LoRaWAN Remote Multicast Setup Specification v1.0.0

NOTICE OF USE AND DISCLOSURE

LoRa Alliance™
5177 Brandin Court
Fremont, CA 94538
United States

LoRaWAN Remote Multicast Setup Specification

Authored by the FUOTA Working Group of the LoRa Alliance Technical Committee

Technical Committee Chairs:
N.SORNIN (Semtech), A.YEGIN (Actility)

Working Group Chairs:
J.CATALANO (Kerlink), N.SORNIN (Semtech)

Editor:
J.CATALANO (Kerlink)

Contributors:
J.CATALANO (Kerlink), J-P.COUPIGNY (STMicroelectronics), J.DELCLEF (STMicroelectronics), A.GRIGORE (Flashnet), J.SCHLARB (Comcast), N.SORNIN (Semtech), J.STOKKING (The Things Network Foundation), A.YEGIN (Actility)

Version: v1.0.0
Date: September 10, 2018
Status: Final release

Contents

Tables

1 Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

The octet order over the air for all multi-octet fields is little endian (Least significant byte is sent first).

2 Introduction

This document defines an application layer messaging package running over LoRaWAN to perform the following operations on a fleet of end-devices:

- Program a multicast distribution window into a group of end-devices
- Having all end-devices of the group switch to ClassB or ClassC temporarily at the beginning of the slot
- Close the distribution window and revert to normal operation (e.g. return to Class A, or change to a different periodicity in Class B)

All messages described in this document are transported as application layer messages. As such, all unicast messages (uplink or downlink) are encrypted by the LoRaWAN MAC layer using the end-device's AppSKey. Downlink multicast messages are encrypted using a multicast group McAppSKey common to all end-devices of the group. The setup of the group as well as means to convey the MCAppSKey are described in the document.

The "**multicast control**" package can be used to:

- Remotely create a multicast group security context inside a group of end-devices
- Report the list of multicast context existing in the end-device
- Remotely delete a multicast security context.
- Program a classC multicast session
- Program a classB multicast session

This package uses a dedicated port to separate its traffic from the rest of the applicative traffic.

3 Multicast group context definition

This package makes the following assumptions.
Inside a given end-device a multicast group is defined by the following parameters (the multicast group context):

1. A McGroupID: an integer in [0:3], the index of the multicast group. This index is used as an end-device specific shortcut to reference one of the multicast groups defined inside the end-device. An end-device supports a maximum of 4 simultaneous multicast groups, and a minimum of 0.
2. Multicast address: the 4 bytes network address of the multicast group, common to all end-devices of the group.
3. A multicast group key (McKey) from which are derived a McAppSKey and a McNwkSKey. The McKey is multicast group specific (different for every multicast group), but all end-devices of a given multicast group have the same McKey associated to this group
4. A frame counter.

Because the end-device can be part of up to 4 multicast groups, every multicast control command MUST first define which multicast group is concerned by the command. To minimize the protocol overhead, a 2-bit McGroupID shortcut is used instead of the full 4 bytes multicast group network address in most of the commands defined in this package.
An end-device MAY support up to 4 multicast groups contexts defined simultaneously. If an end-device supports N simultaneous multicast group contexts where 1<=N<=4 then the McGroupID can only be in the range [0:N-1].

> For example, if an end-device is designed to support only a single multicast group, then this group can only have McGroupID=0.

4 Multicast Control Message Package

The identifier of the multicast control package is 2. The version of this package is version 1.

The following messages are sent to each end-device individually using Unicast downlink on a port specifically used for the multicast package. The default port value is 200. These messages MUST NOT be sent using multicast. If these messages are received on a multicast address the end-device MUST drop them silently.

All unicast control messages use the same format:

Command1	Command1 Payload	Command2	Command2 payload	….

A message MAY carry more than one command. The length of each command's payload is fixed and a function of the command. Commands are executed from first to last. Each command MUST be individually acknowledged by the end-device.

The following table summarizes the list of multicast control messages

CID	Command name	Transmitted by		Short Description
		End-device	server	
0x00	***PackageVersionReq***		x	Used by the AS to request the package version implemented by the end-device
0x00	***PackageVersionAns***	x		Conveys the answer to PackageVersionReq
0x01	***McGroupStatusReq***		x	Asks an end-device to list the multicast groups currently configured
0x01	***McGoupStatusAns***	x		Conveys answer to the McGroupStatus request
0x02	***McGroupSetupReq***		x	Provides an end-device will all necessary information to join a multicast group
0x02	***McGroupSetupAns***	x		
0x03	***McGroupDeleteReq***		x	Used to delete a multicast group from an end-device
0x03	***McGroupDeleteAns***	x		
0x04	***McClassCSessionReq***		x	Conveys information about the next classC multicast session the end-device shall join
0x04	***McClassCSessionAns***	x		
0x05	***McClassBSessionReq***		x	Creates a class B multicast session
0x05	***McClassBSessionAns***	x		

Table 1: Multicast Control messages summary

4.1 PackageVersionReq & Ans

The ***PackageVersionReq*** command has no payload.
The end-device answers with a ***PackageVersionAns*** command with the following payload.

Field	PackageIdentifier	PackageVersion
Size (bytes)	1	1

Table 2: PackageVersionAns

PackageIdentifier uniquely identifies the package. For the "multicast control package" this identifier is 2.
PackageVersion corresponds to the version of the package specification implemented by the end-device.

4.2 McGroupStatusReq & Ans

The McGroupStatusReq command has a single byte payload.

Field	CmdMask
Size (bytes)	1

Table 3: McGroupStatusReq

Where:

CmdMask Fields	RFU	ReqGroupMask
Size (bits)	4bits	4bits

Table 4: McGroupStatusReq CmdMask field

The *ReqGroupMask* bit mask defines the multicast groups whose status should be reported by the end-device. ReqGroupMask[n] = 1 means that the n^{th} multicast group status SHOULD be included in the answer. ReqGroupMask[n] = 0 means that this group SHALL NOT be included in the answer.

The end-device responds to the McGroupStatusReq command with a McGroupStatusAns with the following payload:

Field	status	Optional list of [McGroupID+McAddr]
Size (bytes)	1	5xNbItems

Table 5: McGroupStatusAns

The status field encodes the following information:

Status Fields	RFU	NbTotalGroups	AnsGroupMask
Size (bits)	1bit	3bits	4bits

Table 6: McGroupStatusAns Status field

AnsGroupMask is a bit mask describing which groups are listed in the report. If the end-device cannot report the status of the multicast groups specified by the *ReqGroupMask* field of the request, the end-device SHALL discard the n^{th} last groups (starting with the highest GroupID) until the answer fits. In that case, the *AnsGroupMask* mask is different from the

ReqGroupMask. In that case the server can get the status of the groups not listed by issuing a new McGroupStatusReq command with another *RegGroupMask* field. If all groups requested can be listed, *AnsGroupMask* == *ReqGroupMask.*

NbTotalGroups is the number of multicast groups currently defined in the end-device. The valid range is [0:4].

Each record consists of 5 bytes [McGroupID + McAddr].
McGroupID and McAddr are provided to the end-device by McGroupSetupReq.

4.3 McGroupSetupReq & Ans

This command is used to create or modify the parameters of a multicast group.
The payload of the message is:

Field	McGroupIDHeader	McAddr	McKey_encrypted	minMcFCount	maxMcFCount
Size (bytes)	1	4	16	4	4

Table 7: McGroupSetupReq

Where*:*

McGroupIDHeader Fields	RFU	McGroupID
Size (bits)	6bits	2bits

Table 8: McGroupSetupReq McGroupIDHeader field

McGroupID is the multicast group ID of the multicast context. An end-device MAY support being part of several multicast group simultaneously. Therefore, all multicast related command MUST always contain an identifier (the McGroupID) of the multicast group being affected.

> Note: The McAddr could be used as a multicast group identifier but this would add a systematic 4 bytes overhead, so a more compact McGroupID is used. Additionally, if MultiCast keys are kept in a Hardware Secure Element that can only keep a few keys, the MCU needs to indicate which key memory slot should be used. Therefore, the Multicast group ID concept is required.

An end-device implementing this package SHALL support at least one multicast group. An end-device MAY support up to a maximum of 4 simultaneous multicast contexts.

McKey_encrypted is the encrypted multicast group key from which McAppSKey and McNetSKey will be derived. The McKey_encrypted key can be decrypted using the following operation to give the multicast group's McKey.

McKey = aes128_encrypt(McKEKey, McKey_encrypted)

The McKEKey is a **lifetime end-device specific** key used to encrypt Multicast key transported over the air (it is a Key Encryption Key), and may be either:

- Derived from a new root key (GenAppKey) provisioned in the end-device at any time before the deployment of the end-device in the field. LoRaWAN 1.0.x end-devices SHALL use this scheme.

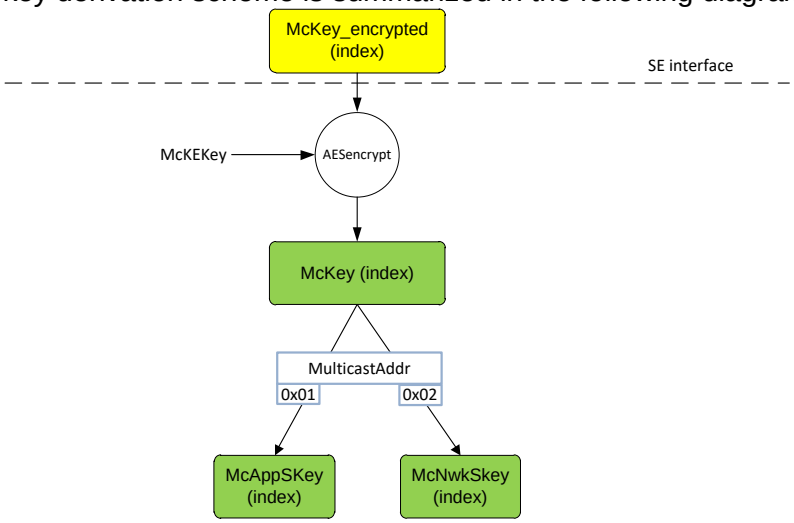
- McRootKey = aes128_encrypt(GenAppKey, 0x00 | pad_{16})
- McKEKey = aes128_encrypt(McRootKey, 0x00 | pad_{16})

- Derived from the AppKey. LoRaWAN 1.1+ end-devices SHALL use this scheme.
  - McRootKey = aes128_encrypt(AppKey, 0x20 | pad_{16})
  - McKEKey = aes128_encrypt(McRootKey, 0x00 | pad_{16})

The McAppSKey and the McNetSKey are then derived from the group's McKey as follow:

McAppSKey = aes128_encrypt(McKey, 0x01 | McAddr | pad_{16})
McNetSKey = aes128_encrypt(McKey, 0x02 | McAddr | pad_{16})

The multicast key derivation scheme is summarized in the following diagram.



> **Note**: using a Key Encryption Key to transport the multicast group McKey allows for a completely secure multicast scheme when using a hardware secure element, when the secure element does not export the McKey, McAppSKey, and McNwkSKey to the outside. It does not increase the security if a full software implementation is used in the end-device. However, for compatibility reason it is recommended to systematically use this scheme.

The *minMcFCount* field is the next frame counter value of the multicast downlink to be sent by the server for this group. This information is required in case an end-device is added to a group that already exists. The end-device MUST reject any downlink multicast frame using this group multicast address if the frame counter is < minMcFCount.

McAddr is the multicast group network address. McAddr is negotiated off-band by the application server with the network server.

maxMcFCount specifies the life time of this multicast group expressed as a maximum number of frames. The end-device will only accept a multicast downlink frame if the 32bits frame counter value minMcFCount ≤ McFCount < maxMcFCount.

The end-device acknowledges the reception of this message by sending an ***McGroupSetupAns*** message with the following payload:

Field	McGroupIDHeader
Size (bytes)	1

Table 9: McGroupSetupAns

Where*:*

McGroupIDHeader Fields	RFU	IDerror	McGroupID
Size (bits)	5	1	2

Table 10: McGroupSetupAns McGroupIDHeader field

When set the *IDerror* bit indicates that the end-device does not support the multicast context indexed by the McGroupID requested by the server. For example, an end-device MAY only support a single multicast group (McGroupID=0). If the server tries to create a second multicast group with McGroupID = 1, the end-device SHALL respond with IDerror=1.

4.4 McGroupDeleteReq & Ans

This message is used to delete a multicast group from an end-device. The command payload is:

Field	McGroupIDHeader
Size (bytes)	1

Table 11: McGroupDeleteReq

Where*:*

McGroupIDHeader Fields	RFU	McGroupID
Size (bits)	6bits	2bits

Table 12: McGroupDeleteReq McGroupIDHeader field

The end-device answers with **McGroupDeleteAns** with payload:

Field	McGroupIDHeader
Size (bytes)	1

Table 13: McGroupDeleteAns

Where*:*

McGroupIDHeader Fields	RFU	MCGroupUndefined	McGroupID
Size (bits)	5bits	1bit	2bits

Table 14: McGroupDeleteAns McGroupIDHeader field

MCGroupUndefined is set 1 if the McGroupID specified by the command is not defined in the end-device (was not created before calling the delete command).

4.5 McClassCSessionReq & Ans

This message is only used to setup a temporary classC multicast session associated with a multicast context.
The payload of the message is:

Field	McGroupIDHeader	Session Time	SessionTimeOut	DLFrequ	DR
Size (bytes)	1	4	1	3	1

Table 15: McClassCSessionReq

Where:

McGroupIDHeader Fields	RFU	McGroupID
Size (bits)	6bits	2bits

Table 16: McClassCSessionReq McGroupIDHeader field

And where:

SessionTimeOut Fields	RFU	TimeOut
Size (bits)	4bits	4bits

Table 17: McClassCSessionReq SessionTimeOut field

McGroupID is the identifier of the multicast group being used.

SessionTime is the start of the Class C window, and is expressed as the time in seconds since 00:00:00, Sunday 6th of January 1980 (start of the GPS epoch) modulo 2^32. Note that this is the same format as the Time field in the beacon frame.

TimeOut encodes the maximum length in seconds of the multicast session (max time the end-device stays in classC before reverting to class A to save battery)
The maximum duration in second is $2^{TimeOut}$ (Example: TimeOut=8 means 256 seconds)
This is a maximum duration because the end-device's application might decide to revert to class A before the end of the session, this decision is application specific.

> For example, the multicast session might be used to broadcast a firmware upgrade file. In that case the end-device might end the multicast session has soon as the full file is received without waiting for TimeOut.

DlFrequ: Encodes the frequency used for the multicast. This field is a 24 bits unsigned integer. The actual channel frequency in Hz is 100 x DlFrequ 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.
This field has the same meaning and coding as LoRaWAN *NewChannelReq* MAC command 'Freq' field.

DR: index of the data rate used for the multicast. Uses the same look-up table than the one used by the LinkAdrReq MAC command of the LoRaWAN protocol.

The end-device acknowledges the reception of this message by sending a ***McClassCSessionAns*** message on the same port with the following payload:

Field	Status&McGroupID	(cond)TimeToStart
Size (bytes)	1	3

Table 18: McClassCSessionAns

Where*:*

Status&McGroupID Fields	RFU	McGroupUndefined	FreqError	DRError	McGroupID
Size (bits)	3bits	1bit	1bit	1bit	2bits

Table 19: McClassCSessionAns Status&McGroupID field

FreqError bit is set to 1 if the DLFrequ frequency set by the network is not usable for the end-device.

DRError bit is set to 1 if the classC downlink Data Rate set by the network is not defined.

McGroupUndefined is set 1 if the McGroupID specified by the command is not defined in the end-device (was not created before calling this command).

If no errors are present, the *TimeToStart* field encodes the number of seconds from the ***McClassCSessionAns*** uplink to the beginning of the multicast session. This allows the server to check that the end-device clock is well synchronized and that the end-device will effectively switch to classC exactly at the right moment (with second accuracy). This is possible because all uplinks are accurately time stamped by the network gateways (at least with an accuracy better than the second).

4.6 McClassBSessionReq & Ans

This message is only used to setup a temporary ClassB multicast session associated with a multicast context.
The payload of the message is:

Field	McGroupIDHeader	Session Time	TimeOutPeriodicity	DLFrequ	DR
Size (bytes)	1	4	1	3	1

Table 20: McClassBSessionReq

Where:

McGroupIDHeader Fields	RFU	McGroupID
Size (bits)	6bits	2bits

Table 21: McClassBSessionReq McGroupIDHeader field

And where:

TimeOutPeriodicity Fields	RFU	Periodicity	TimeOut
Size (bits)	1bits	3bits	4bits

Table 22: McClassBSessionReq TimeOutPeriodicity field

McGroupID is the identifier of the multicast group being used.

SessionTime is the start of the Class B window, and is expressed as the time in seconds since 00:00:00, Sunday 6th of January 1980 (start of the GPS epoch) modulo 2^32. Note that this is the same format as the Time field in the beacon frame. SessionTime MUST be an integer multiple of 128.

TimeOut encodes the maximum length in BeaconPeriods (128seconds) of the multicast fragmentation session (max time the end-device stays in classB before eventually reverting to class A to save battery)
The maximum duration in second is $128*2^{TimeOut}$ (Example: TimeOut=8 corresponds roughly to 9.1hours).

> **Attention**: For classB TimeOut is expressed in BeaconPeriod (128sec), whereas it is expressed in seconds for classC. This is because a classB multicast session is heavily duty-cycled and is likely to last a lot longer than a classC session.

This is a maximum duration because the end-device's application might decide to revert to class A before the end of the session, this decision is application specific.

Periodicity encodes the classB ping slot periodicity for the multicast group. The encoding format is the same than for the Periodicity field of the ***PingSlotInfoReq*** classB MAC command defined in LoRaWAN.

DlFrequ: Encodes the frequency used for the multicast. This field is a 24 bits unsigned integer. The actual channel frequency in Hz is 100 x DlFrequ 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.
This field has the same meaning and coding as LoRaWAN *NewChannelReq* MAC command 'Freq' field.

In regions where the classB beacon is transmitted following a frequency hopping pattern, DlFrequ=0 signals the end-device to use the default classB default frequency hopping scheme. That scheme is defined in the classB section of the LoRaWAN specification. In that case, Class B downlinks use a channel which is a function of the Time field of the last beacon (see Beacon Frame content) and the multicast address McAddr.

$$\text{Class B downlink channel} = \left[\text{McAddr} + \text{floor}\left(\frac{\text{Beacon\_Time}}{\text{Beacon\_period}}\right)\right] \text{ modulo NbChannel}$$

- 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
- McAddr is the 32-bit network address of the multicast group
- NbChannel is the number of channel over which the beacon is frequency hopping

DR: index of the data rate used for the classB multicast. Uses the same look-up table than the one used by the LinkAdrReq MAC command of the LoRaWAN protocol.

The end-device acknowledges the reception of this message by sending a ***McClassBSessionAns*** message on the same port with the following payload:

Field	Status&McGroupID	(cond)TimeToStart
Size (bytes)	1	3

Table 23: McClassBSessionAns

Where*:*

Status&McGroupID Fields	RFU	McGroupUndefined	FreqError	DRError	McGroupID
Size (bits)	3bits	1bit	1bit	1bit	2bits

Table 24: McClassBSessionAns Status&McGroupID field

FreqError bit is set to 1 if the DLFrequ frequency set by the network is not usable for the end-device.

DRError bit is set to 1 if the classB downlink Data Rate set by the network is not defined.

McGroupUndefined is set 1 if the McGroupID specified by the command is not defined in the end-device (was not created before calling this command).

If no errors are present, the *TimeToStart* field encodes the number of seconds from the ***McClassBSessionAns*** uplink to the beginning of the multicast fragmentation session. This allows the server to check that the end-device clock is roughly synchronized and that it will effectively start acquiring the classB beacon at the right moment (before the beginning of the classB multicast session with some margin).

5 Glossary

AS	Application Server
TBD	To Be Done

6 Bibliography

6.1 References

[LoRaWAN 1.0.2]: LoRaWAN™ 1.0.2 Specification, LoRa Alliance, July 2016

[LoRaWAN 1.1]: LoRaWAN™ 1.1 Specification, LoRa Alliance, October 11, 2017

7 NOTICE OF USE AND DISCLOSURE

LoRa Alliance™
5177 Brandin Court
Fremont, CA 94538
United States