TITLE

PLC G3 MAC LAYER SPECIFICATION

TYPE	SPECIFICATION	**PROJECT**	PLC G3 OFDM

ABSTRACT

This document constitutes the specification for PLC communication based on OFDM modulation, and details the MAC and ADP (“Adaptation”) layers of the protocol stack.

[Page intentionally left blank]

Contents

Introduction

The present document constitutes the specification for PLC communication based on OFDM modulation, and details the MAC and ADP (“Adaptation”) layers of the protocol stack. It contains original clauses as well as clauses based on the documents

- 802.15.4-2006: “Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”
- IETF RFC4944: “Transmission of IPv6 Packets over IEEE 802.15.4 Networks”
- IETF draft-daniel-6lowpan-load-adhoc-routing-03: “6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)”
- IETF draft-6lowpan-commissioning-02: “Commissioning in 6LoWPAN”
- IETF RFC4919: “IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals”

Note:
This document is a draft and subject to change.

[Page intentionally left blank]

1 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

[802-2001] IEEE Computer Society, "IEEE 802-200&: IEEE Standard for Local and Metro Area Networks: Overview and Architecture", December 2001.

[802.15.4-2006] IEEE Computer Society, “IEEE 802.15.4-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)”, September 2006.

[rfc0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[rfc2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[rfc2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[rfc2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[rfc2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.

[rfc2865] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[rfc3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004

[rfc3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[rfc3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[rfc4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[rfc4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, January 2007.

[rfc4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[rfc4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[rfc4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[rfc4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[draft-load] Kim, K., Daniel Park, S., Montenegro, G., Yoo, S., Kushalnagar, N., "6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)", draft-daniel-6lowpan-load-adhoc-routing-03, June 2007

[draft-commissionning] Kim, K., Shams, S., Yoo, S., Park, S., Mulligan, G., "Commissioning in 6LoWPAN", draft-6lowpan-commissioning-02, July 2008

[draft-fragment] Thubert, P., "LoWPAN simple fragment Recovery", draft-thubert-6lowpan-simple-fragment-recovery-02, May 2008

[EUI64] "GUIDELINES FOR 64-BIT GLOBAL IDENTIFIER (EUI-64) REGISTRATION AUTHORITY", IEEE http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

[KW03] Karlof, Chris and Wagner, David, "Secure Routing in Sensor Networks: Attacks and Countermeasures", Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols vol 1, issues 2-3, September 2003.

Definitions, abbreviations and conventions

1.1 Definitions

For the purpose of the present document, the definitions given in the following clauses apply:

- Clause 3 of [802.15.4-2006]
- Clause 4 of [draft-load]
- Clause 2 of [draft-commissionning]

1.2 Abbreviations

For the purpose of the present document, the abbreviations given in the following clauses apply:

- Clause 4 of [802.15.4-2006]
- Clause 1.2 of [rfc4944]

1.3 Conventions

In the present document, the status of each requirement from the reference documents is given using the following convention:

- I = "Informative". The statements of the reference document are provided just for information.
- N = “Normative”: The statements of this part of the reference document must be taken as a reference for the present specification, without modifications or remarks.
- M = “Normative with Modifications”: The statements of this part of the reference document must be taken as a reference for the present specification, with the modifications and remarks noted under the part title.
- N/R = “Not Relevant”: The statements of this part of the reference document must be ignored for the present specification. An explanation may be given under the part title.
- A = “Additional”: The statements of this part are not present in the original reference document, and must be taken as a reference for the present specification.

2 MAC Layer

2.1 Services and Primitives

The MAC services and primitives are as given in clause 7.1 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 1 - Modifications and statements to clause 7.1 of the 802.15.4 standard [802.15.4-2006]

Clause	Title & remarks/modifications	Statement
7.1	MAC sublayer service specification	N
7.1.1	MAC data service - MCPS-PURGE primitives are not used in this specification.	M
7.1.1.1	MCPS-DATA.request	N
7.1.1.1.1	Semantics of the service primitive - Additional QualityOfService parameter: Integer ranging from 0x00 to 0x02; The QOS (Quality of Service) parameter of the MSDU to be transmitted by the MAC sublayer entity. This value can take one of the following values: 0 = Normal priority; 1 = High priority; 2 = Contention free; - Only nonbeacon-enabled PAN is used; - Bit b2 of TxOptions parameter must always be 0 See clause 7.5.1.1 in annex 5 of the present document for complete semantics description of this primitive.	M
7.1.1.1.2	Appropriate usage	N
7.1.1.1.3	Effect on receipt - GTS transmission is not used; - Only unslotted CSMA-CA for nonbeacon-enabled PAN is used; - Indirect transmission is not supported	M
7.1.1.2	MCPS-DATA.confirm	N
7.1.1.2.1	Semantics of the service primitive	N
7.1.1.2.2	When generated	N
7.1.1.2.3	Appropriate usage	N
7.1.1.3	MCPS-DATA.indication	N
7.1.1.3.1	Semantics of the service primitive - QualityOfService parameter is added; Integer ranging from 0x00 to 0x02; The QOS (Quality of Service) parameter of the MSDU to be transmitted by the MAC sublayer entity. This value can take one of the following values: 0 = Normal priority; 1 = High priority; 2 = Contention free; See clause 7.5.2.1 in annex 5 of the present document for complete description of the modified MCPS-DATA.indication primitive, including the additional QualityOfService parameter.	M
7.1.1.3.2	When generated	N
7.1.1.3.3	Appropriate usage	N
7.1.1.4	MCPS-PURGE.request - MCSP-PURGE.request is not handled in the present specification.	N/R
7.1.1.5	MCPS-PURGE.confirm - MCSP-PURGE.confirm is not handled in the present specification.	N/R
7.1.1.6	Data service message sequence chart	N
7.1.2	MAC management service	N

Clause	Title & remarks/modifications	Statement
7.1.3	Association primitives	N/R
7.1.3.1	MLME-ASSOCIATE.request - MLME-ASSOCIATE.request is not used in this specification. Association is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.3.2	MLME-ASSOCIATE.indication - MLME-ASSOCIATE.indication is not used in this specification. Association is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.3.3	MLME-ASSOCIATE.response - MLME-ASSOCIATE.response is not used in this specification. Association is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.3.4	MLME-ASSOCIATE.confirm - MLME-ASSOCIATE.confirm is not used in this specification. Association is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.3.5	Association message sequence chart - The association message sequence chart described in figure 31 must be ignored for this specification, as association is performed using the bootstrap mechanism described in [draft-commissioning].	N/R
7.1.4	Disassociation primitive	N/R
7.1.4.1	MLME-DISASSOCIATE.request - MLME-DISASSOCIATE.request is not used in this specification. Disassociation is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.4.2	MLME-DISASSOCIATE.indication - MLME-DISASSOCIATE.indication is not used in this specification. Disassociation is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.4.3	MLME-DISASSOCIATE.confirm - MLME-DISASSOCIATE.confirm is not used in this specification. Disassociation is performed by the LoWPAN Bootstrap Protocol described in clause 5.5 of the present document.	N/R
7.1.4.4	Disassociation message sequence chart - The disassociation message sequence chart described in figure 31 must be ignored for this specification, as disassociation is performed using the bootstrap mechanism described in [draft-commissioning].	N/R
7.1.5	Beacon notification primitive	N/R
7.1.5.1	MLME-BEACON-NOTIFY.indication - Only nonbeacon-enabled PANs are used. - This primitive is generated upon reception of a beacon during an active scan	M
7.1.6	Primitives for reading PIB attributes	N
7.1.6.1	MLME-GET.request	N
7.1.6.1.1	Semantics of the service primitive	N
7.1.6.1.2	Appropriate usage	N
7.1.6.1.3	Effect on receipt	N
7.1.6.2	MLME-GET.confirm	N
7.1.6.2.1	Semantics of the service primitive	N
7.1.6.2.2	When generated	N
7.1.6.2.3	Appropriate usage	N
7.1.7	GTS management primitives - GTS are not used in the present specification	N/R
7.1.8	Primitives for orphan notification - Beacon synchronization is not used in the present specification	N/R

Clause	Title & remarks/modifications	Statement
7.1.9	Primitives for resetting the MAC sublayer	N
7.1.9.1	MLME-RESET.request	N
7.1.9.1.1	Semantics of the service primitive	N
7.1.9.1.2	Appropriate usage	N
7.1.9.1.3	Effect on receipt	N
7.1.9.2	MLME-RESET.confirm	N
7.1.9.2.1	Semantics of the service primitive	N
7.1.9.2.2	When generated	N
7.1.9.2.3	Appropriate usage	N
7.1.10	Primitives for specifying the receiver enable time - The primitives for specifying the receiver enable time are not used in the present application of the norm. The receiver is always enabled	N/R
7.1.11	Primitives for channel scanning	N
7.1.11.1	MLME-SCAN.request	N
7.1.11.1.1	Semantics of the service primitive - The only supported values for the ScanType parameter is 0x01 for active scan. - The ScanChannels parameter is not used, and all of its 27 bits must be set to 0. - The ChannelPage parameter is not used and must always be set to 0. - The SecurityLevel must be 0. Thus the KeyIdMode, KeyIndex and KeySource parameters can be ignored and set to 0.	M
7.1.11.1.2	Appropriate usage - Only active scan is supported - ED scans, passive scans and orphan scans are not used. All devices must be capable of performing active scans.	M
7.1.11.1.3	Effect on receipt - Only active scan is supported - ED scan, passive scan and orphan scan are not supported. - There is no physical channel notion during the scans, as the underlying PHY layer does not support multiple channels.	M
7.1.11.2	MLME-SCAN.confirm	N
7.1.11.2.1	Semantics of the service primitive - The only supported values for the ScanType parameter is 0x01 for active scan. - The UnscannedChannels parameter is not used, and all of its 27 bits must be set to 0. - The ChannelPage parameter is not used and must always be set to 0. - The EnergyDetectList parameter is not used, and must always be null.	M
7.1.11.2.2	When generated - Only active scan is supported - ED scan, passive scan and orphan scan are not supported.	M
7.1.11.2.3	Appropriate usage	N
7.1.11.3	Channel scan message sequence chart - Figure 79 must be ignored (ED scan not supported) - Figure 82 must be ignored (passive scan not supported) - Figure 86 must be ignored (orphan scan not supported) - Active scan message sequence chart is specified in clause 4.7.2 of the present document, and replaces figure 83 of the reference document.	M
7.1.12	Communication status primitive	N
7.1.12.1	MLME-COMM-STATUS.indication	N
7.1.12.1.1	Semantics of the service primitive - Valid values for the status parameters are SUCCESS, CHANNEL_ACCESS_FAILURE, NO_ACK, COUNTER_ERROR, FRAME_TOO_LONG, IMPROPER_KEY_TYPE, IMPROPER_SECURITY_LEVEL, SECURITY_ERROR,	M

Clause	Title & remarks/modifications	Statement
	UNAVAILABLE_KEY, UNSUPPORTED_LEGACY, UNSUPPORTED_SECURITY or INVALID_PARAMETER - Additional valid values for the status parameter could be provided later	
7.1.12.1.2	When generated - This primitive is not used to notify the upper layer about association, disassociation, indirect transmission and transactions management	M
7.1.12.1.3	Appropriate usage	N
7.1.13	Primitives for writing PIB attributes	N
7.1.13.1	MLME-SET.request	N
7.1.13.1.1	Semantics of the service primitive	N
7.1.13.1.2	Appropriate usage	N
7.1.13.1.3	Effect on receipt	N
7.1.13.2	MLME-SET.confirm	N
7.1.13.2.1	Semantics of the service primitive	N
7.1.13.2.2	When generated	N
7.1.13.2.3	Appropriate usage	N
7.1.14	Primitives for updating the superframe configuration - This primitive is only used on the PAN coordinator in case of network formation (see clause 5.7.1 of the present document)	M
7.1.14.1	MLME-START.request - This primitive is only used to initiate a new PAN.	M
7.1.14.1.1	Semantics of the service primitive - Primitive parameters must be set as described in clause 5.7.1 of the present document.	M
7.1.14.1.2	Appropriate usage	N
7.1.14.1.3	Effect on receipt - Primitive parameters must be set as described in clause 5.7.1 of the present document.	M
7.1.14.2	MLME-START.confirm	N
7.1.14.2.1	Semantics of the service primitive	N
7.1.14.2.2	When generated	N
7.1.14.2.3	Appropriate usage	N
7.1.14.3	Message sequence chart for updating the superframe configuration - Figure 38 must be ignored.	N/R
7.1.15	Primitives for synchronizing with a coordinator - This part is used to inform the upper layers in case of a PAN ID conflict or PAN realignment.	M
7.1.15.1	MLME-SYNC.request	N/R
7.1.15.2	MLME-SYNC-LOSS.indication - PAN ID conflict detection is performed by the LoWPAN Bootstrap Protocol as described in clause 5.7.2 of the present document.	N/R
7.1.15.3	Message sequence chart for synchronizing with a coordinator - Synchronization with beacons is not used in the present specification	N/R
7.1.16	Primitives for requesting data from a coordinator - Indirect transmission and transactions are not supported by the present specification	N/R
7.1.17	MAC enumeration description	N

2.2 Frame Formats

The MAC frame formats are as given in clause 7.2 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 2 - Modifications and statements to clause 7.2 of the 802.15.4 standard [802.15.4-2006]

Clause	Title & remarks/modifications	Statement
7.2	MAC frame formats General MAC frame format - Segment Control fields are added to MHR (see Table 2.1) - Detailed description of the Segment Control fields is shown in the Table 2.2	N
7.2.1		M
7.2.1.1	Frame control field	N
7.2.1.1.1	Frame type subfield - The present specification does not use acknowledgement frame type value. The detailed ACK implementation is described in PHY specification associated with the present specification and annex 6 of the present document. An ACK can be sent by invoking the PD-ACK.request primitive.	M
7.2.1.1.2	Security Enabled subfield	N
7.2.1.1.3	Frame Pending subfield - Indirect transmission is not supported, so this bit must always be set to 0.	M
7.2.1.1.4	Acknowledgement Request subfield - The present specification translates the Acknowledgment Request subfield to the proper delimiter type of frame control header. The detailed ACK implementation is described in PHY specification associated with the present specification and annex 6 of the present document. An ACK can be sent by invoking the PD-ACK.request primitive.	M
7.2.1.1.5	PAN ID compression subfield	N
7.2.1.1.6	Destination Addressing Mode subfield	N
7.2.1.1.7	Frame Version subfield - These 2 bits are reserved for future use. In this version of the specification they must be set to 0.	M
7.2.1.1.8	Source Addressing Mode subfield	N
7.2.1.2	Sequence Number field	N
7.2.1.3	Destination PAN Identifier field	N
7.2.1.4	Destination Address field	N
7.2.1.5	Source PAN Identifier field	N
7.2.1.6	Source Address field	N
7.2.1.7	Auxiliary Security Header field - Possible lengths for the Auxiliary Security Header are 0 and 5 bytes (see clause 4.6 of the present document)	M
7.2.1.8	Frame Payload field	N
7.2.1.9	FCS field	N
7.2.2	Format of individual frame types	N
7.2.2.1	Beacon frame format	N
7.2.2.1.1	Beacon frame MHR fields	N
7.2.2.1.2	Superframe Specification field - Beacons are not transmitted at regular time intervals (beaconless network). Therefore the Beacon Order parameter of the Superframe Specification field is not used and must always be set to 0. - The receiver is active all the time when not transmitting. Therefore	M

Clause	Title & remarks/modifications	Statement
	the Superframe Order parameter of the Superframe Specification field is not used and must always be set to 0. - No superframe structure is used for communication, so the Final CAP Slot parameter of the Superframe Specification field is not used and must always be set to 0. - Devices will not be operating on batteries, so the Battery Life Extension subfield of the Superframe Specification field is not used and must always be set to 0. - In case of a 6LoWPAN PLC profile as described in the present specification, the association is performed by the LoWPAN Bootstrap Protocol in the upper layer, so the Association Permit parameter of the Superframe Specification field is meaningless here, and should always be set to 1. If another profile is used, this field should be set as described in clause 7.2.2.1.2 of [802.15.4-2004]	
7.2.2.1.3	GTS Specification field - The GTS Descriptor Count must always be set to 0 (GTS are not supported). - The PAN coordinator never accepts GTS request, therefore the GTS Permit parameter of the GTS Specification field must always be set to 0.	M
7.2.2.1.4	GTS Direction field - The GTS feature is not used, and the GTS Direction field must not be present in the frame	N/R
7.2.2.1.5	GTS List field - The GTS feature is not used, and considering the values of the GTS Specification field described in §7.2.2.1.3, this list must be empty	N/R
7.2.2.1.6	Pending Address specification field - Indirect transmission is not supported in this specification. Consequently, the Number of Short Addresses Pending is always 0, and Number of Extended Addresses Pending is also 0.	M
7.2.2.1.7	Address List field - Indirect transmission is not used, and this field must not be present in beacons.	N/R
7.2.2.1.8	Beacon Payload field - In the current version of this specification, the beacon payload field is empty.	M
7.2.2.2	Data frame format	N
7.2.2.2.1	Data frame MHR fields	N
7.2.2.2.2	Data payload field	N
7.2.2.3	Acknowledgement frame format - Acknowledgement frame format described in clause 7.2.2.3 of [802.15.4-2006] is not relevant; The detailed ACK implementation is described in PHY specification associated with the present specification and annex 6 of the present document. An ACK can be sent by invoking the PD-ACK.request primitive.	M
7.2.2.4	MAC command frame format	N
7.2.2.4.1	MAC command frame MHR fields	N
7.2.2.4.2	Command Frame Identifier field	N
7.2.2.4.3	Command Payload field	N
7.2.3	Frame compatibility - The use of the Frame Version subfield is reserved.	N/R

Table 2.1- General MAC frame format

Octets: 3	**2**	**1**	**0/2**	**0/2/8**	**0/2**	**0/2/8**	**0/5/6/10**	**Variable**	**2**
Segment Control	Frame Control	Sequence Number	Destination PAN	Destination Address	Source PAN	Source Address	Auxiliary Security Header	Frame payload	FCS
MHR								**MAC payload**	**MFR**

Table 2.2- Segment control fields

Field	Byte	Bit number	Bits	Definition
RES	0	7-4	4	Reserved
TMR	0	3	1	Tone map request 1: Tone map is requested 0: Tone map is not requested
CC	0	2	1	Contention Control: 0: contention is allowed in next contention state 1: contention free access
CAP	0	1	1	Channel access priority: 0: Normal 1: High
LSF	0	0	1	Last Segment Flag 0: Not last segment 1: Last segment
SC	1	7-2	6	Segment Count
SL[9-8]	1	1-0	2	Segment Length of MAC frame
SL[7-0]	2	7-0	8	Segment Length of MAC frame

2.3 Command Frames

The MAC command frames are as given in clause 7.3 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 3 - Modifications and statements to clause 7.3 of the 802.15.4 standard [802.15.4-2006]

<table>
<tr><th>Clause</th><th>Title & remarks/modifications</th><th>Statement</th></tr>
<tr><td>7.3</td><td>MAC command frames
- All devices are Full Function Devices
- The supported command list is:
<table>
<tr><th>Command frame identifier</th><th>Command name</th><th>Subclause</th></tr>
<tr><td>0x07</td><td>Beacon request</td><td>7.3.7 of [802.15.4-2006]</td></tr>
<tr><td>0x0a</td><td>Tone map response</td><td>4.3.1 of present document</td></tr>
<tr><td>0x0c–0xff</td><td>Reserved</td><td>—</td></tr>
</table></td><td>M</td></tr>
<tr><td>7.3.1</td><td>Association request command
- In case of a 6LoWPAN PLC profile, association is performed by the LoWPAN Bootstrap protocol described in clause 5.5 of the present document, so the clause 7.3.1 of [802.15.4-2004] is not relevant. If another PLC profile is used, the statement of this clause may change depending on the features of the upper layers.</td><td>N/R</td></tr>
<tr><td>7.3.2</td><td>Association response command
- In case of a 6LoWPAN PLC profile as described in the present specification, association is performed by the LoWPAN Bootstrap protocol described in clause 5.5 of the present document, so the clause 7.3.2 of [802.15.4-2004] is not relevant. If another PLC profile is used, the statement of this clause may change depending on the features of the upper layers.</td><td>N/R</td></tr>
<tr><td>7.3.3</td><td>Disassociation Notification command
- In case of a 6LoWPAN PLC profile as described in the present specification, disassociation is performed by the LoWPAN Bootstrap protocol described in clause 5.5 of the present document, so the clause 7.3.3 of [802.15.4-2004] is not relevant. If another PLC profile is used, the statement of this clause may change depending on the features of the upper layers.</td><td>N/R</td></tr>
<tr><td>7.3.4</td><td>Data Request command</td><td>N/R</td></tr>
<tr><td>7.3.5</td><td>PAN ID Conflict Notification command
PAN ID Conflict Notification is performed by the adaptation layer, see clause 5.7.2 of the present document</td><td>N/R</td></tr>
<tr><td>7.3.6</td><td>Orphan notification command
- Orphan notification is not used in the present specification</td><td>N/R</td></tr>
<tr><td>7.3.7</td><td>Beacon request command
- This command must be implemented in every device</td><td>M</td></tr>
<tr><td>7.3.8</td><td>Coordinator realignment command
- The coordinator realignment command is not used in the present notification</td><td>N/R</td></tr>
<tr><td>7.3.9</td><td>GTS request command
- GTS are not used in the present specification</td><td>N/R</td></tr>
</table>

2.3.1 Tone Map Response command

The MAC sublayer generates Tone Map Response command if Tone Map Request (TMR) bit of received packet Segment Control field is set. It means that a packet originator requested tone map information from destination device. The destination device has to estimate this particular communication link between two points and choose optimal PHY parameters. The tone map information includes the index associated with PHY parameters: number of used tones and allocation (Tone Map), modulation mode and TX power control parameters. The *Tone Map Response* message parameters are described in Table 3.1.

The channel estimation response command shall be formatted as illustrated in Figure 1.

Octets: ***(see 7.2.2.4 of [802.15.4-2006])***	***1***	***7***
MHR fields	Command frame identifier (see table in modified clause 7.3 of [802.15.4-2006])	Tone Map response payload

Figure 1 - Tone map response command format

Table 3.1- Tone Map Response payload

Field	Byte	Bit number	Bits	Definition
TXRES	0	7	1	Tx Gain resolution corresponding to one gain step. 0 : 6 dB 1 : 3 dB
TXGAIN	0	6-3	4	Desired Transmitter gain specifying how many gain steps are requested.
MOD	0	2-1	2	Modulation type: 0 – ROBO; 1 –DBPSK 2 - DQPSK
TM[8]	0	0	1	Tone Map [8]

TM[0:7]	1	7-0	8	Tone Map [7:0]
LQI	2	7-0	8	Link Quality Indicator
TXCOEF[3:0]	3	7-4	4	Specifies number of gain steps requested for 10kHz-20kHz spectrum
TXCOEF[7:4]	3	3-0	4	Specifies number of gain steps requested for 20kHz-30kHz spectrum
TXCOEF[11:8]	4	7-4	4	Specifies number of gain steps requested for 30kHz-40kHz spectrum
TXCOEF[15:12]	4	3-0	4	Specifies number of gain steps requested for 40kHz-50kHz spectrum
TXCOEF[19:16]	5	7-4	4	Specifies number of gain steps requested for 50kHz-60kHz spectrum
TXCOEF[23:20]	5	3-0	4	Specifies number of gain steps requested for 60kHz-70kHz spectrum
TXCOEF[27:24]	6	7-4	4	Specifies number of gain steps requested for 70kHz-80kHz spectrum
TXCOEF[31:28]	6	3-0	4	Specifies number of gain steps requested for 80kHz-90kHz spectrum

On reception of Tone Map Response command frame, the MAC layer updates the neighbor table with the corresponding Tone Map and communication parameters for that device. If no entry already exists in the table for that device a new entry should be added, based on implementation dependant limitations.

The neighbor table is defined in clause 4.5.1 of the present document.

The following procedure shall be used to perform the adaptive tone mapping function:

1. When a station is ready to transmit data it will first check if the neighbor table already has a record related to the destination device address. If the record is not exists or aged (Age counter is 0) the MAC sublayer sets TMR bit of outgoing packet Segment Control field and requests new Tone Map information. In this case the MAC data should be sent in ROBO mode.

2. If a neighbor table record exists and it is not aged the MAC sublayer does not need to send Tone *Map Request* message. In this case MAC sublayer uses an information from the neighbor table to properly configure PHY TX and construct Frame Control Header of outgoing frame.
3. When the destination station receives a data frame it shall check the *Tone Map Request* bit in the Segment Control field. If the bit is set, the destination station shall measure the per-tone quality of the channel, construct and send a *Tone Map Response* command message back to the originator station. The destination station shall not send a *Tone Map Response* message if the *Tone Map Request* bit is not set. The *Tone Map Response* message shall always be transmitted using default ROBO modulation. The destination device PHY uses parameters from the Frame Control Header to decode the MAC data fields.
4. The destination station shall attempt to send a *Tone Map Response* message as soon as possible after receiving a *Tone Map Request* message from the source station.
5. If the source station receives a *Tone Map Response* message it will update a neighbor table record related to the destination address with new Tone Map, modulation and TX gain parameters. If the record is not exists the MAC sublayer will create a new one. The Age counter should be set to desired value. After receiving a *Tone Map Response* message, a station shall begin to use the updated neighbor table information for all transmissions to the associated destination.
6. If the source station does not receive a *Tone Map Response* message after transmitting a *Tone Map Request* message to a certain destination, it shall set the *Tone Map Request* bit in the Segment Control of the next MAC data frame that it wants to transmit to the same destination. In other words, the MAC sublayer will continue to transmit a *Tone Map Request* message to the same destination.
7. The MAC sublayer shall not send a *Tone Map Request* message to the destination device if no data sent to this device.

The Tone Map request/response message sequence chart is shown in Figure 5 of the present document (clause 4.7.4)

2.4 Constants and PIB Attributes

The MAC constants and PIB attributes are as given in clause 7.4 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 4 - Modifications and statements to clause 7.4 of the 802.15.4 standard [802.15.4-2006]

Clause	Title & remarks/modifications	Statement
7.4	MAC constants and PIB attributes	N
7.4.1	MAC constants - The *aBaseSlotDuration* parameter is not used in the present specification and must be set to 0 - The *aBaseSuperframeConfiguration* parameter is not used in the present specification and must be set to 0 - The *aExtendedAddress* parameter must be equal to the EUI-48 address of the device mapped to a EUI-64 address. - The *aSymbolTime* parameter is the duration of one symbol. It is defined in PHY specification. - The *aSlotTime* parameter is the duration of contention window slot. Its value is 2 (symbols) - The *aCIFS* parameter is the contention interframe space. It is defined in annex 4 (clause 7.4.2) and value is 10 (symbols) - The *aRIFS* parameter is the response interframe space. It is defined in annex 4 (clause 7.4.2) and value is 10 (symbols) - The *aEIFS* parameter is the extended interframe space. It is defined in annex 4 (clause 7.4.2) and value is 252 (symbols) - The *aGTSDescPersistenceTime* parameter is not used and must be set to 0. - The *aMaxBeaconOverhead* parameter must be set to 0 - The *aMaxBeaconPayloadLength* parameter must be set to 0. - The *aMaxLostBeacons* parameter is not used and must be set to 0. - The *aMaxMACSafePayloadSize* parameter is not used and must be set to 0. - The *aMaxMACPayloadSize* parameter must be set to 400 (bytes) - The *aMinFrameSize* parameter is the minimum MAC frame size in symbols. It is described in PHY spec, and value is 4 (symbols) - The *aMaxFrameSize* parameter is the maximum MAC frame size in symbols. It is described in PHY spec, and value is 252 (symbols) - The *aMaxMPDUUnsecuredOverhead* parameter must be set to 25 (bytes). - The *aMaxSIFSFrameSize* parameter is not used in the present specification and must be set to 0. - The *aMinCAPLength* parameter is not used and must be set to 0. - The *aMinMPDUOverhead* parameter must be set to 9 (bytes) - The *aNumSuperframeSlots* parameter is not used in the present specification and must be set to 0 - The *aUnitBackoffPeriod* parameter must be set to *aSlotTime*. - The *aAckTime* parameter must be set to 23	M
7.4.2	MAC PIB attributes	M

Clause	Title & remarks/modifications	Statement
	- The *macAckWaitDuration* parameter must be set according to the following formula: *macAckWaitDuration = aRIFS + aAckTime + aCIFS* - The *macMaxFrameTotalWaitTime* parameter is not used. - The *macAssociatedPANCoord* parameter is not used and must be set to FALSE. - The *macAssociationPermit* parameter must always be set to TRUE, and must be read-only. - The *macAutoRequest* parameter is not used and must be set to FALSE. - The *macBattLifeExt* parameter is not used; changing it has no effect on the behavior of the device. Its default value must be FALSE, and must not be changed. - The *macBattLifeExtPeriods* parameter is not used; changing it has no effect on the behavior of the device. Its default value must be 0, and must not be changed. - The *macBeaconPayload* parameter is not used; changing it has no effect on the behavior of the device. Its default value must be NULL, and must not be changed. - The *macBeaconPayload* Length parameter is not used and must be set to 0. - The *macBeaconOrder* parameter is not used; changing it has no effect on the behavior of the device. Its default value must be left to 15, and must not be changed. - When the *macBeaconTxTime* parameter reaches 0xFFFFFF, it must not change anymore. - The *macGTSPermit* parameter is not used; changing it has no effect on the behavior of the device. Its default value must be FALSE, and must not be changed. - The *macMaxBE* parameter must be set to 5 - The *macMaxCSMABackoffs* default value is set to 8 - The *macMinBE* parameter must be set to 3 - The *macMinLIFSPeriod* parameter is not used; changing it has no effect on the behavior of the device. - The *macMinSIFSPeriod* parameter is not used; changing it has no effect on the behavior of the device. - The *macResponseWaitTime* parameter must be set to *macAckWaitDuration* - The *macRxOnWhenIdle* parameter must always be TRUE. - The *macSecurityEnabled* parameter must always be set to TRUE. - The *macShortAddress* parameter must be equal to 0xFFFF when the device does not have a short. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address. - The *macSuperframeOrder* parameter is not used, and must be left to 15. - The *macSyncSymbolOffset* is not used and must be set to 0. - The *macTimestampSupported* parameter must be set to TRUE. - The *macTransactionPersistenceTime* parameter is not used and must be set to 0	

Clause	Title & remarks/modifications	Statement

- Additional set of IB attributes:

macHighPrioirty-WindowSize	Integer	7	The high priority contention window size in number of slots; Default value is *7*aSlotTime*
macTxDataPacketCount	Integer	0x0–0xffffffff	Statistic counter of successfully transmitted MSDUs
macRxDataPacketCount	Integer	0x0–0xffffffff	Statistic counter of successfully received MSDUs
macTxCmdPacketCount	Integer	0x0–0xffffffff	Statistic counter of successfully transmitted command packets
macRxCmdPacketCount	Integer	0x0–0xffffffff	Statistic counter of successfully received command packets
macCSMAFailCount	Integer	0x0–0xffffffff	Statistic counter of failed CSMA transmit attempts
macCSMACollisionCount	Integer	0x0–0xffffffff	Statistic counter of collision due to channel busy or faile transmission
macBroadcastCount	Integer	0x0 –0xffffffff	Statistic counter of the number of broadcast frame sent
macMulticastCount	Integer	0x0 –0xffffffff	Statistic counter of the number of multicast frames sent
macBadCRCCount	Integer	0x0 -0xffffffff	Statistic counter of the number of frames received with bad CRC
macMaxOrphanTimer	Integer	0x0 –0xffffffff	The maximum number of seconds without communication with a particular device after whicl it is declared as an orphan.
macNeighborTable	Set	-	The neighbor table
macPanID	Integer	0x0 –0xffff	The PAN ID to which the device belongs
macNumberOfHops	Integer	0– 8	The number of hops to coordinator

macFreqNotching	Bool	FALSE	63 and 74 kHz frequency notching. Default value is FALSE (disabled)

Clause	Title & remarks/modifications	Statement
	List of MAC PIB attributes with their associated ID used in the current specification:	

macHighPrioirtyWindowSize	0x01000113
macTxDataPacketCount	0x02000101
macRxCmdPacketCount	0x02000202
macTxCmdPacketCount	0x02000201
macRxDataPacketCount	0x02000102
macCSMAFailCount	0x02000103
macCSMACollisionCount	0x02000104
macBroadcastCount	0x02000106
macMulticastCount	0x02000107
macBadCRCCount	0x02000108
macMaxOrphanTimer	0x02000109
macNeighborTable	0x1B000100
macAckWaitDuration	0x01000103
macAssociationPermit	0x01000102
macMaxCSMABackoffs	0x0100010B
macMinBE	0x0100010E
macShortAddress	0x01000112
macAssociatedPANCoord	0x01000104
macMaxBE	0x0100010A
macMaxFrameTotalWaitTime	0x0100010C
macResponseWaitTime	0x01000110
macSecurityEnabled	0x01000111
macPanId	0x0100010F

2.5 Functional Description

The MAC functional description is as given in clause 7.5 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 5 - Modifications and statements to clause 7.5 of the 802.15.4 standard [802.15.4-2006]

Clause	Title & remarks/modifications	Statement
7.5	MAC functional description	M
	- beacon-enabled PAN and GTS are not supported - contention free access is not implemented	
7.5.1	Channel access - See annex 4 of the present document for channel access functional description	M
7.5.1.1	Superframe structure	N/R
7.5.1.2	Incoming and outgoing frame structure	N/R
7.5.1.3	Interframe (IFS) spacing - See annex 4 of the present document for description of Interframe spacing	M
7.5.1.4	CSMA-CA algorithm - See annex 4 clauses 7.4.1 and 7.4.3 of the present document for description of CSMA-CA	M
	Priority (see annex 4 clause 7.4.4 of the present document)	A
	ARQ (see annex 4 clause 7.4.5 of the present document)	A
	Segmentation and reassembly overview (see annex 4 clause 7.4.6 of the present document)	A
7.5.2	Starting and maintaining PANs	N
7.5.2.1	Scanning through channels - Passive scanning is not supported - Orphan scanning is not supported - ED scanning is not supported - Active scanning is the only supported scanning mode - There is no channel page or channel list notion at the PHY level. Consequently a scan request does not care about a particular channel.	M
7.5.2.1.1	ED channel scan - ED channel scan is not supported in this specification	N/R
7.5.2.1.2	Active channel scan - Active channel scan is only used by an unassociated device prior to starting association and by the PAN coordinator prior to starting a new network. - There is no channel page or channel list notion at the PHY level. Consequently a scan request does not care about a particular physical channel.	M
7.5.2.1.3	Passive channel scan - Passive channel scan is not supported in this specification	N/R
7.5.2.1.4	Orphan channel scan - Orphan channel scan is not supported in this specification	N/R
7.5.2.2	PAN identifier conflict resolution	N
7.5.2.2.1	Detection - PAN conflict detection is also performed by scanning all incoming PAN Id of frames received by the devices.	M
7.5.2.2.2	Resolution - On detection of a PAN identifier conflict, a device must generate a CONFLICT frame as described in clause 5.7.2 of the present document.	N/R

Clause	Title & remarks/modifications	Statement
7.5.2.3	Starting and realigning a PAN	N
7.5.2.3.1	Starting a PAN - The only type of device allowed to be a PAN coordinator is a Data Concentrator. A Data Concentrator is always a PAN coordinator, and cannot become a coordinator or a RFD. - A PAN coordinator cannot lose its MAC address. It can however be changed based on criteria out of the scope of this document, for example in case of PAN ID conflict detection.	M
7.5.2.3.2	Realigning a PAN - PAN realignment is not supported by the present specification.	N/R
7.5.2.3.3	Realignment in a PAN - PAN realignment is not supported by the present specification.	N/R
7.5.2.3.4	Updating superframe configuration and channel PIB attributes - The macBeaconOrder parameter must always be set to 15 to have a beaconless PAN - The phyCurrentPage and phyCurrentChannel parameters are not used, and must always be set to 0.	M
7.5.2.4	Beacon generation - Only nonbeacon-enabled PAN are used - Beacon must be transmitted using the RoBo modulation	M
7.5.2.5	Device discovery - Device discovery is done using the active scanning procedure described in 5.5.2.1, to force a coordinator to send a beacon.	M
7.5.3	Association and disassociation	N
7.5.3.1	Association - Association is fully described in clause 5.5 of the present document.	N/R
7.5.3.2	Disassociation - Disassociation is fully described in clause 5.5 of the present document.	N/R
7.5.4	Synchronization	N
7.5.4.1	Synchronization with beacons - Beacon synchronization is not used in the present specification	N/R
7.5.4.2	Synchronization without beacons	N
7.5.4.3	Orphaned device realignment - Orphaned device realignment is not used in the present specification. - Orphaned device detection is performed at the application level using a timer, which is reset each time the device receives a frame with the Destination Address field of the MAC header equal to the MAC address (either short or extended) of the device. If this timer reaches its maximum value (*macMaxOrphanTimer*), then the device loses its short MAC address, and must begin an association procedure.	M
7.5.5	Transaction handling - Transactions are not supported in the present specification.	N/R
7.5.6	Transmission, reception and acknowledgement	N
7.5.6.1	Transmission	N
7.5.6.2	Reception and rejection	N
7.5.6.3	Extracting pending data from a coordinator	N/R
7.5.6.4	Use of acknowledgements and retransmissions	N
7.5.6.4.1	No acknowledgement	N
7.5.6.4.2	Acknowledgement - The present specification implements an acknowledgement differently. The detailed ACK implementation is described in PHY specification associated with the present specification and annex 6 of the present document. An ACK can be sent by invoking the PD-	M

Clause	Title & remarks/modifications	Statement
	ACK.request primitive.	
7.5.6.4.3	Retransmissions	N
7.5.6.5	Promiscuous mode	N
7.5.6.6	Transmission scenario	N
7.5.7	GTS allocation and management - GTS are not used in the present specification	N/R
7.5.8	Frame security	N
7.5.8.1	Security related MAC PIB attributes	N
7.5.8.1.1	Key table	N
7.5.8.1.2	Device table	N
7.5.8.1.3	Minimum security level table	N
7.5.8.1.4	Frame counter	N
7.5.8.1.5	Automatic request attributes	N
7.5.8.1.6	Default key source	N
7.5.8.1.7	PAN coordinator address	N
7.5.8.2	Functional description	N
7.5.8.2.1	Outgoing frame security procedure	N
7.5.8.2.2	Outgoing frame key retrieval procedure	N
7.5.8.2.3	Incoming frame security procedure	N
7.5.8.2.4	Incoming frame security material retrieval procedure	N
7.5.8.2.5	KeyDescriptor lookup table	N
7.5.8.2.6	Blacklist checking procedure	N
7.5.8.2.7	DeviceDescriptor lookup procedure	N
7.5.8.2.8	Incoming security level checking procedure	N
7.5.8.2.9	Incoming key usage policy checking procedure	N

2.5.1 Neighbor Table

Every device must maintain a Neighbor Table, which contains information about all the devices within the POS of a device. This table is actualized each time any frame is received from a neighboring device, and each time a Tone Map Response command is received. This table must be accessible by the Adaptation, MAC and PHY layers. Each entry of this table contains the following fields:

Table 6 - Neighbor Table

Field Name	Size/Type	Description
ToneMap	9 bits	The Tone Map parameter defines which frequency subband can be used for communication with the device. A bit set to 1 means that the frequency subband can be used, and a bit set to 0 means that frequency subband must not be used.
Modulation	2 bits	The modulation type to use for communicating with the device. 0x00: ROBO 0x01: DBPSK 0x02: DQPSK 0x03: Reserved
TxGain	4 bits	The TX Gain to use to transmit frames to that device

TxRes	1 bit	Tx Gain resolution corresponding to one gain step. 0 : 6 dB 1 : 3 dB
TxCoeff	8 x 4 bits	The TX gain for each 10 kHz-wide spectrum band
LQI	8 bits	Link Quality Indicator
Age	8 bits	The remaining lifetime of the device in minutes. When the entry is created, this value must be set to the default value 0; When it reaches 0, a Tone Map request may be issued if data is sent to this device. Upon successful reception of a tone map response, this value is set to *adpMaxAgeTime.*
Is Neighbor	8 bits	Indicate either the device is neighbor or not.

If the device receives a frame whose source address field (in 802.15.4 MAC header) does not exist in the neighbor table, it must add a new entry for that device with default values:

- Modulation = 0 (ROBO)
- ToneMap = (all bits set to 1) AND (*adpToneMask*)
- TxGain = 0000b
- TxCoeff = 0xFFFFFFFF
- LQI = 0
- Age = 0

The Neighbor Table is available in the Information Base under the attribute *macNeighborTable*.

2.6 Security Suite Specification

The MAC security suite specification is as given in clause 7.6 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 7 - Modifications and statements to clause 7.6 of the 802.15.4 standard [802.15.4-2006]

Clause	*Title & remarks/modifications*	*Statement*
7.6	Security suite specification	N
7.6.1	PIB security material	N
7.6.2	Auxiliary security header	N
7.6.2.1	Integer and octet representation	N
7.6.2.2	Security Control field	N
7.6.2.2.1	Security Level subfield - Two values allowed by the present specification: 0x00 = "none" or 0x05 = "ENC-MIC-32"	M
7.6.2.2.2	Key Identifier Mode subfield - One Key Identifier Mode allowed by the present specification: 0x01 = "Key determined from the 1-octet Key Index subfield". The number of keys is limited to 4 (range = 0-3)	M
7.6.2.3	Frame Counter field	N
7.6.2.4	Key Identifier field	N
7.6.2.4.1	Key Source subfield	N/R
7.6.2.4.2	Key Index subfield	N
7.6.3	Security operations	N
7.6.3.1	Integer and octet representation	N
7.6.3.2	CCM* Nonce	N
7.6.3.3	CCM* prerequisites	N
7.6.3.3.1	Authentication field length	N
7.6.3.4	CCM* transformation data representation	N
7.6.3.4.1	Key and nonce data inputs	N
7.6.3.4.2	a data and m data - See clause 7.6.2.2.1	M
7.6.3.4.3	c data output - See clause 7.6.2.2.1	M
7.6.3.5	CCM* inverse transformation data representation	N
7.6.3.5.1	Key and nonce data inputs	N
7.6.3.5.2	c data and a data	N
7.6.3.5.3	m data output	N

2.7 Message Sequence Chart Illustrating MAC-PHY Interaction

The MAC message sequence chart is as given in clause 7.7 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 8 - Modifications and statements to clause 7.7 of the 802.15.4 standard [802.15.4-2006]

Clause	*Title & remarks/modifications*	*Statement*
7.7	Message sequence chart illustrating MAC-PHY interaction Figure 78: Replaced by clause 4.7.1 of this document Figure 79: N/R Figure 80: N/R Figure 81: N/R Figure 82: N/R Figure 83: Replaced by clause 4.7.2 of this document Figure 84 & Figure 85: Replaced by clause 4.7.3 of this document Figure 86: N/R Additional figure about channel estimation in clause 4.7.4 of this document	M

2.7.1 PAN start message sequence chart for PAN coordinators

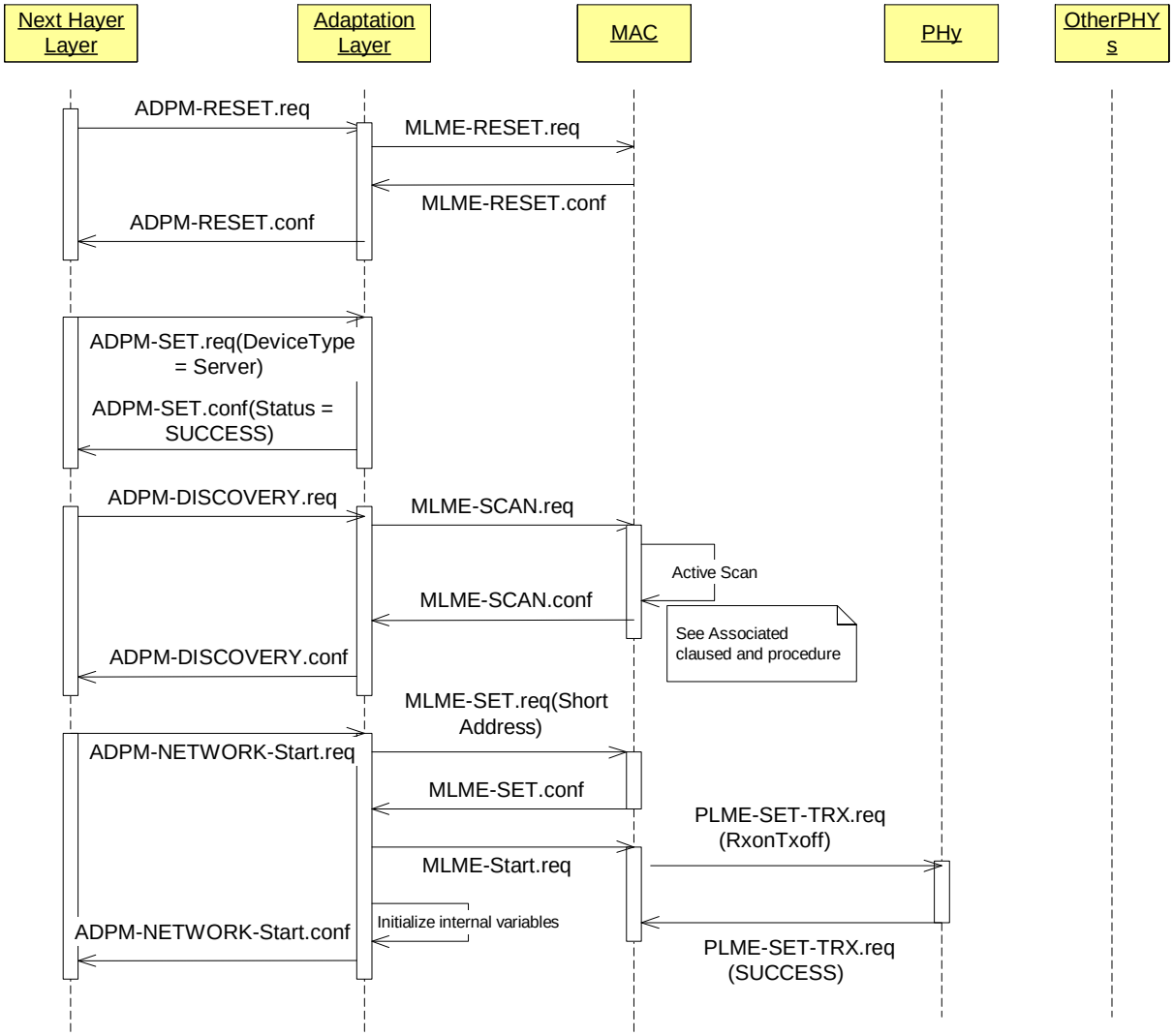


Figure 2 - PAN start message sequence chart

2.7.2 Active Scan message sequence chart

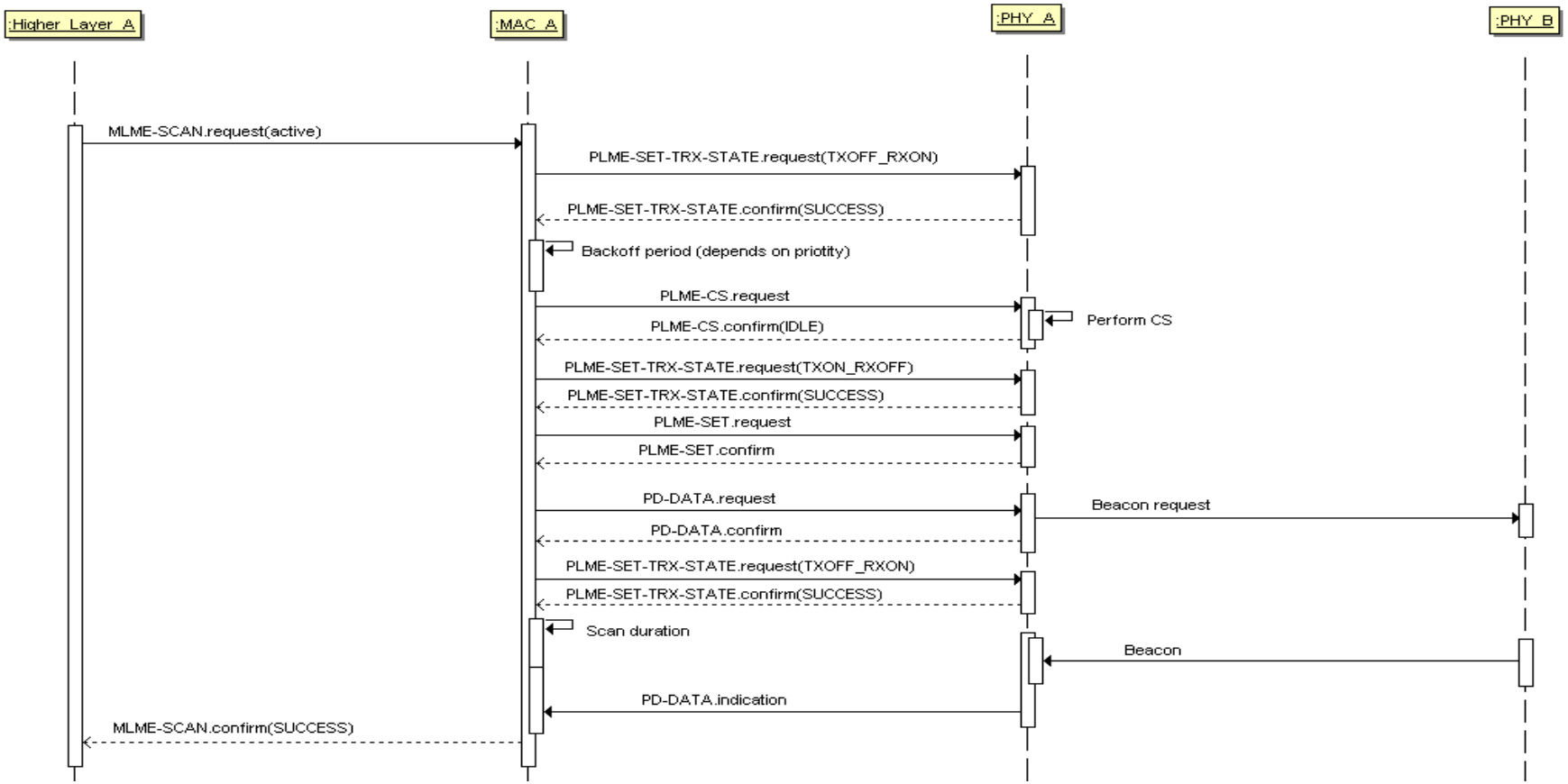


Figure 3 - Active scan message sequence chart

2.7.3 Data Transmission message sequence chart

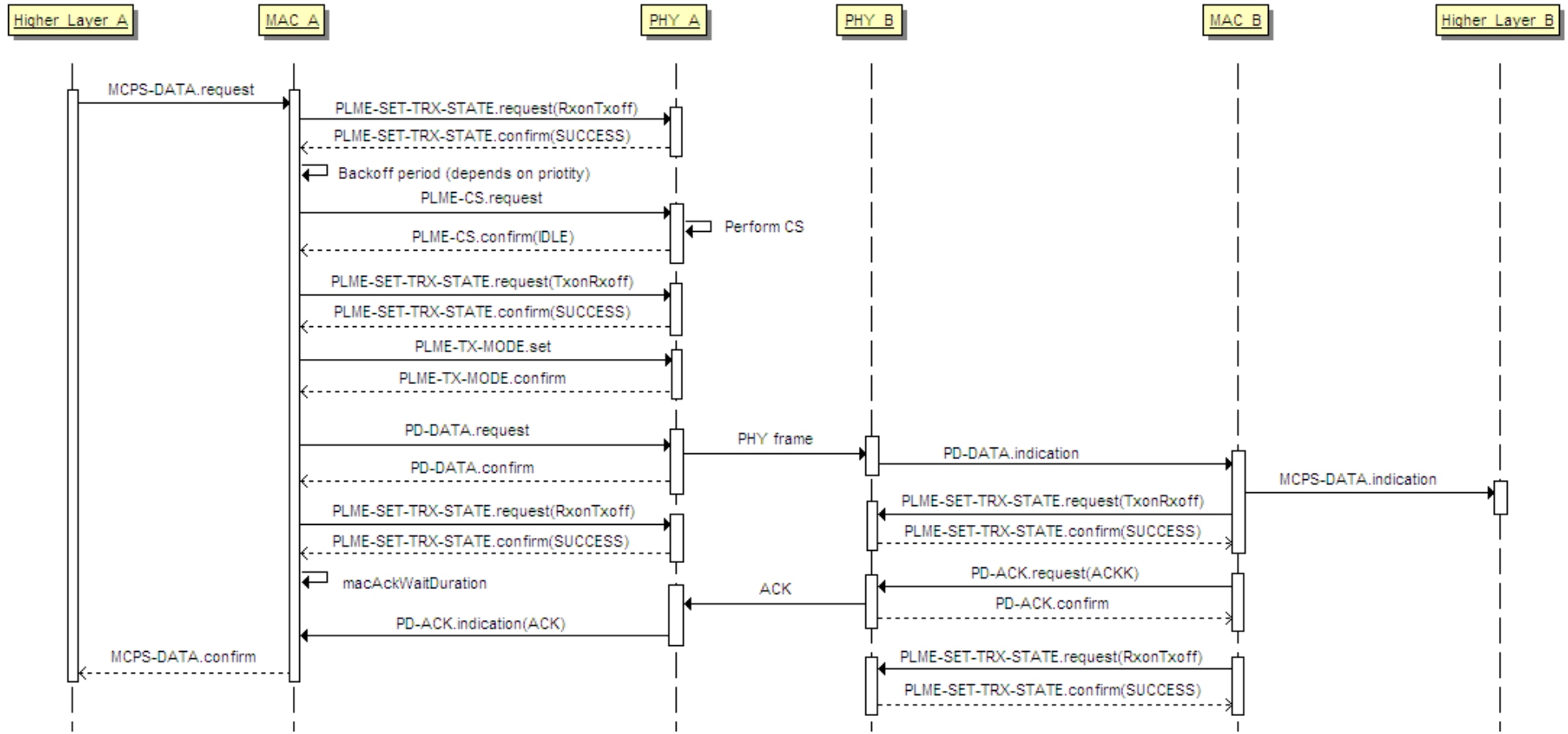


Figure 4 - Data transmission message sequence chart

2.7.4 Channel estimation message sequence chart

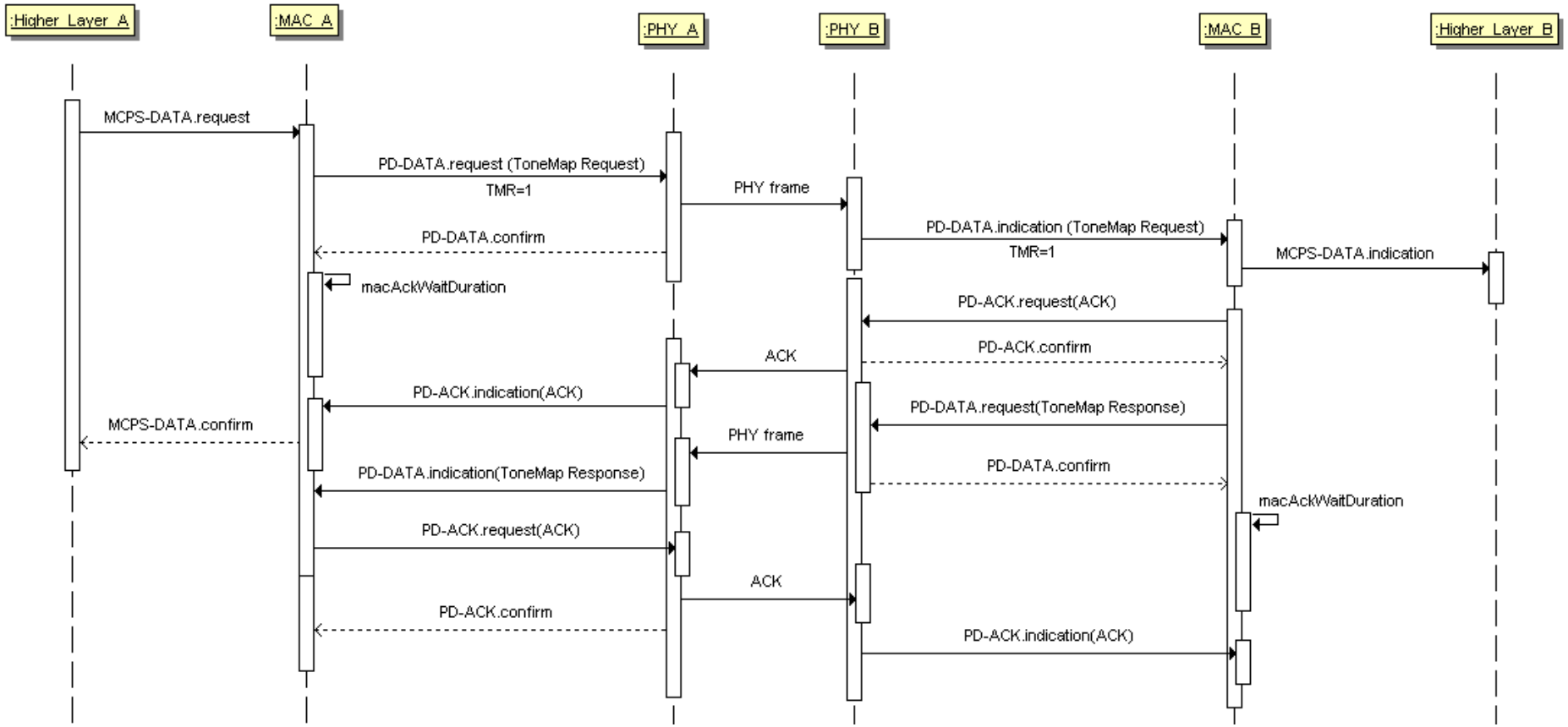


Figure 5 - Channel estimation (tone map request) message sequence chart

2.8 MAC Annexes

The MAC annexes are as given in clause 8 of the 802.15.4-2006 standard [802.15.4-2006] together with the following statements and modifications.

Table 9 - Modifications and statements to clause 8 of the 802.15.4 standard [802.15.4-2006]

Clause	*Title & remarks/modifications*	*Statement*
8	Annexes	N
8.1	Annex A: SSCS - IEEE 802.2 convergence sublayer is not used in the present specification	N/R
8.2	Annex B: CCM* mode of operation	N
8.3	Annex C: Test vectors for cryptographic building blocks	N
8.4	Annex D: PICS - The protocol implementation conformance tables are given in annex 1	M
8.5	Annex E: Coexistence with other IEEE standards and proposed standards - This annex relates to wireless PHY standards and is not relevant for PLC technology	N/R
8.6	Annex F: Regulatory requirements - This annex relates to wireless PHY standards and is not relevant for PLC technology	N/R

[Page intentionally left blank]

3 Adaptation Layer

3.1 Services and Primitives

The Services and Primitives of the ADP layer are described in annex 3 of the present document.

3.2 Information Base Attributes

The following table lists the IB attributes of the adaptation layer.

Table 10 - Adaptation Layer IB Attributes

Attribute	Identifier	Type	Read Only	Range	Description	Default
adpIPv6Address	0x01	IPv6 address	Yes	Any	The IPv6 address obtained from *adpShortAddress*	FE80::FFFF:00FF:FE00:FFFF
adpBroadcastLogTableEntryTTL	0x02	Integer	No	0-3600	The number of seconds an entry in the adpBroadcastLogTable remains active in the table.	10
adpDiscoveryAttemptsSpeed	0x06	Integer	No	1-3600	This value allows to program the maximum wait time between two successive network discoveries.	60
adpPANConflictWait	0x08	Integer	No	0-3600	The number of seconds to wait between two consecutive CONFLICT frames for the same conflicting PAN ID.	1800
adpMaxPANConflictCount	0x09	Integer	No	0-100	The maximum number of CONFLICT frames sent by a device for the same PAN ID.	3
adpActiveScanDuration	0x0A	Integer	No	0-60	The number of seconds an active scan must last.	5
adpBroadcastLogTable	0x0B	Set	Yes	-	The Broadcast Log Table, see 5.3.2 and 5.4.2.1 of this document	Empty
adpRoutingTable	0x0C	Set	Yes	-	The Routing Table, see clause 5.1 in [draft-load] and 5.3.1 of this document	Empty
macNeighborTable	0x0D	Set	Yes	-	The Neighbor Table, see 4.5.1 of this document	Empty
adpGroupTable	0x0E	Set	No	-	The table containing the group addresses to which the device belongs.	Empty
adpToneMask	0x0F	70 bits	No	Any	The Tone Mask to use during symbol formation	All bits set to 1
adpMaxHops	0x10	Integer	No	1-8	The maximum number of hops to be used by the routing algorithm	4
adpDeviceType	0x11	Integer	No	0-2	The Type of the device connected to the modem	2

Attribute	Identifier	Type	Read Only	Range	Description	Default
					([0→Device],[1→Server],[2→ Not_Device,Not_Server])	
adpNetTraversalTime	0x12	Integer	No	0-3600	The Max duration between RREQ and the correspondent RREP.	3000
adpRrtTtl	0x13	Integer	No	0-3600	The time to live of a route request table entry.	10
adpKr	0x14	Integer	No	0-31	The Kr constant to calculate the route cost.	6
adpKm	0x15	Integer	No	0-31	The Km constant to calculate the route cost.	5
adpKc	0x16	Integer	No	0-31	The Kc constant to calculate the route cost.	5
adpKq	0x17	Integer	No	0-31	The Kq constant to calculate the route cost.	5
adpKh	0x18	Integer	No	0-31	The Kh constant to calculate the route cost.	5
adpRREQRetries	0x19	Integer	No	Any	The number of RREQ retransmission in case of RREP reception time out.	3
adpRREQRERRWait	0x1A	Integer	No	Any	The number of seconds to wait between two consecutive RREQ\RRER generations.	400
adpWeakLQIValue	0x1B	Integer	No	Any	The weak Link Value.	63
adpKrt	0x1C	Integer	No	0-31	The Krt constant to calculate the route cost.	5
adpSoftVersion	0x1D	Set	Yes	-	The soft version	-
adpSpyMode	0x1E	Integer	No	0-1	Spy Mode activation/deactivation	0

3.3 IB Attributes Values Description:

In this section we describe some IB attributes Values:

3.3.1 Routing Table Entry:

The following table describes the routing table entry:

Size → 11 bits	*3 bits*	*18 bits*
Next Hop	Status	Life Time

3.3.2 Broadcast Log Table Entry:

The following table describes the broadcast log table entry:

Size → 11 bits	*8 bits*	*13 bits*
Source Address	Sequence Number	TTL

3.3.3 Device Type:

This attribute describe the type of the device connected to the modem. Its default value is (0x02) neither Server nor Device and it can be set to (0x01) server or (0x00) Device.

Note: The range of the IB attributes values is as done in our developpement (minimum values are zeros).

3.4 Data Frame Format, Datagram Transmission and Addressing

The data frame format, the theory of operation for datagram transmission using the IEEE 802.15.4 MAC layer, and the addressing scheme are as given in [rfc4944] together with the following statements and modifications.

Table 11 - Modifications and statements to clauses of [rfc4944]

Clause	Title & remarks/modifications	Statement
1.	Introduction	N
1.1.	Requirements Notation	N
1.2.	Terms Used	N
2.	IEEE 802.15.4 Mode for IP - Data frames should always be acknowledged - Only nonbeacon-enabled network are used	M
3.	Addressing Modes - IPv6 prefixes learning via router advertisements is not supported	M
4.	Maximum Transmission Unit	N
5.	LoWPAN Adaptation Layer and Frame Format - When more than one LoWPAN header is used in the same packet, they MUST appear in the following order: Mesh Addressing Header Broadcast Header Fragmentation Header Command Frame Header (defined later)	M
5.1.	Dispatch Type and Header	N
5.2.	Mesh Addressing Type and Header - The value of the HopsLeft field must not exceed *adpMaxHops*	M
5.3.	Fragmentation Type and Header	N
6.	Stateless Address Autoconfiguration - The Interface Identifier [RFC4291] for an IEEE 802.15.4 interface MUST be based on the EUI-64 identifier [EUI64] assigned to the IEEE 802.15.4 device, the latest being itself based on a EUI-48. - Additional care must be taken when choosing a PAN identifier, so that not to interfere with the I/G and U/L bits of the interface identifier. If the PAN identifiers are chosen randomly, then should be logically ANDed with 0xFCFF.	M
7.	IPv6 Link Local Address	N
8.	Unicast Address Mapping	N
9.	Multicast Address Mapping	N
10.	Header Compression	N
10.1.	Encoding of IPv6 Header Fields	N
10.2.	Encoding of UDP Header Fields	N
10.3.	Non-Compressed Fields	N
10.3.1.	Non-Compressed IPv6 Fields	N
10.3.2.	Non-Compressed and Partially Compressed UDP Fields	N
11.	Frame Delivery in a Link-Layer Mesh - All devices must be FFD	M
11.1.	LoWPAN Broadcast	N
12.	IANA Considerations	N
13.	Security Considerations	N
14.	Acknowledgements	N/R
15.	References	N/R
15.1.	Normative References	N
15.2.	Informative References	I

Clause	Title & remarks/modifications	Statement
Appendix A.	Alternatives for Delivery of Frames in a Mesh	N/R

3.4.1 Adaptation Layer Command Frames

The ADP layer command frames are identified using the ESC header type, followed by an 8-bit dispatch field indicating the type of ADP command. This header must always be in the last position. The command header is fully defined in the following figure:

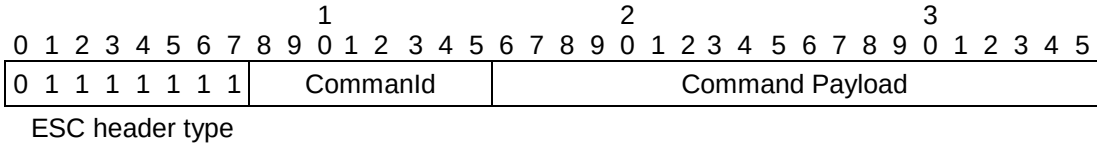


Figure 6 - Adaptation Layer Command Frame Format

The ADP layer command frames are specified in the following table:

Table 12- Adaptation Layer Command Frames

Command	CommandId	Comments	Specified in…
Route Request (RREQ)	0x01	Additional TYPE field set to 0x01 in command payload	Clause 5.2 of the present document
Route Reply (RREP)	0x01	Additional TYPE field set to 0x02 in command payload	Clause 5.2 of the present document
Route Error (RERR)	0x01	Additional TYPE field set to 0x03 in command payload	Clause 5.2 of the present document
Path Request (PREQ)	0x01	Additional TYPE field set to 0x04 in command payload	
Path Reply (PREP)	0x01	Additional TYPE field set to 0x05 in command payload	
LoWPAN Bootstrapping Protocol message	0x02	Additional code in command payload	Clause 5.5.1 of the present document
Contention Free Access Command	0x03	-	

3.4.1.1 Contention Free Access command

The adaptation layer generates CFA (Contention Free Access) command if it receives an ADPD-DATA.request primitive with QualityOfService = 2 (See 7.5.1.1)

The CFA command shall be formatted as illustrated in Figure 7.

Octets: (see7.2.2.4)	1	1
ESC header type	Command Id = 0x03	CFA value

Figure 7- CFA command format

Table 13- CFA value field description

CFA value	Description
0	Request to allow a transmission during contention free slot
1	Request to stop a transmission during contention free slot
2	Response with SUCCESS
3	Response with FAIL

The network coordinator may always use a contention free slot for transmission if other devices are not allowed to use it the same time. Other devices must ask the network coordinator for permission to use a contention free slot (CFS) (see 7.4.4) for transmission by sending CFA command with request. The network coordinator may allow requested device to use a CFS for transmission by sending a confirmation response. After receiving a successful response from the network coordinator requested device can start a transmission during CFS. If the network coordinator denies a request the device should not use a CFS for transmission. The requested device must send a request to stop using CFS when it is done with contention free transmission.

Contention Free Access is an optional feature of this specification, and is currently not implemented. Priority management can be performed using the “Normal” and “High” priority values for QOS parameter of MCPS-DATA.request primitive.

3.5 Mesh Routing

The mesh routing is as given in [draft-load] together with the following statements and modifications.

Table 14 - Modifications and statements to clauses of [draft-load]

Clause	Title & remarks/modifications	Statement
1.	Introduction	N
2.	Requirements notation	N
3.	Overview - Routing is only permitted with 16-bit addresses - LOAD uses the route cost described in annex 2 as a metric of routing	M
4.	Terminology	N
5.	Data Structures	N
5.1	Routing Table Entry - The destination address must be a 16-bit address - The next hop address must be a 16-bit address - The routing table is stored in the IB under the attribute *adpRoutingTable*	M
5.2	Route Request Table Entry - The originator address must be a 16-bit address - The reverse route address must be a 16-bit address	M
5.3	Message Format	N
5.3.1	Route Request (RREQ) - The CT field must be equal to 0x0F, to specify the use of the route cost described in annex 2 - The D bit must be set to 1 - The O bit must be set to 1 - The link layer destination and originator address must be 16-bit addresses	M
5.3.2	Route Reply (RREP) - The CT field must be equal to 0x0F, to specify the use of the route cost described in annex 2 - The D bit must be set to 1 - The O bit must be set to 1 - The link layer destination and originator address must be 16-bit addresses	M
5.3.3	Route Error (RERR) - The D bit must be set to 1 - The O bit must be set to 1 - The unreachable address must be 16-bit addresses	M
6.	Operation	N
6.1	Generating Route Request	N
6.2	Processing and Forwarding Route Request	N
6.3	Generating Route Reply	N
6.4	Receiving and Forwarding Route Reply	N
6.5	Local Repair and RERR - If a link break occurs or a device fails during the delivery of data packets, the upstream node of the link break MUST repair the route locally, and execute the repairing procedure described in the present clause.	M
7.	Configuration Parameters - The values of the configuration parameters must be: NET_TRAVERSAL_TIME = 4000 (to be tuned later after lab	M

Clause	Title & remarks/modifications	Statement
	experimentation) RREQ_RETRIES = 3 RREQ_RERR_WAIT = 2s (to be tuned later after lab experimentation) WEAK_LQI_VALUE = 63	
8.	IANA Consideration	N
9.	Security Considerations	N/R
10.	Acknowledgments	N/R
11.	References	N
11.1	Normative Reference	N
11.2	Informative Reference	I

3.5.1 Unicast Packet Routing

The routing of unicast packet is performed using the following algorithm on reception of a MCPS-DATA.indication from the MAC layer:

- If (MAC Destination address == address of device)
  - If (6LoWPAN Destination Address == address of device)
    - Generate a ADPD-DATA.indication primitive to indicate the arrival of a frame to the upper layer, with the following characteristics:
      - DstAddrMode = 0x02
      - DstAddr = 6LoWPAN Destination address
      - SrcAddr = The originator address in the 6LoWPAN mesh header
      - NsduLength = length of the data
      - Nsdu = the data
      - LinkQualityIndicator = mpduLinkQuality
      - SecurityEnabled = (SecurityLevel != 0)
  - Else if (6LoWPAN Destination Address is in the neighbor table)
    - Forward the packet to the destination address, by invoking an MCPS-DATA.request primitive with the destination address set to the final destination address
  - Else if (6LoWPAN Destination Address is in the routing table and next hop in the neighbor table)
    - Forward the packet to the next hop found in the routing table, by invoking an MCPS-DATA.request primitive, and using the communication parameters to that device contained in the Neighbor Table.
  - Else if (6LoWPAN Destination Address not in routing table)
    - Perform a link repair as described in clause 6.5 of [draft-load]
    - Queue the packet for a sending retry
  - Else
    - Drop the frame
- Else if (MAC Destination address == 0xFFFF)

- This is a broadcast frame: execute algorithm described in clause 5.4.2.1 of the present document

- Else
  - Drop the frame

3.5.2 Multicast / Broadcast

3.5.2.1 Packet Routing

This part is based on clause 11.1 of the [rfc4944] document, and details more precisely the routing of broadcast and multicast packets.

As described in the above clause, each broadcast packet has a BC0 header containing a sequence number. Each time a node sends a broadcast packet, it must increment this sequence number.

Each node must have a Broadcast Log Table. This table is used for routing broadcast packets, and each entry in this table contains the following parameters:

Table 14 - Broadcast Log Table

Field Name	Size	Description
SrcAddr	2 bytes	The 16-bit source address of a broadcast packet. This is the address of the broadcast initiator.
SeqNumber	Integer, 1 byte	The sequence number contained in the BC0 header
TimeToLive	Integer	The remaining time to live of this entry in the Broadcast Log Table, in milliseconds.

Each time a device receives a broadcast address with a HopsLft field strictly greater than 0, it must check if an entry already exists in the Broadcast Log Table having the same SrcAddr and SeqNumber. If an entry exists, the received frame is silently discarded. Else, a new entry is added in the table, and the TimeToLive field is initialized with the value *adpBroadcastLogTableEntryTTL*. When this value reaches 0, the entry is removed from the Multicast Log Table.

When a device receives a broadcast frame, so that it has to create an entry in the Broadcast Log Table, it must decrement its HopsLft field and trigger the emission of the broadcast frame using CSMA/CA. The frame will then be sent as if it were a standard unicast frame using CSMA/CA.

This can be summarized by the following algorithm, executed upon reception of a frame whose destination address is 0xFFFF:

- If (HopsLft == 0)
  - Discard frame and exit
- If ((SrcAddr, SeqNumber) exists in Broadcast Log Table)

    - Discard frame
- Else
    - Create entry (SrcAddr, SeqNumber, *adpBroadcastLogTableEntryTTL*) in Broadcast Log Table, with the corresponding frame characteristics
    - If (Final Destination Address = broadcast address) or (Final Destination Address is found in *adpGroupTable*)
        - Generate an ADPD-DATA.indication primitive with the following characteristics:
            - DstAddrMode = 0x01
            - DstAddr = Destination address in the 6LoWPAN mesh header (multicast or broadcast address)
            - SrcAddr = The originator address in the 6LoWPAN mesh header
            - NsduLength = length of the data
            - Nsdu = the data
            - LinkQualityIndicator = mpduLinkQuality
            - SecurityEnabled = (SecurityLevel != 0)
    - Trigger the sending of the frame using CSMA/CA

Note that in case of a multicast address, the broadcast address 0xFFFF is used at the MAC level as mentioned in clause 3 of [rfc4944]. Multicast frames are routed using the same algorithm as broadcast frames.

The Broadcast Log Table is available in the Information Base with the attribute *adpBroadcastLogTable*.

3.5.2.2 Groups

Each device can belong to one or more group of devices. The IB attribute *adpGroupTable* stores a list of 16-bit group addresses; when the device receives a MAC broadcast message, and if the final destination address in the 6LoWPAN mesh header is equal to one of the 16-bit group addresses in *adpGroupTable*, then an ADPD-DATA.indication primitive is generated (see clause 5.4.2.1 of the present document).

Groups can be added or removed from the *adpGroupTable* using ADPM-SET.request primitives. The size of this table is implementation specific, and must have at least one entry. The way groups are managed by upper layers is beyond the scope of this document.

3.5.3 Route Discovery

Route discovery can be done in 2 ways:

3.5.3.1 Manual Route Discovery

A manual route discovery can be triggered by the upper layers, for maintenance or performance purposes. This is done through the invocation of the ADPM-ROUTE-DISCOVERY.request primitive. The adaptation layer then generates a RREQ frame as described in clause 5.4 of the present document (modified clauses 5.3.1 and 6.1 of [draft-load]), and executes the algorithms described in clause 5.4 of the present document (all of the modified clauses 6.x of [draft-load]).

After the algorithm completes, the adaptation layer generates an ADPM-ROUTE-DISCOVERY.confirm primitive with the corresponding status code, and eventually modify its routing table.

Only one route discovery procedure can be processed in the same time. All other ADPM-ROUTE-DISCOVERY.request will be ignored.

All devices are required to handle RREQ, RREP and RERR frames as described in clause 5.4 of the present document (modified clauses 6.x of [draft-load]), and must modify their routing tables accordingly.

3.5.3.2 Automatic Route Discovery

If an ADPD.DATA.request primitive is invoked with its DiscoverRoute parameter set to TRUE, and if no entry is available in the routing table for the device designated by DstAddr, then the adaptation layer generates a RREQ and executes the algorithms described in clause 5.4 of the present document in order to find a route to the destination. If the route discovery succeeds, then the data frame is send to the destination according to the newly discovered route. If the route discovery fails, then the adaptation layer must generate an ADPD-DATA.confirm primitive with the status code ROUTE_ERROR.
If an ADPD.DATA.request primitive is invoked with its DiscoverRoute parameter set to FALSE, and if no entry is available in the routing table for the device designated by DstAddr, then the adaptation layer must generate an ADPD-DATA.confirm primitive with the status code ROUTE_ERROR.
Route repairing procedures are described in clause 5.4 of the present document (modified clause 6.5 of [draft-load]).

3.5.3.3 RREQ RERR Generation Frequency Limit

A node must wait RREQ_RERR_WAIT second between two successive RREQ/RERR generations to limit the number of broadcast packet in the network.

3.5.4 Path Discovery

3.5.4.1 Operation

A path discovery can be triggered by the upper layers, for maintenance or performance purposes. This is done through the invocation of the ADPM-PATH-DISCOVERY.request primitive. The adaptation layer then generates a PREQ frame and executes the algorithms next.

After the algorithm completes (with the reception of a PREP frame), the adaptation layer generates an ADPM-PATH-DISCOVERY.confirm primitive.

Only one path discovery procedure can be processed in the same time. All other ADPM-PATH-DISCOVERY.request will be ignored.

3.5.4.1.1 Generating a PREQ

During the path discovery period, an originator, a node that requests a path discovery, generates a Path Request (PREQ) message.

A node waits for a PREP, else, and after Go_BACK_TIME milliseconds, the node generate a confirm to the upper layer containing a PREP with HOPS fileds set to 0.

3.5.4.1.2 Processing and forwarding a PREQ

Upon receiving a PREQ, an intermediate FFD node tries to find entry of the same destination address in the routing table. If the entry is found, the node just forwards the PREQ to the next hop toward the destination. Else, the node just discards the PREQ.

3.5.4.1.3 Generating a PREP

A final node, on receiving PREQ, generate a PREP with R flag set to 0, hops to 1 and the Hops1 address set to its own address.

If an intermediate node can’t find a route to the destination of the PREQ, it generates a PREP with R flag set to 1, the HOP to 1 and the Hops1 address to its own address then sends it to the source of the PREQ.

3.5.4.1.4 Processing and forwarding a PREP

Upon receiving a PREP, an intermediate FFD node set the HopN address field in the PREP to its own address, with N = HOPS+1, update the RC and increment the HOPS. If there’s no route to the destination, the node just discard the packet.

3.5.4.2 Path Request Frame

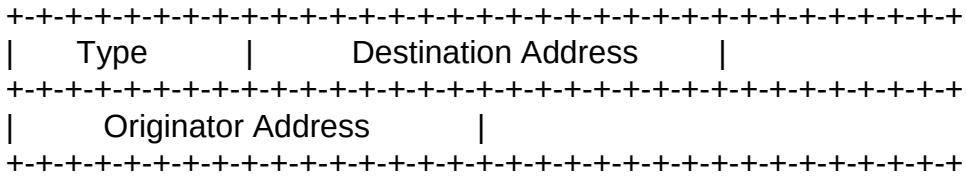


Figure 8 PREQ message format

Type (8bit)
4 for indicating a PREQ message.

RC(Route cost) (8bit)
The accumulated link cost of the reverse route from the originator to the sender of the message.

Destination Address
The 16 bit short link layer address of the destination for which a route is supplied.

Originator Address
The 16 bit short link layer address of node which originated the packet.

3.5.4.3 Path Reply Frame

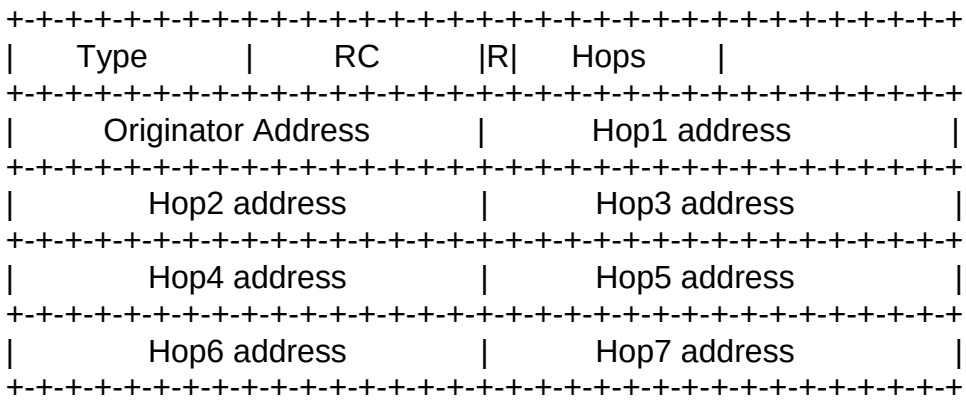


Figure 9 PREP message format

Type (8bit)
= 4 for indicating a PREQ message.

R (1bit)
1 Success of path discovery
0 Failure of path discovery

RC(Route cost) (8bit)
The accumulated link cost of the reverse route from the originator to the sender of the message.

Hops (7bit)
The number of hops of the route.

Originator Address
The 16 bit short link layer address of node which originated the packet.

HopN Address
The 16 bit short link layer address of nodes constituting the path.

3.6 Commissioning of New Devices

The commissioning of new devices on an existing network is as given in [draft-commissioning] together with the following statements and modifications.

Table 16 - Modifications and statements to clauses of [draft-commissioning]

Clause	Title & remarks/modifications	Statement
1.	Introduction	N
2.	Terminology	N
2.1.	Requirements notation	N
3.	Bootstrapping - Obtaining a 16-bit short address and security credentials are mandatory parts of the commissioning process	M
3.1.	Resetting the device	N
3.2.	Scanning through channels - For getting the information of other devices within POS, the device MUST perform an active scan.	M
3.3.	LoWPAN Bootstrapping Mechanism -'LBA discovery phase' is described in section 3.3.3	M
3.3.1.	LoWPAN Bootstrapping Protocol message format	N
3.3.1.1	LBP message - The following clause 5.5.1 of the present document proposes some enhancements and clarifications to LBP message format	M
3.3.2.	LoWPAN Bootstrapping Information Base - PAN_type must always be Secured - Address_of_LBS must be equal to the default address of the PAN coordinator, that is 0x0000 - Short_Addr_Distribution_Mechanism must be 0 for centralized address management	M
3.3.3.	LBA discovering phase - The following clause 5.5.2 of the present document proposes some enhancements and clarifications to 6LoWPAN bootstrapping procedure - The LBD must perform an active scan instead of broadcasting a LBA solicitation message.	M
3.3.4.	LoWPAN Bootstrapping Protocol (LBP)	M
3.3.5.	Bootstrapping in open 6LoWPAN	N/R
3.3.6.	LBP in secured 6LoWPAN - The LBP messages from the LBD to the LBA are sent by invocation of the ADPD-DATA.request primitive with the following attributes: - DstAddrMode = 0x02 - DstAddr = The MAC address of the LBA passed as an argument to the ADPM-NETWORK-JOIN.request primitive - NsduLength = the length of the LBP message - Nsdu = the LBP message itself - NsduHandle = random number - MaxHops = 0 - DiscoverRoute = FALSE - QualityOfService = 0 - SecurityEnabled = FALSE Remark: The LBA is already present in the neighbor table because an active scan must have been performed prior to invoking the ADPM-NETWORK-JOIN.request primitive. Thus the routing algorithm will	M

Clause	Title & remarks/modifications	Statement
	operates correctly as described in clause 5.4.1 of the present document. - The LBP messages from the LBA relayed to the LBS are sent by invocation of the ADPD-DATA.request primitive with the following attributes: - DstAddrMode = 0x02 - DstAddr = The MAC address of the LBS - NsduLength = the length of the LBP message - Nsdu = the LBP message itself - NsduHandle = random number - MaxHops = *adpMaxHops* - DiscoverRoute = TRUE - QualityOfService = 0 - SecurityEnabled = TRUE - The LBP messages from the LBS to the LBD relayed to the LBA are sent by invocation of the ADPD-DATA.request primitive with the following attributes: - SrcAddrMode = 0x02 - DstAddrMode = 0x03 - DstPANId = the PAN ID - DstAddr = 64-bit address of the LBD - msduLength = the length of msdu - msdu = the msdu - msduHandle = a random number - TxOptions = 100b - SecurityLevel = 0 - KeyIdMode = ignored - KeySource = ignored - KeyIndex = ignored	
3.3.7.	Role of Entities in LBP - If a LBD does not find any LBA during the LBA discovery phase, it must still perform LBA discoveries as long as it is not commissioned. Note that LBA discovery is done using active scans rather that broadcasting LBA solicitation messages. - Only secured networks are used	M
3.4.	Assigning the short address - Short addresses are assigned in a centralized fashion by the LBS	M
3.5.	Obtaining IPv6 address - The devices do not need to obtain an IPv6 address prefix, and the procedures described in this clause as well as [rfc4862] must be ignored. Only the IPv6 Link Local Address generated as stated in clause 7 of [rfc4944] is used for communication.	M
3.6.	Configuration Parameters - The values of the configuration parameters must be: CHANNEL_LIST = 0xFFFF800 (not used) SCAN_DURATION = *adpActiveScanDuration* SUPERFRAME_ORDER = 15 BEACON_ORDER = 15 START_RETRY_TIME = 0 (not used) JOIN_RETRY_TIME = 0 (not used) ASSOCIATION_RETRY_TIME = 0 (not used)	M
4.	IANA Consideration	N/R
5.	Security Considerations	N
6.	Contributors	N/R
7.	Acknowledgments	N/R

Clause	Title & remarks/modifications	Statement
8.	References	N
8.1.	Normative References	N
8.2.	Informative References	I

3.6.1 6LoWPAN Bootstrapping Protocol (LBP) messages format

This clause proposes some enhancements and clarifications to LBP messages format

3.6.1.1 LBP message

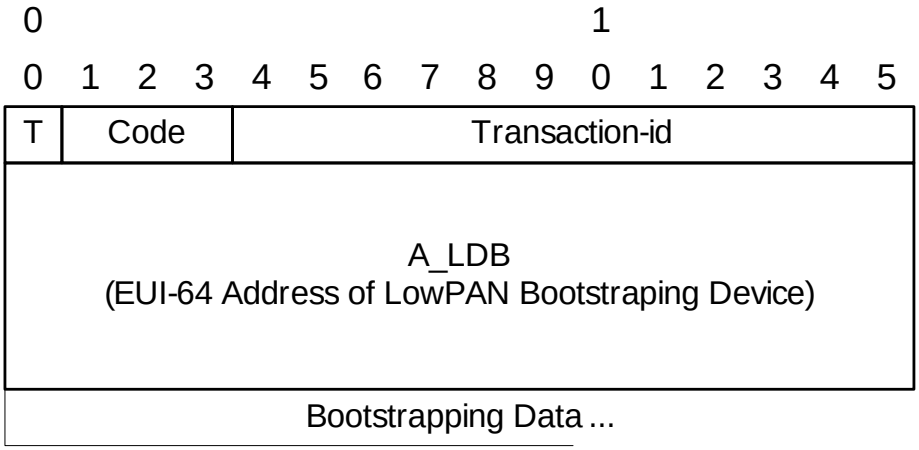


Figure 10- LBP message format

T	The 1-bit T field identifies the type of message 0 message from LDB 1 message to LDB
Code	The 3-bit Code field identifies the message code. See Table 12.
Identifier	The 12-bit Identifier field aids in matching Responses with Request
A_LDB	The A_LDB field is 8 octets and indicates the EUI-64 address of the bootstrapping device (LBD)
Bootstrapping Data	The Bootstrapping Data field is of variable length and contains additional information elements. Two types are defined: - Embedded EAP messages - Configuration parameters

Table 17- T & Code Fields in LBP messages

T	Code	LDB message	Description
0	001	JOINING	The LDB requests joining a PAN and provides necessary authentication material
1	001	ACCEPTED	Authentication succeeded with delivery of Device Specific Information (DSI) to the LDB
1	010	CHALLANGE	Authentication in progress. PAN Specific Information (PSI) may be delivered to the LDB
1	011	DECLINE	Authentication failed
0/1	100	KICK	KICK frame: used by a PAN coordinator to force a device to loose its MAC address, or by any device to inform the coordinator that it left the PAN. On reception of this frame, a device must set its short address to the default value of 0xFFFF, disconnect itself from the network, and perform a reset of the MAC and adaptation layers. See clause 5.5.2.6 of the present document for details about kicking procedure.
0	101	CONFLICT	CONFLICT frame: used by a device to inform the PAN coordinator that it has detected another PAN operating in the same POS. See clause 5.7.2 of the present document for details about PAN ID conflict handling.

3.6.1.2 Embedded EAP messages

LBP messages embed Extended Authentication messages (EAP) as defined by [rfc3748] with a minor modification to fit the generic LBP information element format.

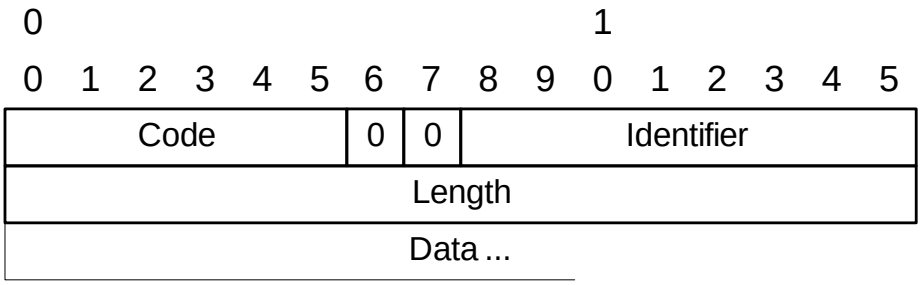


Figure 11- Embedded EAP message format (generic)

Code — The 6-bit Code field identifies the Type of EAP packet. EAP Codes are assigned as follows:

1 Request (sent to the peer = LDB)
2 Response (sent by the peer)
3 Success (sent to the peer)
4 Failure (sent to the peer)

The Code field is slightly different from a regular EAP Code

field as specified in [rfc3748]. The conversion appears straightforward in both directions. The proper conversion must apply when the EAP message is propagated over another protocol (i.e. RADIUS) and in case of integrity protection covering the EAP header.

Identifier	The Identifier field is one octet and aids in matching Responses with Requests
Length	The Length field is two octets and indicates the length, in octets, of the EAP packet including the Code, Identifier, Length, and Data fields. A message with the Length field set to a value larger than the number of received octets MUST be silently discarded.
Data	The Data field is zero or more octets. The format of the Data field is determined by the Code field. Refer to [rfc3748] for more details on: - Specific format for Request / Response messages and the introduction of the Type field (Identity, Nak, etc.) - Specific format for Success / Failure messages with an empty Data field.

3.6.1.3 Configuration parameters

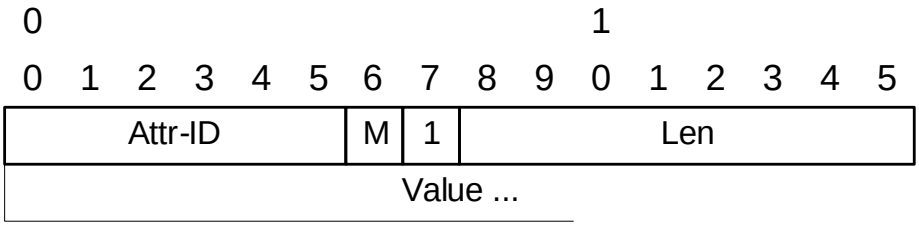


Figure 12 - Configuration parameter format

Attr-ID	The 6-bit Attr-ID field represents the ID of the Attribute in LoWPAN Information Base (LIB)
M	The 1-bit M field identifies the type of the Attribute: 0 Device Specific Information (DSI) 1 PAN Specific Information (PSI)
Len	The Length field is one octet and indicates the length, in octets, of the Value field
Value	The Value field is zero or more octets and contains the value of the Attribute. Its format is defined by Attr-ID.

3.6.2 6LoWPAN bootstrapping procedure

This clause proposes some enhancements and clarifications to 6LoWPAN bootstrapping procedure. This procedure is executed when the ADPM-NETWORK-JOIN.request primitive is invoked.

The following figure provides an overview of the messages exchanged between devices during the Bootstrapping procedure:

Figure 13 – Bootstrapping protocol messages sequence chart

And the following sequence diagram summarizes the forwarded messages involved during a nominal association procedure on a PAN between different protocol layers of the devices, when a single LBP protocol message needs to be exchanged between the LBD and the LBS:

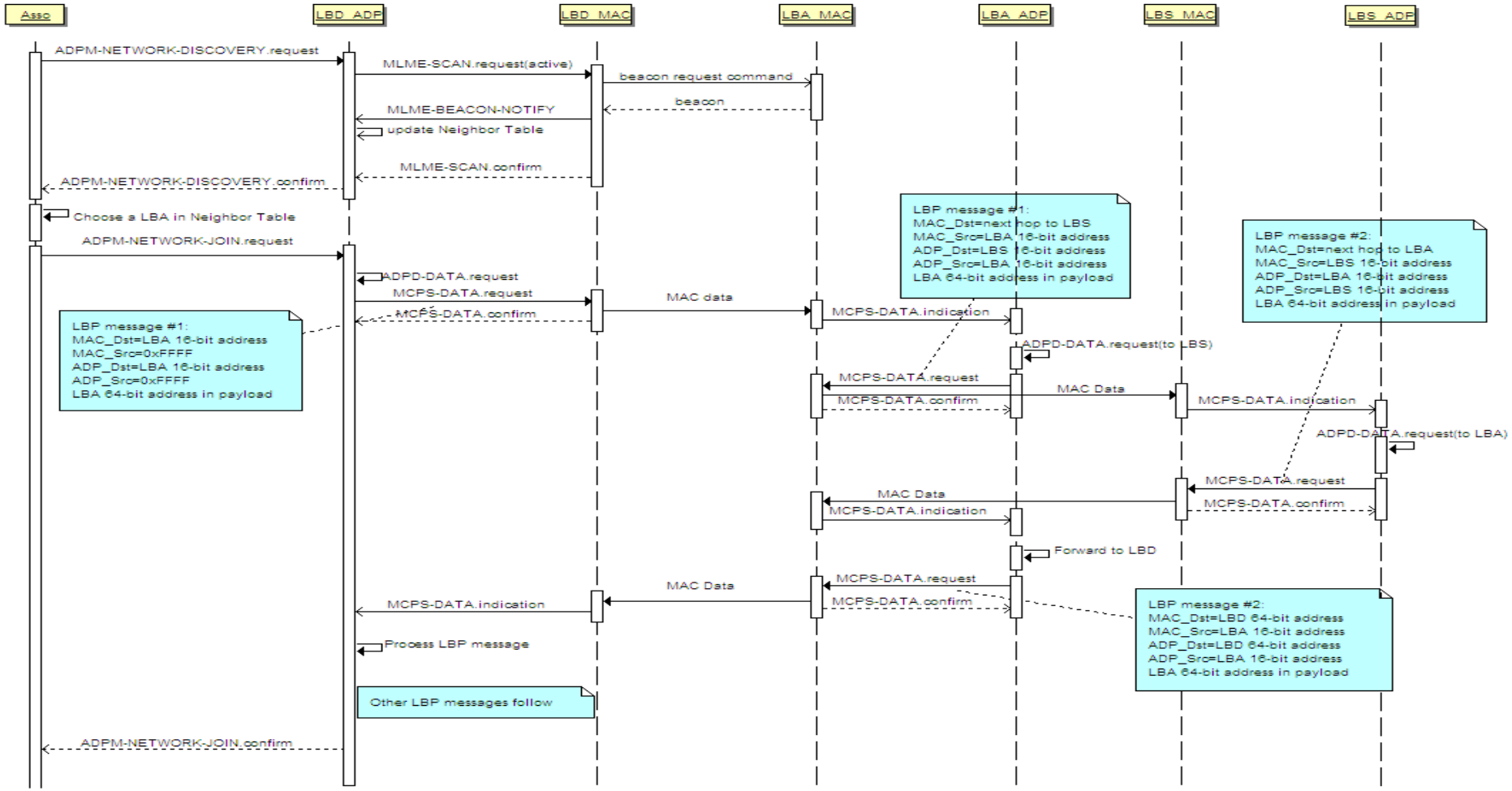


Figure 14 - Bootstrapping protocol messages forwarding

3.6.2.1 Discovering phase

At the beginning of the Bootstrapping procedure, an End Device (aka LoWPAN Bootstrapping Device or LBD) must launch an "active channel scan" (cf. [802.15.4] clause 7.5.2.1.2). The higher layer can start an active scan by invoking the ADPM-DISCOVERY.request primitive, and specifying the duration of the scan. The adaptation layer then invokes the MLME-SCAN.request primitive of the MAC layer with the following parameters:

- ScanType = 0x01
- ScanChannels = all bits to 0 (not used)
- ScanDuration = Duration
- ChannelPage = 0 (not used)
- SecurityLevel = 0
- KeyIdMode = Ignored
- KeySource = Ignored
- KeyIndex = Ignored

The LBD sends a 1-hop broadcast Beacon.request frame and any Full Feature Device in the Neighborhood should reply by sending a Beacon frame with its PAN identifier, short address and capabilities.

Upon completion, the MAC layer issues a MLME-SCAN.confirm primitive, with the list of existing PAN in the PANDescriptorList parameter. In response, the adaptation layer generates an ADPM-DISCOVERY.confirm primitive which contains the PANDescriptorList parameter provided by the MAC layer.

At the end of the scan, the LBD selects one of the Beacon senders. It may be either the PAN coordinator that play the role of LoWPAN Bootstrapping Server (LBS) or another FFD. In the latter case, the FFD (aka LoWPAN Bootstrapping Agent or LBA) is in charge of relaying the LoWPAN Bootstrapping Protocol (LBP) frames between the LBA and the LBS.

The choice is based on the following criteria:

1. Association permit, rejected if negative
2. Link Quality, rejected below a threshold
3. PAN identifier, according to a round robin algorithm
4. PAN coordinator, preferred if positive
5. Short address, according to a round robin algorithm

A device must not perform more than *adpMaxDiscoveryPerHour* network discovery procedures per hour.

3.6.2.2 Access Control phase

Then, the LBD send an LBP JOINING frame to the LBA. This frame includes a field that carries the EUI-64 address of the joining LBD.

This frame, as any other frame during the initial part of the Bootstrapping process, is transmitted between the LBD and the LBA without any additional security at the MAC layer.

When received by the LBA, this frame is relayed by the LBA to the LBS. The LBA is supposed fully bootstrapped with the full capability to directly transmit any message to the LBS in a secure way.

The LBP protocol has been designed to fit two different authentication architectures.

- The authentication function is directly supported by the LBS, and in this case all the authentication material (access lists, credentials, etc.) must be loaded in the LBS.
- The authentication function is supported by a remote (and usually centralized) AAA server, and in this case, the LBS is only in charge of forwarding the EAP messages to the AAA server over a standard AAA protocol (i.e. RADIUS [rfc2865]).

The following procedure description is only based on the first architecture but extension to the second one appears straightforward.

So, when received by the LBS, the EUI-64 address should be compared with an Access Control list (white list or black list). Two possibilities:

- This address does not fit the Access Control list and the LBS send back an LBP DECLINE message, embedding an EAP.failure message.
- This address fit the Access Control List (or the Access Control is not implemented) and the LBS send back an LBP CHALLENGE message, embedding an EAP. Request message. This latter message also carries the first authentication message.

  In the present version of the OFDM CPL specification, the EAP identity phase is skipped as proposed by [rfc3748] to directly move to the authentication phase by sending the first message of the selected EAP method.

  The EAP identity phase could be reintroduced later when the need of roaming features arise.

In both cases, these messages are relayed by the LBA to the LDB.

3.6.2.3 Authentication and Key Distribution phase

The Authentication phase is fully dependant of the EAP method in place. The EAP protocol is very flexible and support various EAP methods (EAP-MD5, EAP-AKA, EAP-TLS, etc.). Each method is characterized by its credentials (shared secret, certificate, SIM cards, etc.) and by its signature and encryption algorithms.

Methods are ordinary based on two round-trip exchanges:

- The first one for mutual authentication and initial exchange of ciphering material
- The second one for mutual control of session keys derivation

At the end, the LBD should be equipped with two sets of session keys:

- Unicast session keys for the end-to-end security of EAP messages. These keys are timely refreshed.
- Group session keys for a basic PAN security. These keys are shared by all the authenticated nodes in the PAN. Every MAC data frame, except those involved in the initial phases of the bootstrapping procedure, is securely transmitted with encryption and decryption at every hop. These Group keys are timely refreshed and when a node is detached from the PAN.

Other keys may be derived for additional security services provided at the Application level.
Refer to §6.2 for further details on the proposed EAP method.

3.6.2.4 Authorization and initial configuration phase

Then, two possibilities:

- The Authentication and Key Distribution process does reach completion and the LBS sends back an LBP DECLINE message, embedding an EAP.failure message and relayed by the LBA to the LBD.
- This process reaches completion and the LBS selects a 16-bit short Address, globally defined and fully routable in the PAN. The LBS sends back an LBP ACCEPTED message, embedding an EAP.success message, at receipt of this message the LBD activate the GMK key. A second LBP ACCEPTED message is sent by LBS embedding the global 16-bit short address and the various parameters (Device Specific and PAN Specific). These messages are relayed by the LBA to the LDB. At this stage, the LDB owns the necessary sessions keys and the messages are securely transferred end-to-end.

At reception of the LBP message, the LBD must set-up an optimized route to the LBS with the help of the LOAD protocol.

3.6.2.5 Joining a PAN for any node except coordinator

The network joining procedure must only be performed by a device which is not a PAN coordinator, and which does not have a short address. It is triggered by invocation of the ADPM-NETWORK-JOIN.request primitive. The algorithm the device must perform is:

- Short_Address = 0xFFFF (= no short address)
- Current_Neighbor_index = 0
- Connected = FALSE
- While (Short_Address == 0xFFFF)
    - Wait for a random number of seconds. The number of seconds the device must wait is *adpNumDiscoveryAttempts**Rnd, where Rnd is a random integer value between 1 and *adpDiscoveryAttemptsSpeed*, and *adpNumDiscoveryAttempts* is the number of times the ADPM-NETWORK-DISCOVERY.request primitive was called in this procedure. *adpNumDiscoveryAttempts* must be reset to 0 when ADPM-NETWORK-DISCOVERY.request succeeds, on device startup, and after a reset of the adaptation layer. The value of *adpNumDiscoveryAttempts* must not be incremented anymore if it reaches 15.
    - Perform the “Discovering Phase” through invocation of a ADPM-NETWORK-DISCOVERY.request primitive (NOTE: the Neighbor Table must be updated for each received beacon)
    - If existing PANs were found
        - Select a PAN and LBA in the neighbor table (criteria beyond the scope of this document)
        - While ((Connected == FALSE) && (Current_Neighbor_index < Size_of_Neighbor_Table)
            - Perform the “Access Control”, “Authentication and Key Distribution” and “Authorization and initial configuration” phases through invocation of the ADPM-NETWORK-JOIN.request primitive using the PAN identifier and MAC address above:
                - PANId = PANId chosen above
                - LBAAddress = Address at index Current_Neighbor_index
            - If ADPM-NETWORK-JOIN.confirm == SUCCESS
                - Short_Address = result of association process
                - Connected = TRUE
            - Else
                - Current_Neighbor_index ++
    - While (Connected == TRUE)
        - Wait for a disconnection

3.6.2.6 Leaving a PAN

3.6.2.6.1 Removal of a Device by the Coordinator

The PAN coordinator may instruct a device to remove itself from the network invoking the ADPM-LBP.request primitive, using a KICK frame. This frame is a standard LBP message frame with its Code field set to 100b. The bootstrapping data in that message should be empty.

When a device receives this message, it must check if the A_LBD field of the LBP message is its own address. If not, the message is silently discarded. Else the device must perform the following steps:

- Acknowledge the frame if necessary
- Set its 16-bit short address to 0xFFFF
- Generate a ADPM-NETWORK-LEAVE.indication containing the 64-bit address of the device
- Invoke a MLME-RESET.request primitive with the SetDefaultPIB parameter set to TRUE
- Invoke its ADPM-RESET.request primitive to reset itself

Figure 15 describes the messages exchanged during removal of a device from the PAN by the coordinator.

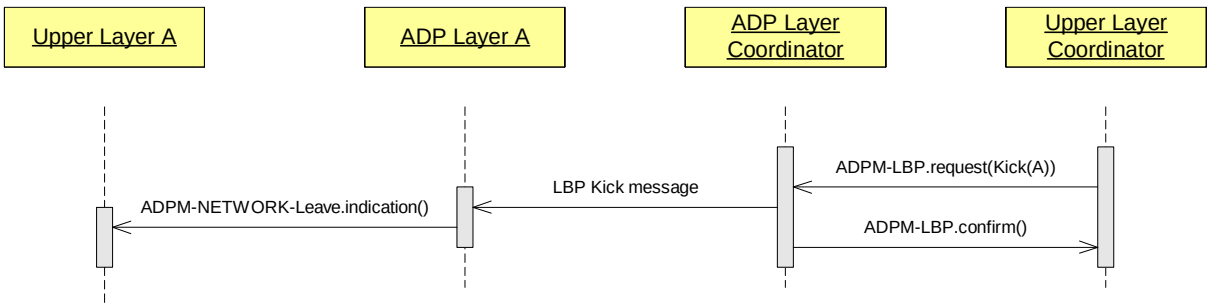


Figure 15 - Message sequence chart during removal of a device by the coordinator

Upon completion of this procedure, the device must restart the joining network procedure described in clause 5.5.2.5 of the present document.

3.6.2.6.2 Removal of a Device by Itself

A device may also call the ADPM-NETWORK-LEAVE.request primitive to remove itself from the network, and notify the PAN coordinator about this removal:
On invocation of the ADPM-NETWORK-LEAVE.request primitive by a device which is not the PAN coordinator, and with a ExtendedAddress parameter not NULL, the adaptation layer must issue an ADPM-NETWORK-LEAVE.confirm primitive with the status INVALID_REQUEST.

On invocation of the ADPM-NETWORK-LEAVE.request primitive by a device which is the PAN coordinator, and with a ExtendedAddress parameter set to NULL, the adaptation layer must issue an ADPM-NETWORK-LEAVE.confirm primitive with the status INVALID_REQUEST.

On invocation of the ADPM-NETWORK-LEAVE.request primitive by a device which is not the PAN coordinator, and with a ExtendedAddress parameter set to NULL, the adaptation layer must

- Send a KICK frame to the PAN coordinator using a ADPD-DATA.request primitive (and setting the T field in the LBP message to 1, to indicate a message from LBD)
- Set its 16-bit short address to 0xFFFF
- Generate a ADPM-NETWORK-LEAVE.indication containing the 64-bit address of the device
- Invoke a MLME-RESET.request primitive with the SetDefaultPIB parameter set to TRUE
- Invoke its ADPM-RESET.request primitive to reset itself

Figure 16 describes the messages exchanged during the removal of a device initiated by the device itself.

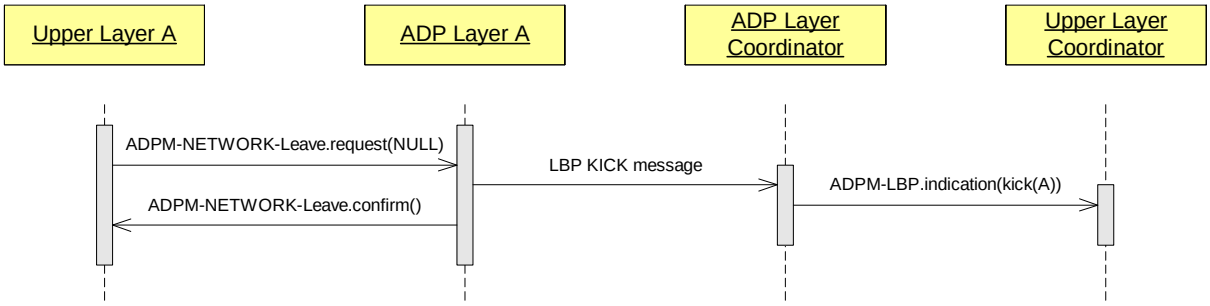


Figure 16 - Message sequence chart during removal of a device by itself

On the PAN coordinator side, an ADPM-LBP.indication containing the KICK message received, this message contain the 64-bit address of the device which removed itself from the PAN is generated to inform the upper layers.

3.7 Fragment Recovery

The fragment recovery is as given in [draft-fragment] together with the following statements and modifications.

Table 18 - Modifications and statements to clauses of [draft-fragment]

Clause	*Title & remarks/modifications*	*Statement*
1.	Introduction	N
2.	Terminology	N
3.	Rationale	N
4.	Requirements	N
5.	Overview	N
6.	New Dispatch types and headers	N
6.1.	Recoverable Fragment Dispatch type and Header	N
6.2.	Fragment Acknowledgement Dispatch type and Header	N
7.	Outstanding Fragments Control	N
8.	Security Considerations	N
9.	IANA Considerations	N
10.	Acknowledgments	N/R
11.	References	N
11.1.	Normative References	N
11.2.	Informative References	I

3.8 Spy Mode

This mode is used to have a spy modem supervising all transmission on its behavior. Once activated; the modem will process all packets like its own. The spy modem will generate an ADPD-DATA.indication for all packets received. It must also, behave like in normal mode; processing and forwarding packets. For example, on reception of an MCPS-DATA.indication which is not mine, the modem, in this case, will generate an ADPD-DATA.indication for upper layer then forward the packet toward the destination.

The route cost of all links with a spy modem is set to 31 so we prevent routing a packet via such modem.

If a spy modem receives a fragment, it'll add an IPv6 fragment header to the packet so the upper layer can detect it. The fragment offset field will be set to the offset of the LOWPAN header and the Identification field to the Datagram_Tag.

3.9 Functional Description

3.9.1 Network Formation

The network formation can only be performed by the PAN coordinator. Any device other than the PAN coordinator must not attempt to perform a network formation.

Prior to the network formation, the PAN coordinator must perform an active scan as described in clause 5.5.2.1 of the present document. If the PANDescriptorList given by the ADPM-DISCOVERY.confirm primitive is empty, then the PAN coordinator can start a new network. If the PANDescriptorList is not empty, the PAN coordinator should inform the rest of the system that a PAN is already operating in the POS of the device, and may start a new network afterwards. The procedures and decisions associated with this behavior are beyond the scope of this document.

After the network discovery, the PAN coordinator must set its PAN ID to the predefined value stored in it. This value can be obtained remotely from a configuration server, or locally computed. The way how this PAN ID is chosen and set in the coordinator is beyond the scope of this document. Additionally, the PAN identifier must be logically ANDed with 0xFCFF, as described in clause 5.3 of the present document (modified clause 6 of [rfc4944]).

Once the PAN identifier has been determined, the adaptation layer must invoke the MLME-START.request with the following parameters

- PANId = the PAN identifier computed above
- LogicalChannel = 0 (not used)
- ChannelPage = 0 (not used)
- StartTime = 0 (not used)
- BeaconOrder = 15 (beaconless network)
- SuperframeOrder = 15 (not used)
- PANCoordinator = TRUE
- BatteryLifeExtension = FALSE (not used)
- CoordRealignment = FALSE
- CoordRealignSecurityLevel, , CoordRealignKeyIdMode, CoordRealignKeySource and CoordRealignKeyIndex: not used, should be set to 0
- BeaconSecurityLevel = 0
- BeaconKeyIdMode, BeaconKeySource, BeaconKeyIndex: not used, should be set to 0

The MAC layer then generates a MLME-START.confirm primitive with the corresponding status code, which is forwarded to the upper layers through the generation of an ADPM-NETWORK-START.confirm.

3.9.2 PAN ID Conflict Detection and Handling

At any time, when a device is associated to a PAN, its MAC layer must analyze the Destination and Source PAN Identifier in the MAC header of any frame it receives. If a frame containing a Destination or Source PAN Identifier is received and does not match the PAN Identifier of the device, it must generate a MLME-SYNC-LOSS.indication primitive with the following characteristics:

- LossReason = PAN_ID_CONFLICT
- PANId = The conflicting PAN ID
- LogicalChannel = 0 (not used)
- ChannelPage = 0 (not used)
- SecurityLevel = 0 (not used)
- KeyIdMode, KeySource and KeyIndex can be ignored

If the adaptation layer receives a MLME-SYNC-LOSS.indication primitive with another LossReason than PAN_ID_CONFLICT, it must ignore it.

In response, the adaptation layer must generate a CONFLICT frame to its PAN coordinator. This frame is a standard LBP message frame with its Code field set to 101b. The bootstrapping data in that message should contain the PAN Id of the detected PAN using the following format defined in clause 3.3.1 of [draft-commissioning]:

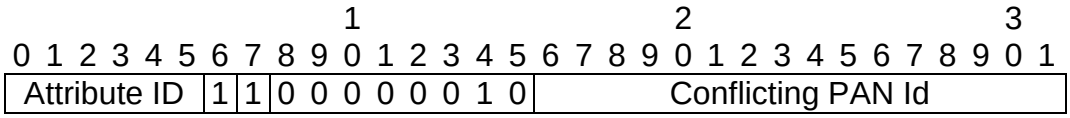


Figure 17 - CONFLICT message format

This frame is sent using an ADPD-DATA.request primitive with the following attributes:

- DstAddrMode = 0x02
- DstAddr = IPv6 destination address, formed with the short address of 0x0000
- NsduLength = the length of the frame
- Nsdu = the frame
- NsduHandle = a random number
- MaxHops = *adpMaxHops*
- DiscoverRoute = TRUE
- QualityOfService = FALSE
- SecurityEnabled = TRUE

A device must wait adpPANConflictWait seconds between two consecutive sendings of a CONFLICT frame for the same conflicting PAN Id, and the total number of CONFLICT frames sent for a given conflicting PAN Id must not exceed adpMaxPANConflitCount. When this value is reached, the device must stop sending CONFLICT frames for this conflicting PAN Id.

When the PAN coordinator receives this frame, it must generate an ADPM-NETWORK-STATUS.indication primitive, with the Status set to PAN_ID_CONFLICT, and the AdditionalInformation set to the conflicting PAN Id.

[Page intentionally left blank]

4 Security

The OFDM CPL network, as defined in the present specification, provides several security services:

Access Control and Authentication:

An End Device (ED) may not access to the network without a preliminary Identification (with comparison to white or black lists) and Authentication. Identification and Authentication are based on two parameters that personalized every ED:

- A EUI-48 MAC address as defined in [802-2001]. This address may be easily converted into a EUI-64 as required by [802.15.4-2006] and related documents. It is considered as public.

- A 128-bit shared secret (aka Pre-Shared Key or PSK) used as a credential during the authentication process. It is shared by the ED itself (aka peer) and an authentication server. The mutual authentication is based on a proof the other party knows the PSK. It is of highest importance, the PSK remains secret.

The Identification and Authentication processes are activated when an ED restarts and may also be launched at any time according to the security policy in place. The related material is carried by the 6LoWPAN Bootstrapping Protocol (LBP) (see §5.5) that embeds the Extensible Authentication Protocol (EAP) (see §6.2).

LBP and EAP have been designed to be relayed by intermediates nodes. Then during the Bootstrapping phase, when an ED (aka LBD) that have not yet acquired a routable 16-bit address, is a 1-hop distance of the PAN Coordinator (aka LBS) they can directly communicate. Otherwise, they must use an intermediate node (aka LBA) located at 1-hop distance of the LBD.

Moreover, two different authentication architectures must be considered:

- The authentication server function is directly supported by the LBS, and in this case all the authentication material (access lists, credentials, etc.) must be loaded in the LBS.

- The authentication server function is supported by a remote (and usually centralized) AAA server, and in this case, the LBS is only in charge of forwarding the EAP messages to the AAA server over a standard AAA protocol (i.e. RADIUS [rfc2865]).

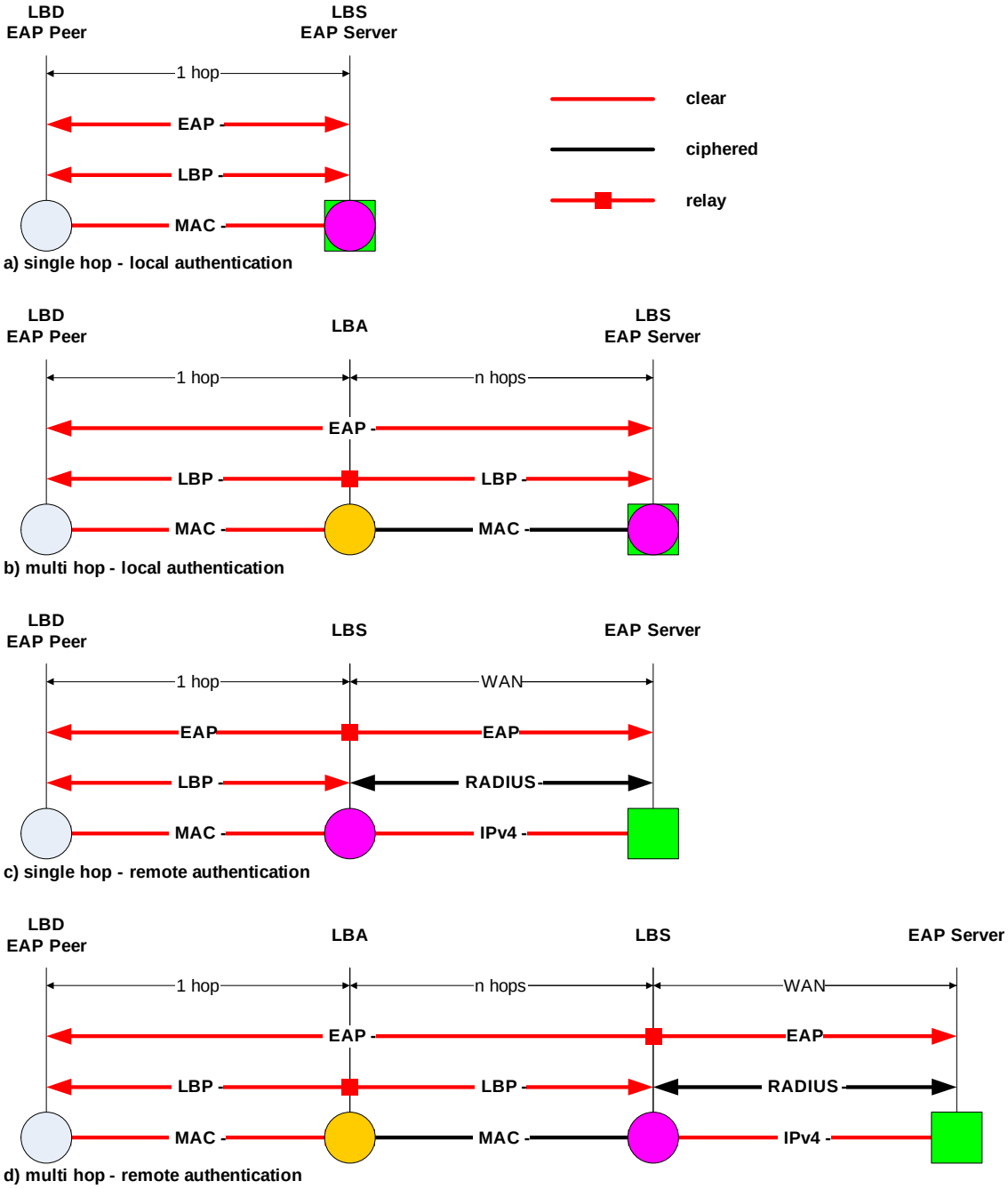


Figure 18 - LBP and EAP Relaying Capabilities

The Authentication process is fully dependant of the EAP method in place. The EAP protocol is very flexible and support various EAP methods (EAP-MD5, EAP-AKA, EAP-TLS, etc.). Each method is characterized by its credentials (shared secret, certificate, SIM cards, etc.) and by its signature and encryption algorithms.

The proposed method for the OFDM CPL network is EAP-PSK (see §6.2), the main design goals of which are:

- Simplicity: it is entirely based on a single credential (a 128-bit Pre-Shared Key) and a single cryptographic algorithm (AES-128).
- Security: it appears very conservative in its design following well-known and improved cryptographic schemes.
- Extensibility: in the OFDM CPL case, it is easily extended to support Group Key distribution (see 6.2.2)

Confidentiality and Integrity

Confidentiality and Integrity services are ensured at different level:

- At MAC level: as defined in [802.15.4-2006] a CCM* type of ciphering is delivered to every frame transmitted between nodes in the network. It's a universal Low Layer Confidentiality and Integrity service (with anti-replay capabilities). The MAC frames are encrypted and decrypted at every hop.
  The only exceptions are some well-controlled frames in the early stages of the Bootstrapping process.
  To fairly support this service, all the nodes in the network receive the same Group session key (GMK). This GMK is individually and securely distributed to every node by using the EAP-PSK Secure Channel.

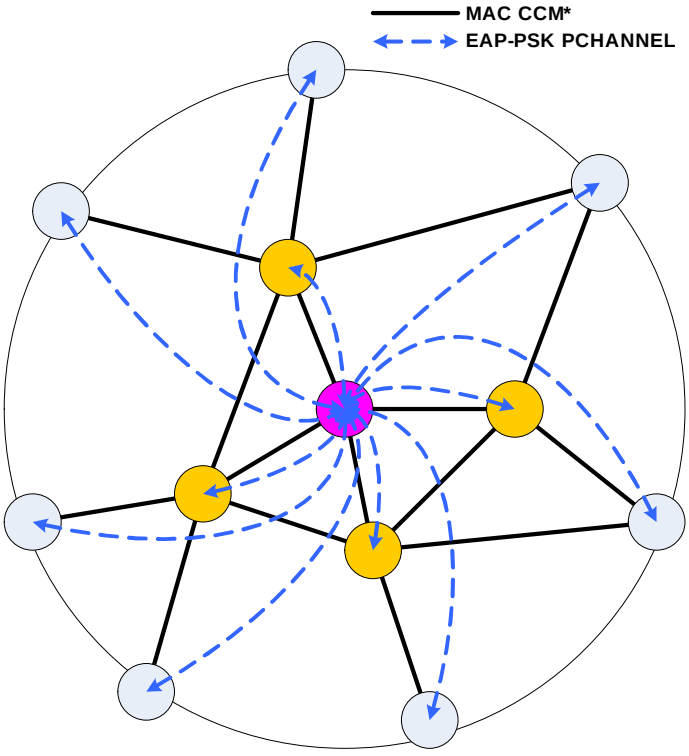


Figure 19 - Confidentiality and Security

- At the EAP-PSK level: as defined in [rfc4764], EAP-PSK provides Confidentiality and Integrity (and Replay Protection) services, also

known as Protected Channel (PCHANNEL) to the messages exchanged over EAP between the EAP server and any peer.

- At the Application layer, but these features are out of the scope of the present document.

Anti-Replay and DoS prevention

It is always difficult to prevent DoS attacks, and especially those targeting the Physical level, but by nature their impact is limited to a small area.

The CCM* ciphering mode is generalized at MAC layer. It prevents unauthenticated devices accessing the network and having malicious actions on routing, provisioning and any other Low Layer processes. The only exception is the well-controlled Bootstrapping process.

Moreover, an anti-replay mechanism is specified at the MAC layer.

4.1 Authentication and Key Distribution protocol

Authentication and Key Distribution are supported by the Extensible Authentication Protocol (EAP) as given in [rfc3748] together with the following statements and modifications.

Table 19- Modifications and statements to clauses of [rfc3748]

Clause	Title & remarks/modifications	Statement
1.	Introduction	N
2.	Extensible Authentication Protocol (EAP) - Initial Identity Request (allow roaming and EAP method negotiation) is let for further study and must be bypassed.	M
2.1	Support for Sequences	N
2.2	EAP Multiplexing Model - Only one EAP method is defined (cf. §6.2)	M
2.3	Pass-Through Behavior - Over LBP, the Code field is slightly different from a regular EAP Code field as specified in [rfc3748]. The conversion appears straightforward in both directions. The proper conversion must apply when the EAP message is propagated over another protocol (i.e. RADIUS) and in case of integrity protection covering the EAP header	M
2.4	Peer-to-Peer Operation	N
3.	Lower Layer Behavior	
3.1	Lower Layer Requirements - LBP and underlying protocols provide: - Reliable transport - Error detection (CRC) - No Lower Layer security when bootstrapping - MTU size greater than 1020 octets (by fragmentation) - No duplication - Ordering guaranties	N

Clause	Title & remarks/modifications	Statement
3.2	EAP Usage Within PPP	N/R
3.3	EAP Usage Within IEEE802	N/R
3.4	Lower Layer Indications	N
4.	EAP Packet Format - Over LBP, the Code field is slightly different from a regular EAP Code field. See §5.5.1.2	M
4.1	Request and Response - Same modification	M
4.2	Success and Failure - Same modification	M
4.3	Retransmission Behavior	N
5.	Initial EAP Request / Response Types - For the Type field, the only available values are 3 (Nak – in Response only) and the value assigned to the EAP method (see §7). Other values are left for further study	M
5.1.	Identity	N/R
5.2.	Notification	N/R
5.3.	Nak	N
5.4.	MD5-Challenge	N/R
5.5.	One-Time Password (OTP)	N/R
5.6.	Generic Token Card (GTC)	N/R
5.7.	Expanded Types	N/R
5.8.	Experimental	N/R
6.	IANA Considerations	N
7.	Security Considerations	N
8.	Acknowledgements	I
9.	References	N
Appendix A.	Changes from RFC2284	I

4.2 EAP Method

The EAP protocol is very flexible and support various EAP methods (EAP-MD5, EAP-AKA, EAP-TLS, etc.). Each method is characterized by its credentials (shared secret, certificate, SIM cards, etc.) and by its signature and encryption algorithms.
For the OFDM CPL case, the recommended method is Pre-Shared Key EAP Method (EAP-PSK) as given in [rfc4764] together with the following statements and modifications.

Table 20 - Modifications and statements to clauses of [rfc4764]

Clause	Title & remarks/modifications	Statement
1.	Introduction	N
2.	Protocol Overview	N
3.	Cryptographic Design of EAP-PSK	N
4.	EAP-PSK Message Flows - EAP-PSK extension capabilities are used for Group Key distribution in full compliance to [rfc4764]. See §6.2.2	N
5.	EAP-PSK Message Format - Same remark	N
6.	Rules of Operation for EAP-PSK Protected Channel	N
7.	IANA Considerations	N

Clause	Title & remarks/modifications	Statement
8.	Security Considerations	N
9.	Security Claims	I
10.	Acknowledgements	I
11.	References	N
Appendix A.	Generation of the PSK from a Password - Discouraged	N/R

4.2.1 Overview of EAP-PSK (informative)

EAP-PSK, according to the EAP specification supports the following key hierarchy:

Pre-Shared Key (PSK)
PSK is the long-term 128-bit credential shared by the EAP server and the peer

Authentication Key (AK)
A 128-bit key derived from the PSK that the EAP peer and server use to mutually authenticate

Key-Derivation Key (KDK)
A 128-bit key derived from the PSK that the EAP peer and server use to derive session keys (such as TEK, MSK and EMSK)

Transient EAP Key (TEK)
A session key that is used to establish a protected channel between the EAP peer and server during the EAP authentication. EAP-PSK uses a 128-bit TEK in conjunction with AES-128 in EAX mode of operation as a cipher suite.

Master Session Key (MSK)
A session key derived between the EAP peer and server. EAP-PSK generates a 512-bit MSK that may be used to provide security at the Application level.

Extended Master Session Key (EMSK)
A session key derived between the EAP peer and server. EAP-PSK generates a 512-bit EMSK. It is not used in OFDM CPL and must not be generated.

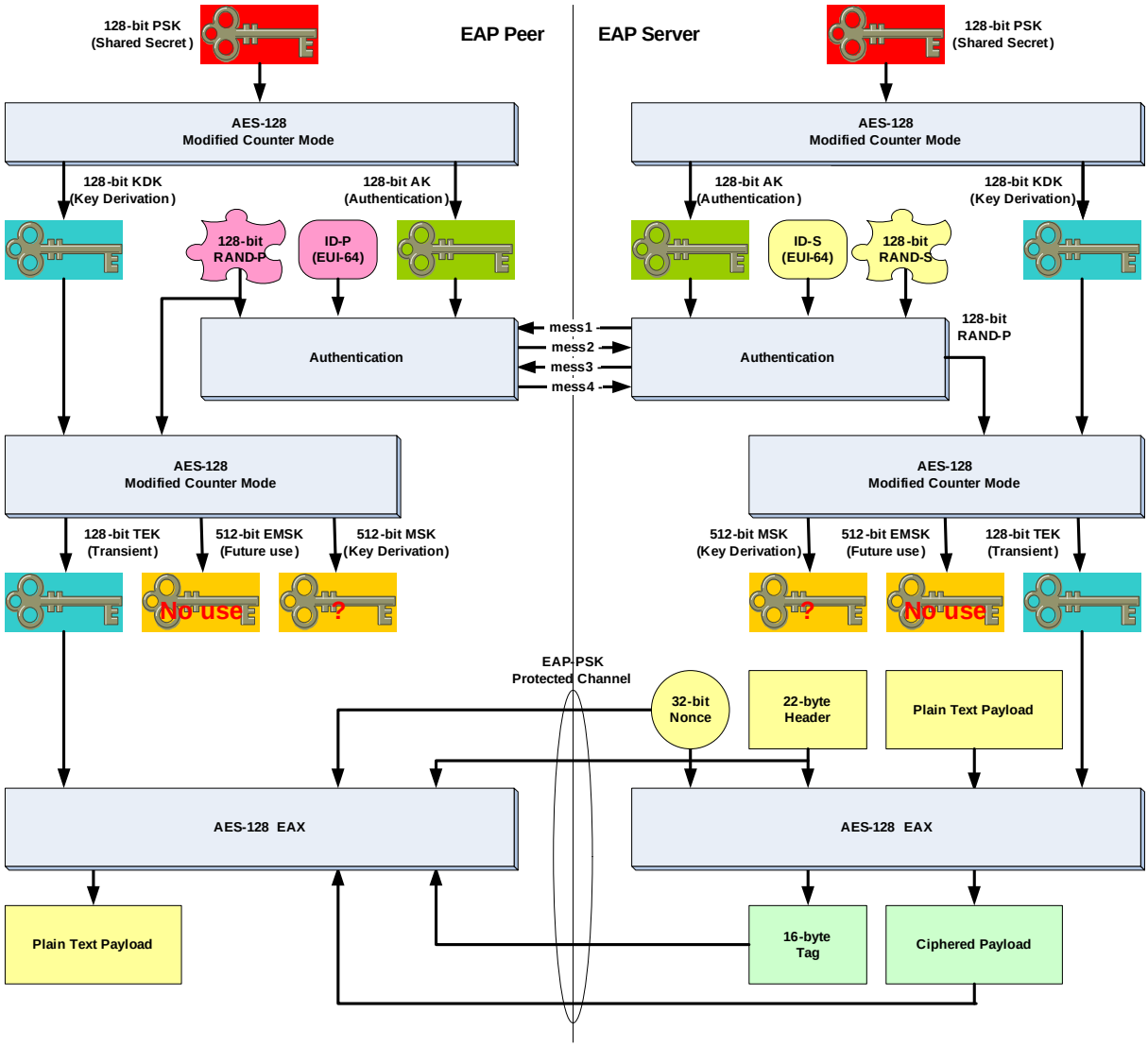


Figure 20 - EAP-PSK Key Hierarchy overview

4.2.2 Group Key distribution (normative)

The 128-bit Group Master Key (GMK) is generated by the EAP Server. Then it is securely and individually delivered to the EAP peers via the EAP-PSK Protected Channel (PCHANNEL).

GMK is assumed being random. GMK generation is considered as purely implementation dependant.

GMK is distributed to the peer in two circumstances:

- During the Bootstrapping process, carried as a regular extension to EAP-PSK message 3.
- During the Re-keying process, carried as a regular extension to EAP-PSK message 5. The GMK lifetime is rather long (several 10s years) due to the 4 byte counter included in the nonce. Nevertheless it's of good policy to timely re-key the network or when a node is leaving it.

4.2.3 GMK field format

The GMK field in message 3 or 5 is defined in compliance with the generic extension field (EXT) (see [rfc4764] clause 5.3.).

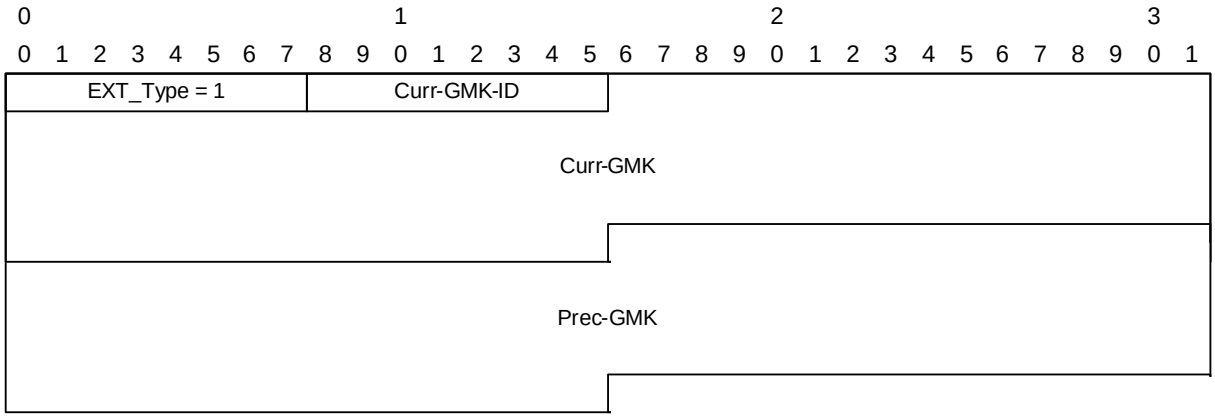


Figure 21 - GMK field format

EXT_Type	The EXT_TYPE field is one octet and indicates the type of the Extension 1 GMK
Curr-GMK-ID	The Curr-GMK-ID field is one octet and represents the Key Identifier of the current GMK
Curr-GMK	The Curr-GMK is 16 octets and contains the value of the current GMK
Prec-GMK	The Prec-GMK is 16 octets and contains the value of the preceding GMK

4.2.4 Peer side procedure

When a peer receives this field embedded in a message 3 in case of Bootstrapping, or a message 5 in case of Re-keying, it installs both keys in their respective slot and process the message according to [rfc4764].

In every case, both keys are immediately available in reception according to the Key Identifier contained in the MAC header of the received frame.

In case of Bootstrapping, the peer keeps sending the frames in clear text up to the reception of an EAP Success message. Then, it starts sending ciphered frames using the current GMK.

In case of Re-keying, the peer keeps sending messages according to the previously assigned policy until reception of an EAP Success message. Then, it starts sending frames using the current GMK.

After switching to the current GMK, a peer may keep receiving some messages encrypted with the preceding GMK during a transient period. The previous GMK must be deleted after a configurable delay (default value = 10 min).

4.2.5 Server side procedure

The Bootstrapping procedure is defined in §5.5.2.

In case of re-keying, the EAP server generates a new GMK.

Then it transmits a LBP challenge message, embedding an EAP Request message that contains the newly generated GMK, coupled with the preceding one, to every formerly authenticated peer.

Upon reception of the corresponding EAP Response, or after a configurable delay (default value = 10 min), the server starts sending EAP Success messages for the validation of the new GMK.

[Page intentionally left blank]

5 Annexes

5.1 Annex 1: Protocol Implementation Conformance Statement

This annex is entirely taken as a reference for the present specification from the document 802.15.4-2006. The protocol implementation conformance tables are the following ones:

Table 21 - PICS table 1 for 802.15.4-2006

	Support			Comments
	N/R	Yes	No	
FD1		X		
FD2			X	
FD3		X		
FD4		X		
FD5		X		
PLF1		X		
PLF2		X		
PLF3	X			Radio specific requirement
PLF4	X			Radio specific requirement
PLF5	X			Radio specific requirement
PLF6		X		
PLF7	X			Radio specific requirement
PLF8		X		
PLF8.1	X			Radio specific requirement
PLF8.2		X		
PLF8.3	X			Radio specific requirement
PLP1		X		
RF1	X			Radio specific requirement
RF1.1	X			Radio specific requirement
RF1.2	X			Radio specific requirement
RF1.3	X			Radio specific requirement
RF1.4	X			Radio specific requirement
RF2	X			Radio specific requirement
MLF1		X		
MLF1.1			X	Indirect transmission is not supported
MLF2		X		
MLF2.1		X		
MLF2.2		X		
MLF2.3		X		
MLF3		X		
MLF3.1		X		
MLF3.2		X		
MLF4		X		
MLF5			X	
MLF5.1			X	
MLF5.2			X	

MLF6		X		
MLF7		X		
MLF8			X	Performed by 6LoWPAN
MLF9		X		
MLF9.1		X		
MLF9.2		X		
MLF9.2.1		X		
MLF9.2.2		X		
MLF10.1	X			Radio specific requirement
MLF10.2		X		
MLF10.3			X	Not necessary for non beacon-enabled networks
MLF10.4			X	
MLF11			X	
MLF12			X	
MLF13			X	

Table 22 - PICS table #2 for 802.15.4-2006

	Support		***Comments***
	Tx	***Rx***	
MF1	Yes	Yes	
MF2	Yes	Yes	
MF3	Yes	Yes	Acknowledgement frames are described in PHY specification associated with the present specification and annex 6 of the present document
MF4	Yes	Yes	
MF4.1	No	No	Association performed by 6LoWPAN
MF4.2	No	No	Association performed by 6LoWPAN
MF4.3	No	No	Association performed by 6LoWPAN
MF4.4	No	No	No transaction support
MF4.5	No	No	Performed by 6LoWPAN
MF4.6	No	No	
MF4.7	Yes	Yes	
MF4.8	No	No	
MF4.9	No	No	

5.2 Annex 2: Routing Cost

This part describes the characteristics a routing cost used in the LOAD routing algorithm (described in [draft-load] and clause 5.4 of the present document) must have.
A route cost is defined as the sum of all the link costs on the route. As described in [draft-load], a route cost is an integer value between 0 and 255, lower values meaning better routes. As there can be at most 8 hops on a route as defined in clause 5.2 of the present document (modified clause 5.2 from [rfc4944]), a link cost is an integer between 0 and 31.

If we note P a route which goes through devices $\{D_0, D_1, ..., D_{N-1}\}$, where N is the number of hops on the route ($0 < N \leq 8$), and $C\{[D_i, D_j]\}$ the link cost between devices D_i and D_j, the route cost $RC(P)$ of P can then be defined as

$$RC(P) = \sum_{i=0}^{N-1} C\{[D_i, D_{i+1}]\}$$

The link cost should take into account PHY transmission parameters, number of hops, etc… Complete description of a link cost is out of the scope of this document.

[Page intentionally left blank]

5.3 Annex 3: Adaptation Layer Service Primitives

5.3.1 ADP Data service

The ADPD is used to transport application layer PDU to other devices on the network, and supports the following primitives:

- ADPD-DATA.request
- ADPD-DATA.confirm
- ADPD-DATA.indication

5.3.1.1 ADPD-DATA.request

This primitive requests the transfer of an application PDU to another device, or multiple devices.

5.3.1.1.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPD-DATA.request	{
	NsduLength,
	Nsdu,
	NsduHandle,
	DiscoverRoute,
	QualityOfService,
	SecurityEnabled
	}

Table specifies the parameters for the ADPD-DATA.request primitive.

Table 23- Parameters of the ADPD-DATA.request primitive

Name	Type	Valid Range	Description
NsduLength	Integer	0 - 1280	The size of the NSDU, in bytes
Nsdu	Set of octets	-	The NSDU to send
NsduHandle	Integer	0x00 - 0xFF	The handle of the NSDU to transmit. This parameter is used to identify in the ADPD-DATA.confirm primitive which request is concerned. It can be randomly chosen by the application layer.
DiscoverRoute	Boolean	TRUE or FALSE	If TRUE, a route discovery procedure will be performed prior to sending the frame if a route to the destination is not available in the routing table. If FALSE, no route discovery is performed.
QualityOfService	Integer	0x00 - 0x01	The required quality of service (QoS) of the frame to send. Allowed values are: 0x00 = standard priority 0x01 = high priority
SecurityEnabled	Boolean	TRUE or FALSE	If TRUE, this parameter enables the ADP layer security for processing the frame.

5.3.1.1.2 When generated

This primitive is generated by the upper layer to request the sending of a given NSDU.

5.3.1.1.3 Effect on receipt

If this primitive is received when the device has not joined a network, the ADP layer will issue an ADPD-DATA.confirm primitive with the status INVALID_REQUEST.

Else, the ADPD constructs a 6LoWPAN frame with the following characteristics, if unicast frame:

- The mesh addressing header is present as described in part 5.2 of [RFC4944], where
  - V must be set to 1, to specify that the originator address is a 16-bit network address
  - F must be set to 1, to specify that the originator address is a 16-bit network address
  - HopsLft = MaxHops
  - Originator address = The 16-bit network address of the sending device, available in the NIB
  - Final destination address = 16-bit destination address of the device designated by the IPv6 address “DstAddr”

- The broadcast header is not present

- If necessary, the fragmentation header must be present to transport NPDU which do not fit in an entire 802.15.4 frame. In that case, clause 5.3 of [RFC4944] applies.

- LOWPAN_HC1 compressed IPv6 header is present with the following parameters:
  - IPv6 Source Address mode = PC-IC (bits 0 and 1 set to 1)
  - IPv6 Destination Address mode = PC-IC (bits 2 and 3 set to 1)
  - Bit 4 = 1 (no Traffic Class and Flow Label)
  - Bits 5 and 6 = value of NsduType

If multicast frame, the ADPD constructs a 6LoWPAN frame with the following characteristics:

- The mesh addressing header is present as described in part 5.2 of [RFC4944], where
  - V must be set to 1, to specify that the originator address is a 16-bit network address
  - F must be set to 1, to specify that the originator address is a 16-bit network address

  - HopsLft = MaxHops
  - Originator address = The 16-bit network address of the sending device, available in the NIB
  - Final destination address = 0xFFFF

- The broadcast header is present with the following values:
  - Sequence Number = previous Sequence Number + 1

- If necessary, the fragmentation header must be present to transport NPDU which do not fit in an entire 802.15.4 frame. In that case, clause 5.3 of [RFC4944] applies.

- LOWPAN_HC1 compressed IPv6 header is present with the following parameters:
  - IPv6 Source Address mode = PC-IC (bits 0 and 1 set to 1)
  - IPv6 Destination Address mode = PC-IC (bits 2 and 3 set to 1)
  - Bit 4 = 1 (no Traffic Class and Flow Label)
  - Bits 5 and 6 = value of NsduType

Once the frame is constructed, it is routed according to the procedures described in clause 5.4 of the present document (modified clauses 6.x of [draft-load]) if the destination address is a unicast address. If the frame is to be transmitted, the MCPS-Data.request primitive is invoked, with the following parameters in case of a unicast sending:

- SrcAddrMode = 0x02, for 16-bit address
- DstAddrMode = 0x02, for 16-bit address
- SrcPANId = DstPANId = the value of macPANId obtained from the MAC PIB
- SrcAddr = the value of macShortAddr obtained from the MAC PIB
- DstAddr = the 16-bit address of the next hop determined by the routing procedure
- msduLength = the length of the frame, or fragment in case of fragmentation, in bytes
- msdu = the frame itself
- msduHandle = NsduHandle
- TxOptions:
  - b0 = 1 if unicast transmission, 0 otherwise
  - b1 = 0
  - b2 = 0
- SecurityLevel = 0 if SecurityEnabled = FALSE, 5 if SecurityEnabled = TRUE
- KeyIdMode, KeySource : Ignored
- KeyIndex : Ignored if SecurityLevel=0; Else depends on security policy

In case of a broadcast (or multicast) frame, the MCPS-Data.request primitive is invoked with the following parameters:

- SrcAddrMode = 0x02, for 16-bit address
- DstAddrMode = 0x02, for 16-bit address

- SrcPANId = DstPANId = the value of macPANId obtained from the MAC PIB
- SrcAddr = the value of macShortAddr obtained from the MAC PIB
- DstAddr = 0xFFFF
- msduLength = the length of the frame, or fragment in case of fragmentation, in bytes
- msdu = the frame itself
- msduHandle = NsduHandle
- TxOptions:
    - b0 = 1 if unicast transmission, 0 otherwise
    - b1 = 0
    - b2 = 0
- SecurityLevel = 0 if SecurityEnabled = FALSE, 5 if SecurityEnabled = TRUE
- KeyIdMode, KeySource : Ignored
- KeyIndex : Ignored if SecurityLevel=0; Else depends on security policy

If security processing fails for that frame, it must be discarded and an ADPD-DATA.confirm primitive must be generated with the status code returned by the security processing suite.

If the DiscoverRoute parameter is set to TRUE, then the route discovery procedure should be initiated prior to sending the frame in case the final destination address is not available in the routing table. For a complete description of this procedure, refer to clause 5.4.3.1 of the present document.

5.3.1.2 ADPD-DATA.confirm

This primitive reports the result of a previous ADPD-DATA.request primitive.

5.3.1.2.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPD-DATA.confirm	{
	Status,
	NsduHandle,
	}

Table specifies the parameters for the ADPD-DATA.confirm primitive.

Table 24- Parameters of the ADPD-DATA.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_IPV6_FRAME, INVALID_REQUEST, NO_KEY, BAD_CCM_OUTPUT, ROUTE_ERROR, BT_TABLE_FULL, FRAME_NOT_BUFFERED or any status values returned from security suite or the MCPS-DATA.confirm primitive	The status code of a previous NDS-DATA.request identified by its NsduHandle
NsduHandle	Integer	0x00 - 0xFF	The handle of the NSDU confirmed by this primitive

5.3.1.2.2 When generated

This primitive is generated in response to a NDS-DATA.request primitive, the Status parameter indicating if the request succeeded, or the reason of failure.

5.3.1.2.3 Effect on receipt

On reception of this primitive, the upper layer is notified of the status of a previous NDS-DATA.request primitive.

5.3.1.3 ADPD-DATA.indication

This primitive is used to transfer received data from the ADP layer to the upper layer.

5.3.1.3.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPD-DATA.indication	{
	NsduLength,
	Nsdu,
	LinkQualityIndicator,
	SecurityEnabled
	}

Table specifies the parameters for the ADPD-DATA.indication primitive.

Table 25- Parameters of the ADPD-DATA.indication primitive

Name	Type	Valid Range	Description
NsduLength	Integer	0-1280	The size of the NSDU, in bytes
Nsdu	Set of octets	-	The received NSDU
LinkQualityIndicator	Integer	0x00-0xFF	The value of the link quality during reception of the frame
SecurityEnabled	Boolean	TRUE or FALSE	TRUE if the frame was received using security.

5.3.1.3.2 When generated

This primitive is generated by the ADP layer when a valid data frame whose final destination is the current station has been received.

5.3.1.3.3 Effect on receipt

On generation of this primitive, the upper layer is notified of the arrival of a data frame.

5.3.2 ADP Management service

The ADPM allows the transport of command frames used for network maintenance. The list of primitives supported by the ADPM is:

- ADPM-DISCOVERY.request, ADPM-DISCOVERY.confirm
- ADPM-NETWORK-START.request, ADPM-NETWORK-START.confirm
- ADPM-NETWORK-JOIN.request, ADPM-NETWORK-JOIN.confirm, ADPM-NETWORK-JOIN.indication
- ADPM-NETWORK-LEAVE.request, ADPM-NETWORK-LEAVE.indication, ADPM-NETWORK-LEAVE.confirm
- ADPM-RESET.request, ADPM-RESET.confirm
- ADPM-GET.request, ADPM-GET.confirm
- ADPM-SET.request, ADPM-SET.confirm
- ADPM-NETWORK-STATUS.indication
- ADPM-ROUTE-DISCOVERY.request, ADPM-ROUTE-DISCOVERY.confirm

5.3.2.1 ADPM-DISCOVERY.request

This primitive allows the upper layer to request the ADPM to scan for networks operating in its POS.

5.3.2.1.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-DISCOVERY.request	{
	Duration,
	}

Table specifies the parameters for the ADPM-DISCOVERY.request primitive.

Table 26- Parameters of the ADPM-DISCOVERY.request primitive

Name	Type	Valid Range	Description
Duration	Integer	0x00-0xFF	The number of seconds the active scan must last

5.3.2.1.2 When generated

This primitive is generated by the next upper layer to get informed of the current networks operating in the POS of the device.

5.3.2.1.3 Effect on receipt

On receipt of this primitive, the ADP layer will initiate an active scan by invoking the MLME-SCAN.request with the following parameters:

- ScanType = 0x01 for active scan
- ScanChannels = all bits set to 0 (not used)
- ScanDuration = Duration
- ChannelPage = 0 (not used)
- SecurityLevel = 0
- KeyIdMode, KeySource and KeyIndex: Ignored

On receipt of the MLME-SCAN.confirm primitive, the ADP layer issues an ADPM-DISCOVERY.confirm primitive containing the PAN ID of all the PANs operating in the POS of the device, or an error code.

5.3.2.2 ADPM-DISCOVERY.confirm

This primitive is generated by the ADP layer upon completion of a previous ADPM-DISCOVERY.request.

5.3.2.2.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-DISCOVERY.confirm	{
	Status,
	PANCount,
	PANDescriptor
	}

Table 27 specifies the parameters for the ADPM-DISCOVERY.request primitive.

Table 27- Parameters of the ADPM-DISCOVERY.request primitive

Name	Type	Valid Range	Description
Status	Status	Any status returned by the MLME-SCAN.confirm primitive	See [802.15.4-2006] for complete list of status codes and their meaning
PANCount	Integer	0x00-0xFF	The number of networks operating in the POS of the device
PANDescriptior	List of PAN descriptors	This list contains PANCount PAN descriptors as described in Table	The PAN operating in the POS of the device

Table 28 specifies the parameters of a PAN descriptor.

Table 28 - PAN descriptor structure specification

Name	Type	Valid Range	Description
ExtendedPANId	List of integers	This list contains the PAN IDs of the network found. The size of the list is PANCount. Each ExtendedPANId must be in the range 0x000000000001-0xFFFFFFFFFFFE	The list of 64-bit PAN identifiers.
PANId	List of integers	This list contains the 16-bit PAN IDs of the network found. The size of the list is PANCount, and its elements appear in the same order as the ExtendedPANId list. Each PANId must be in the range 0x0000-0xFFFF	The list of 16-bit PAN identifiers.

5.3.2.2.2 When generated

This primitive is generated by the ADP layer for the upper layer on completion of a ADPM-DISCOVERY.request primitive.

5.3.2.2.3 Effect on receipt

On receipt of this primitive, the upper layer is notified of the completion of the network scan, and obtains a list of found operating networks.

5.3.2.3 ADPM-NETWORK-START.request

This primitive allows the upper layer to request the starting of a new network. It must only be invoked by device designated as the PAN coordinator during the factory process.

5.3.2.3.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-NETWORK-START.request	{
	PANId
	}

Table specifies the parameters for the ADPM-NETWORK-START.request primitive.

Table 29- Parameters of the ADPM-NETWORK-START.request primitive

Name	Type	Valid Range	Description
PANId	Integer	0x0000-0xFFFF	The PANId of the network to create, determined at the application level

5.3.2.3.2 When generated

This primitive is generated by the upper layer of the PAN coordinator to start a new network.

5.3.2.3.3 Effect on receipt

On receipt of this primitive by a device which is not a PAN coordinator, it must issue an ADPM-NETWORK-START.confirm primitive with the status INVALID_REQUEST.

Prior to invoking this primitive, the upper layer of the PAN coordinator should perform an ADPM-DISCOVERY.request to make sure no other network is currently operating. In case another network is operating, the upper layer may invoke the ADPM-NETWORK-START.request.

On receipt of this primitive by a device which is the PAN coordinator, and if no network has already be formed, the ADP layer must perform the steps described in clause 5.7.1 of the present document.

On receipt of the MLME-START.confirm primitive, the ADP layer must issue a ADPM-NETWORK-START.confirm primitive with the appropriate status code.

5.3.2.4 ADPM-NETWORK-START.confirm

This primitive reports the status of a ADPM-NETWORK-START.request.

5.3.2.4.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-NETWORK-START.confirm	{
	Status
	}

Table 30 specifies the parameters for the ADPM-NETWORK-START.confirm primitive.

Table 30- Parameters of the ADPM-NETWORK-START.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_REQUEST, STARTUP_FAILURE or any status value returned from the MLME-START.confirm primitive	The result of the attempt to create the network

5.3.2.4.2 When generated

This primitive is generated by the ADP layer in response to an ADPM-NETWORK-START.request primitive, and indicates if the network formation was successful or not, and an eventual reason for failure.

5.3.2.4.3 Effect on receipt

On receipt of this primitive, the next higher layer is notified about the status of its previous ADPM-NETWORK-START.request.

5.3.2.5 ADPM-NETWORK-JOIN.request

This primitive allows the next upper layer to join an existing network.

5.3.2.5.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-NETWORK-JOIN.request {
                          PANId,
                          LBAAddress
                          }
```

Table 5 specifies the parameters for the ADPM-NETWORK-START.confirm primitive.

Table 5- Parameters of the ADPM-NETWORK-START.confirm primitive

Name	Type	Valid Range	Description
PANId	Integer	0x0000-0xFFFF	The 16-bit PAN identifier of the network to join
LBAAddress	16-bit address	0x0000-0xFFFF	The 16-bit short address of the device acting as a LoWPAN Bootstrap Agent as defined in [draft-commissioning]

5.3.2.5.2 When generated

The upper layer invokes this primitive when it wishes to join an existing PAN using the MAC association procedure.

5.3.2.5.3 Effect on receipt

On receipt of this primitive by a device which is already joined, the ADP layer generates an ADPM-NETWORK-JOIN.confirm with the status INVALID_REQUEST.

On receipt of this primitive by a device which is not already joined, the ADP layer initiates the MAC association procedure (“bootstrap”) described in clause 5.5.2 of the present document.

On completion, an MLME-SET.request is invoked to set the 16-bit short address of the device which was obtained during the “bootstrapping” phase. Then an ADPM-NETWORK-JOIN.confirm primitive is generated with a status of SUCCESS.

5.3.2.6 ADPM-NETWORK-JOIN.confirm

This primitive is generated by the ADP layer to indicate the completion status of a previous ADPM-NETWORK-JOIN.request.

5.3.2.6.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-NETWORK-JOIN.confirm {
                          Status,
                          NetworkAddress,
                          PANId
                          }
```

Table 32 specifies the parameters for the ADPM-NETWORK-JOIN.confirm primitive.

Table 32- Parameters of the ADPM-NETWORK-JOIN.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_REQUEST, NOT_PERMITTED.	The result of the attempt to join the network
NetworkAddress	Integer	0x0001-0xFFF7 and 0xFFFF	The 16-bit network address that was allocated to the device. If the allocation fails, this address is equal to 0xFFFF.
PANId	Integer	0x0000-0xFFFF	The 16-bit address of the PAN of which the device is now a member.

5.3.2.6.2 When generated

This primitive is generated in response to an ADPM-NETWORK-JOIN.request primitive, and allows the upper layer to obtain information on the status of its request.

The status NOT_PERMITTED is given if the device was unable to authenticate itself to the PAN coordinator.

5.3.2.6.3 Effect on receipt

On receipt of this primitive the upper layer is informed on the status of its request.

5.3.2.7 ADPM-NETWORK-JOIN.indication

This primitive allows the upper layer of the PAN coordinator to be notified when a new device has successfully completed the association procedure, and is now part of the network.

5.3.2.7.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-NETWORK-JOIN.indication	{
	NetworkAddress,
	ExtendedAddress,
	CapabilityInformation,
	}

Table 6 specifies the parameters for the ADPM-NETWORK-JOIN.indication primitive.

Table 6- Parameters of the ADPM-NETWORK-JOIN.indication primitive

Name	Type	Valid Range	Description
NetworkAddress	IPv6 address	See [rfc4944]	The IPv6 network address of the device that was added to the network. This address was given during the association procedure.
ExtendedAddress	64-bit address	0x000000000001-0xFFFFFFFFFFFE	The 64-bit address of the device that was added to the network. This address is unique for any device.
CapabilityInformation	Bitmap	See Figure 56 of document [802.15.4-2006]	The capability information field of the device

5.3.2.7.2 When generated

This primitive is generated by the ADP layer upon successful completion of the association of a new device.

5.3.2.7.3 Effect on receipt

The upper layer is notified of the completion of the association.

5.3.2.8 ADPM-NETWORK-LEAVE.request

This primitive allows the PAN coordinator to remove a device from the network, or allows a device to remove itself from the network.

5.3.2.8.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-NETWORK-LEAVE.request	{
	ExtendedAddress,
	}

Table 34 specifies the parameters for the ADPM-NETWORK-LEAVE.request primitive.

Table 34- Parameters of the ADPM-NETWORK-LEAVE.request primitive

Name	Type	Valid Range	Description
ExtendedAddress	64-bit address	Any	The 64-bit network address of the device to remove from the network. If NULL, the device removes itself from the network.

5.3.2.8.2 When generated

The next higher layer generates this primitive to leave the network, or to request another device to do so.

5.3.2.8.3 Effect on receipt

On receipt of this primitive by a device which is not associated to any network, the ADP layer must issue an ADPM-NETWORK-LEAVE.confirm primitive with the status INVALID_REQUEST.

On receipt of this primitive by a device which is associated to any network, the following steps must be performed:

- If the device is a coordinator
  - If ExtendedAddress == NULL
    - Issue ADPM-NETWORK-LEAVE.confirm with INVALID_REQUEST
  - Else
    - If the device exists
      - Remove the device which has the address ExtendedAddress from the network using the procedure described in 5.5.2.6.1 of the present document

- Issue ADPM-NETWORK-LEAVE.confirm with SUCCESS
    - Issue ADPM-NETWORK-LEAVE.confirm with UNKNOWN_DEVICE

- Else (device is not a coordinator)
    - If ExtendedAddress == NULL
        - The device removes itself from the network, using the procedure described in clause 5.5.2.6.2 of the present document.
        - Issue ADPM-NETWORK-LEAVE.confirm with SUCCESS
    - Else
        - Issue ADPM-NETWORK-LEAVE.confirm with INVALID_REQUEST

5.3.2.9 ADPM-NETWORK-LEAVE.indication

This primitive is generated by the ADP layer to inform the upper layer that a device has been unregistered from the network.

5.3.2.9.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-NETWORK-LEAVE.indication	{
	ExtendedAddress,
	}

Table 3535 specifies the parameters for the ADPM-NETWORK-LEAVE.indication primitive.

Table 35- Parameters of the ADPM-NETWORK-LEAVE.indication primitive

Name	Type	Valid Range	Description
ExtendedAddress	64-bit address	Any	The 64-bit network address of the device removed from the network.

5.3.2.9.2 When generated

This primitive is generated by the ADP layer of a device when it has been removed from the network by the PAN coordinator or by the ADP of the PAN coordinator when a device has decided to leave the network.

5.3.2.9.3 Effect on receipt

On receipt of this primitive, the upper layer of the device is notified that it is no more a part of the PAN.

5.3.2.10 ADPM-NETWORK-LEAVE.confirm

This primitive allows the upper layer to be informed on the status of its previous ADPM-NETWORK-LEAVE.request. This request can be either to leave by itself the network, or to instruct another device to leave (PAN coordinator only)

5.3.2.10.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-NETWORK-LEAVE.confirm	{
	Status,
	ExtendedAddress,
	}

Table 36 specifies the parameters for the ADPM-NETWORK-LEAVE.confirm primitive.

Table 36- Parameters of the ADPM-NETWORK-LEAVE.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_REQUEST, UNKNOWN_DEVICE or any status returned by the MCPS-DATA.confirm primitive	The status of the request
ExtendedAddress	64-bit address	Any	The 64-bit network address of the device removed from the network.

5.3.2.10.2 When generated

This primitive is generated on completion of a device removal. If it is successful, the SUCCESS code is given. Else, an error status is given as explained in clause 5.5.2.6 of the present document.

5.3.2.10.3 Effect on receipt

On receipt, the upper layer is notified of the result of its request.

5.3.2.11 ADPM-RESET.request

This primitive allows the upper layer to request that the ADP layer performs a reset.

5.3.2.11.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-RESET.request	{
	}

This primitive has no parameter

5.3.2.11.2 When generated

This primitive allows a reset of the ADP layer, and allows resetting the NIB attributes.

5.3.2.11.3 Effect on receipt

On receipt of this primitive, the following steps are performed:

- The ADP layer issue a MLME-RESET.request primitive with the SetDefaultPIB parameter set to TRUE, and waits for the MLME-RESET.confirm primitive
- The ADP layer clears all of its internal variables, and flushes its routing and neighbor tables
- The ADP layer issues an ADPM-RESET.confirm primitive with the status SUCCESS, or DISABLE_TRX_FAILURE if the MAC reset operation failed.

5.3.2.12 ADPM-RESET.confirm

This primitive allows the upper layer to be notified of the completion of an ADPM-RESET.request primitive.

5.3.2.12.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-RESET.confirm    {
                      Status,
                      }
```

Table 37 specifies the parameters for the ADPM-RESET.confirm primitive.

Table 37- Parameters of the ADPM-RESET.confirm primitive

Name	Type	Valid Range	Description
Status	Status	Any status value returned from the MLMERESET.confirm primitive	The status of the request

5.3.2.12.2 When generated

This primitive is generated by the ADP layer when a previous ADPM-RESET.request primitive has completed.

5.3.2.12.3 Effect on receipt

The upper layer is notified of the completion of the command.

5.3.2.13 ADPM-GET.request

This primitive allows the upper layer to get the value of an attribute from the information base.

5.3.2.13.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-GET.request	{
	AttributeId,
	AttributeIndex
	}

Table 38 specifies the parameters for the ADPM-GET.request primitive.

Table 38- Parameters of the ADPM-GET.request primitive

Name	Type	Valid Range	Description
AttributeId	Integer	See clause 5.2 of the present document	The identifier of the IB attribute to read
AttributeIndex	Integer	Depends on attribute, see clause 5.2 of the present document	The index within the table of the specified IB attribute to read. This parameter is valid only for IB attributes that are tables.

5.3.2.13.2 When generated

This primitive is generated by the upper layer to read the value of an attribute from the IB.

5.3.2.13.3 Effect on receipt

On receipt of this primitive, the adaptation layer attempts to retrieve the selected attribute in the information base. If the attribute is not found, the adaptation layer generates an ADPM-GET.confirm primitive with the status UNSUPPORTED_ATTRIBUTE. If the attribute is found (and is a table), but the AttributeIndex is out of range, the adaptation layer generates an ADPM-GET.confirm primitive with the status INVALID_INDEX.

Else, the adaptation layer generates an ADPM-GET.confirm primitive with the status SUCCESS, and the value read from the IB in the AttributeValue parameter.

5.3.2.14 ADPM-GET.confirm

This primitive allows the upper layer to be informed of the status of a previously issued ADPM-GET.request primitive.

5.3.2.14.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-GET.confirm{
                    Status,
                    AttributeId,
                    AttributeIndex,
                    AttributeValue
                    }
```

Table specifies the parameters for the ADPM-GET.confirm primitive.

Table 39- Parameters of the ADPM-GET.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, UNSUPPORTED_ATTRIBUTE or INVALID_INDEX	The status of the reading
AttributeId	Integer	See clause 5.2 of the present document	The identifier of the IB attribute read
AttributeIndex	Integer	Depends on attribute, see clause 5.2 of the present document	The index within the table of the specified IB attribute read. This parameter is valid only for IB attributes that are tables.
AttributeValue	Various	Attribute specific	The value of the attribute read from the IB

5.3.2.14.2 When generated

This primitive is generated by the adaptation layer in response to an ADPM-GET.request primitive.

5.3.2.14.3 Effect on receipt

On reception of this primitive, the upper layer is informed on the status of its request, and eventually gets the desired value.

5.3.2.15 ADPM-SET.request

This primitive allows the upper layer to set the value of an attribute in the information base.

5.3.2.15.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-SET.request{
                    AttributeId,
                    AttributeIndex,
                    AttributeValue
                    }
```

Table 40 specifies the parameters for the ADPM-SET.request primitive.

Table 40- Parameters of the ADPM-SET.request primitive

Name	Type	Valid Range	Description
AttributeId	Integer	See clause 5.2 of the present document	The identifier of the IB attribute to write
AttributeIndex	Integer	Depends on attribute, see clause 5.2 of the present document	The index within the table of the specified IB attribute to write. This parameter is valid only for IB attributes that are tables.
AttributeValue	Various	Depends on attribute	The value to write

5.3.2.15.2 When generated

This primitive is generated by the upper layer to write the value of an attribute in the IB.

5.3.2.15.3 Effect on receipt

On receipt of this primitive, the adaptation layer attempts to retrieve the selected attribute in the information base. If the attribute is not found, the adaptation layer generates an ADPM-SET.confirm primitive with the status UNSUPPORTED_ATTRIBUTE. If the attribute is found (and is a table), but the AttributeIndex is out of range, the adaptation layer generates an ADPM-SET.confirm primitive with the status INVALID_INDEX. If the attribute is found but is read only, the adaptation layer generates an ADPM-SET.confirm primitive with the status READ_ONLY. If the attribute is found, can be written, but the AttributeValue is out of range, the adaptation layer generates an ADPM-SET.confirm primitive with the status INVALID_PARAMETER. Else, the adaptation layer generates an ADPM-SET.confirm primitive with the status SUCCESS.

5.3.2.16 ADPM-SET.confirm

This primitive allows the upper layer to be informed about a previous ADPM-SET.request primitive.

5.3.2.16.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-SET.confirm{
                    Status,
                    AttributeId,
                    AttributeIndex
                    }
```

Table 41 specifies the parameters for the ADPM-SET.confirm primitive.

Table 41- Parameters of the ADPM-SET.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, UNSUPPORTED_ATTRIBUTE, READ_ONLY, INVALID_PARAMETER or INVALID_INDEX	The status of the writing
AttributeId	Integer	See clause 5.2 of the present document	The identifier of the IB attribute written
AttributeIndex	Integer	Depends on attribute, see clause 5.2 of the present document	The index within the table of the specified IB attribute written. This parameter is valid only for IB attributes that are tables.

5.3.2.16.2 When generated

This primitive is generated by the adaptation layer in response to an ADPM-SET.request primitive.

5.3.2.16.3 Effect on receipt

On reception of this primitive, the upper layer is informed on the status of its request.

5.3.2.17 ADPM-NETWORK-STATUS.indication

This primitive allows the next higher layer of a PAN coordinator or a coordinator to be notified when a particular event occurs on the PAN.

5.3.2.17.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-NETWORK-STATUS.indication  {
                                Status,
                                AdditionalInformation
                                }
```

Table 42 specifies the parameters for the ADPM-NETWORK-STATUS.indication primitive.

Table 42- Parameters of the ADPM-NETWORK-STATUS.indication primitive

Name	Type	Valid Range	Description
Status	Status	PAN_ID_CONFLICT, or any status code returned by MLME-COMM-STATUS.indication	The status or event to notify
AdditionalInformation	String	Any string	The eventual additional information to the status or event

5.3.2.17.2 When generated

This primitive is generated when the adaptation layer of a PAN coordinator has received a LBP message from a device on the network indicating that a PAN Id conflict is occurring. See clause 5.7.2 of the present document for complete description of the PAN ID conflict handling mechanism. In that case, this primitive is never generated by the adaptation layer of a device which is not a PAN coordinator.

This primitive is also generated if the underlying MAC layer (of a PAN coordinator or a coordinator) generates a MLME-COMM-STATUS.indication (see clause 7.3.3.7 of the present document).

5.3.2.17.3 Effect on receipt

On reception, the upper layer of a PAN coordinator is informed that a PAN Id conflict was detected, or that a MAC event occurred.

5.3.2.18 ADPM-ROUTE-DISCOVERY.request

This primitive allows the upper layer to initiate a route discovery.

5.3.2.18.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-ROUTE-DISCOVERY.request	{
	DstAddr,
	MaxHops
	}

Table 43 specifies the parameters for the ADPM-ROUTE-DISCOVERY.request primitive.

Table 43- Parameters of the ADPM-ROUTE-DISCOVERY.request primitive

Name	Type	Valid Range	Description
DstAddr	Short address	See [rfc4944]	The Short unicast destination address of the route discovery.
MaxHops	Integer	0x00-0x07	This parameter indicates the maximum number of hops allowed for the route discovery.

5.3.2.18.2 When generated

This primitive is generated by the upper layer of a device to obtain a route to another device.

5.3.2.18.3 Effect on receipt

An ADPM-ROUTE-DISCOVERY.confirm with the status INVALID_REQUEST is generated if the DstAddr is not a unicast IPv6 address, or if the MaxHops value is out of range.

On receipt of this primitive, the device will initiate a route discovery procedure as described in clause 5.4.3 of the present document.

5.3.2.19 ADPM-ROUTE-DISCOVERY.confirm

This primitive allows the upper layer to be informed of the completion of a route discovery.

5.3.2.19.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-ROUTE-DISCOVERY.request	{
	Status
	}

Table 44 specifies the parameters for the ADPM-NETWORK-LEAVE.confirm primitive.

Table 44- Parameters of the ADPM-NETWORK-LEAVE.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_REQUEST, ROUTE_ERROR	The status of the route discovery

5.3.2.19.2 When generated

This primitive is generated by the adaptation layer on completion of a route discovery as described in clause 5.4.3 of the present document, and in [draft-load]

5.3.2.19.3 Effect on receipt

On reception of this primitive, the upper layer is informed on the completion of the route discovery. If the Status value is SUCCESS, the routing table has been correctly updated with a brand new route to the desired destination, and the device may begin sending frames to that destination.

5.3.2.20 ADPM-PATH-DISCOVERY.request

This primitive allows the upper layer to initiate a path discovery.

5.3.2.20.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-PATH-DISCOVERY.request	{
	DstAddr,
	}

Table 43 specifies the parameters for the ADPM-ROUTE-DISCOVERY.request primitive.

Table 45- Parameters of the ADPM-PATH-DISCOVERY.request primitive

Name	Type	Valid Range	Description
DstAddr	short address	0-1199	The short unicast destination address of the path discovery.

5.3.2.20.2 When generated

This primitive is generated by the upper layer of a device to obtain the path to another device.

5.3.2.20.3 Effect on receipt

An ADPM-PATH-DISCOVERY.confirm with the status INVALID_REQUEST is generated if the DstAddr is not in the routing table or after the failure of the procedure.

On receipt of this primitive, the device will initiate a path discovery procedure as described in clause 5.4.4 of the present document.

5.3.2.21 ADPM-PATH-DISCOVERY.confirm

This primitive allows the upper layer to be informed of the completion of a path discovery.

5.3.2.21.1 Semantics of the service primitive

The semantics of this primitive are as follows:

ADPM-ROUTE-DISCOVERY.request	{
	DstAddr,
	NSDU
	}

Table 44 specifies the parameters for the ADPM-NETWORK-LEAVE.confirm primitive.

Table 46- Parameters of the ADPM-NETWORK-LEAVE.confirm primitive

Name	Type	Valid Range	Description
DstAddr	Short address	0-1199	The Short unicast destination address of the path discovery.
NsduId	integer	N.C	The buffer containing addresses of nodes constituting the path

5.3.2.21.2 When generated

This primitive is generated by the adaptation layer on completion of a path discovery as described in clause 5.4.4 of the present document.

5.3.2.21.3 Effect on receipt

On reception of this primitive, the upper layer is informed on the completion of the path discovery.

5.3.2.22 ADPM-LBP.request

This primitive allows the upper layer of client to send the LBP message to server modem .

5.3.2.22.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-LBP.request{
                    DstAddrType,
                    DstAddr,
                    NsduLength,
                    Nsdu,
                    NsduType,
                    MaxHops,
                    DiscoveryRoute,
                    QualityOfService,
                    SecurityEnable
                    }
```

Table 44 specifies the parameters for the ADPM-LBP.request primitive.

Table 47- Parameters of the ADPM-LBP.request primitive

Name	Type	Valid Range	Description
DstAddrType	Integer	0x01 - 0x02	The type of destination address contained in the DstAddr parameter. The allowed values are: 0x01 = 2 Bytes address (LBA address) 0x02 = 8 Bytes address (LBD address
DstAddr	Set of octets	-	16 bits address of LBA or 64 bits (extended address of LBD)
NsduLength	Integer	0 - 1280	The size of the NSDU, in bytes
Nsdu	Set of octets	-	The NSDU to send
NsduHandle	Integer	0x00 - 0xFF	The handle of the NSDU to transmit. This parameter is used to identify in the ADPM-LBP.confirm primitive which request is concerned. It can be randomly chosen by the application layer.
NsduType	Integer	0x00-0x03	The type of data contained in the NSDU. 0x00 = any data 0x01 = UDP 0x02 = ICMP 0x03 = TCP
MaxHops	Integer	0x00-0x07	The number of times the frame will be repeated by network routers.
DiscoveryRoute	Boolean	TRUE-FALSE	If TRUE, a route discovery procedure will be performed prior to sending the frame if a route to the destination is not available in the routing table. If FALSE, no route discovery is performed.

Name	Type	Valid Range	Description
QualityOfService	Integer	0x00-0x01	The required quality of service (QoS) of the frame to send. Allowed values are: 0x00 = standard priority 0x01 = high priority
SecurityEnabled	Boolean	TRUE-FALSE	If TRUE, this parameter enables the ADP layer security for processing the frame.

5.3.2.22.2 When generated

This primitive is generated by the client LBPServer to perform the authentication, re-keying and leave procedure.

5.3.2.22.3 Effect on receipt

On reception of this primitive, the modem sends the coming frame to the destination.

5.3.2.23 ADPM-LBP.confirm

This primitive reports the result of a previous ADPM-LBP.request primitive.

5.3.2.23.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-LBP.confirm {
                    Status,
                    NsduHandle,
                    }
```

Table 48 specifies the parameters for the ADPM-LBP.confirm primitive.

Table 48- Parameters of the ADPM-LBP.confirm primitive

Name	Type	Valid Range	Description
Status	Status	SUCCESS, INVALID_REQUEST, NO_KEY, BAD_CCM_OUTPUT, ROUTE_ERROR, BT_TABLE_FULL, FRAME_NOT_BUFFERED or any status values returned from security suite or the MCPS-DATA.confirm primitive	The status code of a previous ADPM-LBP.request identified by its NsduHandle
NsduHandel	Integer	0x00 - 0xFF	The handle of the NSDU confirmed by this primitive

5.3.2.23.2 When generated

This primitive is generated in response to a ADPM-LBP.request primitive, the Status parameter indicating if the request succeeded, or the reason of failure.

5.3.2.23.3 Effect on receipt

On reception of this primitive, the upper layer is notified of the status of a previous ADPM-LBP.request primitive.

5.3.2.24 ADPM-LBP.indication

This primitive is used to transfer received LBP frame from the ADP layer to the upper layer.

5.3.2.24.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-LBP.request{
                    DstAddr,
                    SrcAddr,
                    NsduLength,
                    Nsdu,
                    NsduType,
                    LinkQualityIndicator,
                    SecurityEnabled
                    }
```

Table 49 specifies the parameters for the ADPM-LBP.indication primitive.

Table 49- Parameters of the ADPM-LBP.indication primitive

Name	Type	Valid Range	Description
DstAddr	Integer	0x0000-0xFFFF	16 bits final destination address
SrcAddr	Integer	0x0000-0xFFFF	16 bits original source address
NsduLength	Integer	0 - 1280	The size of the NSDU, in bytes
Nsdu	Set of octets	-	The NSDU to send
NsduType	Integer	0x00-0x03	The type of data contained in the NSDU. 0x00 = any data 0x01 = UDP 0x02 = ICMP 0x03 = TCP
LinkQualityIndicator	Integer	0x00-0xFF	The value of the link quality during reception of the frame
SecurityEnabled	Boolean	TRUE-FALSE	If TRUE, this parameter enables the ADP layer security for processing the frame.

5.3.2.24.2 When generated

This primitive is generated by the ADP layer of client modem when a valid LBP frame whose final destination is the current station has been received.

5.3.2.24.3 Effect on receipt

On generation of this primitive, the upper layer is notified of the arrival of a LBP frame.

5.3.2.25 ADPM-BUFFER.indication

This primitive allows the next higher layer to be notified when the modem reach his limit capability to perform coming frame.

5.3.2.25.1 Semantics of the service primitive

The semantics of this primitive are as follows:

```
ADPM-BUFFER.indication{
                        BufferReady,
                        }
```

Table 50 specifies the parameters for the ADPM-BUFFER.indication primitive.

Table 50- Parameters of the ADPM-BUFFER.indication primitive

Name	Type	Valid Range	Description
BufferReady	Boolean	TRUE-FALSE	TRUE : modem is ready to receipt more data frame FALSE : modem is not ready, stop sending data frame

5.3.2.25.2 When generated

This primitive is generated when the adaptation layer of a modem has reached his limit to perform more Data frame.

5.3.2.25.3 Effect on receipt

On reception, the upper layer should stop the data flow if BufferReady is equal to FALSE and open it if BufferReady is TRUE.

5.3.3 Behavior to MAC Indications

This clause describes the behavior of the adaptation layer in response to an unsolicited indication from the MAC layer.

5.3.3.1 MCPS-DATA.indication

On reception of this indication, the adaptation layer must execute the routing algorithm as described in clause 5.4.2 of the present document.

5.3.3.2 MLME-ASSOCIATE.indication

Nothing must be done upon reception of this primitive by the adaptation layer.

5.3.3.3 MLME-DISASSOCIATE.indication

Nothing must be done upon reception of this primitive by the adaptation layer.

5.3.3.4 MLME-BEACON-NOTIFY.indication

When a MLME-BEACON-NOTIFY.indication is received, and if an ADPM-DISCOVERY.request is currently operating, the adaptation layer must add the PANId to the PANDescriptorList which will be forwarded to the upper layer in the ADPM-DISCOVERY.confirm primitive

5.3.3.5 MLME-GTS.indication

Nothing must be done upon reception of this primitive by the adaptation layer.

5.3.3.6 MLME-ORPHAN.indication

Nothing must be done upon reception of this primitive by the adaptation layer.

5.3.3.7 MLME-COMM-STATUS.indication

On reception of this primitive, the adaptation layer must generate an ADPM-NETWORK-STATUS.indication primitive, with the Status parameter equal to that of the MLME-COMM-STATUS.indication primitive, and the AdditionalInformation parameter equal to the concatenation of the SrcAddr and DstAddr, separated by a “:”.

5.3.3.8 MLME-SYNC-LOSS.indication

Nothing must be done upon reception of this primitive by the adaptation layer.

5.4 Annex 4

5.4.1 Channel access

The channel access is accomplished by using the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism with a random backoff time. The random backoff mechanism spreads the time over which stations attempt to transmit, thereby reducing the probability of collision. Each time a device wishes to transmit data frames, it shall wait for a random period. If the channel is found to be idle, following the random backoff, the device shall transmit its data. If the channel is found to be busy, following the random backoff, the device shall wait for another random period before trying to access the channel again.

A Carrier sense is a fundamental part of the distributed access procedure. Physical Carrier Sense (PCS) is provided by the PHY upon detection of the Preamble. In the latter case, PCS shall stay high long enough to be detected and Virtual Carrier Sense (VCS) to be asserted by the MAC. A virtual carrier sense mechanism is provided by the MAC by tracking the expected duration of channel occupancy. Virtual carrier sense is set by the length of received packet or upon collision. In these cases, virtual carrier sense tracks the expected duration of the Busy state of the medium. The medium shall also be considered Busy when the station is transmitting.

A VCS timer is maintained by all stations to improve reliability of channel access. The VCS timer is set based on received long (data) or short (ACK) frames. The VCS timer is also set upon collision or when the station powers up. Stations use this information to compute the expected Busy condition of the medium or the expected duration of the Contention State and store this information in the VCS timer.

A Collision occurs in each of the following circumstances:

- The transmitting station receives a something other than ACK or NACK response when a response is expected.
- The transmitting station shall infer a Collision from the absence of any response to a transmission when a response is expected. Note that the absence of a response could also be the result of a bad channel. Since there is no way to distinguish between the two causes a Collision is inferred.

5.4.2 Interframe (IFS) Spacing

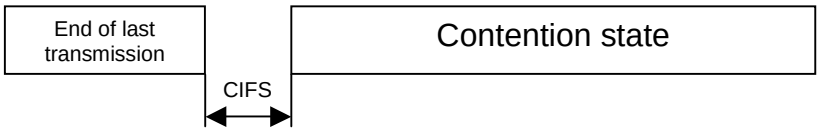
A time intervals between frames on the medium constitute the Interframe Space and are necessary due to propagation and processing time. Three interframe space values are defined. Contention Interframe Space (CIFS) occurs after the end of the previous transmission. The second defined interval is the Response Interframe Space (RIFS).

RIFS is the time between the end of a transmission and the start of its associated response. If no response is expected, the CIFS is in effect.

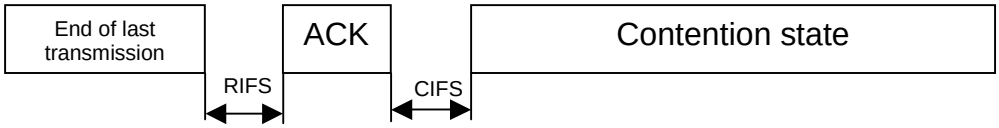
An Extended Interframe Space (EIFS) is defined for conditions when the station does not have complete knowledge of the state of the medium. This can occur when the station initially attaches to the network, when errors in the received frames make them impossible to decode unambiguously. If a packet is received and correctly decoded before the expiration of the EIFS, then the EIFS is cancelled. The EIFS is significantly longer than the other interframe spaces, providing protection from Collision for an ongoing frame transmission or segment burst when any of these conditions occur. The EIFS is calculated as follows:

aEIFS = *aAckTime* + *aCIFS* + *aRIFS* + MaxFrameSize**aSymbolTime*

Interframe space if no response expected



Interframe space if response expected



Extended interframe space (EIFS) period

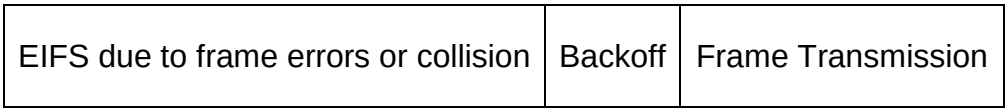


Figure 22- IFS

5.4.3 CSMA-CA

The present specification supports only an unslotted version of the CSMA-CA algorithm for non-beacon PAN described in IEEE 802.15.4

The random backoff mechanism spreads the time over which stations attempt to transmit, thereby reducing the probability of collision, using a truncated binary exponential backoff mechanism.

The CSMA-CA algorithm shall be used before the transmission of data or MAC command frames

The algorithm is implemented using units of time called backoff periods, where one backoff period shall be equal to u*nitBackoffPeriod* symbols.

Each device shall maintain two variables for each transmission attempt: ***N*B** and ***B*E**. ***NB*** is the number of times the CSMA-CA algorithm was required to backoff while attempting the current transmission; this value shall be initialized to 0 before each new transmission attempt.

BE is the backoff exponent, which is related to how many backoff periods a device shall wait before attempting to assess a channel. ***BE*** shall be initialized to the value of ***minB*E**.

Note that if ***minBE*** is set to 0, collision avoidance will be disabled during the first iteration of this algorithm. Figure 69 illustrates the steps of the CSMA-CA algorithm. The MAC sublayer shall first initialize ***NB***, and ***BE*** [step (1)] and then proceed directly to step (2).

The MAC sublayer shall delay for a random number of complete backoff periods in the range 0 to 2^{BE} –1 [step (2)] and then request that the PHY perform a PCS (Physical Carrier Sense) [step (3)].

$$\text{Backoff Time} = \text{Random}(2^{BE} - 1) \times \text{aSlotTime}$$

If the channel is assessed to be busy [step (4)], the MAC sublayer shall increment both ***NB*** and ***BE*** by one, ensuring that ***BE*** shall be no more than ***maxB*E**. Note: for high priority packets ***maxB*E** should be equal to ***minBE.***

If the value of ***NB*** is less than or equal to ***maxCSMABackoff*s**, the CSMA-CA algorithm shall return to step (2).

If the value of ***NB*** is greater than ***maxCSMABackoff*s**, the CSMA-CA algorithm shall terminate with a Channel Access Failure status.

If the channel is assessed to be idle [step (5)], the MAC sublayer shall begin transmission of the frame immediately.

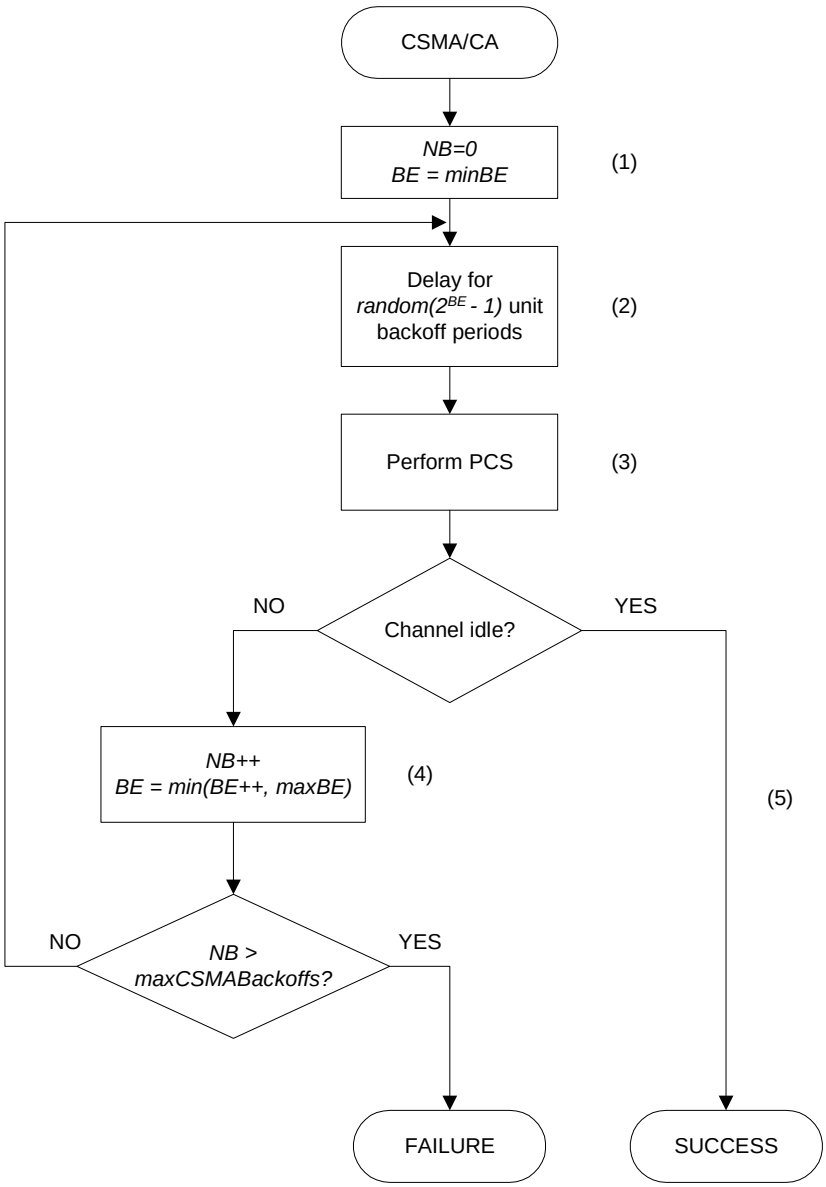


Figure 23 - The CSMA/CA Algorithm

5.4.4 Priority

Prioritized access to the channel can be beneficial for real time application or control application when urgent message should be delivered as soon as possible.

Only two levels of priority (**High and Normal**) will be used to minimize complexity.

Priority resolution is implemented by using two contention time windows during contention state as shown in Figure 24.

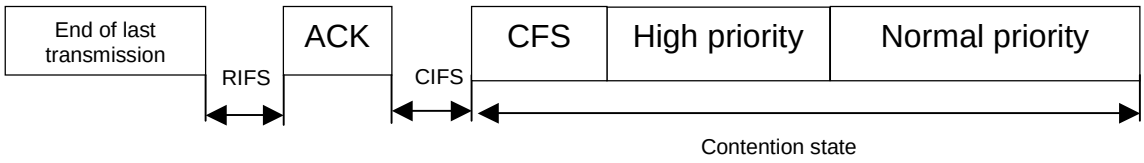


Figure 24 - Priority Contention Windows

First slot of contention window is called Contention Free Slot (CFS). It is used to implement packet bursting without backoff procedure in order to prevent possible interruption from other nodes.

The high and normal priority stations will compete for channel during HPCW and NPCW correspondingly. Since HPCW is located before NPCW high priority stations will get access to the channel before station with normal priority. Duration of HPCW and NPCW are calculated as follow:

HPCW time = *macHighPrioirtyWindowSize * aSlotTime;*

NPCW time = (2^{maxBE} * *aSlotTime)* - HPCW time;

CFS time = *aSlotTime*;

5.4.5 ARQ

ARQ (Automatic Repeat reQuest) is implemented based on acknowledged and unacknowledged retransmission. The MAC uses a response type as part of its **ARQ** mechanism. **ACK** is a traditional positive acknowledgment that when received allows the transmitter to assume successful delivery of the frame. The negative acknowledgment (NACK) is used to inform a packet originator that the receiver received the packet but it was corrupted.

A successful reception and validation of a data can be confirmed with an acknowledgment. If the receiving device is unable to handle the received data frame for any reason, the message is not acknowledged.

If the originator does not receive an acknowledgment after waiting period, it assumes that the transmission was unsuccessful and retries the frame transmission. If an acknowledgment is still not received after several retries, the originator can choose either to terminate the transaction or to try again. When the acknowledgment is not required, the originator assumes the transmission was successful. Also if acknowledgment is not required, the originator can retransmit the same packets few times to increase probability of data delivery. The receiver should be able distinguish and discard redundant copies using the **Sequence Number** and **Segment Count**.

The retransmitted packet will have the same **SequenceNumber** and **Segment Count** as original.

The acknowledgment cannot be requested for broadcast or multicast transmission. On transmit side ARQ requires configurable number of retransmissions (*macMaxFrameRetries* from 7.4.2 of [802.15.4-2006]) as shown in Figure 25.

On receive side ARQ generates acknowledgement for PLC packet with correct FCS (CRC16) if packet corresponds to this address as shown in Figure 26.

The received packet FCS (16 bit) will be sent back to the packet originator as a part of an acknowledgement (Frame Control Header).

All nodes will detect ACK during response time but only one station expecting ACK will accept it as acknowledgement and use 16 bit of FCS from ACK for identification.

MAC acknowledgement is described in details in section 7.6 of this document.

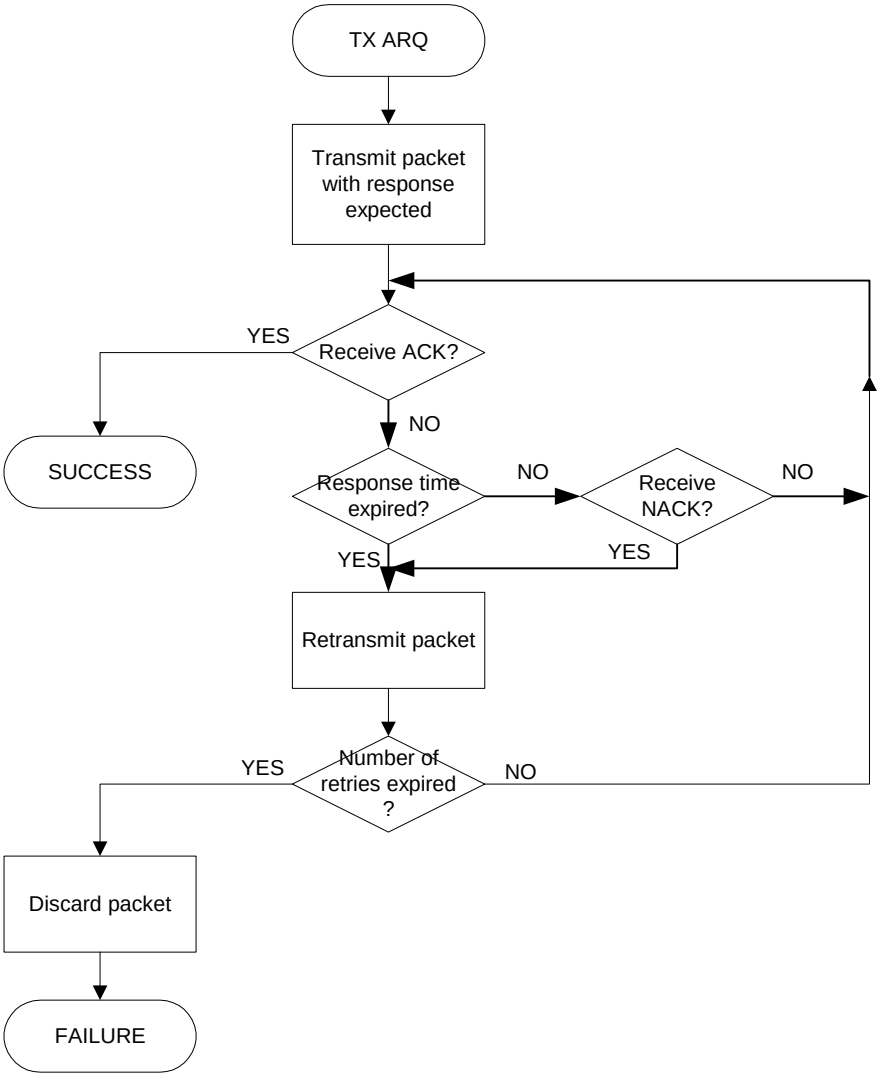


Figure 25 - Transmit ARQ

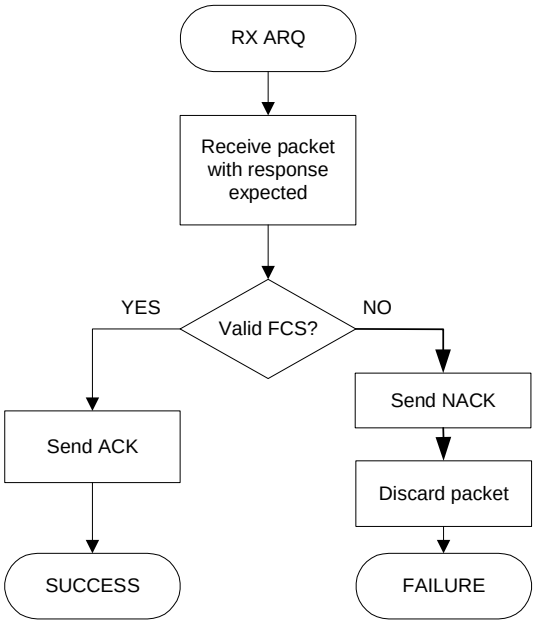


Figure 26 - Receive ARQ

5.4.6 Segmentation and reassembly overview

Since PHY specification supports different types of modulation and tone map a number of data bytes of PHY payload can be changed dynamically based on channel condition. This requires implementing MAC payload fragmentation on MAC sublayer.

If the MAC payload is too large to fit wholly within an MSDU, it must be partitioned into smaller segments that can each fit within an MSDU. This process of partitioning MAC frame into MSDU s is called segmentation.

The segmentation may require to add padding bytes to the last segment in order fill last PHY frame. The reverse process is called reassembly. The segmentation improves the probability of delivery over harsh channels and contributes to better latency characteristics for all stations by restricting the length of each individual transmission.

All forms of addressed delivery (unicast, multicast, and broadcast) are subject to segmentation. Acknowledgments and retransmissions occur independently for each segment.

The Segment Control fields: **SL**, **SC** and **LSF** are used to keep track of segments of fragmented packet and assembly whole packet on receiver side.

Segment control fields:

Field	Byte	Bit number	Bits	Definition
RES	0	7-4	4	Reserved
TMR	0	3	1	Tone map request: 1: Tone map is requested 0: Tone map is not requested
CC	0	2	1	Contention Control: 0: contention is allowed in next contention state 1: contention free access
CAP	0	1	1	Channel access priority: 0: Normal 1: High
LSF	0	0	1	Last Segment Flag is set for last segment only
SC	1	7-2	6	Segment Count
SL[9-8]	1	1-0	2	Segment Length of MAC frame
SL[7-0]	2	7-0	8	Segment Length of MAC frame

5.5 Annex 5: Modified MAC Data Primitives

5.5.1 MCPS-DATA.request

5.5.1.1 Semantics of the service primitive

The semantics of the MCPS-DATA.request primitive are as follows:

The table below specifies the parameters for the MCPS-DATA.request primitive.

MCPS-DATA.request	(SrcAddrMode, DstAddrMode, DstPANId, DstAddr, msduLength, msdu, msduHandle, TxOptions, SecurityLevel, KeyIdMode, KeySource, KeyIndex, QualityOfService)

Table 45 - MCPS-DATA.request parameters

Name	Type	Valid range	Description
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted, see 7.2.1.1.8). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstAddrMode	Integer	0x00–0x03	The destination addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted, see 7.2.1.1.6). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstPANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the entity to which the MSDU is being transferred.

Name	Type	Valid range	Description
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.
msduLength	Integer	*aMaxMACPayloadSize*	The number of octets contained in the MSDU to be transmitted by the MAC sublayer entity.
msdu	Set of octets	—	The set of octets forming the MSDU to be transmitted by the MAC sublayer entity.
msduHandle	Integer	0x00–0xff	The handle associated with the MSDU to be transmitted by the MAC sublayer entity.
TxOptions	Bitmap	3-bit field	The 3 bits (b0 , b1 , b2) indicate the transmission options for this MSDU. For b0 , 1 = acknowledged transmission, 0 = unacknowledged transmission. For b1 , 1 = GTS transmission, 0 = CAP transmission for a beacon-enabled PAN. For b2 , 1 = indirect transmission, 0 = direct transmission. For a nonbeacon-enabled PAN, bit b 1 should always be set to 0.
QualityOfService	Integer	0x00–0x02	The QOS (Quality of Service) parameter of the MSDU to be transmitted by the MAC sublayer entity. This value can take one of the following values: 0 = Normal priority; 1 = High priority; 2 = Contention free;
SecurityLevel	Integer	0x00–0x07	The security level to be used (see Table95 in 7.6.2.2.1).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key to be used (see Table96 in 7.6.2.2.2). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4, or 8octets	As specified by the KeyIdMode parameter	The originator of the key to be used (see 7.6.2.4.1). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key to be used (see 7.6.2.4.2). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

5.5.2 MCPS-DATA.indication

5.5.2.1 Semantics of the service primitive

The semantics of the MCPS-DATA.indication primitive are as follows:

The table below specifies the parameters for the MCPS-DATA.indication primitive.

MCPS-DATA.request	(SrcAddrMode, SrcPANId, SrcAddr, DstAddrMode, DstPANId, DstAddr, msduLength, msdu, msduLinkQuality, DSN, Timestamp, SecurityLevel, KeyIdMode, KeySource, KeyIndex, QualityOfService)

Table 46 - MCPS-DATA.request parameters

Name	Type	Valid range	Description
SrcAddrMode	Integer	0x00–0x03	The source addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted, see 7.2.1.1.8 of [802.15.4-2006]). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
SrcPANId	Integer	0x0000-0xffff	The 16-bit PAN identifier of the device from which the frame was received.
SrcAddr	Device address	As specified by the SrcAddrMode parameter	The address of the device which sent the message.

Name	Type	Valid range	Description
DstAddrMode	Integer	0x00–0x03	The destination addressing mode for this primitive and subsequent MPDU. This value can take one of the following values: 0x00 = no address (addressing fields omitted, see 7.2.1.1.6 of [802.15.4-2006]). 0x01 = reserved. 0x02 = 16-bit short address. 0x03 = 64-bit extended address.
DstPANId	Integer	0x0000–0xffff	The 16-bit PAN identifier of the entity to which the MSDU is being transferred.
DstAddr	Device address	As specified by the DstAddrMode parameter	The individual device address of the entity to which the MSDU is being transferred.
msduLength	Integer	*aMaxMACPayloadSize*	The number of octets contained in the MSDU to be indicated to the upper layer
msdu	Set of octets	—	The set of octets forming the MSDU received by the MAC sublayer entity.
msduLinkQuality	Integer	0x00–0xff	The LQI value measured during reception of the message.
DSN	Integer	0x00-0xff	The DSN of the received frame.
Timestamp	Integer	0x00000000-0xffffffff	The time, in symbols, at which the frame was received.
QualityOfService	Integer	0x00–0x02	The QOS (Quality of Service) parameter of the MSDU received by the MAC sublayer entity. This value can take one of the following values: 0 = Normal priority; 1 = High priority; 2 = Contention free;
SecurityLevel	Integer	0x00–0x07	The security level used (see Table95 in 7.6.2.2.1 of [802.15.4-2006]).
KeyIdMode	Integer	0x00–0x03	The mode used to identify the key used (see Table96 in 7.6.2.2.2 of [802.15.4-2006]). This parameter is ignored if the SecurityLevel parameter is set to 0x00.
KeySource	Set of 0, 4, or 8octets	As specified by the KeyIdMode parameter	The originator of the key used (see 7.6.2.4.1 of [802.15.4-2006]). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.
KeyIndex	Integer	0x01–0xff	The index of the key used (see 7.6.2.4.2 of [802.15.4-2006]). This parameter is ignored if the KeyIdMode parameter is ignored or set to 0x00.

5.6 Annex 6: MAC acknowledgement

The present specification does not use IEEE802.15.4-2006 MAC acknowledgment frame but specifies a positive and negative acknowledgments using Frame Control Header (section 5.5 of PHY specification).

The Frame Control Header contains an information used by all stations in the network for channel access, as well as PHY receiver information used by the destination. For this reason, Frame Control Header has specific physical layer encoding and modulation as defined in PHY specification.

Only Frame Control Header will be used as positive (ACK) or negative (NACK) acknowledgement.

The packet originator may request an acknowledgment by setting Delimiter Type field of Frame Control Header (section 5.5 of PHY specification).

The receiver will send ACK to the originator if it is requested and the MAC frame was decoded correctly by PHY.

The receiver will send NACK to the originator if it is requested and the received MAC frame is corrupted and cannot be recovered by PHY.

ACK and NACK frames contain the 16-bit CRC (MAC FCS field) received in the MAC frame for which the ACK or NACK response is being sent. These 16 bits are used as ACK or NACK identifier and located in 2 bytes of FCH (TM[0:7] and PDC) (see 5.5 of PHY specification). The transmitter will compare against transmitted FCS to determine validity of the response. If it matches of transmitted FCS, the response is accepted. If it does not match the FCS, the response is ignored and treated as a collision.

5.7 Annex 7: Device Starting Sequence of messages

Each device should start on Not_Device_Server status and then the following procedure is performed:

1- Reset the equipment by sending the ADPM-RESET.request.

2- Set the type of the device to switch it on Device or Server mode and optionally set the PIB parameters to configure it.

- If the equipment is a device it should perform :

i. Discovery procedure by invoking the ADPM-NETWORK-DISCOVERY.request.

ii. If there is a device or a server in its pose, it must then invoke the ADPM-NETWORK-JOINNING.request to perform the bootstrapping procedure.

- Else (the equipment is a server) it should perform:

i. Discovery procedure by invoking the ADPM-NETWORK-DISCOVERY.request.

ii. If there is a device in the server's pose, it should invoke the ADPM-NETWORK-START; else it switches to joining status.

Equipment can't send or receive data or load packet unless it is joined.

3- [End of document]