

1 **LoRaWAN Remote Multicast Setup Specification v1.0.0**

2 Copyright © 2018 LoRa Alliance, Inc. All rights reserved.

3

4 **NOTICE OF USE AND DISCLOSURE**

5 Copyright © LoRa Alliance, Inc. (2018). All Rights Reserved.

6

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

10

11 Elements of LoRa Alliance specifications may be subject to third party intellectual property rights, including
12 without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of LoRa
13 Alliance). The Alliance is not responsible and shall not be held responsible in any manner for identifying or failing
14 to identify any or all such third party intellectual property rights.

15

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

22

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

28

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

30

31 LoRa Alliance™
32 5177 Brandin Court
33 Fremont, CA 94538
34 United States

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



38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

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

64 Contents

65	1	Conventions	4
66	2	Introduction	5
67	3	Multicast group context definition	6
68	4	Multicast Control Message Package	7
69	4.1	PackageVersionReq & Ans	8
70	4.2	McGroupStatusReq & Ans	8
71	4.3	McGroupSetupReq & Ans	9
72	4.4	McGroupDeleteReq & Ans	11
73	4.5	McClassCSessionReq & Ans	12
74	4.6	McClassBSessionReq & Ans	13
75	5	Glossary	16
76	6	Bibliography	17
77	6.1	References.....	17
78	7	NOTICE OF USE AND DISCLOSURE.....	18
79			

80 Tables

81	Table 1: Multicast Control messages summary	7
82	Table 2: PackageVersionAns	8
83	Table 3: McGroupStatusReq.....	8
84	Table 4: McGroupStatusReq CmdMask field	8
85	Table 5: McGroupStatusAns	8
86	Table 6: McGroupStatusAns Status field.....	8
87	Table 7: McGroupSetupReq.....	9
88	Table 8: McGroupSetupReq McGroupIDHeader field.....	9
89	Table 9: McGroupSetupAns	11
90	Table 10: McGroupSetupAns McGroupIDHeader field	11
91	Table 11: McGroupDeleteReq.....	11
92	Table 12: McGroupDeleteReq McGroupIDHeader field.....	11
93	Table 13: McGroupDeleteAns	11
94	Table 14: McGroupDeleteAns McGroupIDHeader field	11
95	Table 15: McClassCSessionReq.....	12
96	Table 16: McClassCSessionReq McGroupIDHeader field.....	12
97	Table 17: McClassCSessionReq SessionTimeOut field	12
98	Table 18: McClassCSessionAns	13
99	Table 19: McClassCSessionAns Status&McGroupID field	13
100	Table 20: McClassBSessionReq	13
101	Table 21: McClassBSessionReq McGroupIDHeader field.....	13
102	Table 22: McClassBSessionReq TimeOutPeriodicity field.....	13
103	Table 23: McClassBSessionAns	15
104	Table 24: McClassBSessionAns Status&McGroupID field	15

105

106 1 Conventions

107

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

111

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

114

115 **2 Introduction**

116

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

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

124

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

130 The “**multicast control**” package can be used to:

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

136

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

139

140 **3 Multicast group context definition**

141

142 This package makes the following assumptions.

143 Inside a given end-device a multicast group is defined by the following parameters (the
144 multicast group context):

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

156 Because the end-device can be part of up to 4 multicast groups, every multicast control
157 command MUST first define which multicast group is concerned by the command. To
158 minimize the protocol overhead, a 2-bit McGroupID shortcut is used instead of the full 4
159 bytes multicast group network address in most of the commands defined in this package.

160 An end-device MAY support up to 4 multicast groups contexts defined simultaneously. If an
161 end-device supports N simultaneous multicast group contexts where $1 \leq N \leq 4$ then the
162 McGroupID can only be in the range [0:N-1].

163

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

166

167

168 4 Multicast Control Message Package

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

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

175 All unicast control messages use the same format:
 176

Command1	Command1 Payload	Command2	Command2 payload
----------	---------------------	----------	---------------------	------

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

181 The following table summarizes the list of multicast control messages
 182

183 The following table summarizes the list of multicast control messages

184

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

185 **Table 1: Multicast Control messages summary**

186

187 4.1 PackageVersionReq & Ans

188

189 The *PackageVersionReq* command has no payload.

190 The end-device answers with a *PackageVersionAns* command with the following payload.

191

Field	PackageIdentifier	PackageVersion
Size (bytes)	1	1

192

Table 2: PackageVersionAns

193 *PackageIdentifier* uniquely identifies the package. For the “multicast control package” this
 194 identifier is 2.

195 *PackageVersion* corresponds to the version of the package specification implemented by the
 196 end-device.

197 4.2 McGroupStatusReq & Ans

198

199 The McGroupStatusReq command has a single byte payload.

200

Field	CmdMask
Size (bytes)	1

201

Table 3: McGroupStatusReq

202

203 Where:

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

204

Table 4: McGroupStatusReq CmdMask field

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

209

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

212

213

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

214

Table 5: McGroupStatusAns

215

216 The status field encodes the following information:

217

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

218

Table 6: McGroupStatusAns Status field

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

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

226
 227 *NbTotalGroups* is the number of multicast groups currently defined in the end-device. The
 228 valid range is [0:4].

229
 230 Each record consists of 5 bytes [*McGroupID* + *McAddr*].
 231 *McGroupID* and *McAddr* are provided to the end-device by *McGroupSetupReq*.

232 4.3 *McGroupSetupReq* & Ans

233
 234 This command is used to create or modify the parameters of a multicast group.
 235 The payload of the message is:

Field	<i>McGroupIDHeader</i>	<i>McAddr</i>	<i>McKey_encrypted</i>	<i>minMcFCount</i>	<i>maxMcFCount</i>
Size (bytes)	1	4	16	4	4

237 **Table 7: *McGroupSetupReq***

238
 239 Where:

<i>McGroupIDHeader</i> Fields	RFU	<i>McGroupID</i>
Size (bits)	6bits	2bits

241 **Table 8: *McGroupSetupReq* *McGroupIDHeader* field**

242
 243 *McGroupID* is the multicast group ID of the multicast context. An end-device MAY support
 244 being part of several multicast group simultaneously. Therefore, all multicast related
 245 command MUST always contain an identifier (the *McGroupID*) of the multicast group being
 246 affected.

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

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

255
 256 *McKey_encrypted* is the encrypted multicast group key from which *McAppSKey* and
 257 *McNetSKey* will be derived. The *McKey_encrypted* key can be decrypted using the following
 258 operation to give the multicast group's *McKey*.

$$259 \text{McKey} = \text{aes128_encrypt}(\text{McKEKey}, \text{McKey_encrypted})$$

260
 261 The *McKEKey* is a **lifetime end-device specific** key used to encrypt Multicast key
 262 transported over the air (it is a Key Encryption Key), and may be either:

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

- 266 ○ McRootKey = aes128_encrypt(GenAppKey, 0x00 | pad₁₆)
- 267 ○ McKEKey = aes128_encrypt(McRootKey, 0x00 | pad₁₆)
- 268
- 269 • Derived from the AppKey. LoRaWAN 1.1+ end-devices SHALL use this scheme.
- 270 ○ McRootKey = aes128_encrypt(AppKey, 0x20 | pad₁₆)
- 271 ○ McKEKey = aes128_encrypt(McRootKey, 0x00 | pad₁₆)
- 272

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

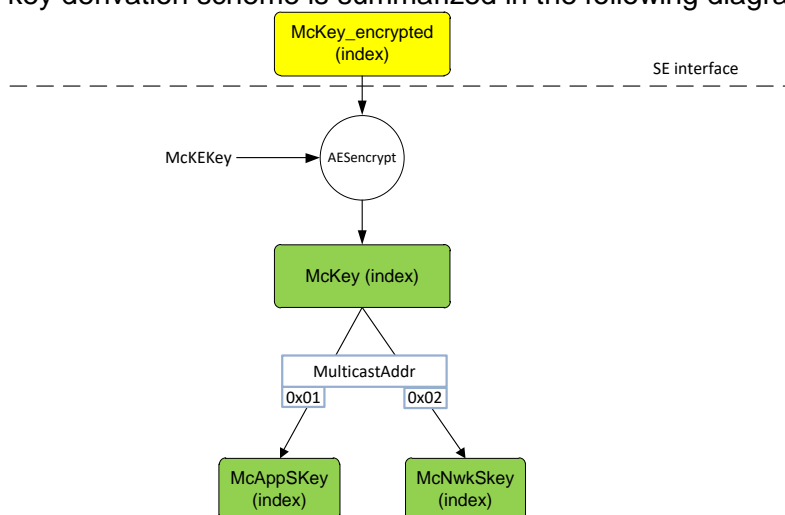
274

275 McAppSKey = aes128_encrypt(McKey, 0x01 | McAddr | pad₁₆)

276 McNetSKey = aes128_encrypt(McKey, 0x02 | McAddr | pad₁₆)

277

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



279

280

281 **Note:** using a Key Encryption Key to transport the multicast group

282 McKey allows for a completely secure multicast scheme when using a

283 hardware secure element, when the secure element does not export

284 the McKey, McAppSKey, and McNwkSKey to the outside. It does not

285 increase the security if a full software implementation is used in the

286 end-device. However, for compatibility reason it is recommended to

287 systematically use this scheme.

288

289 The *minMcFCount* field is the next frame counter value of the multicast downlink to be sent

290 by the server for this group. This information is required in case an end-device is added to a

291 group that already exists. The end-device MUST reject any downlink multicast frame using

292 this group multicast address if the frame counter is < minMcFCount.

293

294 *McAddr* is the multicast group network address. *McAddr* is negotiated off-band by the

295 application server with the network server.

296

297 *maxMcFCount* specifies the life time of this multicast group expressed as a maximum

298 number of frames. The end-device will only accept a multicast downlink frame if the 32bits

299 frame counter value $\text{minMcFCount} \leq \text{McFCount} < \text{maxMcFCount}$.

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

Field	McGroupIDHeader
Size (bytes)	1

306 **Table 9: McGroupSetupAns**

307 Where:

McGroupIDHeader Fields	RFU	IDerror	McGroupID
	Size (bits)	5	1

309 **Table 10: McGroupSetupAns McGroupIDHeader field**

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

314 4.4 McGroupDeleteReq & Ans

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

Field	McGroupIDHeader
Size (bytes)	1

321 **Table 11: McGroupDeleteReq**

322 Where:

McGroupIDHeader Fields	RFU	McGroupID
	Size (bits)	6bits

324 **Table 12: McGroupDeleteReq McGroupIDHeader field**

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

Field	McGroupIDHeader
Size (bytes)	1

327 **Table 13: McGroupDeleteAns**

328 Where:

McGroupIDHeader Fields	RFU	McGroupUndefined	McGroupID
	Size (bits)	5bits	1bit

330 **Table 14: McGroupDeleteAns McGroupIDHeader field**

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

333

334 **4.5 McClassCSessionReq & Ans**

335

336 This message is only used to setup a temporary classC multicast session associated with a
337 multicast context.

338 The payload of the message is:

339

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

340

Table 15: McClassCSessionReq

341 *Where:*

McGroupIDHeader Fields	RFU	McGroupID
	Size (bits)	6bits

342

Table 16: McClassCSessionReq McGroupIDHeader field

343

344 *And where:*

345

SessionTimeOut Fields	RFU	TimeOut
	Size (bits)	4bits

346

Table 17: McClassCSessionReq SessionTimeOut field

347

348 *McGroupID* is the identifier of the multicast group being used.

349

350 *SessionTime* is the start of the Class C window, and is expressed as the time in seconds
351 since 00:00:00, Sunday 6th of January 1980 (start of the GPS epoch) modulo 2³². Note that
352 this is the same format as the Time field in the beacon frame.

353

354 *TimeOut* encodes the maximum length in seconds of the multicast session (max time the
355 end-device stays in classC before reverting to class A to save battery)

356 The maximum duration in second is 2^{TimeOut} (Example: TimeOut=8 means 256 seconds)

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

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

363

364

365 *DLFreque*: Encodes the frequency used for the multicast. This field is a 24 bits unsigned
366 integer. The actual channel frequency in Hz is 100 x DIFrequ whereby values representing
367 frequencies below 100 MHz are reserved for future use. This allows setting the frequency of
368 a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps.

369 This field has the same meaning and coding as LoRaWAN *NewChannelReq* MAC command
370 'Freq' field.

371

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

374

375 The end-device acknowledges the reception of this message by sending a
 376 **McClassCSessionAns** message on the same port with the following payload:
 377

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

378 **Table 18: McClassCSessionAns**

379 Where:
 380

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

381 **Table 19: McClassCSessionAns Status&McGroupID field**

382
 383 *FreqError* bit is set to 1 if the DLFreq frequency set by the network is not usable for the
 384 end-device.

385
 386 *DRError* bit is set to 1 if the classC downlink Data Rate set by the network is not defined.
 387

388 *McGroupUndefined* is set 1 if the McGroupID specified by the command is not defined in the
 389 end-device (was not created before calling this command).
 390

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

397 4.6 McClassBSessionReq & Ans

398
 399 This message is only used to setup a temporary ClassB multicast session associated with a
 400 multicast context.

401 The payload of the message is:
 402

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

403 **Table 20: McClassBSessionReq**

404 Where:

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

405 **Table 21: McClassBSessionReq McGroupIDHeader field**

406 And where:
 407

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

408 **Table 22: McClassBSessionReq TimeOutPeriodicity field**

409

410 *McGroupID* is the identifier of the multicast group being used.

411

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

416

417 *TimeOut* encodes the maximum length in BeaconPeriods (128seconds) of the multicast
 418 fragmentation session (max time the end-device stays in classB before eventually reverting
 419 to class A to save battery)

420 The maximum duration in second is $128 * 2^{TimeOut}$ (Example: *TimeOut*=8 corresponds roughly
 421 to 9.1hours).

422

423

424

425

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.

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

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

431

432 *DIFrequ*: Encodes the frequency used for the multicast. This field is a 24 bits unsigned
 433 integer. The actual channel frequency in Hz is $100 \times DIFrequ$ whereby values representing
 434 frequencies below 100 MHz are reserved for future use. This allows setting the frequency of
 435 a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps.

436 This field has the same meaning and coding as LoRaWAN *NewChannelReq* MAC command
 437 'Freq' field.

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

443 Class B downlink channel = $\left[McAddr + \text{floor} \left(\frac{\text{Beacon_Time}}{\text{Beacon_period}} \right) \right] \text{ modulo NbChannel}$

444

445

446

447

448

449

450

451

452

453

454

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.

455 The end-device acknowledges the reception of this message by sending a
 456 **McClassBSessionAns** message on the same port with the following payload:
 457

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

458 **Table 23: McClassBSessionAns**

459 Where:
 460

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

461 **Table 24: McClassBSessionAns Status&McGroupID field**

462
 463 *FreqError* bit is set to 1 if the DLFreq frequency set by the network is not usable for the
 464 end-device.

465
 466 *DRError* bit is set to 1 if the classB downlink Data Rate set by the network is not defined.
 467

468 *McGroupUndefined* is set 1 if the McGroupID specified by the command is not defined in the
 469 end-device (was not created before calling this command).
 470

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

476 **5 Glossary**

477

478 AS Application Server

479

480 TBD To Be Done

481

482 **6 Bibliography**483 **6.1 References**

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

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

486

487 7 NOTICE OF USE AND DISCLOSURE

488 Copyright © LoRa Alliance, Inc. (2018). All Rights Reserved.

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

492 Elements of LoRa Alliance specifications may be subject to third party intellectual property
493 rights, including without limitation, patent, copyright or trademark rights (such a third party
494 may or may not be a member of LoRa Alliance). The Alliance is not responsible and shall
495 not be held responsible in any manner for identifying or failing to identify any or all such third
496 party intellectual property rights.

497 This document and the information contained herein are provided on an “AS IS” basis and
498 THE ALLIANCE DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING
499 BUT NOT LIMITED TO (A) ANY WARRANTY THAT THE USE OF THE INFORMATION
500 HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES (INCLUDING WITHOUT
501 LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT,
502 COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF
503 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR
504 NONINFRINGEMENT.

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

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

513 LoRa Alliance™
514 5177 Brandin Court
515 Fremont, CA 94538
516 United States

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