ITSS Interface 2 Lite

Protocol Specification

Versions

Version	Date	Changes	Author
1.0-Draft	2020-08-14	Initial Draft	Lars Thiemann, Mario Köster, Simon Hügi, Wolfgang Windolf
1.0 Rev01	2020-09-04	Reviewed Version	Lars Thiemann, Mario Köster, Simon Hügi, Wolfgang Windolf
1.0 Rev02	2020-11-20	Fixed a typo	Lars Thiemann
1.0 Rev03	2020-11-26	Fixed formatting in some tables	Lars Thiemann
1.0 Rev04	2021-07-12	Changed order of Table 6 to comply with data order Specified tolerance of nwkFlarePeriod Specified FW Update restart when receiving FirmwareUpdateStart while update in progress Updated Profiles reference version Added comment column to Table 11 Fixed MAC frame format	Lars Thiemann, Simon Hügi
1.0 Rev05	2021-08-19	Disclaimer added	Simon Hügi

Notice of Use and Disclosure

This document and the information contained herein is provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY) AND THE ITSS PRACICE GROUP DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE ITSS PRACTICE GROUP OR ANY OF THEIR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS INFORMATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Without limiting the foregoing, the ITSS practice group makes no warranty that:

- the information will meet your requirements
- the information will be uninterrupted, timely, secure or error-free
- the results that may be obtained from the use of this information will be effective, accurate or reliable
- the quality of the information will meet your expectations
- any errors in the information will be corrected

The information made available within this document:

- could include technical or other mistakes, inaccuracies or typographical errors
- may be out of date and the ITSS practice group and its contributors make no commitment to update such materials
- is not intended to be used in safety relevant applications
- is not approved for use in safety-relevant applications

In no event shall the ITSS practice group or its contributors be liable to you or any third parties for any special, punitive, incidental, indirect or consequential damages of any kind, or any damages whatsoever, including, without limitation, those resulting from loss of use, data or profits, whether or not ITSS practice group or its contributors has been advised of the possibility of such damages, and on any theory of liability, arising out of or in connection with the use of the information contained herein.

Table of Contents

1 Introduction

1.1 General

1.1.1 Scope

The purpose of this document is to provide a description of the PHY and MAC requirements of 802.15.4 and to define the upper layers network and application including the security and firmware update mechanisms. It describes the internal functionalities of the higher layers and the interface to lower layers. Based on IEEE 802.15.4, this standard provides ultra-low complexity, ultra-low cost and ultra-low power consumption for low data rate radio devices.

1.1.2 References

IEEE802.15.4. (5. December 2003). IEEE Standard for Low-Rate Wireless Networks. *IEEE 802.15.4-2003*. IEEE.

ITSS-Interface 2 Specification - Profiles, Version 1.1. ITSS practice group.

2 Network Layer

2.1 General

2.1.1 Scope

The network layer described in this chapter extends the functionality of the MAC layer of the 802.15.4 specification (IEEE802.15.4, 2003) and specifies its usage and configuration to meet the requirements of the ITSS IF2 Lite.

2.2 Conventions

2.2.1 Number Formats

In this specification, hexadecimal numbers are prefixed with the designation “0x” and binary numbers are prefixed with the designation “0b”. All other numbers are assumed to be decimal unless indicated otherwise within the associated text.

2.2.2 Transmission Order

The frames in this specification are described as a sequence of fields in a specific order. All frame formats are depicted in the order in which they are transmitted by the PHY, from left to right, where the leftmost bit is transmitted first in time, or top to bottom, where the topmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are sent by the MAC in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits (little endian).

2.2.3 Reserved Values

Unless otherwise specified, all reserved fields appearing in a frame structure shall be set to zero on transmission and ignored upon reception. Reserved values appearing in multi-value fields shall not be used.

All reserved fields shall be set to 0 to avoid conflicts with future extensions.

2.2.4 Acronyms & Abbreviations

Abbreviation	**Description**
CCA	Clear channel assessment
CSMA/CA	Carrier sense multiple access with collision avoidance
MIC	Message integrity check
PAN	Personal area network

Table 1: Acronyms & abbreviations.

2.2.5 Terminology

There are two types of devices:

- The coordinator that operates as a PAN coordinator.
- The end device that shall neither operate as a PAN coordinator nor as a coordinator.

There are two directions of communication:

- Download: From the coordinator to an end device.
- Upload: From an end device to the coordinator.

There are two types of communication:

- Unicast: Transfer of data from one device to another device.
- Broadcast: Transfer of data from one device to all devices in the network, usually done by the coordinator.

2.3 *PHY Layer (IEEE 802.15.4)*

2.3.1 General

The requirements for the physical layer are listed in Table 2.

Parameter	Value	Comment
IEEE radio communication standard	802.15.4-2003	
Frequency	2.4 GHz	2400.0–2483.5 MHz
Used channel	11-26	
Data rate	250 kBit/s	
Modulation	O-QPSK	
Symbol rate	62.5 kBaud	Symbol duration = 16µs
Maximum TX power	10 dBm	Shall be as small as possible to reduce possible collisions with frames of devices from other PANs
Minimum RX sensitivity	-95 dBm	A worse sensitivity reduces the radio range
Maximum packet size	127 bytes	PDU size including all MAC header and trailer but without physical header 6 byte
Maximum allowed clock deviation	±20 ppm	over the temperature range -40°C … +85°C

Table 2: Requirements for the physical layer.

2.3.2 PHY Protocol Data Unit (PPDU)

2.3.2.1 PPDU Structure

Octets				
4	**1**	**1**		**≤127**
Preamble	Start-of-frame delimiter (SFD)	Frame length (7 bits)	Reserved (1 bit)	PHY service data unit (PSDU)
Synchronization header (SHR)		PHY header (PHR)		PHY Payload

Table 3: Format of the PPDU.

2.3.2.2 PPDU Duration

The dependency of transition time and PPDU frame length is shown in Figure 1.

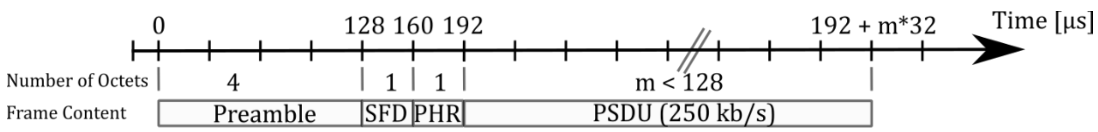


Figure 1: Timing figure for a PPDU frame.

A data rate of 250 kBit/s and a symbol rate of 62.5 kSymbols/s leads to 16µs per symbol. One octet consists of two symbols and will take 32µs for transmission.

The physical header is 6 octets in size; the maximum available size for a PSDU (MAC-Frame) is 127 octets in size which means the physical frame size can be up to 133 octets. Consequently, the transmission will take about 4.26 ms for a frame of maximum length.

2.4 MAC Layer (IEEE 802.15.4)

2.4.1 Clear Channel Assessment (CCA)

Clear channel assessment is a logical function determining the current state of use of a wireless medium. The algorithm determines if a channel is free or already used by another device by scanning the channel for a certain period of time.

For this specification, all devices that access the network in a contention access period (CAP) shall employ the unslotted CSMA/CA algorithm before any access to the wireless medium. A channel shall be observed for a duration of 8 symbols as defined in (IEEE802.15.4, 2003).

For this specification *macMinBE* and *macMaxBE* shall be used as shown in Table 4 (exceptions from using these default values are explicitly mentioned).

MAC Variable	Value	Comment
macMaxCsmaBackoffs	4	The maximum number of backoffs the CSMA/CA algorithm will attempt before declaring a channel access failure
macMinBE	3	The minimum value of the backoff exponent BE of the CSMA/CA algorithm
macMaxBE	5	The maximum value of the backoff exponent BE of the CSMA/CA algorithm

Table 4: CSMA/CA configuration.

2.4.2 MAC Data Frames

All communication is based on MAC data frames as specified in (IEEE802.15.4, 2003). Table 5 shows the format of a data frame. Since the different network frames differ in addressing and security, some fields vary in size.

Octets: 2	**1**	**2**	**2/8**	**0/2**	**8**	**var**	**2**
Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Data Payload	FCS
		Addressing Fields					
MAC header (MHR)						MAC Payload	MFR

Table 5: MAC data frame format.

2.4.2.1 MAC Header (MHR)

The MAC Header shall be configured as described in Table 6.

Field		**Description**
Frame Control	Frame Type	Set to MAC Data Frame (0b001)
	Security Enabled	When security is used, set to one (see chapter 3)
	Frame Pending	Always set to zero
	Ack. Request	Set to one for all frames except those with broadcast addressing
	Intra-PAN	Set to one when possible to suppress the *Source PAN ID* field
	Dest./Source Addressing Mode	Set depending on included addresses (see section 2.8.5)
	Reserved	Set to 0x00 to indicate IEEE 802.15.4-2003 used for backward compatibility
Sequence Number		Shall contain the current value of *macDSN*
Addressing fields		Set to included addresses (see section 2.8.5)

Table 6: MAC header configuration.

2.4.2.2 MAC Payload

The payload of a MAC data frame shall contain the sequence of octets that the network layer has assembled to transmit flare, data and join frames as described in the following.

If security is enabled, the MAC data payload has the format shown in Table 7. Refer to chapter 3 for further details.

Octets: 4	1	Variable	4
Frame counter	Key sequence counter	Encrypted payload	Encrypted integrity code

Table 7: MAC payload.

2.5 Network Architecture

Up to *nMaxNumberOfEndDevices* end devices are connected in a so called star topology to one coordinator, see Figure 2. Consequently, each end device is directly connected to one coordinator only. There is no connection between two end devices nor two coordinators. The coordinator is responsible for network organization like allowing end devices to join or leave its PAN or coordination of communication and security.

Different networks shall operate as independently as possible with respect to all networks operating in vicinity. This will be achieved by choosing different channels for the communication regions and by using different personal area network identifiers (PAN ID).

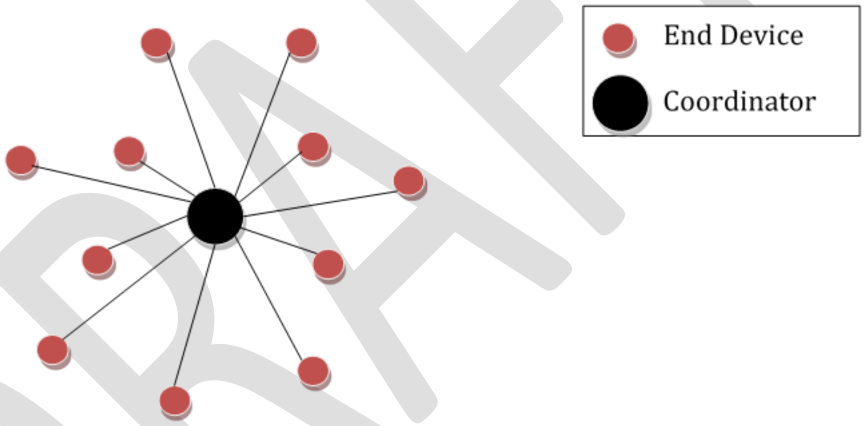


Figure 2: Star topology.

2.6 *Parameterization*

2.6.1 Constants

Name	Description	Value
nProtocolVersion	The version of the protocol	0
nMaxNumberOfEndDevices	Maximum number of supported end devices	15
nFlareChannel	Channel used to transmit flares	20
nFlaresPerSuperframe	Number of flares per superframe	8
nJoinWindowDuration	Duration of the join window	10 ms
nMaxMissedMainFlares	Maximum number of missed main flares until disconnection	5
nRegionOffset	Time difference between start of the flare period and start of the region, containing guard times and the join window	100 ms

Table 8: Constants of the network layer.

2.6.2 Attributes

Name	Description	Default
nwkFlarePeriod	Duration of a flare period (100ppm tolerance)	8 s
nwkSearchDurationMargin	Include this margin to be sure to reach one flare of a related coordinator when scanning, assuming very bad clock difference between coordinator and end device (12500 ppm)	100 ms
nwkMaxDataFramesPerUpload	Maximum number of data frames an end device is allowed to send per upload region	3
nwkMaxFrameRetries	Maximum number of retries per frame, until acknowledged or failed (if MAC ACK used) corresponding to *macMaxFrameRetries* from IEEE 802.15.4	3

Table 9: Attributes of the network layer.

2.7 *Superframe Structure*

Each coordinator maintains a superframe structure consisting of *nFlaresPerSuperframe* flare periods as shown in Figure 3. It sends flare frames every flare period of duration *nwkFlarePeriod*. The first flare of a superframe is called main flare while subsequent flares are called sub flares. This schedule is repeated after *nwkFlarePeriod * nFlaresPerSuperframe*.

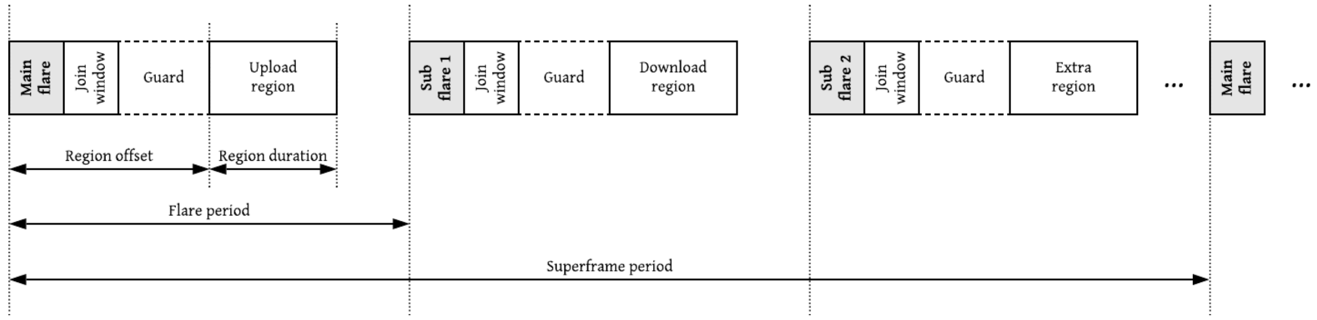


Figure 3: Superframe structure.

2.7.1 Flares

Flares are used to broadcast the presence of a coordinator and to keep the wagon network synchronized. Flares are sent without CSMA/CA in the fixed channel *nFlareChannel*.

Receiving flares enables end devices to determine the start of the next superframe after loss of tracking or to join a network during initial network setup (commissioning phase).

Each flare

- indicates which region is active in the current flare period and configures details like channel, duration and data access for end devices,
- indicates when the next flare is expected (the start of the next superframe / main flare is implicitly given by counting the remaining flares).

If a flare has been missed, the following region must be ignored since the channel is not known.

2.7.2 Main Flare

In addition to the previously mentioned flare contents, each main flare

- contains the current *SystemTime* of the coordinator, allowing the end devices to set their internal UTC clock,
- contains the *NetworkInformation* (metadata of the coordinator),
- indicates which regions are active for the current superframe.

2.7.3 Join Window

Each flare is followed by a join window of length *nJoinWindowDuration* as shown in Figure 4. It is available to end devices to send a *JoinRequest* or *RejoinRequest* to a coordinator and to coordinators to reply with a *JoinResponse*. The reason to have a join window in every flare period is to keep the implementation of the end device simple and to optimize its power consumption during scanning. This choice goes at the cost of an increased power consumption of the coordinator.

All frames in the join window are sent in the fixed channel *nFlareChannel*. The purpose of the join window is explained in section 2.9.

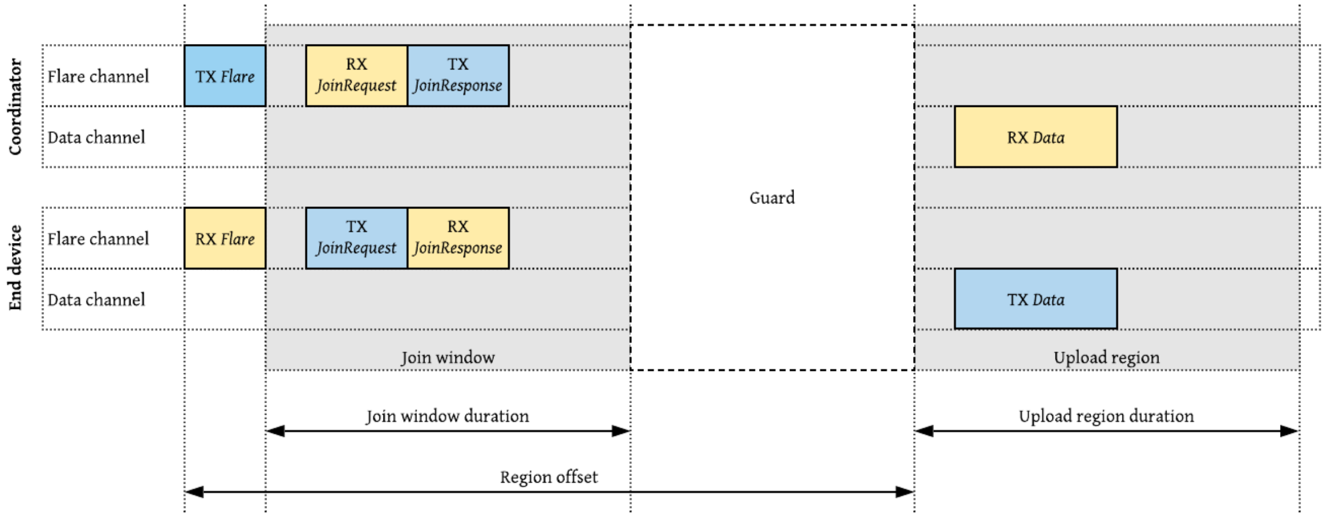


Figure 4: Example of an upload region with preceding join window.

2.7.4 Regions

A region starts after a guard time. The region's starting point is defined by *nRegionOffset* describing the difference between the start of the flare period and the start of the region.

The following region types are defined:

- Upload: Available to the end devices to communicate to their coordinator.
- Download: Available to the coordinator to communicate to its end devices.
- Extra: Available to the coordinator and end devices for fast alternating communication like firmware update.
- Empty: Nothing is done in this region, so the flare frame is only followed by the join window and all devices can skip this region.

It is up to the coordinator to configure the active regions of a superframe – they can be static or change over time. There must be at least one upload region per superframe. The preferred superframe structure is to have an upload region in the first flare period and a download region in the second flare period. The remaining flare periods shall be empty or configured as temporary extra regions during firmware updates. Having all empty regions at the end of a superframe allows the end devices to sleep for a long duration. Not following this preferred superframe structure can result in a higher power consumption and is therefore not recommended.

If the region is not empty, the *ChannelCode* field in the related flare determines the channel to be used for the communication between coordinator and end devices. This enables the coordinator to specify the desired data channel and to introduce some variation by a self-defined algorithm.

The *Duration* field specifies the duration of the active part of the region.

The coordinator is allowed to enter a low-power mode during the remaining time until the next flare period starts. An end device shall always wake up for main flares and for sub flares followed by a download or extra region. The subflare's regions are indicated by the main flare. The end device is allowed to skip sub flares and regions completely, if they are indicated as empty or upload region (while not having any data to send).

2.7.4.1 Upload Region

The upload region can be used by end devices wishing to send data to their coordinator as shown in Figure 3. An end device shall use CSMA/CA to reduce contention while competing with other end devices for sending data. An end device is only allowed to send up to *nwkMaxDataFramesPerUpload* in one upload region.

The field *UploadAllowed* in the preceding flare determines the end devices that are allowed to send data in the upload region. This can be used by the coordinator to distribute the end devices across several upload regions to reduce collisions. Every end device must be allowed to send data at least once per superframe.

2.7.4.2 Download Region

The download region is used by the coordinator to send data to its end devices. The coordinator shall perform CSMA/CA before transmission in order to minimize collisions with potential other PANs in the radio range (collision domain). For download regions, *macMinBE* shall be set to 0 since we assume that no other devices are competing to send at the region start.

The coordinator indicates in its flare if data is pending for an end device using the field *DataPending*. This allows an end device to immediately enter a low power state if there is no data for itself to receive.

2.7.4.3 Extra Region

The extra region can be used by both the coordinator and end devices to transfer data in alternation (ping-pong) for the duration of the region. This enables an exclusive communication between two devices for larger data transfers like firmware updates. Both devices need to perform CSMA/CA before transmitting messages in order to minimize collisions with messages of other PANs being potentially in the radio range (collision domain).

The coordinator indicates in its flare which end device is allowed to make use of this region using the field *DataPending*. This allows all other end devices to enter a low-power state immediately.

2.7.4.4 Empty Region

When no data should be transferred after a sub flare, empty regions shall be used. They are defined for completeness, as no data exchange takes place so devices can sleep during empty regions.

2.8 Frame Formats

The following section describes the structure of the different frame types.

2.8.1 General

This section describes the general frame format. Data types are one of the following:

- union (can only have anonymous name),
- struct,
- array (all corresponding to the well-known C language),
- U/S-8/16/32/64 integers (corresponding to unsigned/signed 8/16/32/64-bit integers, possibly combined with void boxing on the application layer).

Following rules apply:

- For bitfields, 1 means “true” and 0 means “false”, if not stated differently.
- The key word OPT means the field is optional and can be left out.
- The indentation level marked with “··” shown under name represents the nesting of the data structure.

2.8.1.1 Implementation Hint

All data structures need to be packed by the compiler (no alignment) and must use little-endian for number types, see section 2.2.2. When these conditions are technically met by the platform, the structure can simply be casted from a byte buffer using pointers (indirect addressing) and does not need to have a parser routine at all, saving valuable CPU cycles and thus energy. The optional and variable parts are either all at the end of the frame structure to left them out easily or are unified by unions. Hence, there is always a static space allocation for the complete structure possible, avoiding the need of a parser in a resource limited environment. This approach is followed in the following sub chapters.

2.8.1.2 NetworkFrameType

Name	Value (in Hex)	Description
Flare	0x0	A flare frame broadcasted by the coordinator
Join	0x1	A join frame with content from the network layer
Data	0x2	A data frame with content from the application layer
Reserved	0x3	Reserved

2.8.1.3 NetworkFrameStructure

Type	Octets	Name	Description
bitfield	1	FrameControl	Flags regarding the frame, structured into an octet
b2 b1 b0		··ProtocolVersion	Protocol version matching *nProtocolVersion*
b4 b3		··FrameType	Frame type, see *NetworkFrameType*
b7 b6 b5		··Reserved	Reserved
union		(anonymous)	Unification of the following structures by space, only one of the following structures can be used at a time
struct		··Flare	Flare frame, see section 2.8.2
struct		··Join	Join frame, see section 2.8.3
struct		··Data	Data frame, see section 2.8.3.6

2.8.2 Flare Frame

2.8.2.1 NetworkFlareType

Name	Value (in Hex)
MainFlare	0x0
SubFlare	0x1

2.8.2.2 NetworkRegionType

Name	Value (in Hex)
EmptyRegion	0x0
UploadRegion	0x1
DownloadRegion	0x2
ExtraRegion	0x3

2.8.2.3 *NetworkFlareStructure*

Type	Octets	Name	Description
bitfield	2	FlareControl	Flags of the current flare, structured into an octet
b0		··Type	Flare type, see *NetworkFlareType*
b3 b2 b1		··SubflareNumber	Counting the flares inside a superframe (0 Main, 1-7 Subs)
b5 b4		··RegionType	Region type, see *NetworkRegionType*
b8 b7 b6		··DeviceListRevision	Revision counter of the device list for identifying changes
b9 – b15		··Reserved	Reserved
U8	1	FlarePeriod	Length of the flare period in 1/8 second (125 ms resolution) [Default: 64 = 8 s]
struct		RegionConfig	Configuration of the region following the current flare, see *RegionConfig*
struct	OPT	NetworkConfig	Configuration of the current network, see *NetworkConfig*

2.8.2.4 *RegionConfigStructure*

Type	Octets	Name	Description
union	4	(anonymous)	Unification of the following structures by space, only one of them can be used at a time
struct	4	··EmptyRegion	Empty region for synchronization purposes
U16		····Empty1	Reserved
U16		····Empty2	Reserved
bitfield	4	··UploadRegion	Upload region for data transfer from end device to coordinator using CAP
b3 … b0		····ChannelCode	ChannelCode 0...15 + Offset 11 = Channel 11...26
b15 … b4		····Duration	Values 10-4095 [Unit 1ms]
b31 ... b16		····UploadAllowed	Bitfield indicating for each end device if upload is allowed; if the bit is set, the related end device with index described in section 2.8.3.5 is allowed to send
bitfield	4	··DownloadRegion	Download region for data transfer from coordinator to end device
b3 … b0		····ChannelCode	ChannelCode 0...15 + Offset 11 = Channel 11...26
b15 … b4		····Duration	Values 10-4095 [Unit 1ms]
b31 ... b16		····DataPending	Bitfield indicating for each end device if data from coordinator is pending; if the bit is set, data is pending for the related end device with index described in section 2.8.3.5
bitfield	4	··ExtraRegion	Extra region for data transfer between coordinator and end device (e.g., firmware update)
b3 … b0		····ChannelCode	ChannelCode 0...15 + Offset 11 = Channel 11...26
b15 … b4		····Duration	Values 10-4095 [Unit 1ms]
b31 ... b16		····DataPending	Bitfield indicating for each end device if data from coordinator is pending; if the bit is set, data is pending for the related end device with index described in section 2.8.3.5

2.8.2.5 *NetworkConfigStructure*

This structure is only present in main flares.

Type	Octets	Name	Description
U48	6	SystemTime	Coordinator’s UTC Unix system time in ms for synchronization of system time (time since 1970-01-01 00:00:00 UTC)
U8	1	NetworkInformation	Coordinator’s metadata, see section 2.8.6
U16	2	FlaresRegions	Region type for each of the 8 configurable flare periods, see *NetworkRegionType*

2.8.3 Join Frame

2.8.3.1 *NetworkJoinType*

Name	Value (in Hex)
JoinRequest	0x0
JoinResponse	0x1
RejoinRequest	0x2
Reserved	0x3 – 0xFF

2.8.3.2 *NetworkJoinStatusType*

Name	Value (in Hex)	Description
Accept	0x0	Accept a requested join
Reject	0x1	Reject a requested join

2.8.3.3 *JoinStructure*

Type	Octets	Name	Description
U8	1	Type	Join type, see *NetworkJoinType*
struct		Data	Data
union		··(anonymous)	Unification by space, can use only one
struct		····JoinRequest	See section 2.8.3.4
struct		····JoinResponse	See section 2.8.3.5
struct		····RejoinRequest	See section 2.8.3.6

2.8.3.4 *JoinRequest*

Sent in the join window by an end device to a coordinator intending to join a network.

Type	Octets	Name	Description
			This command has no further parameters

2.8.3.5 *JoinResponse*

Sent in the join window by a coordinator as a response to an end device requesting to join or rejoin its network.

Type	Octets	Name	Description
bitfield	1	Result	
b3 b2 b1 b0		··DeviceIndex	The coordinator assigns an index to each accepted end device. This index is used to determine the position in all array and bitfield data structures that are used as a list of end devices. Values 0 - 14
b4		··Status	Result of join request, see *NetworkJoinStatusType*
b7 b6 b5		··Reserved	

2.8.3.6 *RejoinRequest*

Sent in the join window by an end device to a coordinator intending to rejoin a network.

Type	Octets	Name	Description
			This command has no further parameters

2.8.4 Data Frame

2.8.4.1 *DataStructure*

Type	Octets	Name	Description
U8	1	PacketsPendingCount	End device: Indication of how many further packets are pending on the end device. Could be used by the coordinator to configure upload regions. Coordinator: Set to zero
U8	1	Length	Data length in bytes 0…92
array[0-92] of U8	0-92	Data	Data buffer

2.8.5 Addressing and PAN ID

For transferring data and join frames, the full 64-bit unicast device address is used as shown in Table 10.

Type	Value	Comment
Source PAN ID	0xNNNN (16 bits)	Derived from the 16-bit LSB part of EUI of coordinator
Destination PAN ID	equals “Source PAN ID”	Set PAN ID compression flag accordingly
Source address	0xNNNNNNNNNNNNNNNN (64 bits)	
Destination address	0xNNNNNNNNNNNNNNNN (64 bits)	

Table 10: Addressing and PAN ID for data and join frames.

For transferring flare frames, the 16-bit broadcast address shall be used as shown in Table 11. The destination PAN ID is set to the specific value in order to filter frames on PAN ID level. This avoids reception of frames from non-ITSS PANs.

Type	Value	Comment
Source PAN ID	0xNNNN (16 bits)	Derived from the 16-bit LSB part of EUI of coordinator
Destination PAN ID	0xFFF0 (16 bits) ITSS flare PAN ID	Set PAN ID compression flag accordingly
Source address	0xNNNNNNNNNNNNNNNN (64 bits)	
Destination address	0xFFFF (16 bits) broadcast address	

Table 11: Addressing and PAN ID for flare frames.

2.8.6 Network Information

The *NetworkInformation* is a bitfield containing meta information from the coordinator which is required by end devices to alter their measurement process.

Type	Octets	Name	Description
bitfield	1	NetworkInformation	
b0		··MovementState	0 = wagon standing 1 = wagon moving
b7 .. b1		·· Reserved	Reserved

2.9 Functional Description

2.9.1 Device List

For the following sections, it is assumed that a coordinator is active and sending flare frames to broadcast its presence. The coordinator of a network maintains an internal list of end devices that are part of its network. It is out of the scope of this specification how the coordinator gets this list.

To allow changes to this list and notify end devices, the coordinator maintains the field *DeviceListRevision* in its flares. Its value is incremented (and modulo wrapped) to signal end devices that they should rejoin. As long as the internal index positions of all devices do not change, the value does not need to be incremented. Consequently, there is usually no need to increment the value when adding a new end device since the other end device's indices do not change and new end devices are in scanning mode anyway. This also reduces the number of required rejoins, and thus, potential collisions. Removing an end device always requires incrementing the value to notify the removed end device.

All end devices detecting a changed value of *DeviceListRevision* in a received flare need to rejoin the network. Further details are outlined in the following sections.

2.9.2 Scanning Modes

Since the scanning process of an end device consumes a lot of energy, the scanning strategy in case of no found network is crucial.

One iteration of the scanning process lasts for about *nwkFlarePeriod* + *nwkSearchDurationMargin*. This gives an end device the chance to receive flares of potential coordinators in range.

It is recommended that the scanning process is repeated according to the following rules (every x seconds for y seconds for z times) which are applied successively. After one rule times out, the end device shall back off to the next rules scanning period.

The activation of an end device by a peripheral (e. g. magnet) causes it to start with mode 0 again.

Mode	Every	For	Times	Description
0	60 s	900 s	15	Fast scanning mode (commissioning phase)
1	1 h	6 h	6	Slow scanning mode
2	6 h	forever	-	Warehouse scanning mode

Table 12. Recommended scanning modes.

2.9.3 Joining a Network

This section describes the case when an end device is not yet associated with a coordinator so far, i.e., there is no relation to and no knowledge about a coordinator. Consequently, the end device needs to scan for possible coordinators.

An end device intending to join a network starts scanning for flares. Upon reception of a flare, the end device sends a *JoinRequest* to the related coordinator in the join window as shown in Figure 4. This is done with CSMA/CA and MAC layer acknowledgement enabled, as it is likely that other end devices are doing the same. The end device needs to listen for the response from the coordinator in the same join window. The end device repeats this process spanning many subsequent flares, until a join accept response has been received or a timeout as defined in section 2.9.2 occurs. End devices must not exclude a coordinator after receiving a rejected *JoinResponse* since the coordinator's device list may change after the initial request.

A coordinator receiving a *JoinRequest* decides whether the end device is allowed to join the network by responding with accept or reject in its *JoinResponse*. This is also done with CSMA/CA and MAC layer acknowledgement enabled.

The coordinator assigns a unique *DeviceIndex* to each accepted end device. This index is used to determine the position in all array and map data structures that are used as a list of end devices.

Upon a successful *JoinResponse*, the end device stores information associated with the network:

- The PAN ID of the network.
- The 64-bit MAC address of the coordinator.
- Its own associated *DeviceIndex*.

The end device starts tracking the coordinator's flares and has successfully joined the network.

2.9.4 Maintaining a Network

This section describes the case when an end device is associated with its coordinator. Consequently, the PAN ID and the source address of the coordinator are known.

2.9.4.1 Maintaining the Connection

Since an end device knows the address of its coordinator, it ignores flares from other coordinators during the scanning phase and keeps track of its coordinator's flares.

As soon as the end device receives a flare from its coordinator, it must check whether the *DeviceListRevision* has changed compared to the last known one. If it has changed, the end device needs to rejoin. Before sending the first *RejoinRequest*, each end device has to wait for *DeviceIndex* modulo *nFlaresPerSuperframe* frames by either sleeping for the corresponding time or counting the flares. This reduces the number of potential collisions in the join window. After this time, the join procedure using *RejoinRequest* frames as analogous to section 2.9.3 is executed. If the *RejoinRequest* is rejected, the end device has to follow the rules described in section 2.9.5.

On acceptance, the end device starts tracking the coordinator's flares again and has rejoined the network. The end device shall send periodic *ApplicationEndDeviceConnected* messages. This enables the coordinator to detect the loss of end devices.

2.9.4.2 Disconnecting from a Network

If an end device is not able to receive any main flare for *nMaxMissedMainFlares* superframes, it considers itself as disconnected from the network and starts the scanning phase as described above. This assures end devices can go to sleep mode even if not explicitly decommissioned by their coordinator.

2.9.5 Leaving a Network

Only the coordinator of a network is able to remove end devices from the network, by updating its internal device list accordingly and by incrementing (and modulo wrapping) the *DeviceListRevision* field in its flares. All end devices detecting a changed value of the *DeviceListRevision* need to rejoin the network by sending a new *RejoinRequest.*

An end device shall evaluate the *JoinResponse* in the following way:

- If the *JoinResponse* indicates that the request is accepted, the end device stays connected to its current coordinator.
- If the *JoinResponse* indicates that the request is rejected, the end device shall leave the network, delete all associated information about the coordinator, purge all configurations from the application layer, and enter the warehouse scanning mode.

2.9.5.1 Warehouse Scanning Mode

When entering the warehouse scanning mode while an end device has an associated coordinator, it shall leave the network and delete all associated information. The rationale of this behavior is to cover the case when a coordinator is removed while the end device is not connected.

In case the end device detects the same coordinator again, it repeats the process described in section 2.9.3 as it has lost the knowledge about this coordinator. However, the coordinator accepts the end device back into the network normally when a *JoinRequest* is made.

2.9.6 Data Transfer

Data transfer is done in the upload CAP regions for uploads and in the download or extra regions for downloads.

All data frames sent shall use acknowledged transmission of the MAC layer. If a frame is not acknowledged, it shall be retransmitted by the MAC layer in the same region for at most *nwkMaxFrameRetries*.

A successful CSMA/CA (performed in the MAC layer) is necessary before retransmission.

The application layer is responsible to handle unsuccessful transmissions – either resend later or discard data.

3 Security

3.1 General

Security of this specification is based on the MAC layer security using AES-128 based encryption and message integrity check (MIC) as defined in (IEEE 802.15.4, 2003).

3.2 Link Key

AES-128 encryption is based on a pre-shared 128-bit link key, i.e., a secret that is shared between an end device and its coordinator. The process of sharing this key is out of scope of this specification.

3.3 Security Suites

This specification uses two security suites of the MAC layer as specified in (IEEE 802.15.4, 2003):

- Security suite 0 disables all security features.
- Security suite 4 / AES-CCM-32 encrypts outgoing frames (data confidentiality) and uses a 4 byte MIC for data authenticity and replay protection.

The MAC layer decrypts and authenticates received frames if their security suite is set to 4. Frames with decryption or authentication error are silently discarded.

3.4 Flares

Flares are sent as intra PAN broadcasts and shall use the security suite 0. This removes the need of a network key and the key exchange procedure between coordinator and end devices willing to join a PAN.

3.5 Join Frames

Join requests are sent by end devices intending to join a network, while coordinators reply with join responses. All join request frames shall use the security suite 0. Join response frames used to reject an end device shall use the security suite 0, too. This is necessary since the coordinator might not know the end device and its related link key when receiving join requests.

Join response frames used to accept an end device shall use the security suite 4 for the following reason: In case the coordinator and the end device do not have the same link key, the join response would fail immediately and consequently, the end device would continue to scan for new networks as described in section 2.9.

3.6 Data Frames

Data frames are used for unicast communication between an end device and the coordinator. All data frames shall use the security suite 4. Received data frames having a different security suite than 4 shall be rejected.

4 Application Layer

4.1 General

This layer provides access to application-specific storage and transmission of measured values and process configurations over the network, using the underlying network layer. For communication upload and download regions shall be used.

Although the firmware update functions and its message formats are part of the application layer, they are described separately in chapter 5.

4.1.1 Terminology

An end device has up to *aMaxEndpoints* endpoints associated to it. An endpoint corresponds to a software driver controlling a sensing element inside or outside of the housing of an end device or some other measurement source. At the same time, it is an instance of a sensor profile that performs measurements.

4.1.2 Attributes

Attributes are single parameters which can be read or written (R/W) by the application layer and mandatory or optional (M/O). They are either configuration or measurement values of a profile. In this chapter, attributes are also called keys.

The first two bits (MSBs) of the keys (AttributeIds) carry a priority information used by profiles, see *ITSS-Interface 2 Specification - Profiles*.

Value (in Hex)	Priority
0x00	High priority
0x40	Low priority
0x80	Configuration

Table 13: Priority information of attributes.

4.2 Parameterization

4.2.1 Constants

Name	Description	Value
aKeepAlivePeriod	Number of superframes between two ApplicationEndDeviceConnected messages	30
aMaxEndpoints	Maximum number of endpoints per end device	8

Table 14: Constants of the application layer.

4.3 Services

The following services have been defined for application configuration and measurement (transfer directions included). They are transferred as application layer messages.

`ApplicationEndDeviceConnected()`	`End device -> Coordinator`
`ApplicationEndpointReportRequest()`	`Coordinator -> End device`
`ApplicationEndpointReportResponse(List(EndpointNr, ProfileId))`	`End device -> Coordinator`
`ApplicationEndpointStatusRequest(EndpointNr)`	`Coordinator -> End device`
`ApplicationEndpointStatusResponse(List(Parameters)`	`End device -> Coordinator`
`ApplicationEndpointConfigure(EndpointNr, List(Parameters))`	`Coordinator -> End device`
`ApplicationEndpointControl(List(EndpointNr, Active))`	`Coordinator -> End device`
`ApplicationEndpointMeasure(List(Values))`	`End device -> Coordinator`

4.4 Message Formats

The following structures describe the message format used by the application layer. Each application message type is encoded by a numerical value.

4.4.1 ApplicationMessageType

Name	Value (in Hex)	Description
ApplicationEndDeviceConnected	0x00	End device connected notification
ApplicationEndpointReportRequest	0x01	Endpoint report request
ApplicationEndpointReportResponse	0x02	Endpoint report response
ApplicationEndpointStatusRequest	0x03	Endpoint status request
ApplicationEndpointStatusResponse	0x04	Endpoint status response
ApplicationEndpointConfigure	0x05	Endpoint configure request
ApplicationEndpointControl	0x06	Endpoint control request
ApplicationEndpointMeasure	0x07	Endpoint measure notification
Reserved	0x08-0xEF	Reserved
FirmwareUpdateStart	0xF0	Firmware update start request
FirmwareBlockRequest	0xF1	Firmware update block request
FirmwareBlockResponse	0xF2	Firmware update block response
FirmwareUpdateFinished	0xF3	Firmware update finished notification
FirmwareUpdateAbort	0xF4	Firmware update abort request
Reserved	0xF5-0xFF	Reserved

4.4.2 ApplicationEndpointStatusType

Name	Value (in Hex)	Description
Inactive	0x00	Endpoint is inactive
Active	0x01	Endpoint is active
Reserved	0x02-0xFF	Reserved

4.4.3 ApplicationMessageStructure

Type	Octets	Name	Description
U8	1	MessageType	Message type, see *ApplicationMessageType*
struct		MessageData	Message data
union		··(anonymous)	Unification by space, can use only one
struct		····ApplicationEndDeviceConnected	See section 4.5.1
struct		····ApplicationEndpointReportRequest	See section 4.5.2
struct		····ApplicationEndpointReportResponse	See section 4.5.3
struct		····ApplicationEndpointStatusRequest	See section 4.5.4
struct		····ApplicationEndpointStatusResponse	See section 4.5.5
struct		····ApplicationEndpointConfigure	See section 4.5.6
struct		····ApplicationEndpointControl	See section 4.5.7
struct		····ApplicationEndpointMeasure	See section 4.5.8
struct		····FirmwareUpdateStart	See section 5.6.1
struct		····FirmwareBlockRequest	See section 5.6.2
struct		····FirmwareBlockResponse	See section 5.6.3
struct		····FirmwareUpdateFinished	See section 5.6.4
struct		····FirmwareUpdateAbort	See section 5.6.5

4.5 Commands

The following commands are packed inside a message of the application layer. All commands of this chapter shall be sent by the network layer in data frames using upload regions by end devices and download regions by the coordinator, see section 2.7.4 and section 2.8.3.6.

4.5.1 ApplicationEndDeviceConnected

An end device shall send this command to the coordinator once it has synchronized as described in section 2.9. After it has synchronized, the end device shall send this message every *aKeepAlivePeriod*.

Type	Octets	Name	Description
			This command has no further parameters

4.5.2 ApplicationEndpointReportRequest

A coordinator can ask an end device to return a full list of endpoints and their associated sensor profiles.

Type	Octets	Name	Description
			This command has no further parameters

4.5.3 ApplicationEndpointReportResponse

An end device responds with the following list of endpoints when requested by its coordinator.

Type	Octets	Name	Description
U8	1	Count	Number of elements in the following list
array[0-8] of struct	0-16	Endpoints	List of endpoints
U8		··EndpointNr	Number of the endpoint
U8		··ProfileId	Associated sensor profile ID

4.5.4 ApplicationEndpointStatusRequest

A coordinator can ask an end device to return a full list of parameters of the current configuration of one of its endpoints.

Type	Octets	Name	Description
U8	1	EndpointNr	Number of the endpoint

4.5.5 ApplicationEndpointStatusResponse

An end device responds with the following list of parameters when requested by its coordinator.

Type	Octets	Name	Description
U8	1	Count	Number of elements in the following list
list	var	Parameters	List of parameters
U8		··Key	The key of the parameter
depends on key		··Value	The value of the parameter of length given by key, see *ITSS-Interface 2 Specification - Profiles*

4.5.6 ApplicationEndpointConfigure

A coordinator can ask an end device to set a list of parameters for configuring one of its endpoints.

Type	Octets	Name	Description
U8	1	EndpointNr	Number of the endpoint
U8	1	Count	Number of elements in the following list
list	var	Parameters	List of parameters
U8		··Key	The key of the parameter
depends on key		··Value	The value of the parameter of length given by key, see *ITSS-Interface 2 Specification - Profiles*

4.5.7 ApplicationEndpointControl

A coordinator can ask an end device to activate or deactivate some or all of its endpoints which then start or stop measurement. This can also be used to start or stop the operation of all sensors in order to save energy. When activating an endpoint, the measurement shall start immediately.

Type	Octets	Name	Description
U8	1	Count	Number of elements in the following list
array[0-8] of struct	0-16	Endpoints	List of endpoints
U8		··EndpointNr	Number of the endpoint
U8		··Status	Status to be set, see *ApplicationEndpointStatusType*

4.5.8 ApplicationEndpointMeasure

An end device can report a list of measured parameters to a coordinator for one of its endpoints.

Type	Octets	Name	Description
U8	1	EndpointNr	Number of the endpoint
U8	1	Count	Number of elements in the following list
list	var	Parameters	List of parameters
U8		··Key	The key of the parameter
depends on key		··Value	The value of the parameter of length given by key, see *ITSS-Interface 2 Specification - Profiles*

4.6 Functional Description

After an end device has connected to a coordinator, it shall send an *ApplicationEndDeviceConnected* notification to inform about its availability. If the coordinator has no knowledge about the end device yet (due to a reset or a newly joined end device), it sends an *ApplicationEndpointReportRequest* to request information about the endpoints associated with the end device.

The coordinator may send several *ApplicationEndpointConfigure* messages, followed by a single message *ApplicationEndpointControl* to enable the desired endpoints.

The measurement process starts and the end device begins to report measurement data using *ApplicationEndpointMeasure* messages.

During operation, the coordinator can reconfigure certain endpoints of an end device by sending *ApplicationEndpointConfigure* messages.

Optionally, the coordinator can request *ApplicationEndpointStatusRequest* in advance to get an overview of the current end device's configuration. In order to stop or restart certain endpoints during operation, *ApplicationEndpointControl* can be sent accordingly.

4.6.1 Acknowledgement

The commands are acknowledged through the MAC layer, i.e., commands that have no explicit application response are expected to work if the MAC ACK has been successful.

4.6.2 Example

```
(coordinator (C) is running)
(join of end device (ED))
ApplicationEndDeviceConnected() [ED->C]
ApplicationEndpointReportRequest() [C->ED]
ApplicationEndpointReportResponse(List(EndpointNr, ProfileId)) [ED->C]
ApplicationEndpointConfigure(EndpointNr, List(Parameters)) [C->ED]
ApplicationEndpointControl(List(EndpointNr, Active=true)) [C->ED]

(reconnection of end device (ED))
ApplicationEndDeviceConnected() [ED->C]

(operation)
ApplicationEndpointMeasure(List(Values)) [ED->C]
ApplicationEndpointMeasure(List(Values)) [ED->C]
ApplicationEndpointMeasure(List(Values)) [ED->C]
...

(operation with reconfigure)
ApplicationEndpointStatusRequest(EndpointNr) [C->ED] {optional}
ApplicationEndpointStatusResponse(List(Parameters)) [ED->C] {optional}
ApplicationEndpointConfigure(EndpointNr, List(Parameters)) [C->ED]
ApplicationEndpointMeasure(EndpointNr, List(Values)) [ED->C]
ApplicationEndpointMeasure(EndpointNr, List(Values)) [ED->C]
ApplicationEndpointMeasure(EndpointNr, List(Values)) [ED->C]
...

(operation with disable/enable)
ApplicationEndpointControl(List(EndpointNr, Active=false)) [C->ED]
ApplicationEndpointControl(List(EndpointNr, Active=true)) [C->ED]
```

5 Firmware Update

5.1 General

This chapter describes the firmware update process which provides a standardized way to upgrade end devices in the network over the air. All communication for firmware updates shall be done in extra regions instead of download and upload regions.

5.2 Parameterization

5.2.1 Attributes

Name	Description	Default
fwuTimeout	Timeout to abort the firmware update	1800 s

Table 15: Attributes of the firmware update.

5.3 Metadata

5.3.1 Manufacturer ID

The ITSS manufacturer ID (5 byte String Value) is a manufacture specific code which is unique for each manufacturer. This ID consists of exactly 5 digits of printable ASCII codes from ‘0’ to ‘9’ and from ‘A’ to ‘Z’. Lower case letters or special characters are not allowed.

The ID shall be administered and assigned to each manufacturer from the ITSS practice group and shall not be generated by the manufacture itself.

The ID shall be identical to the ITSS_ManufacturerID used in ITSS Interface 1 communication. Please check the ITSS manufacturer ID document under http://www.innovative-freight-wagon.de/download-bereich/ section ITSS for application rules.

For development purposes, the ITSS has reserved the ID of C0FFE which can be used freely by any manufacturer during development. Productively used coordinators shall never indicate or serve OTA processes with this ID.

5.3.2 Image Variant

The manufacturer should assign an appropriate and unique image variant value to each of its end devices in order to distinguish the product. This is a manufacturer specific value.

5.3.3 Firmware Version

The firmware version of a firmware image shows the major and minor version and the build.

5.4 Services

The following services have been defined for the firmware update (transfer directions included). They are transferred as application layer messages in the extra region, see section 1.1.2.

`FirmwareUpdateStart(TransferId, ImageSize, MetaInfo)`
`FirmwareBlockRequest(TransferId, BlockNumber)`
`FirmwareBlockResponse(TransferId, BlockNumber, Data)`
`FirmwareUpdateFinished(TransferId, Status)`
`FirmwareUpdateAbort(TransferId)`
	`Coordinator -> End device`
	`End device -> Coordinator`
	`Coordinator -> End device`
	`End device -> Coordinator`
	`Coordinator -> End device`

5.5 Message Formats

The following structures describe the message format used by the firmware update. They are a subset of application layer messages as defined in section 4.4.3.

5.5.1 FirmwareStatusType

Name	Value (in Hex)	Description
Success	0x00	Success
FailManufacturer	0x01	Failed manufacturer
FailImageType	0x02	Failed image type
FailVersion	0x03	Failed version
FailChecksum	0x04	Failed checksum
FailGeneric	0x05	Generic failure
Reserved	0x06-0xFF	Reserved

5.6 Commands

The following commands are packed inside a message of the application layer. All commands of this chapter shall be sent by the network layer in data frames using extra regions, see section 2.7.4 and section 2.8.3.6.

5.6.1 FirmwareUpdateStart

Initiates the firmware update by sending the message from the coordinator to the end device. Contains a unique *TransferId*, shared throughout the whole session, *ImageSize* for the end device to know how much to request, and a *MetaInfo* structure with information like manufacturer, variant, version and checksum.

Type	Octets	Name	Description
U32	4	TransferId	Unique transfer ID used during the whole transfer for reference
U32	4	ImageSize	Total size of the firmware image in bytes
struct		MetaInfo	
array[5] of U8	5	··Manufacturer	ITSS manufacturer ID
U16	2	··Variant	Manufacturer specific and unique image variant
struct	4	··Version	Firmware version Major.Minor.BuildNumber
U8		····Major	
U8		····Minor	
U16		····BuildNumber	
U32	4	ImageChecksum	CRC-32 calculated over image (standard polynomial $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$)

5.6.2 FirmwareBlockRequest

The end device requests a block from the coordinator.

Type	Octets	Name	Description
U32	4	TransferId	Unique Transfer ID, used during the whole transfer for reference
U16	2	BlockNumber	Number of the requested block; FW image max 4 Mbyte

5.6.3 FirmwareBlockResponse

The coordinator responds to a request from the end device with one block of data.

If the last block does not contain 64 bytes, the coordinator shall fill the remaining data with 0xFF.

Type	Octets	Name	Description
U32	4	TransferId	Unique transfer ID used during the whole transfer for reference
U16	2	BlockNumber	Number of the transferred block
array[64] of U8	64	Data	The data of the 64 bytes firmware block

5.6.4 FirmwareUpdateFinished

The end device finishes the firmware update by reporting on the success / failure status.

Type	Octets	Name	Description
U32	4	TransferId	Unique transfer ID used during the whole transfer for reference
U8	1	Status	States if firmware update was successful or if an error occurred, see *FirmwareStatusType*

5.6.5 FirmwareUpdateAbort

The coordinator informs the end device to abort the firmware update process.

Type	Octets	Name	Description
U32	4	TransferId	Unique transfer ID, used during the whole transfer for reference

5.7 Functional Description

The firmware image can be different by vendor and sensor (hardware) type. The coordinator transfers the firmware update to its end devices separately and one after another. It is done by a straightforward concept using unicast addressing in the extra regions of a superframe. A coordinator can allocate several extra regions inside one superframe. To find a good compromise between total update duration and channel occupation, the recommended duration of extra regions is 500ms. If the end of a region is reached, the update is resumed in the next region. A fixed size of 64 bytes per block is used during the whole transfer.

The firmware update process consists of the following steps:

- It is started by the coordinator sending *FirmwareUpdateStart*. When receiving this message, the end device verifies the received metadata: Vendor (needs to match), variant (needs to match). If *FirmwareUpdateStart* is received while an update is already running, the currently running update is canceled and the new one is started.
- If the check is correct, the image transfer is executed by the end device sending *FirmwareBlockRequests* and the coordinator sending *FirmwareBlockResponses* in alternation.
- The processes is finished by the end device sending *FirmwareUpdateFinished* with the corresponding status as shown in section 5.5.1.

The commands are acknowledged through the MAC layer, there is no request + indication + response + confirm mechanism involved.

The end device shall abort the firmware update after *fwuTimeout* without *FirmwareBlockResponse.* The coordinator shall abort the firmware update after *fwuTimeout* without *FirmwareBlockRequest.*

If an end device has the requested firmware version already locally available as it may be the case when downgrading, it is allowed to skip the step of the image transfer.

5.7.1 Example Transfer

```
Extra Region:
	FirmwareUpdateStart(123, 1024, (Manufacturer, Version, ...)) [C->ED]

	FirmwareBlockRequest(123, 0) [ED->C]
	FirmwareBlockResponse(123, 0, ABCD...) [C->ED]

	FirmwareBlockRequest(123, 1) [ED->C]
	FirmwareBlockResponse(123, 1, EFGH...) [C->ED]

	...

	FirmwareBlocksRequest(123, 7) [ED->C]
	FirmwareBlockResponse(123, 7, JBHG...) [C->ED]

	(end of extra region)

Next Extra Region:
	FirmwareBlocksRequest(123, 8) [ED->C]
	FirmwareBlockResponse(123, 8, IJKL...) [C->ED]

	(further requests and responses...)
	(end of extra region)

(verify checksums)

Next Extra Region:
	FirmwareUpdateFinished(123, Success=true) [ED->C]
```

6 ANNEX

6.1 MAC Frame Lengths

Octets: 2	1	2	2/8	0/2	8	var	2
Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Data Payload	FCS
		Addressing Fields					
MAC header (MHR)						MAC Payload	MFR

6.1.1 Main Flare

MHR (inter PAN broadcast) 17 octets	MAC Payload (unencrypted) 16 octets	MFR 2 octets

6.1.2 Sub Flare

MHR (inter PAN broadcast) 17 octets	MAC Payload (unencrypted) 7 octets	MFR 2 octets

6.1.3 Join Request / Rejoin Request

MHR (intra PAN unicast) 21 octets	MAC Payload (unencrypted) 1 octets	MFR 2 octets

6.1.4 Join Response

MHR (intra PAN unicast) 21 octets	MAC Payload (unencrypted / encrypted) 2 / 11 octets	MFR 2 octets

6.1.5 Data Frame

MHR (intra PAN unicast) 21 octets	MAC Payload (unencrypted / encrypted) 12 … 104 octets	MFR 2 octets

6.2 Energy Consumption

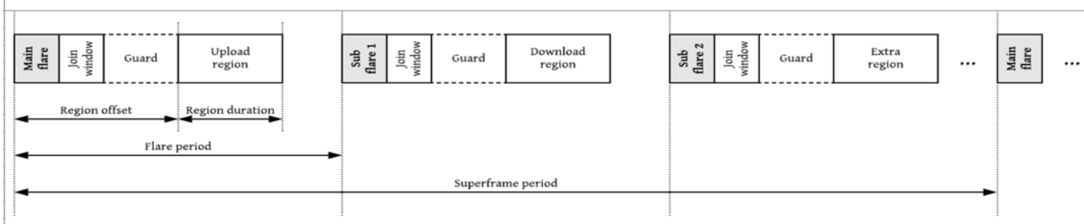
6.2.1 Coordinator

The indicative energy consumption of a coordinator employing the ITSS IF2 Lite protocol is based on the following assumptions:

- Power supply of transceiver used: 1.8 V
- Current consumption while sending frames at higher power regime: 12 mA
- Current consumption while listening or receiving frames: 6 mA
- Transceiver startup duration not considered, as these depend on the sleep mode employed
- Additional consumption caused by other onboard components not considered
- Only operational mode is considered because this covers about 99 % of the lifecycle

Calculation of energy consumption

ITSS IF2 Lite operational mode: Coordinator



ITSS IF2 Lite reference device (Coordinator)

current consumption at 1V8	Pulse [mA]	Duration [ms]	period [s]	Average Current [µA]	Consumption [mAh/day]	Consumption [mAh/year]	Comment
TX (send flare)	12	1,312	8	1,97	0,05	17	
RX to listen / receive in join window	6	10,000	8	7,50	0,18	66	joinResponse not considered
RX to listen / receive in upload region	6	150,000	64	14,06	0,34	123	actual length of region could be adopted to number of End Devices joint
TX in download region: one message per End Device	12	30,000	64	5,63	0,14	49	assumption: one configuration frame per Superframe per End Device
Total				**29,156**	**0,700**	**255**	

frame / region type	mode	octets	duration [ms]	Comment
flare	TX	41	1,312	phy header: 6 octets; MAC-header+FCS+data: 35 octets
join window	RX	n.a.	10	
guard	sleep	n.a.	90	sleep consumption not considered, as it is very device dependant
upload (15 end devices)	RX	n.a.	150	we give every End Device 10 ms to perform CSMA and send data
configuration data sent in download region	TX	50	2	duration of one frame includes ACK received from End Device

clock drift [ppm]
80

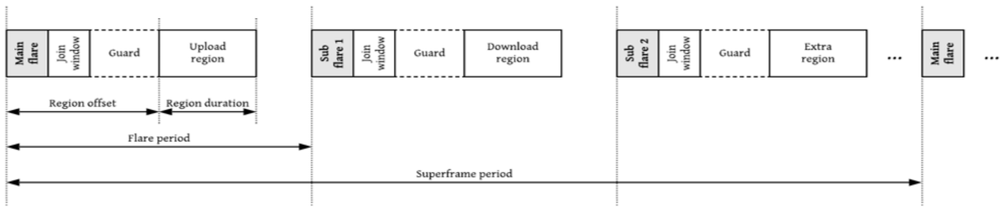
Figure 5: Indicative energy consumption of coordinator.

6.2.2 End Device

The indicative energy consumption of a typical end device employing the ITSS IF2 Lite protocol is based on the following assumptions:

- Power supply of transceiver used: 1.8 V
- Current consumption while sending frames at higher power regime: 12 mA
- Current consumption while listening or receiving frames: 6 mA
- Transceiver startup duration not considered, as these depend on the sleep mode employed
- Additional consumption caused by other onboard components not considered
- Additional consumption caused by the actual sensor element(s) not considered
- Only operational mode is considered because this covers about 99 % of the lifecycle

Calculation of energy consumption — ITSS IF2 Lite operational mode: End Device



ITSS IF2 Lite reference device (End Device) — clock drift [ppm]: 40

current consumption at 1V8	Pulse [mA]	Duration [ms]	Duration incl. drift [ms]	period [s]	Average Current [µA]	Consumption [mAh/day]	Consumption [mAh/year]	
TX (send data frame 127 bytes) + preamble + CSMA/CA + ACK	12	6	6	64	1,13	0,03	9,86	worst case: max data frame length used (127 octets)
RX (track Main flare)	6	2	7	64	0,67	0,02	5,85	
RX (track Sub flare)	6	2	3	64	0,25	0,01	2,17	assumption: End Device tracks one sub flare in average
RX (receive config data frame in download region)	6	30	31	960	0,19	0,00	1,70	assumption: configuration data available every 15 minutes frame size 60 octetcs; 2 ms / frame
RX (look up the coordinator) (4 times a day)	6	8100	8100	21600	2,25	0,05	19,71	even joined, the End Device shall "re-join" with Coordinator every 6 hours
Total					**4,48**	**0,11**	**39,28**	

overall allowed clock drift between two devices	drift [ppm]	period [s]	total drift [ms]
tracking of Main flare: End Device - Coordinator	**80**	64	5,12
tracking of sub flare: End Device - Coordinator	**80**	8	0,64

Figure 6 Indicative energy consumption of end device.

6.3 Commissioning Phase

The definition of the commissioning phase is currently under work and will be described in a separate document.

6.4 Decommissioning Phase

The definition of the decommissioning phase is currently under work and will be described in a separate document.