LoRaWAN Version and Status Specification

Versioning

Version	Author	Description
0.1	A Mille	First draft
0.2	A Mille	Added *erase* spec; Added *string id* info; Corrections; Removing Annexes

Table of Contents

Table of Figures

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.

Abbreviations:

RFU: Reserved for Future Use

Introduction

This document proposes an application layer messaging package, running over LoRaWan to get information about an end-device status and availability. This is supposed to be used in a FUOTA context.

What is later called SLOT correspond to a memory storage area used to store a firmware image. This can also be used to store data received from a fragmentation session for example.

The various proposed commands allow the server to have more information about the status of the device and make decision about the FUOTA process.

All messages described in this document are transported as application layer messages. As such, all unicast messages are encrypted by the LoRaWan MAC layer using the and-device AppSKey.

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

Version & status Verification packages

Summary

The identifier of the Version and status package is 10. The version of this package is version 1.

The Following messages are sent to/from each end-device individually using Unicast uplink or downlink on the specified port. The default port value used is 111. These messages MUST NOT be sent in multicast. If messages are received on a multicast address, the end device MUST drop them silently.

CID	Command Name	Sender		Short Description
		End Device	Server	
0x00	PackageVersionReq		x	Used by the server of request to the device the version of the implemented package
0x00	PackageVersionAns	x		Conveys the answer to the PackageVersionReq
0x01	VersionRunningReq		x	Used by the server to ask for the running version of the end device
0x01	VersionRunningAns	x		Conveys the answer to the VersionRunningReq
0x02	VersionStoredReq		x	Used by the server to ask for the stored version of software the end device, and the status of slots
0x02	VersionStoredAns	x		Conveys the answer to the VersionStoredReq
0x03	SpaceStatusReq		x	Used by the server to ask for the space and memory availability on the end device
0x03	SpaceStatusAns	x		Conveys the answer to the SpaceStatusReq
0x04	UptimeReq		x	Used by the server to get uptime since (hard of soft) reboot from end device
0x04	UptimeAns	x		Conveys the answer to the SpaceStatusReq
0x05	EraseSlotReq		x	Used by the server to request the deletion of a stored fw
0x06	DeviceDescriptionReq		x	Used by the server to request String ID of device manufacturer, more features can be added
0x06	DeviceDescriptionAns	x		Conveys the answer to the DeviceDescriptionReq

Table 1 – Version & Status Messages summary

PackageVersionReq & Ans

The ***PackageVersionReq*** command has no payload but SHOULD trigger an answer.

The end-device answers with a ***PackageVersionAns***, using the following payload.

Field	*PackageIdentifier*	*PackageVersion*	*VersionInfo*
Size (bytes)	1	1	1

Table 2 – PackageVersionAns

PackageIdentifier uniquely identifies the package. For the "Version&Status package" this identifier is: 10.

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

VersionInfo

In this context *VersionInfo* is composed of:

VersionInfo Field	*Versioning Type*	*Slot Used*
Size (Bits)	4bits	4bits

Table 3 – VersionInfo fields

Versioning Type

In this context *Versioning Type* Values can be:

Value	*Signification*	*Structure*				*Description*
0x0	Not supported	/				/
0x1	Major/Minor Patch	RFU	Major	Minor	Patch	Commonly used versioning system encoded on bytes.
		uint8	uint8	uint8	uint8	
0x2	Sec Timestamp	uint32				Version stored as time since GPS Epoch
0x3 – 0x7	RFU	/				/

Table 4 – Versionning Type

And *Versioning Type* encodes on 4bits (as *uint4)*, the number of firmware slots prepared on the device to store firmware. If value is 0, SHOULD default to consider 3 slots. If otherwise it MUST be specified.

VersionRunningReq & Ans

The ***VersionRunningReq*** command has no payload but SHOULD trigger an answer.

The end-device answers with a ***VersionRunningAns***, using the following payload.

Field	*StatusInfo*	*RunningVersion*
Size (bytes)	1	4 (uint32)

Table 5 – VersionRunningAns

Where *StatusInfo* is some info about the current running setup, the slot of memory where is image is stored (encoded as *uint4*) for example:

StatusInfo Field	*RFU*	*Slot of running*
Size (Bits)	4bits	4bits

Table 6 – StatusInfo Field

And where *RunningVersion* is the current running version. The value can have multiple formats, as described in: *Versioning Type*. The server SHOULD have asked for the format beforehand, to be able to understand the format of the answer.

VersionStoredReq & Ans

The ***VersionStoredReq*** command has a one-byte payload. This is used to specify the expected size of the answer.

Field	*StoredInfoParam*
Size (bytes)	1

Table 7 – VersionStoredReq

Where *StoredInfoParam* contains some info about the expected number of slot info the answer MUST contains (*nbSlots*). If the value has not been asked before and or is known as 0, the end-device MUST return 3 slots information's.

StoredInfoParam Field	*RFU*	*nbSlots*
Size (Bits)	4bits	4bits

Table 8 – StatusInfoParam Field

The end-device answers with a ***VersionStoredAns***, using the following payload.

		(times nbSlots)
Field	*StoredInfoData*	*SlotVersion*
Size (bytes)	1	4

Table 9 – VersionStoredAnd

Where *StoredInfoData* contains info on the usage of slots firmware. This is composed as flag, the slot corresponding should have a bit-flag to 1, if the firmware stored is considered runnable, and the version make sense to read, 0 otherwise. If the flag is 0, no part of the answer is dedicated to this slot. The flags are used to inform the server of the slot corresponding to the info sent.

i.e.: if 3 slots have been requested, the length of the answer corresponds to 3 slots, but the only the slot at *value: 1* have their information given in order. So: $HammingWeight(SlotsFlags) <= nbSlots$

StoredInfoData Field	*Slot0*	*Slot1*	*Slot2*	*Slot3*	*Slot4*	*Slot5*	*Slot6*	*Slot7*
Size (Bits)	1	1	1	1	1	1	1	1

Table 10 – StoredInfoData Fields

And where *SlotVersion* is the version of the slot corresponding. The value can have multiple formats, as described in: *Versioning Type*. The server SHOULD have asked for the format beforehand, to be able to understand the format of the answer.

SpaceStatusReq & Ans

The ***SpaceStatusReq*** command has no payload.

The end-device answers with a ***SpaceStatusAns***, 8bytes payload, using the following structure:

Field	*Heap Available*	*Slot Size*
Size (bytes)	4 (uint32)	4 (uint32)

Table 11 – SpaceStatusAns

The *Heap Available* field contains number of bytes (encoded on an uint32) available at the time on the device. This can be used by the server to make decisions about the updates process. Value 0 means that the value could not be read. Value 1 is default to minimum value; the server SHOULD interpret as 0.

The *Slot Size* field contains the number of bytes (encode on a uint32) in one slot. This can be used by the server to extrapolate the size of each slot, and of the final firmware usable on device.

UptimeReq & Ans

The ***UptimeReq*** command has no payload.

The end-device answers with a ***UptimeAns***, 8bytes payload, using the following structure:

Field	*Current Uptime*
Size (bytes)	4 (uint32)

Table 12 – UptimeAns

The *Current Uptime* field contains number of sec (encoded on an uint32), since the last restart of the device.

EraseSlotReq

The ***EraseSlotReq*** command has a 1-byte payload.

Field	*StatusInfo*
Size (bytes)	1

Table 13 – EraseSlotReq

Where *StatusInfo* is some info about the requested slot to erase (encoded as *uint4*):

StatusInfo Field	*RFU*	*Slot Specified*
Size (Bits)	4bits	4bits

Table 14 – StatusInfo Field

The end-device is not expected to answer, if the server wants to validate the action, it can ask again the versions in the device slot.

DeviceDescriptionReq

The ***DeviceDescriptionReq*** command has a 1-byte payload.

Field	*InfoRequest*
Size (bytes)	1

Table 15 – DeviceDescriptionReq

Where *InfoRequest* is some info about the requested asked data, encoded as flags.

InfoRequest Field	*RFU*	*DescriptionFlags*
Size (Bits)	4bits	4bits

Table 16 – StatusInfo Field

DescriptionFlags Field	*RFU*	*Manufacturer ID*	*Device ID*
Size (Bits)	6bits	1bit	1bit

Table 17 – DescriptionFlags Fields

The end-device answers with a ***DeviceDescriptionAns***, which has fluctuating length payload. The first 2 bytes specify the answer content and length. The first byte is a direct answer to the request, InfoRequest, (a list of bit, flags of the content of the answer). The next bytes help know the length of the string received. For each flag at 1, and in order, a byte encodes as uint8 the length of the string in char in the answer.

The following bytes are the string encoded in char[]

		(times nbFlags (i))	
Field	*InfoRequest*	$String_i\ Size$	$String_i$
Size (bytes)	1	1 byte	$String_i\ Size$ Bytes

Table 18 – DeviceDescriptionReq

For example: “flags up, on 0b0011, means two strings are encrypted and are Manufacturer ID and Device ID :

InfoRequest	$String_1\ Size$	$String_1$	$String_2\ Size$	$String_2$
0x03 = 0b0011	0x04	“LTEK”	0x06	“FF1705”

Table 19 – Device Description example