of operating in the 902 to 928 MHz frequency band and should feature a channel data structure to store the parameters of 72 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

If using the over-the-air activation procedure, the end-device should broadcast the JoinReq message alternatively on a random 125 kHz channel amongst the 64 channels defined using **DR0** and a random 500 kHz channel amongst the 8 channels defined using **DR4**. The end-device should change channel for every transmission.

Personalized devices shall have all 72 channels enabled following a reset.

### 8.2.3 US902-928 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (**DR**) and End-point Output Power (**TXPower**) in the US902-928 band:

DataRate	Configuration	Indicative physical bit rate [bit/sec]	TXPower	Configuration
0	LoRa: SF10 / 125 kHz	980	0	30 dBm – 2*TXpower
1	LoRa: SF9 / 125 kHz	1760	1	28 dBm
2	LoRa: SF8 / 125 kHz	3125	2	26 dBm
3	LoRa: SF7 / 125 kHz	5470	3 : 9	....
4	LoRa: SF8 / 500 kHz	12500	10	10 dBm
5:7	RFU		11:15	RFU
8	LoRa: SF12 / 500 kHz	980		
9	LoRa: SF11 / 500 kHz	1760		
10	LoRa: SF10 / 500 kHz	3900		
11	LoRa: SF9 / 500 kHz	7000		
12	LoRa: SF8 / 500 kHz	12500		
13	LoRa: SF7 / 500 kHz	21900		
14:15	RFU			

**Table 19: Data rate and TX power table**

DR4 is identical to DR12, DR8..13 must be implemented in end-devices and are reserved for future applications

## 8.2.4 US902-928 JoinAccept CFList

The US902-928 LoRaWAN supports the use of the optional **CFList** appended to the JoinResp message. If the **CFList** is not empty then the CFListType field shall contain the value one (0x01) to indicate the CFList contains a series of ChMask fields. The ChMask fields are interpreted as being controlled by a virtual ChMaskCntl that initializes to a value of zero (0) and increments for each ChMask field to a value of five (5).

Size (bytes)	[2]	[2]	[2]	[2]	[2]	[2]	[3]	[1]
CFList	ChMask	ChMask	ChMask	ChMask	ChMask	ChMask	RFU	CFListType

For example, the first ChMask 16bits field contains the channel mask controlling the channels 0 to 15.

## 8.2.5 US902-928 LinkAdrReq command

For the US902-928 version the **ChMaskCntl** field of the **LinkADRReq** command has the following meaning:

ChMaskCntl	ChMask applies to
0	Channels 0 to 15
1	Channels 16 to 31
..	..
4	Channels 64 to 71
5	RFU
6	All 125 kHz ON ChMask applies to channels 64 to 71
7	All 125 kHz OFF ChMask applies to channels 64 to 71

**Table 20: ChMaskCntl value table**



If **ChMaskCntl** = 6 (resp 7) then 125 kHz channels are enabled (resp disabled). Simultaneously the channels 64 to 71 are set according to the **ChMask** bit mask.

**Note:** FCC regulation requires hopping over at least 50 channels when using maximum output power. It is possible to have end-devices with less channels (at least six 125 kHz channels) when limiting the end-device transmit power to 21 dBm.

## 8.2.6 US902-928 Maximum payload size

The maximum **MACPayload** size length ( $M$ ) is given by the following table. It is derived from the maximum allowed transmission time at the PHY layer taking into account a possible repeater encapsulation. The maximum application payload length in the absence of the optional **FOpt** MAC control field ( $N$ ) is also given for information only. The value of  $N$  might be smaller if the **FOpt** field is not empty:

DataRate	$M$	$N$
0	19	11
1	61	53
2	134	126
3	250	242
4	250	242
5:7	Not defined	
8	41	33
9	117	109
10	230	222
11	230	222
12	230	222
13	230	222
14:15	Not defined	

Table 21: US902-928 maximum payload size (repeater compatible)

The greyed lines correspond to the data rates that may be used by an end-device behind a repeater.

If the end-device will never operate under a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	$M$	$N$
0	19	11
1	61	53
2	134	126
3	250	242
4	250	242
5:7	Not defined	
8	61	53
9	137	129
10	250	242
11	250	242
12	250	242
13	250	242
14:15	Not defined	

Table 22 : US902-928 maximum payload size (not repeater compatible)

## 8.2.7 US902-928 Receive windows

- The RX1 receive channel is a function of the upstream channel used to initiate the data exchange. The RX1 receive channel can be determined as follows.
  - RX1 Channel Number = Transmit Channel Number modulo 8
- The RX1 window data rate depends on the transmit data rate (see Table 23 below).
- The RX2 (second receive window) settings uses a fixed data rate and frequency. Default parameters are 923.3Mhz / DR8

Upstream data rate	Downstream data rate			
RX1DROffset	0	1	2	3
DR0	DR10	DR9	DR8	DR8
DR1	DR11	DR10	DR9	DR8
DR2	DR12	DR11	DR10	DR9
DR3	DR13	DR12	DR11	DR10
DR4	DR13	DR13	DR12	DR11

**Table 23: US902-928 data rate RX offset**

The allowed values for RX1DROffset are in the [0:3] range. Values in the range [4:7] are reserved for future use.

## 8.2.8 US902-928 Default Settings

The following parameters are recommended values for the US902-928 band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE\_DELAY1 & 2 latency), those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

## 8.3 China 779-787MHz ISM Band

### 8.3.1 CN779-787 Preamble Format

The following synchronization words should be used :

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols
GFSK	0xC194C1	5 bytes

Table 24: CN779-787 synch words

### 8.3.2 CN779-787 ISM Band channel frequencies

The LoRaWAN can be used in the Chinese 779-787MHz band as long as the radio device EIRP is less than 10mW (or 10dBm).

The end-device transmit duty-cycle should be lower than 1%.

The LoRaWAN channels center frequency can be in the following range:

- Minimum frequency : 779.5Mhz
- Maximum frequency : 786.5 MHz

CN780Mhz end-devices should be capable of operating in the 779 to 787 MHz frequency band and should feature a channel data structure to store the parameters of at least 16 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

The first three channels correspond to 779.5, 779.7 and 779.9 MHz with DR0 to DR5 and must be implemented in every end-device. Those default channels cannot be modified through the **NewChannelReq** command and guarantee a minimal common channel set between end-devices and gateways of all networks. Other channels can be freely distributed across the allowed frequency range on a network per network basis.

The following table gives the list of frequencies that should be used by end-devices to broadcast the JoinReq message. The JoinReq message transmit duty-cycle shall respect the condition described in chapter 9.

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	779.5 779.7 779.9 780.5 780.7 780.9	DR0 – DR5 / 0.3-5 kbps	6	<0.1%

Table 25: CN780 JoinReq Channel List

### 8.3.3 CN779-787 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (DR) and End-point Output Power (TXPower) in the CN780 band:

DataRate	Configuration	Indicative physical bit rate [bit/s]	TXPower	Configuration
0	LoRa: SF12 / 125 kHz	250	0	10 dBm
1	LoRa: SF11 / 125 kHz	440	1	7 dBm
2	LoRa: SF10 / 125 kHz	980	2	4 dBm
3	LoRa: SF9 / 125 kHz	1760	3	1 dBm
4	LoRa: SF8 / 125 kHz	3125	4	-2 dBm
5	LoRa: SF7 / 125 kHz	5470	5	-5 dBm
6	LoRa: SF7 / 250 kHz	11000	6..15	RFU
7	FSK: 50 kbps	50000		
8..15	RFU			

Table 26: Data rate and TX power table

### 8.3.4 CN779-787 JoinAccept CFList

The CN780 ISM band LoRaWAN implements an optional **channel frequency list** (CFList) of 16 octets in the JoinAccept message.

In this case the CFList is a list of five channel frequencies for the channels four to eight whereby each frequency is encoded as a 24 bits unsigned integer (three octets). All these channels are usable for DR0 to DR5 125kHz LoRa modulation. The list of frequencies is followed by a single CFListType octet for a total of 16 octets. The CFListType shall be equal to zero (0) to indicate that the CFList contains a list of frequencies.

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	<b>CFListType</b>

The actual channel frequency in Hz is 100 x frequency whereby values representing frequencies below 100 Mhz are reserved for future use. This allows setting the frequency of a channel anywhere between 779 and 787MHz in 100 Hz steps. Unused channels have a frequency value of 0. The **CFList** is optional and its presence can be detected by the length of the join-accept message. If present, the **CFList** replaces all the previous channels stored in the end-device apart from the three default channels as defined in Chapter 6.

The newly defined channels are immediately enabled and usable by the end-device for communication.

### 8.3.5 CN779-787 LinkAdrReq command

The CN780 LoRaWAN only supports a maximum of 16 channels. When **ChMaskCntl** field is 0 the ChMask field individually enables/disables each of the 16 channels.

ChMaskCntl	ChMask applies to
0	Channels 1 to 16
1	RFU
..	..
4	RFU
5	RFU
6	All channels ON The device should enable all currently defined channels independently of the ChMask field value.
7	RFU

Table 27: ChMaskCntl value table

If the ChMask field value is one of values meaning RFU, then end-device should reject the command and unset the “**Channel mask ACK**” bit in its response.

### 8.3.6 CN779-787 Maximum payload size

The maximum **MACPayload** size length ( $M$ ) is given by the following table. It is derived from limitation of the PHY layer depending on the effective modulation rate used taking into account a possible repeater encapsulation layer. The maximum application payload length in the absence of the optional **FOpt** control field ( $N$ ) is also given for information only. The value of  $N$  might be smaller if the **FOpt** field is not empty:

DataRate	$M$	$N$
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	250	242
7	230	222
8:15	Not defined	

Table 28: CN780 maximum payload size

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	$M$	$N$
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242

8:15	Not defined
------	-------------

**Table 29 : CN780 maximum payload size (not repeater compatible)**

### 8.3.7 CN779-787 Receive windows

The RX1 receive window uses the same channel than the preceding uplink. The data rate is a function of the uplink data rate and the RX1DROffset as given by the following table. The allowed values for RX1DROffset are in the [0:5] range. Values in the range [6:7] are reserved for future use

RX1DROffset Upstream data rate	0	1	2	3	4	5
Downstream data rate in RX1 slot						
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2

The RX2 receive window uses a fixed frequency and data rate. The default parameters are 786 MHz / DR0.

### 8.3.8 CN779-787 Default Settings

The following parameters are recommended values for the CN779-787MHz band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE\_DELAY1 and RECEIVE\_DELAY2 latency), those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

## 8.4 EU 433MHz ISM Band

### 8.4.1 EU433 Preamble Format

The following synchronization words should be used :

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols
GFSK	0xC194C1	5 bytes

Table 30: EU433 synch words

### 8.4.2 EU433 ISM Band channel frequencies

The LoRaWAN can be used in the ETSI 433-434 MHz band as long as the radio device EIRP is less than 10 mW (or 10 dBm).

The end-device transmit duty-cycle should be lower than 1%<sup>1</sup>.

The LoRaWAN channels center frequency can be in the following range:

- Minimum frequency : 433.175 MHz
- Maximum frequency : 434.665 MHz

EU433 end-devices should be capable of operating in the 433.05 to 434.79 MHz frequency band and should feature a channel data structure to store the parameters of at least 16 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

The first three channels correspond to 433.175, 433.375 and 433.575 MHz with DR0 to DR5 and must be implemented in every end-device. Those default channels cannot be modified through the **NewChannelReq** command and guarantee a minimal common channel set between end-devices and gateways of all networks. Other channels can be freely distributed across the allowed frequency range on a network per network basis.

The following table gives the list of frequencies that should be used by end-devices to broadcast the JoinReq message. The JoinReq message transmit duty-cycle shall respect the condition described in chapter 9.

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	433.175 433.375 433.575	DR0 – DR5 / 0.3-5 kbps	3	<1%

Table 31: EU433 JoinReq Channel List

### 8.4.3 EU433 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (DR) and End-point Output Power (TXPower) in the EU433 band:

DataRate	Configuration	Indicative physical	TXPower	Configuration
----------	---------------	---------------------	---------	---------------

<sup>1</sup> The EN300220 ETSI standard limits to 10% the maximum transmit duty-cycle in the 433MHz ISM band. The LoRaWAN requires a 1% transmit duty-cycle lower than the legal limit to avoid network congestion.

		bit rate [bit/s]		
0	LoRa: SF12 / 125 kHz	250	0	10 dBm
1	LoRa: SF11 / 125 kHz	440	1	7 dBm
2	LoRa: SF10 / 125 kHz	980	2	4 dBm
3	LoRa: SF9 / 125 kHz	1760	3	1 dBm
4	LoRa: SF8 / 125 kHz	3125	4	-2 dBm
5	LoRa: SF7 / 125 kHz	5470	5	-5 dBm
6	LoRa: SF7 / 250 kHz	11000	6..15	RFU
7	FSK: 50 kbps	50000		
8..15	RFU			

Table 32: Data rate and TX power table

#### 8.4.4 EU433 JoinAccept CFList

The EU433 ISM band LoRaWAN implements an optional **channel frequency list** (CFList) of 16 octets in the JoinAccept message.

In this case the CFList is a list of five channel frequencies for the channels four to eight whereby each frequency is encoded as a 24 bits unsigned integer (three octets). All these channels are usable for DR0 to DR5 125 kHz LoRa modulation. The list of frequencies is followed by a single CFListType octet for a total of 16 octets. The CFListType shall be equal to zero (0) to indicate that the CFList contains a list of frequencies.

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	CFListType

The actual channel frequency in Hz is 100 x frequency whereby values representing frequencies below 100 Mhz are reserved for future use. This allows setting the frequency of a channel anywhere between 433.175 to 434.665MHz in 100 Hz steps. Unused channels have a frequency value of 0. The **CFList** is optional and its presence can be detected by the length of the join-accept message. If present, the **CFList** replaces all the previous channels stored in the end-device apart from the three default channels as defined in Chapter 6.

The newly defined channels are immediately enabled and usable by the end-device for communication.



#### 8.4.5 EU433 LinkAdrReq command

The EU433 LoRaWAN only supports a maximum of 16 channels. When **ChMaskCntl** field is 0 the ChMask field individually enables/disables each of the 16 channels.

ChMaskCntl	ChMask applies to
0	Channels 1 to 16
1	RFU
..	..
4	RFU
5	RFU
6	All channels ON The device should enable all currently defined channels independently of the ChMask field value.
7	RFU

Table 33: ChMaskCntl value table

If the ChMask field value is one of the values meaning RFU, then end-device should reject the command and unset the “**Channel mask ACK**” bit in its response.

#### 8.4.6 EU433 Maximum payload size

The maximum **MACPayload** size length (*M*) is given by the following table. It is derived from limitation of the PHY layer depending on the effective modulation rate used taking into account a possible repeater encapsulation layer. The maximum application payload length in the absence of the optional **FOpt** control field (*N*) is also given for information only. The value of *N* might be smaller if the **FOpt** field is not empty:

DataRate	<i>M</i>	<i>N</i>
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222
8:15	Not defined	

Table 34: EU433 maximum payload size

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	<i>M</i>	<i>N</i>
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242

8:15	Not defined
------	-------------

**Table 35 : EU433 maximum payload size (not repeater compatible)**

### 8.4.7 EU433 Receive windows

The RX1 receive window uses the same channel than the preceding uplink. The data rate is a function of the uplink data rate and the RX1DROffset as given by the following table. The allowed values for RX1DROffset are in the [0:5] range. Values in the range [6:7] are reserved for future use.

RX1DROffset Upstream data rate	0	1	2	3	4	5
	Downstream data rate in RX1 slot					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2

The RX2 receive window uses a fixed frequency and data rate. The default parameters are 434.665MHz / DR0 (SF12 , 125kHz)

### 8.4.8 EU433 Default Settings

The following parameters are recommended values for the EU433band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE\_DELAY1 & 2 latency) , those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

## 8.5 Australia 915-928MHz ISM Band

### 8.5.1 AU915-928 Preamble Format

The following synchronization words should be used:

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols

LoRaWAN does not make use of GFSK modulation in the AU915-928 ISM band.

### 8.5.2 AU915-928 Channel Frequencies

The AU ISM Band shall be divided into the following channel plans.

- Upstream – 64 channels numbered 0 to 63 utilizing LoRa 125 kHz BW varying from DR0 to DR3, using coding rate 4/5, starting at 915.2 MHz and incrementing linearly by 200 kHz to 927.8 MHz
- Upstream – 8 channels numbered 64 to 71 utilizing LoRa 500 kHz BW at DR6 starting at 915.9 MHz and incrementing linearly by 1.6 MHz to 927.1 MHz
- Downstream – 8 channels numbered 0 to 7 utilizing LoRa 500 kHz BW at DR8 to DR13) starting at 923.3 MHz and incrementing linearly by 600 kHz to 927.5 MHz

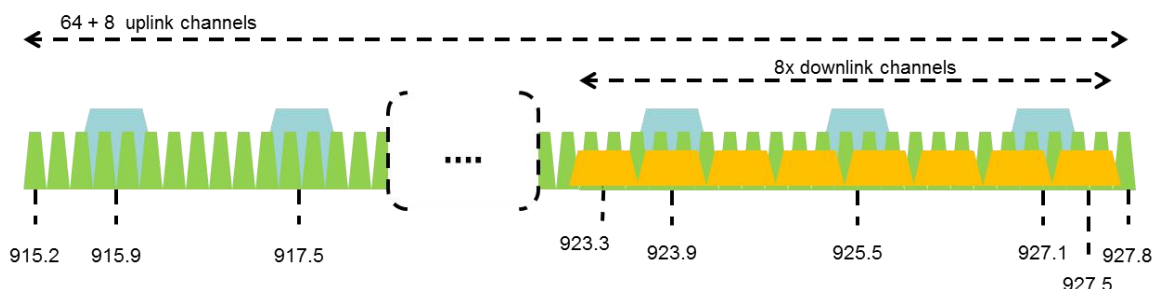


Figure 11: AU915-928 channel frequencies

AU ISM band end-devices should use the following default parameters:

- Default radiated transmit output power: 20 dBm
  - All Devices may use a maximum of +30 dBm.
  - Devices, when transmitting with 125 kHz BW must frequency hop using a minimum of 20 channels.

AU915-928 end-devices should be capable of operating in the 915 to 928 MHz frequency band and should feature a channel data structure to store the parameters of 72 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

If using the over-the-air activation procedure, the end-device should broadcast the JoinReq message alternatively on a random 125 kHz channel amongst the 64 channels defined using **DR0** and a random 500 kHz channel amongst the 4 channels defined using **DR6**. The end-device should change channel for every transmission.

Personalized devices shall have all 68 channels enabled following a reset.

### 8.5.3 AU915-928 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (**DR**) and End-point Output Power (**TXPower**) in the AU915-928 band:

DataRate	Configuration	Indicative physical bit rate [bit/sec]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF8 / 500 kHz	12500
7	RFU	
8	LoRa: SF12 / 500 kHz	980
9	LoRa: SF11 / 500 kHz	1760
10	LoRa: SF10 / 500 kHz	3900
11	LoRa: SF9 / 500 kHz	7000
12	LoRa: SF8 / 500 kHz	12500
13	LoRa: SF7 / 500 kHz	21900
14:15	RFU	

Table 36: AU915 Data rate table

TXPower	Configuration
0	30 dBm – 2*TXpower
1	28 dBm
2	26 dBm
3 : 9	...
10	10 dBm
11:15	RFU

Table 37 : AU915 TX power table

DR6 is identical to DR12, DR8...13 must be implemented in end-devices and are reserved for future applications.

#### 8.5.4 AU915-928 JoinAccept CFList

The AU915-928 LoRaWAN does not support the use of the optional **CFlist** appended to the JoinAccept message. If the **CFlist** is not empty it is ignored by the end-device.

#### 8.5.5 AU915-928 LinkAdrReq command

For the AU915-928 version the **ChMaskCntl** field of the **LinkADRReq** command has the following meaning:

ChMaskCntl	ChMask applies to
0	Channels 0 to 15
1	Channels 16 to 31
..	..

ChMaskCntl	ChMask applies to
4	Channels 64 to 71
5	RFU
6	All 125 kHz ON ChMask applies to channels 64 to 71
7	All 125 kHz OFF ChMask applies to channels 64 to 71

**Table 38: ChMaskCntl value table**

If **ChMaskCntl** = 6 (resp 7) then 125 kHz channels are enabled (resp disabled). Simultaneously the channels 64 to 67 are set according to the **ChMask** bit mask.

**Note:** ACMA regulation requires hopping over at least 20 channels when using channels that do not meet a minimum 6 dB bandwidth of 500 kHz.

### 8.5.6 AU915-928 Maximum payload size

The maximum **MACPayload** size length ( $M$ ) is given by the following table. It is derived from the maximum allowed transmission time at the PHY layer taking into account a possible repeater encapsulation. The maximum application payload length in the absence of the optional **FOpt** MAC control field ( $N$ ) is also given for information only. The value of  $N$  might be smaller if the **FOpt** field is not empty:

DataRate	$M$	$N$
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	Not defined	
8	61	53
9	137	129
10	250	242
11	250	242
12	250	242
13	250	242
14:15	Not defined	

**Table 39: AU915-928 maximum payload size**

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	$M$	$N$
0	59	51
1	59	51

2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	Not defined	
8	61	53
9	137	129
10	250	242
11	250	242
12	250	242
13	250	242
14:15	Not defined	

**Table 40: AU915-928 maximum payload size (not repeater compatible)**

### 8.5.7 AU915-928 Receive windows

- The RX1 receive channel is a function of the upstream channel used to initiate the data exchange. The RX1 receive channel can be determined as follows.
  - RX1 Channel Number = Transmit Channel Number modulo 8
- The RX1 window data rate depends on the transmit data rate (see Table 23 below).
- The RX2 (second receive window) settings uses a fixed data rate and frequency. Default parameters are 923.3Mhz / DR8

Upstream data rate RX1DROffset	Downstream data rate			
	0	1	2	3
DR0	DR10	DR9	DR8	DR8
DR1	DR11	DR10	DR9	DR8
DR2	DR12	DR11	DR10	DR9
DR3	DR13	DR12	DR11	DR10
DR4	DR13	DR13	DR12	DR11
DR5	DR13	DR13	DR13	DR12
DR6	DR13	DR13	DR13	DR13

**Table 41: AU RX1DROffset**

The allowed values for RX1DROffset are in the [0:3] range. Values in the range [4:7] are reserved for future use.

### 8.5.8 AU915-928 Default Settings

The following parameters are recommended values for the AU915-928 band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

1 If the actual parameter values implemented in the end-device are different from those default  
2 values (for example the end-device uses a longer RECEIVE\_DELAY1 & 2 latency), those  
3 parameters must be communicated to the network server using an out-of-band channel  
4 during the end-device commissioning process. The network server may not accept  
5 parameters different from those default values.  
6

## 8.6 CN 470-510MHz Band

### 8.6.1 CN470-510 Preamble Format

The following synchronization words should be used:

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols

### 8.6.2 CN470-510 Channel Frequencies

In China, this band is defined by SRRC to be used for civil metering applications.

The 470 MHz ISM Band shall be divided into the following channel plans:

- Upstream – 96 channels numbered 0 to 95 utilizing LoRa 125 kHz BW varying from DR0 to DR5, using coding rate 4/5, starting at 470.3 MHz and incrementing linearly by 200 kHz to 489.3 MHz.

Channel Index 6 to 38 and 45 to 77 are mainly used by China Electric Power. In the areas where these channels are used by China Electric Power, they should be disabled.

- Downstream – 48 channels numbered 0 to 47 utilizing LoRa 125 kHz BW varying from DR0 to DR5, using coding rate 4/5, starting at 500.3 MHz and incrementing linearly by 200 kHz to 509.7 MHz

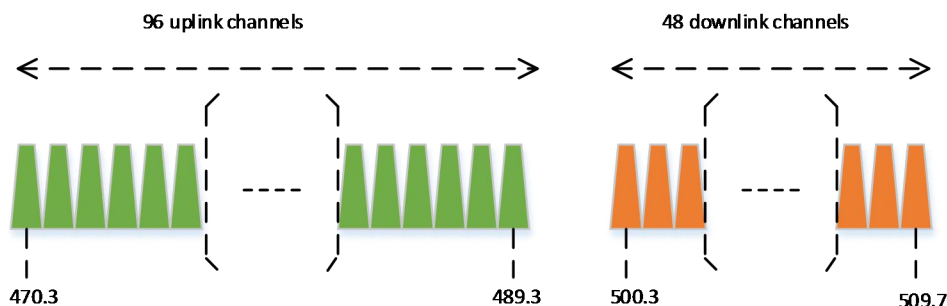


Figure 12: CN470-510 channel frequencies

The LoRaWAN can be used in the Chinese 470-510MHz band as long as

- The radio device EIRP is less than 50mW (or 17dBm).
- The transmission never lasts more than 5000 ms.

CN470 MHz band end-devices should use the following default parameters:

- Default radiated transmits output power: 14 dBm.

CN470-510 end-devices should be capable of operating in the 470 to 510 MHz frequency band and should feature a channel data structure to store the parameters of 96 uplink channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.



If using the over-the-air activation procedure, the end-device should broadcast the JoinReq message on a random 125 kHz channel amongst the 96 uplink channels defined using **DR5 to DR0**.

Personalized devices shall have all 96 channels enabled following a reset.

### 8.6.3 CN470-510 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (**DR**) and End-point Output Power (**TXPower**) in the CN470-510 band:

DataRate	Configuration	Indicative physical bit rate [bit/sec]	TXPower	Configuration
0	LoRa: SF12 / 125 kHz	250	0	17 dBm
1	LoRa: SF11 / 125 kHz	440	1	16 dBm
2	LoRa: SF10 / 125 kHz	980	2	14 dBm
3	LoRa: SF9 / 125 kHz	1760	3	12 dBm
4	LoRa: SF8 / 125 kHz	3125	4	10 dBm
5	LoRa:SF7 / 125 kHz	5470	5	7 dBm
			6	5 dBm
6:15	RFU		7	2 dBm
			8...15	RFU

Table 42: CN470 Data rate and TX power table

### 8.6.4 CN470-510 JoinResp CFList

The CN470-510 LoRaWAN does not support the use of the optional **CFlist** appended to the JoinAccept message. If the **CFlist** is not empty it is ignored by the end-device.

### 8.6.5 CN470-510 LinkAdrReq command

For the CN470-510 version the **ChMaskCntl** field of the **LinkADRReq** command has the following meaning:

ChMaskCntl	ChMask applies to
0	Channels 0 to 15
1	Channels 16 to 31
2	Channels 32 to 47
3	Channels 48 to 63
4	Channels 64 to 79
5	Channels 80 to 95
6	All channels ON The device should enable all currently defined channels independently of the ChMask field value.
7	RFU

Table 43: CN470 ChMaskCntl value table

If the ChMask field value is one of the values meaning RFU, then end-device should reject the command and unset the “**Channel mask ACK**” bit in its response.

## 8.6.6 CN470-510 Maximum payload size

The maximum **MACPayload** size length ( $M$ ) is given by the following table. It is derived from the maximum allowed transmission time at the PHY layer taking into account a possible repeater encapsulation. The maximum application payload length in the absence of the optional **FOpt** MAC control field ( $N$ ) is also given for information only. The value of  $N$  might be smaller if the **FOpt** field is not empty:

DataRate	$M$	$N$
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6:15	Not defined	

Table 44: CN470-510 maximum payload size

## 8.6.7 CN470-510 Receive windows

- The RX1 receive channel is a function of the upstream channel used to initiate the data exchange. The RX1 receive channel can be determined as follows.
  - RX1 Channel Number = Uplink Channel Number modulo 48, for example, when transmitting channel number is 49, the rx1 channel number is 1.
- The RX1 window data rate depends on the transmit data rate (see Table Table 45: CN470-510 Data rate offset below).
- The RX2 (second receive window) settings uses a fixed data rate and frequency. Default parameters are 505.3 MHz / DR0

RX1DROffset	0	1	2	3	4	5
Upstream data rate	Downstream data rate in RX1 slot					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0

Table 45: CN470-510 Data rate offset

The allowed values for RX1DROffset are in the [0:3] range. Values in the range [4:7] are reserved for future use.

## 8.6.8 CN470-510 Default Settings

The following parameters are recommended values for the CN470-510 band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s

1	MAX_FCNT_GAP	16384
2	ADR_ACK_LIMIT	64
3	ADR_ACK_DELAY	32
4	ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

5 If the actual parameter values implemented in the end-device are different from those default  
6 values (for example the end-device uses a longer RECEIVE\_DELAY1 & 2 latency), those  
7 parameters must be communicated to the network server using an out-of-band channel  
8 during the end-device commissioning process. The network server may not accept  
9 parameters different from those default values.

10  
11  
12  
13

## 9 Retransmissions back-off

Uplink frames that:

- Require an **acknowledgement or an answer** by the network or an application server, and are **retransmitted** by the device if the acknowledgement or answer is not received.

and

- can be triggered by an **external** event causing **synchronization** across a large (>100) number of devices (power outage, radio jamming, network outage, earthquake...)

can trigger a catastrophic, self-persisting, radio network overload situation.

Note: An example of such uplink frame is typically the JoinRequest if the implementation of a group of end-devices decides to reset the MAC layer in the case of a network outage.

The whole group of end-device will start broadcasting JoinRequest uplinks and will only stops when receiving a JoinResponse from the network.

For those frame retransmissions, the interval between the end of the RX2 slot and the next uplink retransmission shall be random and follow a different sequence for every device (For example using a pseudo-random generator seeded with the device's address) .The transmission duty-cycle of such message shall respect the local regulation and the following limits, whichever is more constraining:

Aggregated during the first hour following power-up or reset	$T_0 < t < T_0 + 1$	Transmit time < 36Sec
Aggregated during the next 10 hours	$T_0 + 1 < t < T_0 + 11$	Transmit time < 36Sec
After the first 11 hours , aggregated over 24h	$T_0 + 11 + N < t < T_0 + 35 + N$ $N \geq 0$	Transmit time < 8.7Sec per 24h

## **CLASS B – BEACON**

---

Class B must be considered as experimental in this version of the specification

## 10 Introduction to Class B

This section describes the LoRaWAN Class B layer which is optimized for battery-powered end-devices that may be either mobile or mounted at a fixed location.

End-devices should implement Class B operation when there is a requirement to open receive windows at fixed time intervals for the purpose of enabling server initiated downlink messages.

LoRaWAN Class B option adds a synchronized reception window on the end-device.

One of the limitations of LoRaWAN Class A is the Aloha method of sending data from the end-device; it does not allow for a known reaction time when the customer application or the server wants to address the end-device. The purpose of Class B is to have an end-device available for reception on a predictable time, in addition to the reception windows that follows the random uplink transmission from the end-device of Class A. Class B is achieved by having the gateway sending a beacon on a regular basis to synchronize the all the end-devices in the network so that the end-device can opening a short extra reception window (called “ping slot”) at a predictable time during a periodic time slot.

**Note:** The decision to switch from Class A to Class B comes from the application layer of the end-device. If this class A to Class B switch needs to be controlled from the network side, the customer application must use one of the end-device’s Class A uplinks to send back a downlink to the application layer, and it needs the application layer on the end-device to recognize this request – this process is not managed at the LoRaWAN level.

## 11 Principle of synchronous network initiated downlink (Class-B option)

For a network to support end-devices of Class B, all gateways must synchronously broadcast a beacon providing a timing reference to the end-devices. Based on this timing reference the end-devices can periodically open receive windows, hereafter called “ping slots”, which can be used by the network infrastructure to initiate a downlink communication. A network initiated downlink using one of these ping slots is called a “ping”. The gateway chosen to initiate this downlink communication is selected by the network server based on the signal quality indicators of the last uplink of the end-device. For this reason, if an end-device moves and detects a change in the identity advertised in the received beacon, it must send an uplink to the network server so that the server can update the downlink routing path database.

All end-devices start and join the network as end-devices of Class A. The end-device application can then decide to switch to Class B. This is done through the following process:

- The end-device application requests the LoRaWAN layer to switch to Class B mode. The LoRaWAN layer in the end-device searches for a beacon and returns either a BEACON\_LOCKED service primitive to the application if a network beacon was found and locked or a BEACON\_NOT\_FOUND service primitive. To accelerate the beacon discovery the LoRaWAN layer may use the “BeaconTimingReq” message described later.
- Based on the beacon strength and the battery life constraints, the end-device application selects a ping slot data rate and periodicity, this is then requested them from the end-device LoRaWAN layer.
- Once in Class B mode, the MAC layer sets to 1 the *Class B* bit of the FCTRL field of every uplink frame transmitted. This bit signals to the server that the device has switched to Class B. The MAC layer will autonomously schedule a reception slot for each beacon and each ping slot. When the beacon reception is successful the end-device LoRaWAN layer forwards the beacon content to the application together with the measured radio signal strength. The end-device LoRaWAN layer takes into account the maximum possible clock drift in the scheduling of the beacon reception slot and ping slots. When a downlink is successfully demodulated during a ping slot, it is processed similarly to a downlink as described in the LoRaWAN Class A specification.
- A mobile end-device must periodically inform the network server of its location to update the downlink route. This is done by transmitting a normal (possibly empty) “unconfirmed” or “confirmed” uplink. The end-device LoRaWAN layer will appropriately set the *Class B* bit to 1. Optimally this can be done more efficiently if the application detects that the node is moving by analyzing the beacon content. In that case the end-device must apply a random delay (as defined in Section 17.5 between the beacon reception and the uplink transmission to avoid systematic uplink collisions.
- If no beacon has been received for a given period (as defined in Section 14.2), the synchronization with the network is lost. The MAC layer must inform the application layer that it has switched back to Class A. As a consequence the end-device LoRaWAN layer stops setting the *Class B* bit in all uplinks and this informs the network server that the end-device is no longer in Class B mode. The end-device application can try to switch back to Class B periodically. This will restart this process starting with a beacon search.

1 The following diagram illustrates the concept of beacon reception slots and ping slots.

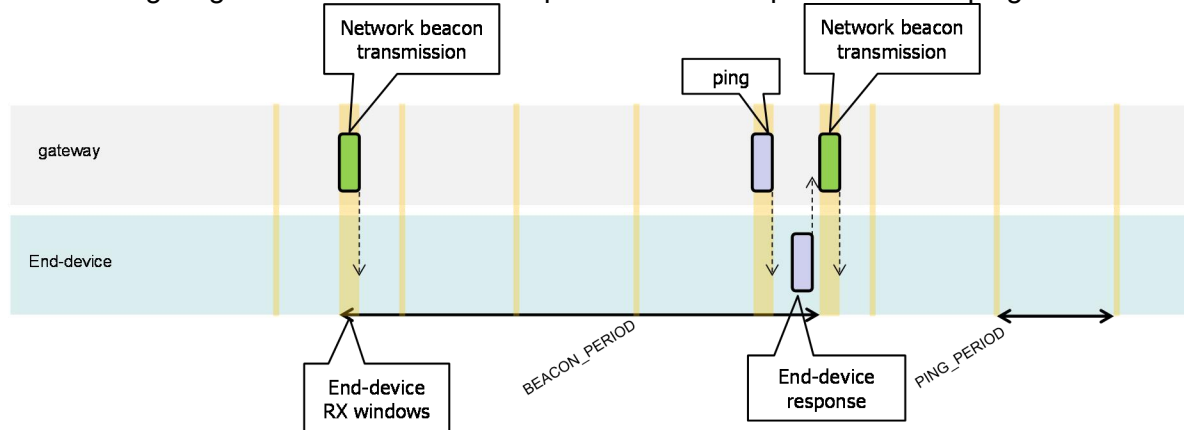


Figure 13: Beacon reception slot and ping slots

2  
3

4 In this example, given the beacon period is 128 s, the end-device also opens a ping reception  
5 slot every 32 s. Most of the time this ping slot is not used by the server and therefore the  
6 end-device reception window is closed as soon as the radio transceiver has assessed that no  
7 preamble is present on the radio channel. If a preamble is detected the radio transceiver will  
8 stay on until the downlink frame is demodulated. The MAC layer will then process the frame,  
9 check that its address field matches the end-device address and that the Message Integrity  
10 Check is valid before forwarding it to the application layer.



## 12 Uplink frame in Class B mode

The uplink frames in Class B mode are same as the Class A uplinks with the exception of the RFU bit in the FCtrl field in the Frame header. In the Class A uplink this bit is unused (RFU). This bit is used for Class B uplinks.

Bit#	7	6	5	4	3..0
FCtrl	ADR	ADRACKReq	ACK	Class B	FOptsLen

The *Class B* bit set to 1 in an uplink signals the network server that the device as switched to Class B mode and is now ready to receive scheduled downlink pings.

The signification of the FPending bit for downlink is unaltered and still signals that one or more downlink frames are queued for this device in the server and that the device should keep is receiver on as described in the Class A specification.

## 13 Downlink Ping frame format (Class B option)

### 13.1 Physical frame format

A downlink Ping uses the same format as a Class A downlink frame but might follow a different channel frequency plan.

### 13.2 Unicast & Multicast MAC messages

Messages can be “unicast” or “multicast”. Unicast messages are sent to a single end-device and multicast messages are sent to multiple end-devices. All devices of a multicast group must share the same multicast address and associated encryption keys. The LoRaWAN Class B specification does not specify means to remotely setup such a multicast group or securely distribute the required multicast key material. This must either be performed during the node personalization or through the application layer.

#### 13.2.1 Unicast MAC message format

The MAC payload of a unicast downlink **Ping** uses the format defined in the Class A specification. It is processed by the end-device in exactly the same way. The same frame counter is used and incremented whether the downlink uses a Class B ping slot or a Class A “piggy-back” slot.

#### 13.2.2 Multicast MAC message format

The Multicast frames share most of the unicast frame format with a few exceptions:

- They are not allowed to carry MAC commands, neither in the **FOpt** field, nor in the payload on port 0 because a multicast downlink does not have the same authentication robustness as a unicast frame.
- The **ACK** and **ADRACKReq** bits must be zero. The **MType** field must carry the value for Unconfirmed Data Down.
- The **FPending** bit indicates there is more multicast data to be sent. If it is set the next multicast receive slot will carry a data frame. If it is not set the next slot may or may not carry data. This bit can be used by end-devices to evaluate priorities for conflicting reception slots.

## 14 Beacon acquisition and tracking

Before switching from Class A to Class B, the end-device must first receive one of the network beacons to align his internal timing reference with the network.

Once in Class B, the end-device must periodically search and receive a network beacon to cancel any drift of its internal clock time base, relative to the network timing.

A Class B device may be temporarily unable to receive beacons (out of range from the network gateways, presence of interference, ..). In this event, the end-device has to gradually widen its beacon and ping slots reception windows to take into account a possible drift of its internal clock.

**Note:** For example, a device which internal clock is defined with a  $\pm 10\text{ppm}$  precision may drift by  $\pm 1.3\text{mSec}$  every beacon period.

### 14.1 Minimal beacon-less operation time

In the event of beacon loss, a device shall be capable of maintaining Class B operation for 2 hours (120 minutes) after it received the last beacon. This temporary Class B operation without beacon is called “beacon-less” operation. It relies on the end-device’s own clock to keep timing.

During beacon-less operation, unicast, multicast and beacon reception slots must all be progressively expanded to accommodate the end-device’s possible clock drift.

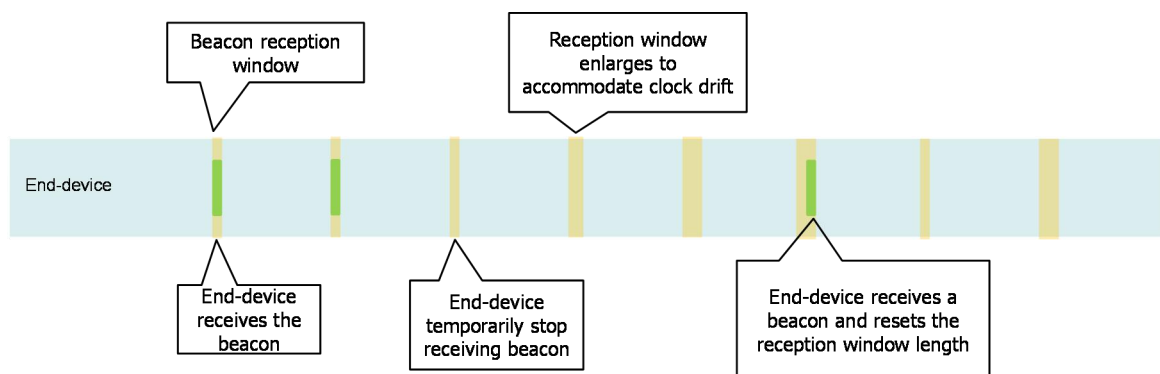


Figure 14 : beacon-less temporary operation

### 14.2 Extension of beacon-less operation upon reception

During this 120 minutes time interval the reception of any beacon directed to the end-device, should extend the Class B beacon-less operation further by another 120 minutes as it allows to correct any timing drift and reset the receive slots duration.

### 14.3 Minimizing timing drift

The end-devices may use the beacon’s (when available) precise periodicity to calibrate their internal clock and therefore reduce the initial clock frequency imprecision. As the timing oscillator’s exhibit a predictable temperature frequency shift, the use of a temperature sensor could enable further minimization of the timing drift.

## 15 Class B Downlink slot timing

### 15.1 Definitions

To operate successfully in Class B the end-device must open reception slots at precise instants relative to the infrastructure beacon. This section defines the required timing.

The interval between the start of two successive beacons is called the beacon period. The beacon frame transmission is aligned with the beginning of the BEACON\_RESERVED interval. Each beacon is preceded by a guard time interval where no ping slot can be placed. The length of the guard interval corresponds to the time on air of the longest allowed frame. This is to insure that a downlink initiated during a ping slot just before the guard time will always have time to complete without colliding with the beacon transmission. The usable time interval for ping slot therefore spans from the end of the beacon reserved time interval to the beginning of the next beacon guard interval.

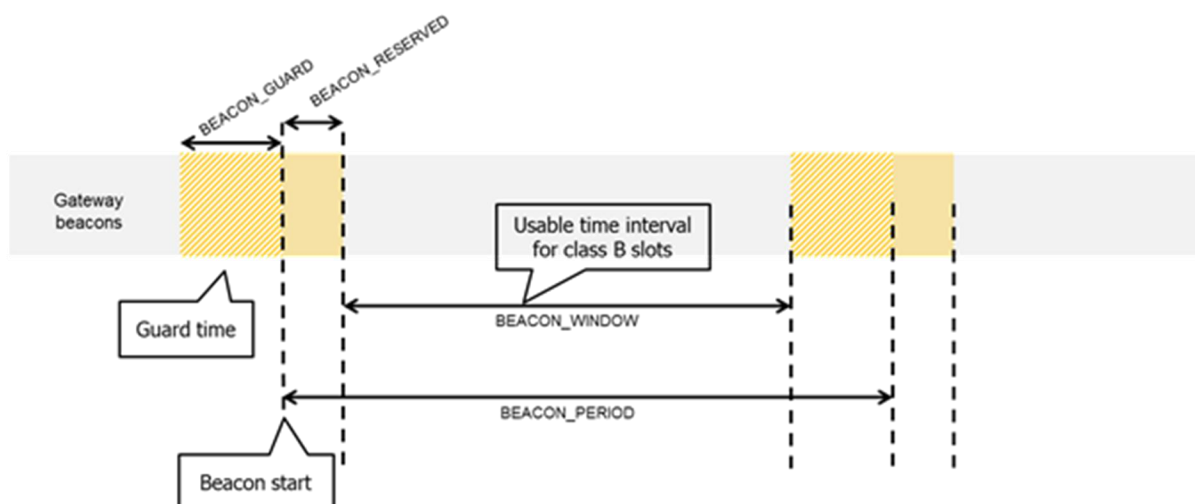


Figure 15: Beacon timing

Beacon_period	128 s
Beacon_reserved	2.120 s
Beacon_guard	3.000 s
Beacon-window	122.880 s

Table 46: Beacon timing

The beacon frame time on air is actually much shorter than the beacon reserved time interval to allow appending network management broadcast frames in the future.

The beacon window interval is divided into  $2^{12} = 4096$  ping slots of 30 ms each numbered from 0 to 4095.

An end-device using the slot number  $N$  must turn on its receiver exactly  $T_{on}$  seconds after the start of the beacon where:

$$T_{on} = \text{beacon\_reserved} + N * 30 \text{ ms}$$

$N$  is called the *slot index*.

The latest ping slot starts at  $\text{beacon\_reserved} + 4095 * 30 \text{ ms} = 124\,970 \text{ ms}$  after the beacon start or 3030 ms before the beginning of the next beacon.

## 15.2 Slot randomization

To avoid systematic collisions or over-hearing problems the slot index is randomized and changed at every beacon period.

The following parameters are used:

<b>DevAddr</b>	Device 32 bit network unicast or multicast address
<i>pingNb</i>	Number of ping slots per beacon period. This must be a power of 2 integer: $pingNb = 2^k$ where $1 \leq k \leq 7$
<i>pingPeriod</i>	Period of the device receiver wake-up expressed in number of slots: $pingPeriod = 2^{12} / pingNb$
<i>pingOffset</i>	Randomized offset computed at each beacon period start. Values can range from 0 to ( $pingPeriod - 1$ )
<i>beaconTime</i>	The time carried in the field <b>BCNPPayload</b> . Time of the immediately preceding beacon frame
<i>slotLen</i>	Length of a unit ping slot = 30 ms

At each beacon period the end-device and the server compute a new pseudo-random offset to align the reception slots. An AES encryption with a fixed key of all zeros is used to randomize:

$Key = 16 \times 0x00$

$Rand = aes128\_encrypt(Key, beaconTime \parallel DevAddr \parallel pad16)$

$pingOffset = (Rand[0] + Rand[1] \times 256) \text{ modulo } pingPeriod$

The slots used for this beacon period will be:

$pingOffset + N \times pingPeriod$  with  $N=[0:pingNb-1]$

The node therefore opens receive slots starting at :

First slot	$Beacon\_reserved + pingOffset \times slotLen$
Slot 2	$Beacon\_reserved + (pingOffset + pingPeriod) \times slotLen$
Slot 3	$Beacon\_reserved + (pingOffset + 2 \times pingPeriod) \times slotLen$
...	...

If the end-device serves simultaneously a unicast and one or more multicast slots this computation is performed multiple times at the beginning of a new beacon period. Once for the unicast address (the node network address) and once for each multicast group address.

In the case where a multicast ping slot and a unicast ping slot collide and cannot be served by the end-device receiver then the end-device should preferentially listen to the multicast slot. If there is a collision between multicast reception slots the FPending bit of the previous multicast frame can be used to set a preference.

The randomization scheme prevents a systematic collision between unicast and multicast slots. If collisions happen during a beacon period then it is unlikely to occur again during the next beacon period.

## 16 Class B MAC commands

All commands described in the Class A specification shall be implemented in Class B devices. The Class B specification adds the following MAC commands.

CID	Command	Transmitted by		Short Description
		End-d evice	Gateway	
0x10	<b><i>PingSlotInfoReq</i></b>	x		Used by the end-device to communicate the ping unicast slot data rate and periodicity to the network server
0x10	<b><i>PingSlotInfoAns</i></b>		x	Used by the network to acknowledge a PingInfoSlotReq command
0x11	<b><i>PingSlotChannelReq</i></b>		x	Used by the network server to set the unicast ping channel of an end-device
0x11	<b><i>PingSlotFreqAns</i></b>	x		Used by the end-device to acknowledge a <b><i>PingSlotChannelReq</i></b> command
0x12	<b><i>BeaconTimingReq</i></b>	x		Used by end-device to request next beacon timing & channel to network
0x12	<b><i>BeaconTimingAns</i></b>		x	Used by network to answer a <b><i>BeaconTimingReq</i></b> uplink
0x13	<b><i>BeaconFreqReq</i></b>		x	Command used by the network server to modify the frequency at which the end-device expects to receive beacon broadcast
0x13	<b><i>BeaconFreqAns</i></b>	x		Used by the end-device to acknowledge a BeaconFreqReq command

### 16.1 PingSlotInfoReq

With the ***PingSlotInfoReq*** command an end-device informs the server of its unicast ping slot periodicity and expected data rate. This command must only be used to inform the server of the parameters of a UNICAST ping slot. A multicast slot is entirely defined by the application and should not use this command.

<b>Size (bytes)</b>	1
<b>PingSlotInfoReq Payload</b>	Periodicity & data rate

<b>Bit#</b>	7	[6:4]	[3:0]
<b>Periodicity &amp; data rate</b>	RFU	Periodicity	Data rate

The **Periodicity** subfield is an unsigned 3 bits integer encoding the ping slot period currently used by the end-device using the following equation.  
in seconds

- **Periodicity** = 0 means that the end-device opens a ping slot every second
- **Periodicity** = 7 , every 128 seconds which is the maximum ping period supported by the LoRaWAN Class B specification.

The **Data rate** subfield encodes the data rate at which the end-point expects any ping. This uses the same encoding scheme that the ***LinkAdrReq*** command described in the Class A specification.

The server needs to be aware of the end-device ping slot periodicity or expected data rate else Class B downlinks will not happen successfully. For that purpose the ***PingSlotInfoReq*** MAC command **must be acknowledged** with a ***PingSlotInfoAns*** before the device can switch from class A to Class B. To change its ping slot scheduling or data rate a device should first revert to Class A , send the new parameters through a ***PingSlotInfoReq*** command and get an acknowledge from the server through a ***PingSlotInfoAns*** . It can then switch back to Class B with the new parameters.

This command can be concatenated with any other MAC command in the **FHDRFOpt** field as described in the Class A specification frame format.

## 16.2 BeaconFreqReq

This command is sent by the server to the end-device to modify the frequency on which this end-device expects the beacon.

Octets	3
<b>PingSlotChannelReqPayload</b>	Frequency

The Frequency coding is identical to the ***NewChannelReq*** MAC command defined in the Class A.

**Frequency** is a 24bits unsigned integer. The actual beacon channel frequency in Hz is 100 x frequ. This allows defining the beacon channel anywhere between 100 MHz to 1.67 GHz by 100 Hz step. The end-device has to check that the frequency is actually allowed by its radio hardware and return an error otherwise.

A valid non-zero Frequency will force the device to listen to the beacon on a fixed frequency channel even if the default behavior specifies a frequency hopping beacon (i.e US ISM band).

A value of 0 instructs the end-device to use the default beacon frequency plan as defined in the “Beacon physical layer” section. Where applicable the device resumes frequency hopping beacon search.

Upon reception of this command the end-device answers with a ***BeaconFreqAns*** message. The MAC payload of this message contains the following information:

Size (bytes)	1
<b>BeaconFreqAns Payload</b>	Status

The **Status** bits have the following meaning:

Bits	7:1	0
Status	RFU	Beacon frequency ok

	Bit = 0	Bit = 1
<b>Beacon frequency ok</b>	<u>The device cannot use this frequency, the previous beacon frequency is kept</u>	<u>The beacon frequency has been changed</u>

### 16.3 PingSlotChannelReq

This command is sent by the server to the end-device to modify the frequency on which this end-device expects the downlink pings.

Octets	3	1
PingSlotChannelReq Payload	Frequency	DrRange

The Frequency coding is identical to the **NewChannelReq** MAC command defined in the Class A.

**Frequency** is a 24bits unsigned integer. The actual ping channel frequency in Hz is 100 x frequ. This allows defining the ping channel anywhere between 100MHz to 1.67GHz by 100Hz step. The end-device has to check that the frequency is actually allowed by its radio hardware and return an error otherwise.

A value of 0 instructs the end-device to use the default frequency plan.

**DrRange** is the data rate range allowed on this channel. This byte is split in two 4-bit indexes.

Bits	7:4	3:0
DrRange	Max data rate	Min data rate

Following the convention defined in the “Physical layer” section of the Class A specification, the “Min data rate” subfield designates the lowest data rate allowed on this channel. For example 0 designates DR0 / 125 kHz in the EU physical layer. Similarly “Max data rate” designates the highest data rate. For example in the EU spec, DrRange = 0x77 means that only 50 kbps GFSK is allowed on a channel and DrRange = 0x50 means that DR0 / 125 kHz to DR5 / 125 kHz are supported.

Upon reception of this command the end-device answers with a **PingSlotFreqAns** message. The MAC payload of this message contains the following information:

Size (bytes)	1
pingSlotFreqAns Payload	Status

The **Status** bits have the following meaning:

Bits	7:2	1	0
Status	RFU	Data rate range ok	Channel frequency ok

	Bit = 0	Bit = 1
<b>Data rate range ok</b>	The designated data rate range exceeds the ones currently defined for this end device, the previous range is kept	The data rate range is compatible with the possibilities of the end device
<b>Channel frequency ok</b>	The device cannot use this frequency, the previous ping frequency is kept	The device is able to use this frequency.



## 16.4 BeaconTimingReq

This command is sent by the end-device to request the next beacon timing and channel. This MAC command has no payload. The **BeaconTimingReq** & **BeaconTimingAns** mechanism is only meant to accelerate the initial beacon search to lower the end-device energy requirement.

The network may answer only a limited number of requests per a given time period. An end-device must not expect that **BeaconTimingReq** is answered immediately with a **BeaconTimingAns**. Class A end-devices wanting to switch to Class B should not transmit more than one **BeaconTimingReq** per hour.

End-devices requiring a fast beacon lock must implement an autonomous beacon finding algorithm.

## 16.5 BeaconTimingAns

This command is sent by the network to answer a **BeaconInfoReq** request.

Size (bytes)	2	1
<b>BeaconInfoReqPayload</b>	Delay	Channel

The “**Delay**” field is a 16bits unsigned integer. If the remaining time between the end of the current downlink frame and the start of the next beacon frame is noted *RTime* then:

$$30 \text{ ms} \times (\mathbf{Delay} + 1) > RTime \geq 30 \text{ ms} \times \mathbf{Delay}$$

In networks where the beacon uses alternatively several channels, the “**Channel**” field is the index of the beaoning channel on which the next beacon will be broadcasted. For networks where the beacon broadcast frequency is fixed then this field content is 0.

## 17 Beaconing (Class B option)

### 17.1 Beacon physical layer

Besides relaying messages between end-devices and network servers, gateways may participate in providing a time-synchronization mechanisms by sending beacons at regular fixed intervals. All beacons are transmitted in radio packet implicit mode, that is, without a LoRa physical header and with no CRC being appended by the radio.

<b>PHY</b>	Preamble	BCNPayload
------------	----------	------------

The beacon Preamble shall begin with (a longer than default) 10 unmodulated symbols. This allows end-devices to implement a low power duty-cycled beacon search.

The beacon frame length is tightly coupled to the operation of the radio Physical layer. Therefore the actual frame length might change from one region implementation to another..

#### 17.1.1 EU 863-870MHz ISM Band

The beacons are transmitted using the following settings

DR	3	Corresponds to SF9 spreading factor with 125 kHz BW
CR	1	Coding rate = 4/5
Signal polarity	Non-inverted	As opposed to normal downlink traffic which uses inverted signal polarity
frequency	869.525 MHz	This frequency allows up to +27 dBm EIRP.

The beacon frame content is:

<b>Size (bytes)</b>	2	4	2	7	2
<b>BCNPayload</b>	RFU	Time	CRC	GwSpecific	CRC

#### 17.1.2 US 902-928MHz ISM Band

The beacons are transmitted using the following settings:

DR	10	Corresponds to SF10 spreading factor with 500kHz bw
CR	1	Coding rate = 4/5
Signal polarity	Non-inverted	As opposed to normal downlink traffic which uses inverted signal polarity
frequencies	923.3 to 927.5MHz with 600kHz steps	Beaconing is performed on the same channel that normal downstream traffic as defined in the Class A specification

The downstream channel used for a given beacon is:

Channel =

- whereby beacon\_time is the integer value of the 4 bytes "Time" field of the beacon frame
- whereby beacon\_period is the periodicity of beacons , 128 seconds
- whereby  $\text{floor}(x)$  designates rounding to the integer immediately inferior or equal to  $x$

Example: the first beacon will be transmitted on 923.3Mhz , the second on 923.9MHz, the 9<sup>th</sup> beacon will be on 923.3Mhz again.

Beacon channel nb	Frequency [MHz]
0	923.3
1	923.9
2	924.5
3	925.1
4	925.7
5	926.3
6	926.9
7	927.5

The beacon frame content is:

Size (bytes)	3	4	2	7	1	2
<b>BCNPayload</b>	RFU	Time	<b>CRC</b>	GwSpecific	<b>RFU</b>	<b>CRC</b>

## 17.2 Beacon frame content

The beacon payload **BCNPayload** consists of a network common part and a gateway-specific part.

Size (bytes)	2/3	4	1/2	7	0/1	2
<b>BCNPayload</b>	RFU	Time	CRC	GwSpecific	RFU	CRC

The common part contains an RFU field equal to 0, a timestamp **Time** in seconds since 00:00:00, Sunday 5<sup>th</sup> of January 1980 (start of the GPS time) modulo  $2^{32}$ . The integrity of the beacon's network common part is protected by a 16 bits CRC. The CRC-16 is computed on the RFU+Time fields as defined in the IEEE 802.15.4-2003 section 7.2.1.8.

For example: This is a valid EU868 beacon frame:

00 00 | 00 00 02 CC | A2 7E | 00 | 01 20 00 | 00 81 03 | DE 55

Bytes are transmitted left to right. The corresponding field values are:

Field	RFU	Time	<b>CRC</b>	InfoDesc	lat	long	<b>CRC</b>
<b>Value Hex</b>	0000	CC020000	<b>7EA2</b>	0	002001	038100	<b>55DE</b>

The gateway specific part provides additional information regarding the gateway sending a beacon and therefore may differ for each gateway. The RFU field when applicable (region specific) should be equal to 0. The optional part is protected by a CRC-16 computed on the GwSpecific+RFU fields. The CRC-16 definition is the same as for the mandatory part.

For example: This is a valid US900 beacon:

Field	RFU	Time	<b>CRC</b>	InfoDesc	lat	long	<b>RFU</b>	<b>CRC</b>
<b>Value Hex</b>	000000	CC020000	<b>7E A2</b>	0	002001	038100	<b>00</b>	<b>D450</b>

Over the air the bytes are sent in the following order:

00 00 00 | 00 00 02 CC | 7E C8 | 00 | 01 20 00 | 00 81 03 | 00 | 50 D4

Listening and synchronizing to the network common part is sufficient to operate a stationary end-device in Class B mode. A mobile end-device should also demodulate the gateway specific part of the beacon to be able to signal to the network server whenever he is moving from one cell to another.

**Note:** As mentioned before, all gateways participating in the beaoning process send their beacon simultaneously so that for network common part there are no visible on-air collisions for a listening end-device even if the end-device simultaneously receives beacons from several gateways. Not all gateways are required to participate in the beaoning process. The participation of a gateway to a given beacon may be randomized. With respect to the gateway specific part, collision occurs but an end-device within the proximity of more than one gateway will still be able to decode the strongest beacon with high probability.

### 17.3 Beacon GwSpecific field format

The content of the **GwSpecific** field is as follow:

Size (bytes)	1	6
GwSpecific	InfoDesc	Info

The information descriptor **InfoDesc** describes how the information field **Info** shall be interpreted.

InfoDesc	Meaning
0	GPS coordinate of the gateway first antenna
1	GPS coordinate of the gateway second antenna
2	GPS coordinate of the gateway third antenna
3:127	RFU
128:255	Reserved for custom network specific broadcasts

For a single omnidirectional antenna gateway the **InfoDesc** value is 0 when broadcasting GPS coordinates. For a site featuring 3 sectorized antennas for example, the first antenna broadcasts the beacon with **InfoDesc** equals 0, the second antenna with **InfoDesc** field equals 1, etc ...

#### 17.3.1 Gateway GPS coordinate: InfoDesc = 0, 1 or 2

For **InfoDesc** = 0, 1 or 2, the content of the **Info** field encodes the GPS coordinates of the antenna broadcasting the beacon

Size (bytes)	3	3
Info	Lat	Lng

The latitude and longitude fields (**Lat** and **Lng**, respectively) encode the geographical location of the gateway as follows:

- The north-south latitude is encoded using a signed 24 bit word where  $-2^{23}$  corresponds to 90° south (the South Pole) and  $2^{23}$  corresponds to 90° north (the North Pole). The equator corresponds to 0.
- The east-west longitude is encoded using a signed 24 bit word where  $-2^{23}$  corresponds to 180° west and  $2^{23}$  corresponds to 180° east. The Greenwich meridian corresponds to 0.

## 17.4 Beacons precise timing

The beacon is sent every 128 seconds starting at 00:00:00, Sunday 5<sup>th</sup> – **Monday 6<sup>th</sup>** of January 1980 (start of the GPS time) plus TBeaconDelay. Therefore the beacon is sent at

$$B_T = k * 128 + T_{\text{BeaconDelay}}$$

seconds after ~~00:00:00 Coordinated Universal Time (UTC), 1 January 1970~~ the GPS epoch.

whereby  $k$  is the smallest integer for which

$$k * 128 > T$$

whereby

$T$  = seconds since 00:00:00, Sunday 5<sup>th</sup> of January 1980 (start of the GPS time).

**Note:**  $T$  is GPS time and unlike Unix time,  $T$  is strictly monotonically increasing and is not influenced by leap seconds.

Whereby TBeaconDelay is 1.5 mSec +/- 1uSec delay. .

TBeaconDelay is meant to allow a slight transmission delay of the gateways required by the radio system to switch from receive to transmit mode..

All end-devices ping slots use the beacon transmission start time as a timing reference, therefore the network server as to take TBeaconDelay into account when scheduling the class B downlinks.

## 17.5 Network downlink route update requirements

When the network attempts to communicate with an end-device using a Class B downlink slot, it transmits the downlink from the gateway which was closest to the end-device when the last uplink was received. Therefore the network server needs to keep track of the rough position of every Class B device.

Whenever a Class B device moves and changes cell, it needs to communicate with the network server in order to update its downlink route. This update can be performed simply by sending a “confirmed” or “unconfirmed” uplink, possibly without applicative payload.

The end-device has the choice between 2 basic strategies:

- Systematic periodic uplink: simplest method that doesn’t require demodulation of the “gateway specific” field of the beacon. Only applicable to slowly moving or stationery end-devices. There are no requirements on those periodic uplinks.
- Uplink on cell change: The end-device demodulates the “gateway specific” field of the beacon, detects that the ID of the gateway broadcasting the beacon it demodulates has changed, and sends an uplink. In that case the device should respect a pseudo random delay in the [0:120] seconds range between the beacon demodulation and the uplink transmission. This is required to insure that the uplinks of multiple Class B devices entering or leaving a cell during the same beacon period will not systematically occur at the same time immediately after the beacon broadcast.

- 1 Failure to report cell change will result in Class B downlink being temporary not operational.
- 2 The network server may have to wait for the next end-device uplink to transmit downlink
- 3 traffic.
- 4
- 5

## 18 Class B unicast & multicast downlink channel frequencies

### 18.1 EU 863-870MHz ISM Band

All unicast&multicastClass B downlinks use a single frequency channel defined by the “***PingSlotChannelReq***” MAC command. The default frequency is 869.525MHz

### 18.2 US 902-928MHz ISM Band

By default Class B downlinks use a channel function of the Time field of the last beacon (see Beacon Frame content) and the DevAddr.

- Whereby Beacon\_Time is the 32 bit Time field of the current beacon period
- Beacon\_period is the length of the beacon period (defined as 128sec in the specification)
- Floor designates rounding to the immediately lower integer value
- DevAddr is the 32 bits network address of the device

Class B downlinks therefore hop across 8 channels in the ISM band and all Class B end-devices are equally spread amongst the 8 downlink channels.

If the “***PingSlotChannelReq***” command with a valid non-zero argument is used to set the Class B downlink frequency then all subsequent ping slots should be opened on this single frequency independently of the last beacon frequency.

If the “***PingSlotChannelReq***” command with a zero argument is sent, the end-device should resume the default frequency plan, id Class B ping slots hoping across 8 channels.

The underlying idea is to allow network operators to configure end-devices to use a single proprietary dedicated frequency band for the Class B downlinks if available, and to keep as much frequency diversity as possible when the ISM band is used.



1

## **CLASS C – CONTINUOUSLY LISTENING**

---

## 19 Class C: Continuously listening end-device

The end-devices implementing the Class C option are used for applications that have sufficient power available and thus do not need to minimize reception time.

Class C end-devices cannot implement Class B option.

The Class C end-device will listen with RX2 windows parameters as often as possible. The end-device listens on RX2 when it is not either (a) sending or (b) receiving on RX1, according to Class A definition. To do so, it will open a short window on RX2 parameters between the end of the uplink transmission and the beginning of the RX1 reception window and it will switch to RX2 reception parameters as soon as the RX1 reception window is closed; the RX2 reception window will remain open until the end-device has to send another message.

**Note:** There is not specific message for a node to tell the server that it is a Class C node. It is up to the application on server side to know that it manages Class C nodes based on the contract passed during the join procedure.

### 19.1 Second receive window duration for Class C

Class C devices implement the same two receive windows as Class A devices, but they do not close RX2 window until they need to send again. Therefore they may receive a downlink in the RX2 window at nearly any time. A short listening window on RX2 frequency and data rate is also opened between the end of the transmission and the beginning of the RX1 receive window.

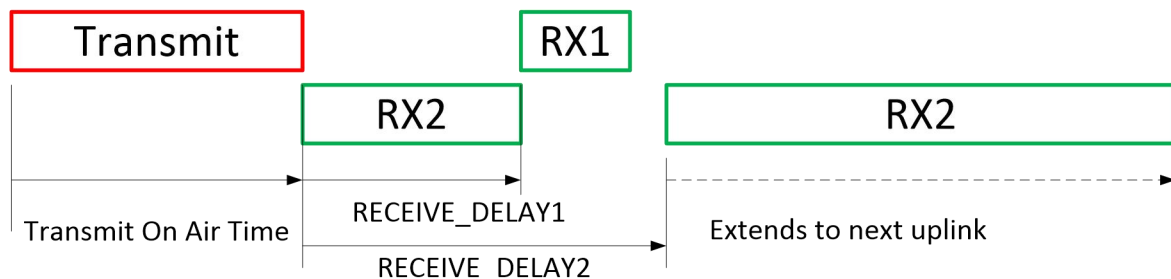


Figure 16: Class C end-device receive slot timing.

### 19.2 Class C Multicast downlinks

Similarly to Class B, Class C devices may receive multicast downlink frames. The multicast address and associated network session key and application session key must come from the application layer. The same limitations apply for Class C multicast downlink frames:

- They are not allowed to carry MAC commands, neither in the **FOpt** field, nor in the payload on port 0 because a multicast downlink does not have the same authentication robustness as a unicast frame.
- The **ACK** and **ADRACKReq** bits must be zero. The **MType** field must carry the value for Unconfirmed Data Down.
- The **FPending** bit indicates there is more multicast data to be sent. Given that a Class C device keeps its receiver active most of the time, the **FPending** bit does not trigger any specific behavior of the end-device.

## 20 Class C MAC commands

All commands described in the Class A specification shall be implemented in Class C devices. The Class C specification adds the following MAC commands.

CID	Command	Transmitted by		Short Description
		End-d evice	Gateway	
0x20	<b><i>DeviceModelInd</i></b>	x		Used by the end-device to indicate its current operating mode (Class A or C)
0x20	<b><i>DeviceModeConf</i></b>		x	Used by the network to acknowledge a <b><i>DeviceModelInd</i></b> command

### 20.1 Device Mode (***DeviceModelInd***, ***DeviceModeConf***)

With the ***DeviceModelInd*** command, an end-device indicates to the network that it wants to operate either in class A or C. The command has a one byte payload defined as follows:

<b>Size (bytes)</b>	1
<b>DeviceModelInd Payload</b>	Class

With the classes defined for the above commands as:

Class	Value
Class A	0x00
RFU	0x01
Class C	0x02

When a ***DeviceModelInd*** command is received by the network server, it responds with a ***DeviceModeConf*** command. The device shall include the ***DeviceModelInd*** command in all uplinks until the ***DeviceModeConf*** command is received.

The device shall switch mode as soon as the ***DeviceModelInd*** command is transmitted.

**Note:** When transitioning from class A to class C, it is recommended for battery powered devices to implement a time-out mechanism in the application layer to guarantee that it does not stay indefinitely in class C mode if no connection is possible with the network.

The ***DeviceModeConf*** command has a 1 byte payload.

<b>Size (bytes)</b>	1
<b>DeviceModeConf Payload</b>	Class

With the classes defined as for the ***DeviceModelInd*** command

## SUPPORT INFORMATION

---

1  
2  
3

This sub-section is only a recommendation.

## 21 Examples and Application Information

Examples are illustrations of the LoRaWAN spec for information, but they are not part of the formal specification.

### 21.1 Uplink Timing Diagram for Confirmed Data Messages

The following diagram illustrates the steps followed by an end-device trying to transmit two confirmed data frames (Data0 and Data1):

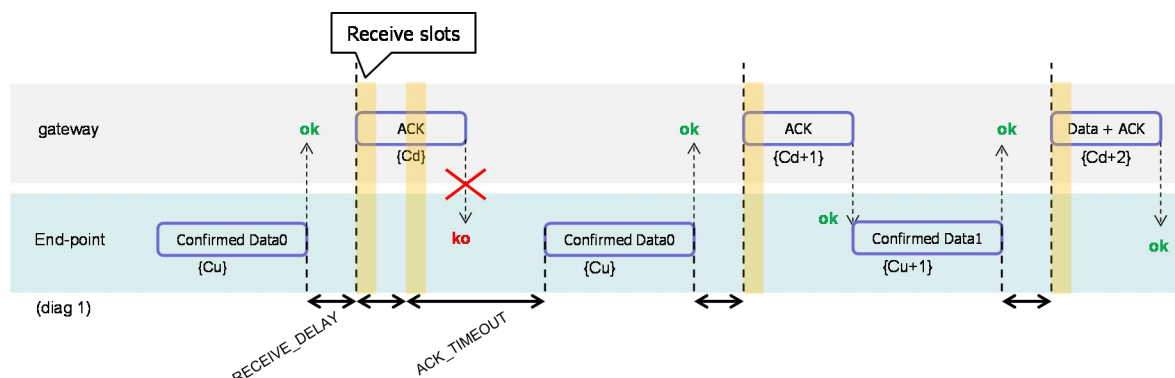


Figure 17: Uplink timing diagram for confirmed data messages

The end-device first transmits a confirmed data frame containing the Data0 payload at an arbitrary instant and on an arbitrary channel. The frame counter Cu is simply derived by adding 1 to the previous uplink frame counter. The network receives the frame and generates a downlink frame with the ACK bit set exactly RECEIVE\_DELAY1 seconds later, using the first receive window of the end-device. This downlink frame uses the same data rate and the same channel as the Data0 uplink. The downlink frame counter Cd is also derived by adding 1 to the last downlink towards that specific end-device. If there is no downlink payload pending the network shall generate a frame without a payload. In this example the frame carrying the ACK bit is not received.

If an end-point does not receive a frame with the ACK bit set in one of the two receive windows immediately following the uplink transmission it may resend the same frame with the same payload and frame counter again at least ACK\_TIMEOUT seconds after the second reception window. This resend must be done on another channel and must obey the duty cycle limitation as any other normal transmission. If this time the network receives the ACK downlink during its first receive window, as soon as the ACK frame is demodulated, the end-device is free to transmit a new frame on a new channel.

### 21.2 The third ACK frame in this example also carries an application payload. A downlink frame can carry any combination of ACK, MAC control commands and payload. Downlink Diagram for Confirmed Data Messages

The following diagram illustrates the basic sequence of a “confirmed” downlink.

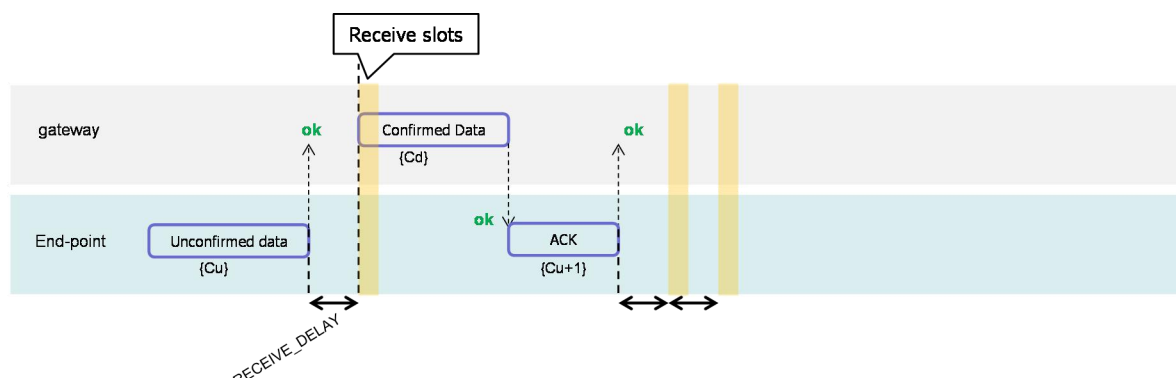


Figure 18: Downlink timing diagram for confirmed data messages

The frame exchange is initiated by the end-device transmitting an “unconfirmed” application payload or any other frame on channel A. The network uses the downlink receive window to transmit a “confirmed” data frame towards the end-device on the same channel A. Upon reception of this data frame requiring an acknowledgement, the end-device transmits a frame with the ACK bit set at its own discretion. This frame might also contain piggybacked data or MAC commands as its payload. This ACK uplink is treated like any standard uplink, and as such is transmitted on a random channel that might be different from channel A.

**Note:** To allow the end-devices to be as simple as possible and have keep as few states as possible it may transmit an explicit (possibly empty) acknowledgement data message immediately after the reception of a data message requiring an acknowledgment. Alternatively the end-device may defer the transmission of an acknowledgement to piggyback it with its next data message.

## 21.3 Downlink Timing for Frame-Pending Messages

The next diagram illustrates the use of the **frame pending** (FPending) bit on a downlink. The FPending bit can only be set on a downlink frame and informs the end-device that the network has several frames pending for him; the bit is ignored for all uplink frames.

If a frame with the FPending bit set requires an acknowledgement, the end-device shall do so 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.

**Note:** The FPending bit is independent to the acknowledgment scheme.

(\*) F\_P means 'frame pending' bit set

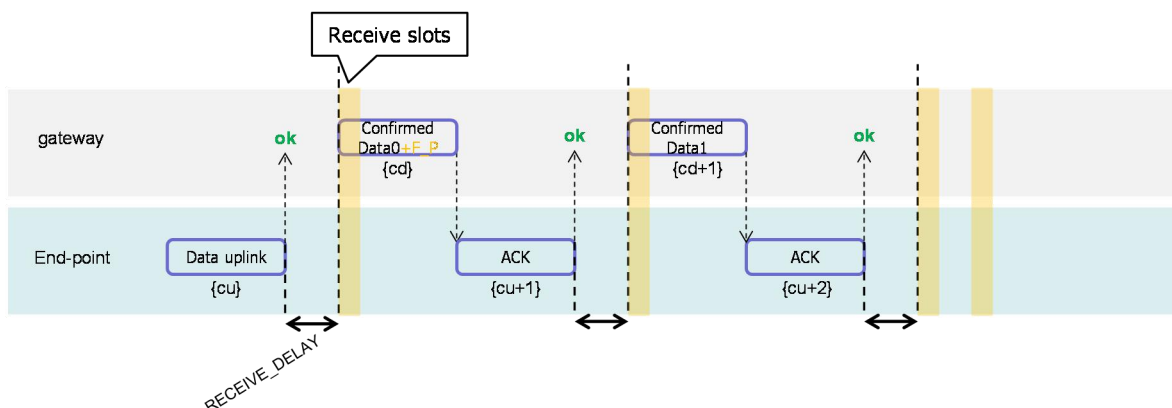


Figure 19: Downlink timing diagram for frame-pending messages, example 1

In this example the network has two confirmed 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 the Data0 with the 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. RECEIVE\_DELAY1 seconds later, the network transmits the second frame Data1 on channel B, again using a confirmed data message but with the FPending bit cleared. The end-device acknowledges on channel C.

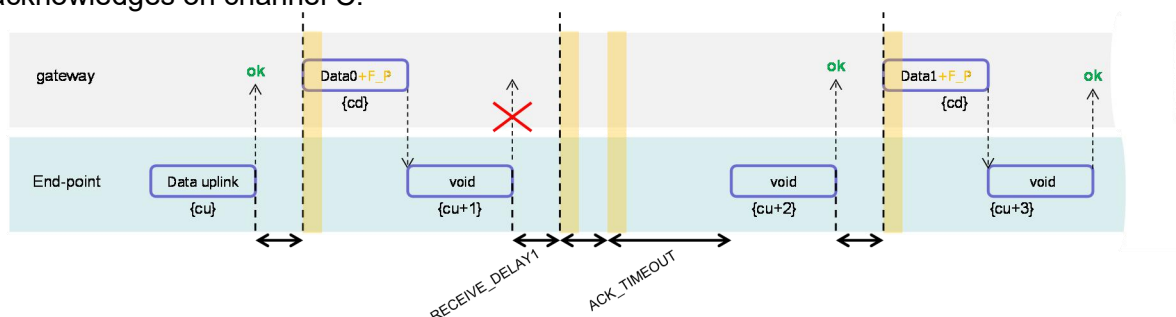


Figure 20: Downlink timing diagram for frame-pending messages, example 2

In this example, the downlink frames are “unconfirmed” frames, the end-device does not need to send back and acknowledge. Receiving the Data0 unconfirmed frame with the FPending bit set the end-device sends an empty data frame. This first uplink is not received by the network. If no downlink is received during the two receive windows, the network has to wait for the next spontaneous uplink of the end-device to retry the transfer. The end-device can speed up the procedure by sending a new empty data frame.

**Note:** An acknowledgement is never sent twice.

The FPending bit, the ACK bit, and payload data can all be present in the same downlink. For example, the following frame exchange is perfectly valid.

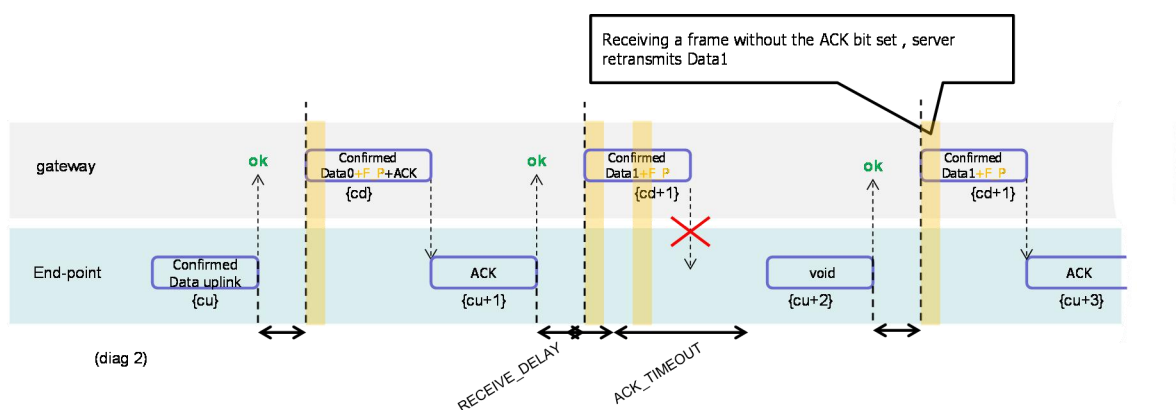


Figure 21: Downlink timing diagram for frame-pending messages, example 3

The end-device sends a “confirmed data” uplink. The network can answer with a confirmed downlink containing Data + ACK + “Frame pending” then the exchange continues as previously described.

## 21.4 Data-Rate Adaptation during Message Retransmissions

When an end-device attempts the transmission of a “confirmed” frame toward the network it expects to receive an acknowledgement in one of the subsequent reception slot. In the absence of the acknowledgement it will try to re-transmit the same data again. This

re-transmission happens on a new frequency channel, but can also happen at a different data rate (preferable lower) than the previous one. It is strongly recommended to adopt the following re-transmission strategy.

The first transmission of the “confirmed” frame happens with a data rate DR.

Transmission nb	Data Rate
1 (first)	DR
2	DR
3	$\max(\text{DR}-1,0)$
4	$\max(\text{DR}-1,0)$
5	$\max(\text{DR}-2,0)$
6	$\max(\text{DR}-2,0)$
7	$\max(\text{DR}-3,0)$
8	$\max(\text{DR}-3,0)$

The Data Rate  $\max(a,b)$  stands for maximum of a and b values.

If after a recommended 8 transmissions, the frame has not been acknowledged the MAC layer should return an error code to the application layer.

**Note:** For each re-transmission, the frequency channel is selected randomly as for normal transmissions.

Any further transmission uses the last data rate used.

For example if an end-device sends a “confirmed” frame first using DR5 and has to retransmit 3 times (twice at DR5 then twice at DR4), the next frame transmitted will use DR4

Other example, if an end-device sends a “confirmed” frame first using DR5 and does not receive an acknowledge after 8 transmissions (2 at DR5, 2 at DR4, ... , 2 at DR2), and the application of this end-device re-initiates a “confirmed” transmission a little later, the first two transmission will be tentatively at DR2, then switch to DR1, then to DR0.



## **22 Recommendation on contract to be provided to the network server by the end-device provider at the time of provisioning**

Configuration data related to the end-device and its characteristics must be known by the network server at the time of provisioning. –This provisioned data is called the “contract”. This contract cannot be provided by the end-device and must be supplied by the end-device provider using another channel (out-of-band communication).

This end-device contract is stored in the network server. It can be used by the application server and the network controller to adapt the algorithms.

This data will include:

- End-device specific radio parameters (device frequency range, device maximal output power, device communication settings - RECEIVE\_DELAY1, RECEIVE\_DELAY2)
- Application type (Alarm, Metering, Asset Tracking, Supervision, Network Control)

## **23 Recommendation on finding the locally used channels**

End-devices that can be activated in territories that are using different frequencies for LoRaWAN will have to identify what frequencies are supported for join message at their current location before they send any message. The following methods are proposed:

- A GPS enabled end-device can use its GPS location to identify which frequency band to use.
- End-device can search for a beacon and use its frequency to identify its region
- End-device can search for a beacon and if this one is sending the antenna GPS coordinate, it can use this to identify its region
- End-device can search for a beacon and if this one is sending a list of join frequencies, it can use this to send its join message

## 24 Revisions

### 24.1 Revision 1.0

- Approved version of LoRaWAN1.0

### 24.2 Revision 1.1

- Fixed the RX window start time definition
- Introduced the ResetInd / ResetConf MAC commands
- Fixed the maximum payload size for DR2 in the NA section
- Corrected the typo on the downlink data rate range in 7.2.2
- Introduced a requirement for using coding rate 4/5 in 7.2.2 to guarantee a maximum time on air < 400mSec
- Corrected the JoinAccept MIC calculation in 6.2.5
- Clarified the NbRep field and renamed it to NbTrans in 5.2
- Removed the possibility to not encrypt the Applicative payload in the MAC layer , removed the paragraph 4.3.3.2. If further security is required by the application , the payload will be encrypted, using any method, at the application layer then re-encrypted at the MAC layer using the specified default LoRaWAN encryption
- Corrected FHDR field size typo
- Corrected the channels impacted by ChMask when chMaskCntl equals 6 or 7 in 7.2.5
- Clarified 6.2.5 sentence describing the RX1 slot data rate offset in the JoinResp message
- Split Data rate and txpower table in 7.1.3 for clarity
- Added DeviceTimeReq/Ans MAC command to class A
- Changed Class B time origin to GPS epoch
- Aligned all beacons of class B to the same time slot. Class B beacon is now common to all networks.

## 25 Glossary

ADR	Adaptive Data Rate
AES	Advanced Encryption Standard
AFA	Adaptive Frequency Agility
AR	Acknowledgement Request
CBC	Cipher Block Chaining
CMAC	Cipher-based Message Authentication Code
CR	Coding Rate
CRC	Cyclic Redundancy Check
DR	Data Rate
ECB	Electronic Code Book
ETSI	European Telecommunications Standards Institute
EIRP	Equivalent Isotropically Radiated Power
FSK	Frequency Shift Keying modulation technique
GPRS	General Packet Radio Service
HAL	Hardware Abstraction Layer
IP	Internet Protocol
LBT	Listen Before Talk
LoRa™	Long Range modulation technique
LoRaWAN™	Long Range Network protocol
MAC	Medium Access Control
MIC	Message Integrity Code
RF	Radio Frequency
RFU	Reserved for Future Usage
Rx	Receiver
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SNR	Signal Noise Ratio
SPI	Serial Peripheral Interface
SSL	Secure Socket Layer
Tx	Transmitter
USB	Universal Serial Bus

## 1    **26 Bibliography**

### 2    **26.1 References**

- 3    [IEEE802154]: IEEE Standard for Local and Metropolitan Area Networks—Part 15.4:  
4    Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4TM-2011  
5    (Revision of IEEE Std 802.15.4-2006), September 2011.
- 6    [RFC4493]: The AES-CMAC Algorithm, June 2006.

## 27 NOTICE OF USE AND DISCLOSURE

Copyright © LoRa Alliance. All Rights Reserved.

The information within this document is the property of the LoRa Alliance. The “LoRa Alliance” and its use and disclosure are subject to LoRa Alliance Corporate Bylaws, Intellectual Property Rights, Privacy Policy and Membership Agreements.

Elements of LoRa Alliance Specifications may be subject to third party intellectual property rights. Inclusion without limitation of patent, copyright or trademark rights. A third party may or may not be a member of LoRa Alliance. The Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

This document and the information contained herein are provided on an “AS IS” basis and THE ALLIANCE DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT, COPYRIGHT OR TRADEMARK RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT.

IN NO EVENT WILL THE ALLIANCE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND IN CONTRACT OR IN TORT IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The above notice and this paragraph must be included on all copies of this document that are made.

LoRa Alliance. Inc.

2400 Gammon Road Suite 375

San Ramon, CA 94583

*Note: All company, brand and product names may be trademarks that are the sole property of their respective owners.*