LoRaWAN® Multi Package Access Protocol Specification TS007-1.0.0-rc4

NOTICE OF USE AND DISCLOSURE

Copyright © 2021 LoRa Alliance, Inc. All rights reserved. The information within this document is the property of the LoRa Alliance ("The Alliance") and its use and disclosure are subject to LoRa Alliance Corporate Bylaws, Intellectual Property Rights (IPR) Policy and Membership Agreements.

Elements of LoRa Alliance specifications may be subject to third-party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of the 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 (A) 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 (B) ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT. IN NO EVENT WILL THE ALLIANCE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTIAL, 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.

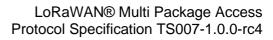
- LoRa Alliance®
- 33 5177 Brandin Court
- 34 Fremont, CA 94538
- 35 United States

Note: LoRa Alliance® and LoRaWAN® are trademarks of the LoRa Alliance, used by permission. All company, brand and product names may be trademarks that are the sole property of their respective owners.





41	
42	
43	LoRaWAN® Multi Package Access
44	Protocol Specification
45	TS007-1.0.0-rc4
46	
47	Authored by the FUOTA Working Group of the LoRa Alliance Technical Committee
48	
49	Technical Committee Chair and Vice-Chair:
50	A. YEGIN (Actility), O. SELLER (Semtech)
51	
52	Working Group Chair:
53	J. CATALANO (Kerlink)
54	
55	Editor:
56 57	J. CATALANO (Kerlink)
58	Contributors:
59	J. CATALANO (Kerlink), R. HOUDE (Senet), M. LE GOURRIEREC (Sagemcom),
60	N. SORNIN (Semtech), J. SWETINA (NEC)
61	
62	Version : 1.0.0-rc4
63	Date : May 28, 2021
64	Status: RELEASE CANDIDATE
65	





66	Contents	
67	1 Revisions (Remove in final version) Error! Bookmark not of	defined.
68	2 Conventions	<u>4</u> 5
69	3 Introduction	
70	4 Multi-Package Access Protocol	
71	4.1 Multi-Package Access (usable on FPort 225)	
72	5 Multi-package Access Package	<u>9</u> 10
73	5.1 Package Version Commands (Package Version Req, Package Version Ans)	
74	5.2 Device Package Commands (DevPackageReq, DevPackageAns)	
75	5.3 Multi-Package Buffer Fragmentation (MultiPackBufferFrag)	<u>10</u> 11
76	5.4 Multi-Package Buffer Command (MultiPackBufferReq)	
77	6 Glossary	
78	7 Bibliography	
79 80	7.1 References	<u>15</u> 1€
81 82	Tables Table 1: Multi- package access command format	67
_	Table 1: Multi- package access command format	<u>0</u> 7
83 84	Table 2: Command Header byte format Error! Bookmark not c	<u>terinea.</u> 4
	Table 2: Dealers Identifier field format	lofinad (
	Table 3: PackageIdentifier field format	
85	Table 4: ANS buffer format	<u>7</u> 8
85 86	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format	<u>7</u> 8 <u>9</u> 10
85 86 87	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format Table 6: Multi-package commands and command-answers	<u>7</u> 8 <u>9</u> 10 <u>9</u> 10
85 86 87 88	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format Table 6: Multi-package commands and command-answers Table 7: PackageVersionAns	<u>7</u> 8 <u>9</u> 10 <u>9</u> 10
85 86 87 88 89	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format Table 6: Multi-package commands and command-answers Table 7: PackageVersionAns Table 8: Payload of DevPackageAns	<u>7</u> 8 <u>9</u> 10 <u>10</u> 11
85 86 87 88 89 90	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format Table 6: Multi-package commands and command-answers Table 7: PackageVersionAns Table 8: Payload of DevPackageAns Table 9: DevPackageAns Nb Packages Fields	<u>7</u> 8 <u>9</u> 10 <u>10</u> 11 <u>10</u> 11
85 86 87 88 89	Table 4: ANS buffer format	<u>7</u> 8 <u>9</u> 10 <u>10</u> 11 <u>10</u> 11 <u>10</u> 12
85 86 87 88 89 90	Table 4: ANS buffer format	<u>7</u> 8 910 1011 1014 1012 1112
85 86 87 88 89 90 91	Table 4: ANS buffer format	
85 86 87 88 89 90 91 92 93	Table 4: ANS buffer format	
85 86 87 88 89 90 91 92 93 94	Table 4: ANS buffer format	
85 86 87 88 89 90 91 92 93 94 95	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format Table 6: Multi-package commands and command-answers Table 7: PackageVersionAns Table 8: Payload of DevPackageAns Table 9: DevPackageAns Nb Packages Fields Table 10: MultiPackBufferFrag payload Table 11: MultiPackBufferFrag command example Table 12: MultiPackBufferFrag command example Table 13: MultiPackBufferFrag command example Table 14: MultiPackBufferReq payload	78 910 910 1011 1014 1012 1112 1112 1112 1213
85 86 87 88 89 90 91 92 93 94 95 96	Table 4: ANS buffer format Table 5: Multi-package commands and command-answer format Table 6: Multi-package commands and command-answers Table 7: PackageVersionAns Table 8: Payload of DevPackageAns Table 9: DevPackageAns Nb Packages Fields Table 10: MultiPackBufferFrag payload Table 11: MultiPackBufferFrag command example Table 12: MultiPackBufferFrag command example Table 13: MultiPackBufferFrag command example Table 14: MultiPackBufferReq payload Table 15: MultiPackBufferReq payload when request is invalid Table 16: MultiPackBufferReq command example Table 17: MultiPackBufferReq command example	78 910 1011 1014 1012 1112 1112 1213 1213
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99	Table 4: ANS buffer format	78 910 910 1011 1014 1012 1112 1112 1112 1114 1114
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99	Table 4: ANS buffer format	78 910 910 1011 1014 1014 1112 1112 1112 11142 11213 11314 11314
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99	Table 4: ANS buffer format	78 910 910 1011 1014 1014 1112 1112 1112 11142 11213 11314 11314





1 Conventions

104 105 106

107

108

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF Best Current Practice 14 (BCP14 [RFC2119] [RFC8174]) when, and only when, they appear in all capitals, as shown here.

109 110 111

The tables in this document are normative. The figures and notes in this document are informative.

112113114

Document titles are written as *LoRaWAN Link Layer Specification* and sections within a document are written as "Frequency-Hopping Beacon Transmission".

115 116 117

118

120

121

122

Commands are written *PackageVersionReq*, bits and bit fields are written PackageIdentifier, constants are written RECEIVE_DELAY1, variables are written *N*.

119 In this document:

- The octet order for all multi-octet fields SHALL be little endian.
- EUI are 8-octet fields and SHALL be transmitted as little endian.
- By default, RFU bits are Reserved for Future Use and SHALL be set to 0 by the transmitter of the packet and SHALL be silently ignored by the receiver.



2 Introduction

125126127

128

129

130 131

132

133

The FUOTA Working Group inside the LoRa Alliance® is defining several application layer packages serving various purposes. Packages defined for the moment are *LoRaWAN Application Layer Clock Synchronization Specification* [TS003], *LoRaWAN Fragmented Data Block Transport Specification* [TS004], *LoRaWAN Remote Multicast Setup Specification* [TS005], *LoRaWAN Firmware Management Protocol Specification* [TS006] and more will probably come in the future. Each of those packages will also probably evolve and feature several versions.

134 135 This specification defines a standard application layer allowing to query what are the packages implemented on the end-device and on which FPort they are running.

136 137

138

This specification also defines a way to send several commands belonging to the same package or different packages to an end-device inside a single downlink payload to limit the number of downlinks required.



3 Multi-Package Access Protocol

142 143

144

146

141

A command implemented by one of the packages running on the end-device MAY be invoked by two methods:

- Dedicated package access
 - Multi-package access
- 147 If implemented the Multi-Package Access Protocol SHALL use FPort 225.
- Other packages MAY use any FPort value from those not assigned by the LoRa Alliance[®] and excluding FPort 225 for dedicated-package access. Every package SHALL use a different
- 150 FPort number, and this FPort number SHALL NOT be used by the end-device's application.

3.1 Multi-Package Access (usable on FPort 225)

151 152 153

This method SHALL be used on FPort 225 only.

The Application Server sends an applicative downlink to the end-device on the multi-package FPort (225), with the following syntax:

155 156

154

PackageID	CommandID	Command1	PackageID	CommandID	Command2		Command
1 (opt)	1	Payload	2 (opt)	2	payload	•••	Token

Table 1: Multi- package access command format

158

157

The Command Token field is a 1-byte field that follows the following format:

159

	Bits	7:2	1:0
Command	Token Fields	RFU	Token

160

Table 2: Command Token field format

The Command Token field SHALL be appended to all downlink command sets except if the

161 162 163

164

downlink only contains a *MultiPackBufferReq* command (Refer to section 4.45.4). The Command Token field is used by the server to differentiate the end-device's answers to different consecutive downlink commands. The end-device SHALL systematically append the Command Token field into the uplink answer(s) to a Multi-Package server command.

165 166 167

168

169

170

171

172

The multi-package FPort MAY be used to access any package. A PackageID field is used to differentiate among packages to which individual commands belong, that are sent via multi-package access.

The first command (CommandID1) SHALL be prefixed with a PackageID field if the first package identifier is different from 0 (Multi-package access package). If the first package identifier is 0 (Multi-package access package) it SHALL NOT be prefixed with a PackageID field.

173 174 175

176

177

For the next commands, a PackageID field is REQUIRED when the command does not belong to the same package than the preceding command. If several consecutive commands belong to the same package, only the first command of the series SHALL be prefixed with a PackageID field. The PackageID field SHALL NOT be used for the following commands.

178 179 180

PackageID field is a 1-byte field that follows the following format:

Bits	7	6:0



PackageID Fields 1 PackageIdentifier

Table 3: PackageID field format

The Most Significant bit (MSb) set to 1 differentiates a PackageID field from a CommandID field. This means that the CommandID SHALL be less than 128 to be used on the FPort 225. A CommandID greater or equal to 128 will be interpreted as a PackageID and SHALL NOT be used on FPort 225.

The PackageIdentifier of the multi-package access package is 0. The PackageVersion of this package is version 1. Multi-package access protocol implements the commands described in section 45 of this document.

The PackageIdentifier field is defined in the specification document of each package. This method introduces an overhead but allows to group several commands belonging to different packages inside a single downlink. For example, setup a temporary Class C slot and a FUOTA fragmentation session in a single downlink.

When a set of commands are received on FPort 225 the end-device SHALL process all package commands in the order they are received (pkg1/cmd1 then pkg2/cmd2). The answer of each command is appended to a buffer (the ANS buffer) using the following format.

		Command1			Command2	
PackageID	CommandID	answer	PackageID	CommandID	answer	 Command Token
1 (opt)	1	payload	2 (opt)		payload	Token

Table 4: ANS buffer format

Whenever a command is prefixed with a PackageID field in the request, the PackageID field SHALL also be copied in the ANS buffer.

The size of the ANS buffer SHALL NOT exceed 128 bytes.

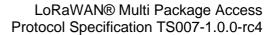
Note: Each command is processed as if there was no limitation on the maximum answer payload size. In general, the maximum size of 128 bytes of the *ANS* buffer is sufficient for firmware management purposes. In case the concatenation of all the answers would exceed 128 bytes, only the first 128 bytes are copied into the *ANS* buffer, the rest is discarded.

Once all commands have been processed and all answers have been gathered in the ANS buffer the end-device transmits it on FPort 225 using one of the two following methods:

- If (ANS buffer length + 1) is smaller or equal to the maximum payload length authorized for the currently used data rate (DR), the complete buffer is transmitted on FPort 225 without any modification.
- If the ANS buffer length is greater than the maximum payload length authorized, the buffer is fragmented over several uplinks and each fragment is sent using the **MultiPackBufferFrag** command.

All Multi-Package uplinks (fragmented or not) on FPort 225 SHALL end with a Command Token field. The Token subfield SHALL copy the Token value of the last valid command set received.

An example of fragmented ANS buffer transmission is provided later in this document.





The server MAY request the retransmission of any part of the buffer using the *MultiPackBufferReq* command described in the next section of this document.

The ANS buffer and Command Token field SHALL be kept in memory until a new multipackage set of commands is received, to allow on-demand retransmissions.

229230231

232

233

234

226

227 228

The size of each command answer payload SHALL be derivable from the command being answered, so that the server can identify the next command byte if present (or the message end is reached).



238239240

241

242

243

244

245246

247

248

249250

251

252

253254255

4 Multi-package Access Package

The PackageIdentifier of the multi-package access package is 0. The PackageVersion of this package is version 1.

The following messages are sent to each end-device individually using Unicast downlink on a FPort specifically used for the package. If this package is implemented by the end-device, then it SHALL use FPort 225. These messages SHALL NOT be sent using multicast. If these messages are received on a multicast address the end-device SHALL drop them silently. These commands SHALL be invoked using the **multi-package access protocol only**. Similarly, all answers SHALL also use the multi-package protocol format:

potentially	PackageID	CommandID	Command1	potentially
preceding	= 0	1	(answer) payload	additional multi-
packages	(opt)		71 7	package access
	(0)			commands and/or
				following different
				packages

Table 5: Multi-package commands and command-answer format

The following table summarizes the list of multi-package access messages.

CID	Command name	Transmitted by		Short Description
		End- device	App Server	
0x00	PackageVersionReq		X	Used by the Application Server to request the package version implemented by the end-device
0x00	PackageVersionAns	X		Used by the end-device to convey the answer to <i>PackageVersionReq</i>
0x01	DevPackageReq		х	Requests the list of packages running on the end-device and on which ports
0x01	DevPackageAns	Х		Used by the end-device to convey the answer to DevPackagesReq
0x02	MultiPackBufferFrag	х		Used by the end-device to convey fragment of the ANS buffer if buffer exceeds the max applicative payload length. Also convey the answer to the Multi-Package Buffer Command (MultiPackBufferReqMulti-Package Buffer Command (MultiPackBufferReq request)
0x02	MultiPackBufferReq		х	Used by server to request a retransmission of a part of the last multi-package answer buffer

Table 6: Multi-package commands and command-answers

4.1 Package Version Commands (*PackageVersionReq*, *PackageVersionAns*)

The *PackageVersionReq* command has no payload.



The end-device SHALL answer this command with a *PackageVersionAns* command with the following payload.

258

261

263264265

266267

Size (octets)	1	1
Field	PackageIdentifier	PackageVersion
'	Table 7: PackageVersion	Ans

259

260 PackageIdentifier uniquely identifies the package.

 ${\tt PackageVersion} \ \textbf{corresponds} \ \textbf{to} \ \textbf{the version of the package specification implemented by}$

the end-device.

4.2 Device Package Commands (DevPackageReg, DevPackageAns)

The **DevPackageReg** command has no payload.

The end-device responds to the **DevPackageReq** command with a **DevPackageAns** with the following payload:

268269270

Size (octets) Field	ĺ
Field	I
	l

;)	1	1	1	1	1	1	1	
d	Nb Packages	_	Package A version	portA	Package B identifier	Package B version	portB	

Table 8: Payload of DevPackageAns

Nb packages: This field contains the total number of packages implemented by the enddevice (including "multi-package").

274

275

276

271

		Bits
Nb	Packages	Fields

	_
7:4	3:0
RFU	NbTotalPackages

Table 9: DevPackageAns Nb Packages Fields

NbTotalPackages is the total number of packages implemented by the end-device.

277278279

Package X identifier is the package identifier (PackageIdentifier) of a package implemented on the end-device. This package identifier is unique per package and is defined in the specification document of each package.

280 281 282

Package X version is the major version (PackageVersion) of the package implemented on the end-device. The capabilities of a package version are defined in the specification document of each package.

284 285 286

283

Port is the FPort at which the end-device can receive or answer commands for this package.

287 288

289 290

4.3 Multi-Package Buffer Fragmentation (*MultiPackBufferFrag*)

This command is used by the end-device to signal it is transmitting a fragment of the ANS buffer. The format of the command payload is:

Size (octets) 1 Variable: 1 to MaxPayloadLen - 3 1



292

293

294

295 296

297

298

299

300

301 302

303 304

305 306

307

308

309 310

311

316

317

318

319 320

321

322

323

324

325

LoRaWAN® Multi Package Access Protocol Specification TS007-1.0.0-rc4

Field			
ricia	BaseByte	ANS buffer[BaseByte:BaseByte + X]	Command Token

Table 10: MultiPackBufferFrag payload

The BaseByte field encodes the index of the first byte of the buffer being transmitted. The BaseByte value is always less than or equal to 127, due to the limitation imposed on the maximum buffer size (128 bytes).

This field is followed by as many bytes of the ANS buffer as possible until one of the two following conditions is met:

- All (requested) bytes of the ANS buffer starting from byte BaseByte can fit in the payload.
- The maximum payload length is reached, at least another fragment will have to be sent to complete the transmission of the (requested) ANS buffer.

Note: The maximum payload length (MaxPayloadLen) depends on the LoRaWAN Data Rate used for transmission.

The transmission of the fragmented ANS buffer is aborted whenever a valid Multi-Package access command or command set (different from the *MultiPackBufferReq* command) is received.

- If a MultiPackBufferReq command is received, the device SHALL abort the previous fragmented transmission and starts transmitting the newly requested ANS buffer segment(s)
- If any other command is received, the device SHALL abort the previous fragmented transmission, clears the ANS buffer and starts processing the new command set.

Example:

Let's assume that following a multi-package command set with Token=3 the ANS buffer contains 20 bytes (noted ANS[0..19]). The maximum payload length that can be transported is 11 bytes. The end-device will transmit the following message sequence on FPort 225.

First message (11 bytes):

CommandID	BaseByte	Buffer bytes (8)	Command Token
0x02	0	ANS[07]	0x03

Table 11: MultiPackBufferFrag command example

Second message (11 bytes):

CommandID	BaseByte	Buffer bytes (8)	Command Token
0x02	8	ANS[815]	0x03

Table 12: MultiPackBufferFrag command example

Third & last message (7 bytes):

CommandID	BaseByte	Buffer bytes (4)	Command Token	
0x02	16	ANS[1619]	0x03	

Table 13: MultiPackBufferFrag command example



4.4 Multi-Package Buffer Command (MultiPackBufferReq)

The MultiPackBufferReq command payload is:

Size (bytes)	1	1
Field	StartByte	StopByte

Table 14: MultiPackBufferReg payload

When used, this command SHALL be the single command present in a Multi-Package access downlink. If the same downlink carries another command the end-device SHALL discard the full command set silently.

The Command Token field SHALL NOT be appended after a *MultiPackBufferReq* command.

With this command the Application Server instructs the end-device to retransmit the content of the ANS buffer starting with byte StartByte and ending with byte StopByte (included).

The valid range for StartByte is [0:BufferLength -1]. If StartByte is greater than BufferLength-1 the end-device responds with the error code.

If StopByte is greater than the bufferLength-1 the command is valid, the buffer is retransmitted starting from byte StartByte to the end, and no padding is added.

StopByte SHALL be greater or equal to StartByte. If StopByte is less than StartByte the end-device SHALL respond with the error code.

BufferLength indicates the number of bytes of the ANS buffer.

If StartByte and/or StopByte fields are invalid, the end-device responds to the *MultiPackBufferReq* command with the *MultiPackBufferFrag* with the following payload:

Size (octets) 1
Field Error=0xFF

Table 15: MultiPackBufferAns payload when request is invalid

Where the Error field value is 0xFF.

If StartByte and StopByte fields are valid, the end-device responds to the *MultiPackBufferReq* command with one or several *MultiPackBufferFrag*.

Example 1:

©2021 LoRa Alliance®

Let's assume that the ANS buffer is 13 bytes long (noted ANS[0..12]) and that the last command set received had a Token = 2.

The maximum applicative payload size that can be transported is 10 bytes and the end-device received the request on FPort 225.

CommandID	Command payload
0x02	0x01 0x05

Table 16: MultiPackBufferReg command example



The StartByte parameter is 1, StopByte is 5, therefore the end-device must respond with the following applicative payload sent on FPort 225:

CommandID	BaseByte	Command payload	Command Token
0x02	1	ANS[15]	0x02

Table 17: MultiPackBufferFrag example

The length of the applicative payload is 8 bytes, the payload contains:

- The command identifier (CommandID): 0x02 corresponding to MultiPackBufferFrag
- The BaseByte field which is the index of the first byte being transmitted
- The ANS buffer content, starting at byte 1 (because StartByte==1) and stopping at byte 5.

(Note: the byte numbering for the ANS buffer always starts at byte 0)

377378

379

380

381

382

383

384 385

386

387

388 389

390

366

367 368

369

370

371

372

373 374

375

376

Example 2:

The ANS buffer is still 13 bytes long. The maximum applicative payload size that can be transported is unchanged (10 bytes), the end-device received the request on FPort 225.

CommandID	Command payload	
0x02	0x01 0x0C	

Table 18: MultiPackBufferReq command example

The StartByte parameter is 1, StopByte is 12, therefore the end-device must respond with the following applicative payloads sent on FPort 225:

CommandID	BaseByte	Command payload	Command Token
0x02	1	ANS[17]	0x02
Table 19 : MultiPackBufferFrag example (first message)			

CommandID	BaseByte	Command payload	Command Token
0x02	8	ANS[812]	0x02

Table 20 : MultiPackBufferFrag example (second message)

The end-device automatically fragments the answer because it cannot fit in a single payload. The first fragment sent contains bytes ANS[1..7] because of the payload length limitation.

391 392

393

394

This mechanism allows the server to request the retransmission of any portion of the ANS buffer if the buffer could not be transmitted in a single uplink payload and a fragment has been lost.





396	5 Glos	sary
397		-
398	DR	Data Rate
399	EUI	Extended Unique Identifier
400	FM	Firmware Management
401	FUOTA	Firmware Update Over-The-Air
402	MSb	Most Significant bit
403	RFU	Reserved for Future Use
<i>4</i> 0 <i>4</i>		



6 Bibliography

405

406	6.1 References
407 408	[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCF 14, RFC 2119, March 1997
409 410	[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCF 14, RFC8174, May 1997
411	[TS001] LoRaWAN [™] Link Layer 1.0.4 Specification, LoRa Alliance, October 2020
412	[TS002] LoRaWAN® Backend Interfaces 1.0 Specification, LoRa Alliance, October 11, 2017
413 414	[TS003] LoRaWAN® Application Layer Clock Synchronization Specification TS003-2.0.0 LoRa Alliance, $\overline{\mbox{TBD}}$
415	[TS004] LoRaWAN [©] Fragmented Data Block Transport Specification TS004-2.0.0, <u>TBD</u>
416	[TS005] LoRaWAN® Remote Multicast Setup Specification TS005-2.0.0, LoRa Alliance, TBD
417 418	[TS006] LoRaWAN [©] Firmware Management Protocol Specification TS006-1.0.0, LoRa Alliance, to be published