System protocol description

Revision history

Version	Date	Initials	Description
1.0.21	2010.11.29	JMH	Added Leave command
1.0.20	2010.11.23	LAJ	Transport protocol header updated.
1.0.19	2010.11.09	LAJ	Network and Store key bit size changed
1.0.18	2010.10.12	AKP + LAJ	Added device type to ESL status message and changed the status message sequence -Comment in SetNightMode (Table 2) updated -"Firmware version development increment" added to status message
1.0.17	2010.09.20	BPL	Changed Remote Firmware Upload message to a generic Remote File Upload message, and added a description of the file format.
1.0.16	2010.09.08	LAJ	-NumOfFreeBuffers in Table 9 updated. -Minor changes in message names in Table 9.
1.0.15	2010.09.07	LAJ	-In the status message (see Table 2) the label "Remaining hours" is changed to "Operating hours" and storage mode is removed, Rail alarm changed to tilt sensor. -The message SetStorageMode marked as obsolete (see Table 2). -Comment added to channel select (RequestActiveCh) (see Table 9) specifying that a returned 0x00 indicated no valid channel selected.
1.0.14	2010.08.31	LAJ	Error in HASH result corrected in section 7.1.5
1.0.13	2010.08.26	JMH	Added section 7.1.4 as an example encoding of an ESL data message. And section 7.1.5 to describe the calculation of the hash codes in the status message
1.0.12	2010.08.25	LAJ	DebugCmd added to Table 2, Hub2EslApplication protocol commands
1.0.11	2010.07.12	LAJ	GetFirmwareVer added to Table 9
1.0.10	2010.07.08	LAJ	SetTime added to Table 1
1.0.9	2010.06.30	JMH	Added new commands to the network controller in section 7.4.1: RequestActiveCh, ResponseActiveCh, RequestSetChMask, ResponseChMask, ScanProbeNotify and Reset
1.0.8	2010.06.11	HDM	- Remote Firmware Update description added.
1.0.7	2010.06.02	HDM HDM LAJ LAJ	- Frame size changed from 255 to 252 in section 7.4 and table 10. - Tidy up Id's in Table 9, Hub2Esl protocol commands. - NumFreeBuffers in Table 9, Hub2Esl protocol commands changed - Table 2, Hub2EslApplication protocol commands updated
1.0.6	2010.05.30	JMH	Added *Beacon order and Superframe order* to Scan probe response
1.0.5	2010.05.27	JMH+LAJ	Added ESL network control messages in section 7.6 - Table 9, Hub2Esl protocol commands updated
1.0.4	2010.05.14	LAJ	- Table 3, ESL data commands updated - Table 13, EslNetCtrl2Esl protocol commands updated - Minor update ex. 16 bit -> 2 bytes etc.

Version	Date	Initials	Description
1.0.3	2010.05.12	LAJ	- Doc updated according to phone meeting 11/5-2010 with Fabio
1.0.2	2010.05.10	LAJ	-Reserved bit added in Table 3 -Comment regarding virtual UART over USB added to chapter 7.5
1.0.1	2010.05.04	LAJ	-Req/Ack for TI_IEEE_EUI_64 -Added missing bit to EslStatus -Minor changes in Table 6, Hub2EslTransport protocol field description
1.0.0	2010.05.03	LAJ	More details
0.0.2	2010.04.19	LAJ	More details
0.0.1	2010.04.12	LAJ	Document created

Introduction

Purpose	Describe the protocols used in the system
Scope	
Limitation	
Usage	The system protocol description is part of the design phase in development of software.
Preconditions	
Responsible	Project manager

Contents:

1 Abbreviations

Abbreviation	Explanation
AES	Advanced Encryption Standard
CCM	Counter with CBC-MAC
ESL	Electronic Shelf Label
FCS	Frame Check Sum
LQI	Link Quality Indication
MIC	Message Integrity Code
HENM	Hub ESL network management
NA	Not available
PDU	Protocol Data Unit
SOP	Start Of Package
TBD	To be decided
UART	Universal Asynchronous Receiver-Transmitter
UC	Under construction

2 References

[R1] 802.11s (http://en.wikipedia.org/wiki/IEEE_802.11s)

[R2] JGroups (http://www.jgroups.org/)

[R3] Extensible Markup Language (XML) (http://www.w3.org/XML/)

[R4] Efficient XML Interchange (EXI) Format 1.0 (http://www.w3.org/TR/exi/)

[R5] Open-Mesh community (http://www.open-mesh.net/)

[R6] RabbitMQ (http://www.rabbitmq.com/)

[R7] CCM (http://en.wikipedia.org/wiki/CCM_mode)

[R8] Nonce (http://en.wikipedia.org/wiki/Cryptographic_nonce)

[R9] 802.15.4 standard (http://www.ieee802.org/15/pub/TG4.html)

[R10] Server MQ Message descriptions Server MG Messages on dev.s5tech.com

[R11] Server MQ XML Schema and examples XML Zip file on dev.s5tech.com

3 ESL system protocols

The ESL system is physical built up of a backend server, a HUB Gateway, a numbers of HUB's and ESL's (see Figure 1).

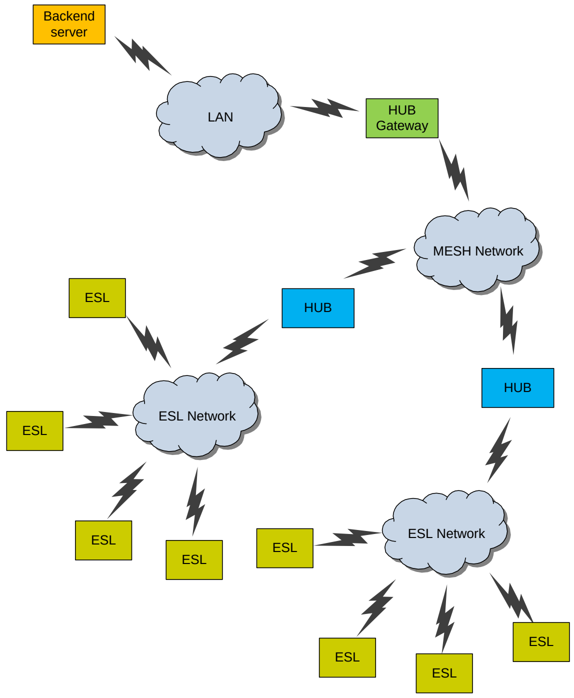


Figure 1, ESL system topology

On each item one or more applications are running to handle the functionality of the system. To support the communication among the applications a number of protocols are needed. Some of these protocols are available as open source, whereas others are designed especially to support the functionality in the ESL system. An overview of the communication layers used in the system is given in Figure 2.

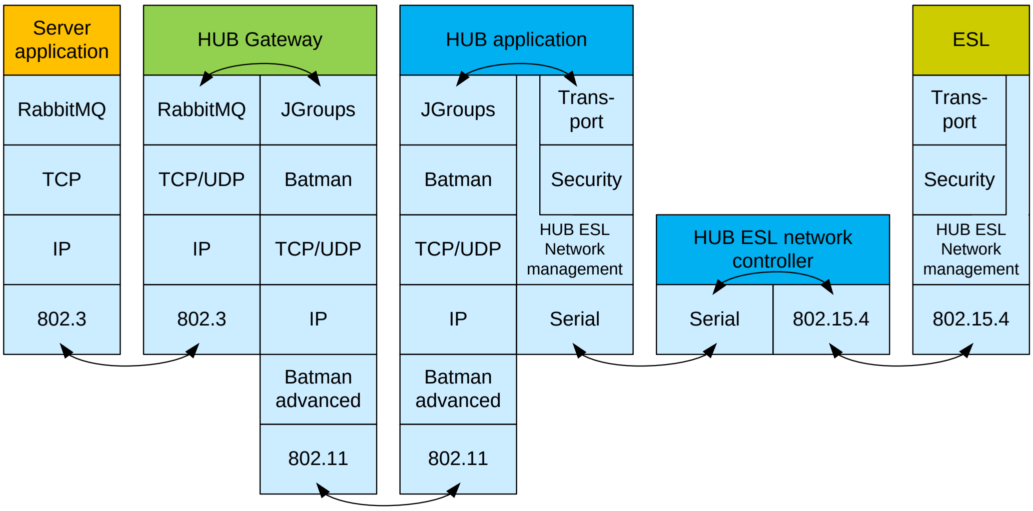


Figure 2, Communication overview for the ESL system

The protocols used to exchange data between the applications are described in the following chapters.

All numbers are codified as little-endian (intel). For example, the hex number 0x12345678 is codified as four bytes {0x78, 0x56, 0x34, 0x12}.

All bit fields are aligned high-to-low. For example, if the number of pages is 2 and the reserved field is 0, the value of the byte in the record is 0x20. The first nibble is the number of pages and is stored in bits 7-4, the second nibble is reserved and is stored in bits 3-0.

4 MESH network

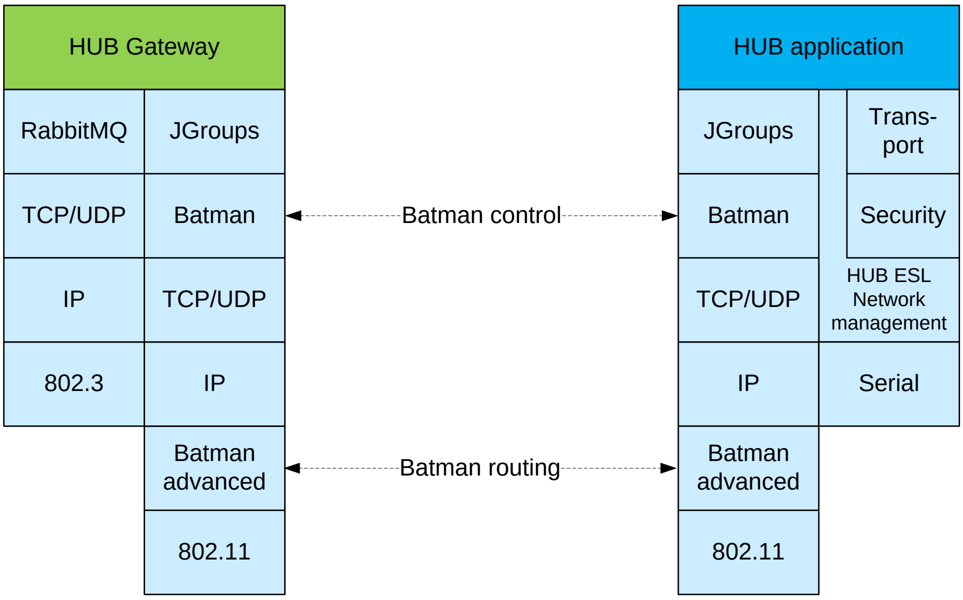


Figure 3, MESH protocol

4.1 Batman protocol

Batman is an open source protocol used for implementing MESH network on a Linux platform.

For more information about Batman please refer to [R5].

5 Server2Gateway protocols

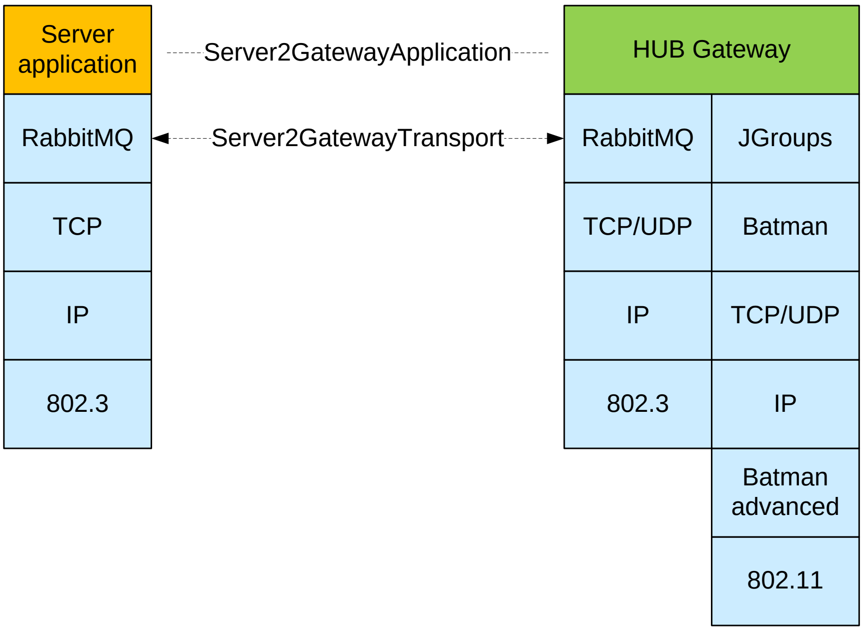


Figure 4, Server to HUB Gateway communication

5.1 Server2GatewayApplication protocol

The Server2GatewayApplication is based on xml messages sent via RabbitMQ.
See [R10] for description of the functionality and [R11] for the xml schema and message examples.

5.2 Server2GatewayTransport protocol

The Server2GatewayTransport protocol is based on the open source RabbitMQ. RabbitMQ is a complete and highly reliable enterprise messaging system based on the emerging AMQP standard. It is licensed under the open source Mozilla Public License and has a platform-neutral distribution, plus platform-specific packages and bundles for easy installation.

For more information about RabbitMQ please refer to [R6].

6 Gateway2Hub protocols

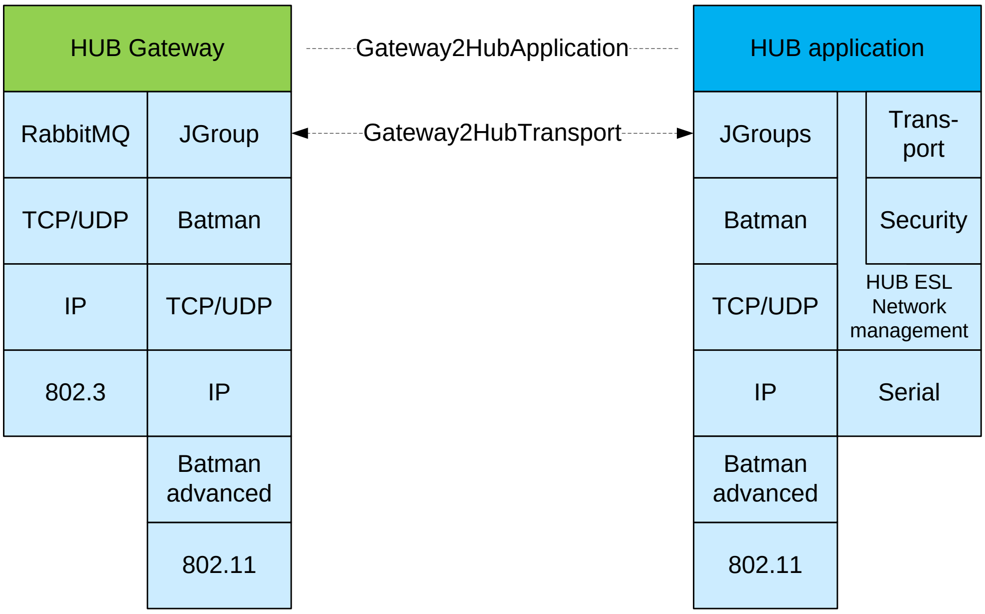


Figure 5, Gateway to HUB communication

6.1 Gateway2HubApplication protocol

This protocol consists of serialized java objects. The specific object to be transferred will be defined during development.
Some of the application protocol functionality will be based on JGroups, since it handles detection of new and missing nodes (hubs) and partly supports automatic data distribution.

6.2 Gateway2HubTransport protocol

Inter-hub communication is based on Jgroups which is an open source toolkit for reliable message communication.

Jgroups handles reliable delivery of messages as well as efficient bandwidth usage (by multicasting).

The detailed configuration of Jgroups will be done in conjunction with the configuration of the mesh network, as it's performance and bandwidth requirements are closely tied together.

For more information about Jgroups please refer to [R2].

7 Hub2Esl protocols

This chapter describes the communication stack used to exchange information between the HUB application and the ESL application. Figure 6, HUB to ESL communication shows the protocols identified.

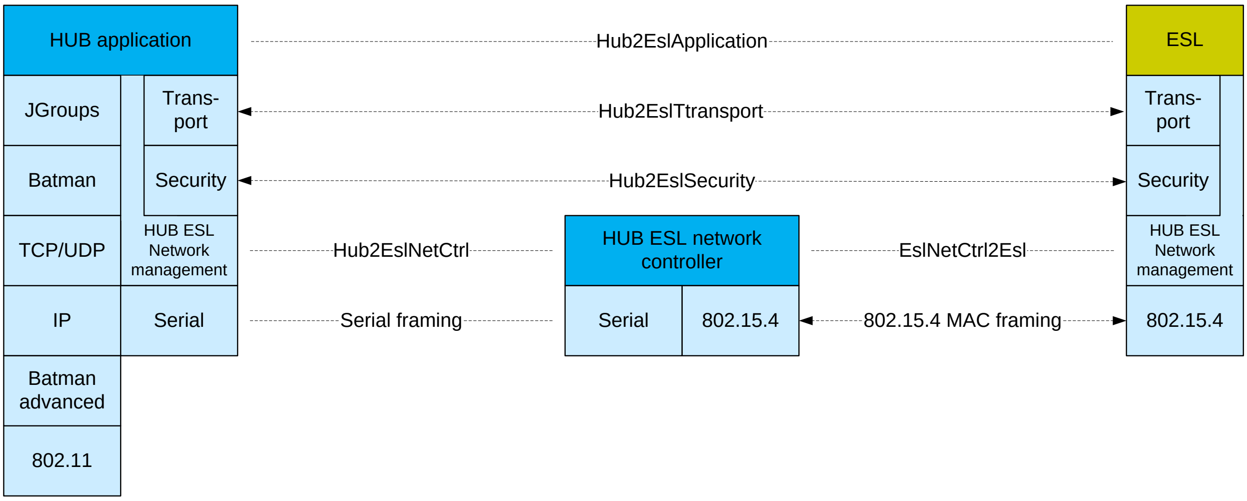


Figure 6, HUB to ESL communication

7.1 Hub2EslApplication protocol

The frame layout is shown in Figure 7, Hub2EslApplication frame structure. Table 1, Hub2EslApplication protocol field description explains the fields building up the frame and Table 2, Hub2EslApplication protocol commands describes the commands. There are no retransmissions in this layer.

Frame control	Command Id	App. PDU

Figure 7, Hub2EslApplication frame structure

Name	Field	Size	Description / Value(s)
Frame control	Protocol version	3 bit	
	Reserved	5 bit	
Command Id		1 byte	See Table 2
App. PDU		See Table 2	See Table 2

Table 1, Hub2EslApplication protocol field description

7.1.1 Commands

CMD		Structure	
Description	**Id**	**Size**	**Description / Value(s)**
Reserved	0x00		
EslData	0x01	See Table 3	Binary format (See Table 3)
EslStatus	0x02	1 byte	ESL device type
		1 byte	Firmware version major
		1 byte	Firmware version minor
		1 byte	Firmware version build
		1 byte	Firmware version development increment
		1 byte	Battery level
		1 byte	LQI for the associated HUB
		1 byte	Temperature
		2 bytes	Operating hours
		4 bytes	Hash code for Active price data (Refer to section 7.1.5 for ex.)
		4 bytes	Hash code for Pending price data (Refer to section 7.1.5 for ex.)
NB! *ESL must send a status before going into storage mode.*		1 bit	0 – Tilt sensor upside down 1 – Tilt sensor upside up
		1 bit	0 – Night mode off 1 – Night mode on
		2 bit	reserved
		4 bit	Num of alternate hubs in range
		0 – Num of alternate hubs in range	
		{	
		8 bytes	EUI64 of hub in range
		1 byte	LQI
		}	
SetNightMode	0x03	1 bit	0 – Night mode off 1 – Night mode on
		7 bit	Reserved
		4 bytes	Activation time in sec. Since 01012000
		4 bytes	Duration time in sec.
(obsolete) SetStorageMode	0x04	-	
N_KeyUpdate	0x05	16 bytes	Network key
S_KeyUpdate	0x06	16 bytes	Store key
UpdateFirmware	0x07		Binary format (see section Remote File Upload 7.1.6)
SetTime	0x08	4 byte	Time in sec. since 01012000
DebugCmd	0xFF	1 byte	Debug command id
		8 bytes	Raw data which can be used as wished, by the specific debug commands. See source code for the description of the individual debug commands.

Table 2, Hub2EslApplication protocol commands

7.1.2 ESL data commands

CMD		Structure	
Description	**Id**	**Size**	**Description / Value(s)**
Reserved	0x00		
SetLCD_PriceUpdate *NB!* *Num of pages <= 3* *NB!* *Page data size (X) depends on ESL type*	0x01	1 byte 4 bytes 4 bit 4 bit *0 – Num of pages* *{* 4 bit 4 bit X bytes *}*	ESL type (see Table 4 for assigned identification numbers) Activation time in sec. Since 01012000 Number of pages Reserved Page visible in duration of x sec. Reserved ESL data structures (see Table 4)

SetE_PaperPriceUpdate
 	
NB!
PNG image size (X) depends on image
	0x02	1 byte
		
		4 bytes
		1 byte
		{
		2 byte
		
		2 byte
		
		2 byte
		(X) bytes
		}
			ESL type (see Table 4 for assigned identification numbers)
			Activation time in sec. Since 01012000
			Number of images (max 3)
			
			X-position (zero-based 16 bit unsigned int)
			Y-position (zero-based 16 bit unsigned int)
			Image length in bytes (16 bit unsigned int)
			PNG image

Table 3, ESL data commands

7.1.3 ESL types

LCD		Structure	
Type	**Identification number**	**Size**	**Description**
DM0567YT	1	4 bytes	Price
		3 bytes	Free text
		1 bit	Dollar
		1 bit	Euro
		1 bit	Each

LCD		**Structure**	
Type	**Identification number**	**Size**	**Description**
		1 bit	PZ
		1 bit	OFFERTA
		1 bit	SCONTO
		1 bit	X
		1 bit	%
		1 bit	Sale
		7 bit	Reserved
DM0558YT DM0568YT	2 3	4 bytes	Price 1
		7 bytes	Free text 1
		3 bytes	Free text 2
		4 bytes	Price 2
		1 bit	Dollar
		1 bit	Euro
		1 bit	Each
		1 bit	PZ
		1 bit	X
		1 bit	%
		1 bit	Inverse Dot
		1 bit	SOTTOCOSTO
		1 bit	Dollar (small)
		1 bit	Euro (small)
		1 bit	MT
		1 bit	LT
		1 bit	KG
		1 bit	FT
		1 bit	LB
		1 bit	GAL
EG020AS012	4		

Table 4, ESL data structures

7.1.4 ESL data example encoding

To help clarify the byte and bit order of the various fields and example ESL data is constructed. This is example is also references in the section describing the hash code for the status message.

7.1.4.1 Example

CMD: 01 (SetLCD_PriceUpdate)
ESL Type: 03 (DM0568YT)
Activation time: 32 d6 08 14 (336123442 (2010-08-26T07:37:22))
Number of pages: 10 (one page)
Page 0:

- Page visible : f0 (visible in f seconds)
- Price 1: 49 21 00 00 (85.21)
- Text 1: 20 53 43 4f 4e 54 4f (“ SCONTO”)
- Text 2: 20 31 30 (" 10")
- Price 2: e6 00 00 00 (2.30)
- Icons: 50 48 (“Euro”, “PZ”, “Euro (Small)”, KG)

7.1.4.2 The resulting ESL data

01 03 32 d6 08 14 10 f0 49 21 00 00 20 53 43 4f 4e 54 4f 20 31 30 e6 00 00 00 50 48

7.1.5 Hash codes

The Status message described in Table 2 contains two hash codes. These hash codes are calculated using the Matayas Meyer Oases hash algorithm with AES 128 as the encryption engine.

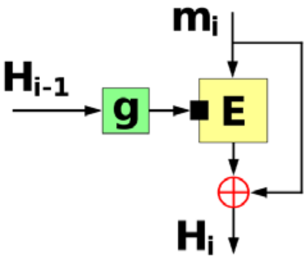


Figure 8 – Hash algorithm block diagram

This is that same algorithm specified for use in ZigBee protocols. And it was chosen because the chipset used on the ESL's are developed as a ZigBee chipset and therefore platform support SW for this algorithm is available.

The data to be hash is padded according to the following formula in order to ensure that the message is always of a length dividable to n 128 bit blocks

Pad (M) to obtain M' that is has a (length % block size == 0)

from ZigBee spec:
"Right-concatenate to the message M the binary consisting of the bit '1'
followed by k '0' bits, where k is the smallest non-negative solution to the
equation:
l+1+k 7n (mod 8n)"

7.1.5.1 Calculating the hash code for the LCD ESL types

The hash codes for LCD ESL's are calculated on the entire data structure contained in the ESL data message. Refer to Table 3, ESL data commands.

Starting from the CMD byte and including all the remaining bytes in the price update.

The following hash code is calculated from the example ESL data in section 7.1.4.2

Hash 128-bit result:
be f4 61 e9 e6 44 da 92 e2 0f b1 bc f5 7f 49 f7

The first 4 bytes of the hash codes is used in the status message:

be f4 61 e9 (3203686889 little endian decimal) (3915510974 big endian decimal)

7.1.5.2 Calculating the hash code for the Epaper ESL types

The hash codes for E-paper ESL's are calculated on the TBD parts of the data structure.

7.1.6 Remote File Upload

The ESL application will automatically receive a message when there is a new file ready in the flash file area (upper half of the address range). The message has no information about the uploaded file. The information must be extracted from the files header. The file header layout is shown in **Figure 9**. Table 5 explains the fields building up the header. There are no retransmissions in this layer.

FileType	File specific	Page start	File Size	Magic A	Magic B	CRC16 Content	CRC16 File	RLE flag	Esc Char	File data

Figure 9, Rfu frame structure

Name	Field	Size	Description
File Type		1 byte	0x03: Firmware 0x0C: Price update image
File Specific	File Type: Firmware	(6 bytes)	Data which is specific for the file type. Must always be present regardless of file type to keep the file header size constant at 18 bytes.
	Version Major	1 byte	
	Version Minor	1 byte	
	Version Maintenance	1 byte	
	Version Build	1 byte	
	Manufacturer ID	1 byte	
	Device Type	1 bytes	
	File Type: Picture	(6 bytes)	
	Reserved	6 bytes	
Page Start		1 byte	Start page in flash where file is stored.
File Size		2 bytes	Number of words (16 bit) in file (max size is 64K words or 128K bytes).
Magic A		1 byte	Signature for locating the file in flash.
Magic B		1 byte	
CRC16 Content		2 bytes	CRC of the decompressed file payload.
CRC16 File		2 bytes	CRC of file data.
RLE flag		1 byte	The byte has the value 'R' if the file is run length encoded (RLE). If the value is not 'R', then the file is plain binary.
Esc Char		1 byte	Escape character used in the file to mark places where the data has been RLE compressed.
File data		(File Size-18)*2 bytes	Array of bytes with file data

Table 5, Rfu protocol field description

The RLE format uses the RLE flag and ESC char to mark compressed areas of the file. The RLE decoder in the ESL will read the bytes sequentially from the start of the file. The following rules are obeyed:

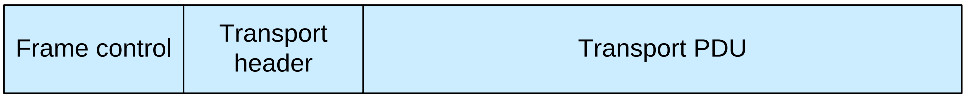
1) If the RLE flag is not 'R' then the file is not compressed. Copy all bytes as-is.
2) If the RLE flag is 'R', then use the following rules:

ESC, 0x00 -> ESC
ESC, Cnt, Val -> Val_1, ... Val_{Cnt}

a. If the current byte is equal to the Esc Char then read the next byte. If the next byte is 0x00, then copy the Esc Char to the output stream.
b. If the next byte is not 0x00, then use the byte value as a counter. Read the next byte.
c. Copy the next byte to the output stream the number of times the counter value specifies.

7.2 Hub2EslTransport protocol

The frame layout is shown in Figure 10. The size of the each frame is limited to 93 bytes. Table 6, Hub2EslTransport protocol field description explains the fields building up the frame. There are no retransmissions of packages in this layer.



Frame control	Transport header	Transport PDU

Figure 10, Hub2EslTransport frame structure

Name	Field(s)	Size	Description / Value(s)
Frame control	Protocol version	3 bit	
	Fragmentation enabled	1 bit	0 – Not fragmented 1 – Fragmented When not fragmented the transport header is not included.
	Sequence number	4 bit	Unique ID identifying a collection of frames.
Transport header	Total number of packages	2 bytes	Only available if fragmented. 16 bit unsigned int
	Current package number	2 bytes	Only available if fragmented. 16 bit unsigned int
Transport PDU		0 – 88 (92) bytes	88 – fragmented 92 – not fragmented

Table 6, Hub2EslTransport protocol field description

Transport layer is responsible for delivering data to the recipient in the upper layer. However, in some situation the Transport layer has to deliver a big chunk of data which that impossible to store with the amount of available RAM. In this situation the Transport layer has to store the chunk of data in flash memory. The chunk of data is composed of small fragments of data and will only be delivered when it is complete.

7.3 Hub2EslSecurity protocol

The frame layout is shown in Figure 11. The size of the each frame is limited to 102 bytes. Table 7, Hub2EslSecurity protocol field descriptionexplains the fields building up the frame. There are no retransmissions of packages in this layer.

Security control	Auxiliary Security header	Security PDU	MIC

Figure 11, Hub2EslSecurity frame structure

Name	Field	Size	Description/Value(s)
Security control	Protocol version	3 bit	
	Security / Encryption type	2 bit	0 – Security disabled Neither auxiliary Security header nor MIC is present in the Enhanced Frame Header. 1 – CCM 128 bit AES Auxiliary Security header and MIC are both present in the Enhanced Frame Header. As specified 2 – Reserved 3 – Reserved
	Reserved	3 bit	
Auxiliary Security header	Frame Counter	4 bytes	32 bit unsigned int
Security PDU		0 – 93 bytes	
MIC		4 bytes	

Table 7, Hub2EslSecurity protocol field description

7.3.1 Nonce

The nonce [R8] is constructed by concatenating the EUI-64 (MAC) address of the transmitting device with the Frame counter from the Auxiliary Security Header.

7.4 Hub2EslNetCtrl protocol

The frame layout is shown in Figure 12. The size of each frame is limited to 252 bytes. Table 8, Hub2EslCtrl protocol field description explains the fields building up the frame and Table 9, Hub2Esl protocol commands describes the commands. There are no retransmissions in this layer.

Frame control	Command Id	Ctrl. PDU

Figure 12, Hub2EslCtrl frame structure

Name	Field	Size	Description/Value(s)
Frame control	Protocol version	3 bit	
	Reserved	5 bit	
Command Id		1 byte	See Table 9
Ctrl. PDU		See Table 9	See Table 9

Table 8, Hub2EslCtrl protocol field description

7.4.1 Commands

CMD	Structure

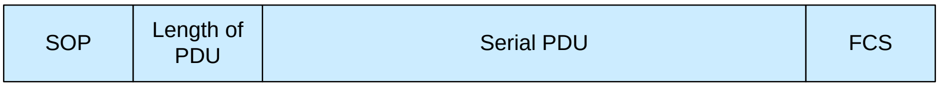
Description	Id	Size	Description / Value(s)
Reserved	0x00		
ValidateDeviceAuthorization	0x01	8 bytes	IEEE EUI-64
		1 byte	Capability information (802.15.4 specified, with the slight modification that the security bit will have the additional meaning that the device already has a store key.)
DeviceAuthorized	0x02	8 bytes	IEEE EUI-64
		2 bytes	Short network address (FFFF indicates error)
DataPending	0x03	2 bytes	ESL network address
		2 bytes	(unsigned int) Number of pending messages
PendingStatus	0x04	2 bytes	ESL network address Group Mask (ex. 0x001A)
ESL_Message	0x05	Options/Framecontrol {	
		3 bit	Version
		1 bit	16/64 bit addr
		1 bit	MAC Ack (enable retransmission)
		3 bit	Reserved
		}	
		2/8 bytes	ESL address
		0-102 bytes	PDU
NumOfFreeBuffers	0x06	1 byte	(unsigned int) number of free slots in broadcast queue
		1 byte	(unsigned int) number of free slots in pending data command queue
		1 byte	(unsigned int) number of free slots in the pending status command queue
		Data queue {	Repeated 14 times
		2 bytes	Short network address. 0x0000 indicates unassigned.
		1 byte	Free slots in data queue for specified short network address.
		}	
Req_IEEE_EUI	0x07		Request HENM IEEE EUI
IEEE_EUI	0x08	8 bytes	IEEE EUI-64 address of HENM
ReqActiveChannel	0x09	-	Request the current logical channel
ActiveChannel	0x0A	1 byte	Current logical channel Note! A value of 0x00 indicates no channel selected.
ReqSetChannelMask	0x0B	4 byte	Get/Set that Allowed channel mask 0x00000000 - read mask in device All others – set new mask (note only bits 11-26 are valid)
ChannelMask	0x0C	4 byte	Current channel mask Bit number indicates channel number
ScanProbeNotify	0x0D	8 bytes	EUI-64 of a device that just probed the coordinator (This is for production test)
ReqFirmwareVersion	0x0E	-	Request software version

CMD		**Structure**	
Description	**Id**	**Size**	**Description / Value(s)**
FirmwareVersion	0x0F	1 byte 1 byte 1 byte	Firmware version major Firmware version minor Firmware version build
ReqNumOfFreeBuffers	0x10		Request the current number of free buffers (slots), see CMD NumOfFreeBuffers.
SetTime	0x11	4 bytes	Set time in seconds
FirmwareUpdate	0x12	See 0x05	Same format as “ESL_Message (0x05) ” only this is intended for the coordinator
Leave	0xFD	2 bytes	Network address of the device that shall be asked to leave.
Reset	0xFE	-	Reset the coordinator

Table 9, Hub2Esl protocol commands

7.5 Serial framing

The frame layout is shown in Figure 13. The size of the each frame is limited to 258 bytes. Table 10, Serial communication parameters explains the fields building up the frame and Table 11, Serial protocol field description shows the serial communication set up parameters. There are no retransmissions in this layer. The interface is based on a virtual UART interface over USB.



SOP	Length of PDU	Serial PDU	FCS

Figure 13, Serial frame structure

Baud rate	*TBD*
Data	*8 bit/ char*
Parity	*No parity*
Stop bit	*1*

Table 10, Serial communication parameters

Name	**Field(s)**	**Size**	**Description / Value(s)**
SOP		1 byte	0x02
Length of PDU		1 byte	0 – 252
Serial PDU		0 – 252 bytes	Any value
FCS		1 byte	XOR value of each byte except SOP and FCS

Table 11, Serial protocol field description

7.6 EslNetCtrl2Esl

The frame layout is shown in Figure 14. The size of each frame is limited to 99 bytes. Table 12, EslNetCtrl2Esl protocol field description explains the fields building up the frame and Table 13, EslNetCtrl2Esl protocol commands describes the commands. There are no retransmissions in this layer.

Frame control	Command Id	PDU

Figure 14, EslNetCtrl2ESL frame structure

Name	**Field**	**Size**	**Description**
Frame control	Protocol version	3 bit	
	Reserved	1 bit	
	Frame type	4 bit	0 - Data other - CMD (See Table 13)
PDU		0 – 101 bytes	See Table 13

Table 12, EslNetCtrl2Esl protocol field description

7.6.1 Commands

CMD		**Structure**	
Description	**Id**	**Size**	**Description / Value(s)**
Reserved for data	0x00		
Time slot allocation	0x01	4 bit	Time slots for pending data (TSPD)
NB! *TSPD + TSSG <= 15*		4 bit	Time slots for status groups (TSSG)
		0 – TSPD of *{*	
		4 bit	Time slot
		4 bit	Superframe order (Duration)
		2 bytes	ESL network address

NB!
ESL network address Group Mask interpretation.
0x001A addresses the following ESLs: 0x01A0, 0x01A1, 0x01A2 … 0x01AE, 0x01AF
		}
		0 – TSSG of
		{	

		4 bit	Time slot
		4 bit	Reserved
		2 bytes	ESL network address Group Mask (ex. 0x001A)
		}	
Scan probe	0x02	*8 bytes*	*IEEE of ESL*
Scan probe response	0x03	*4 bytes*	*Symbols to next beacon*
		1 byte	*Link quality indication (of received Scan probe msg)*
		4 bit	*Beacon order*
		4 bit	*Superframe order*
Leave	0x04	*0 – no payload*	*ESL receiving this command from the coordinator shall leave the network and try to rejoin*

Table 13, EslNetCtrl2Esl protocol commands

7.7 802.15.4 MAC framing

Please refer to 802.15.4 standard [R9].